SEC504 | HACKER TOOLS, TECHNIQUES, EXPLOITS, AND INCIDENT HANDLING

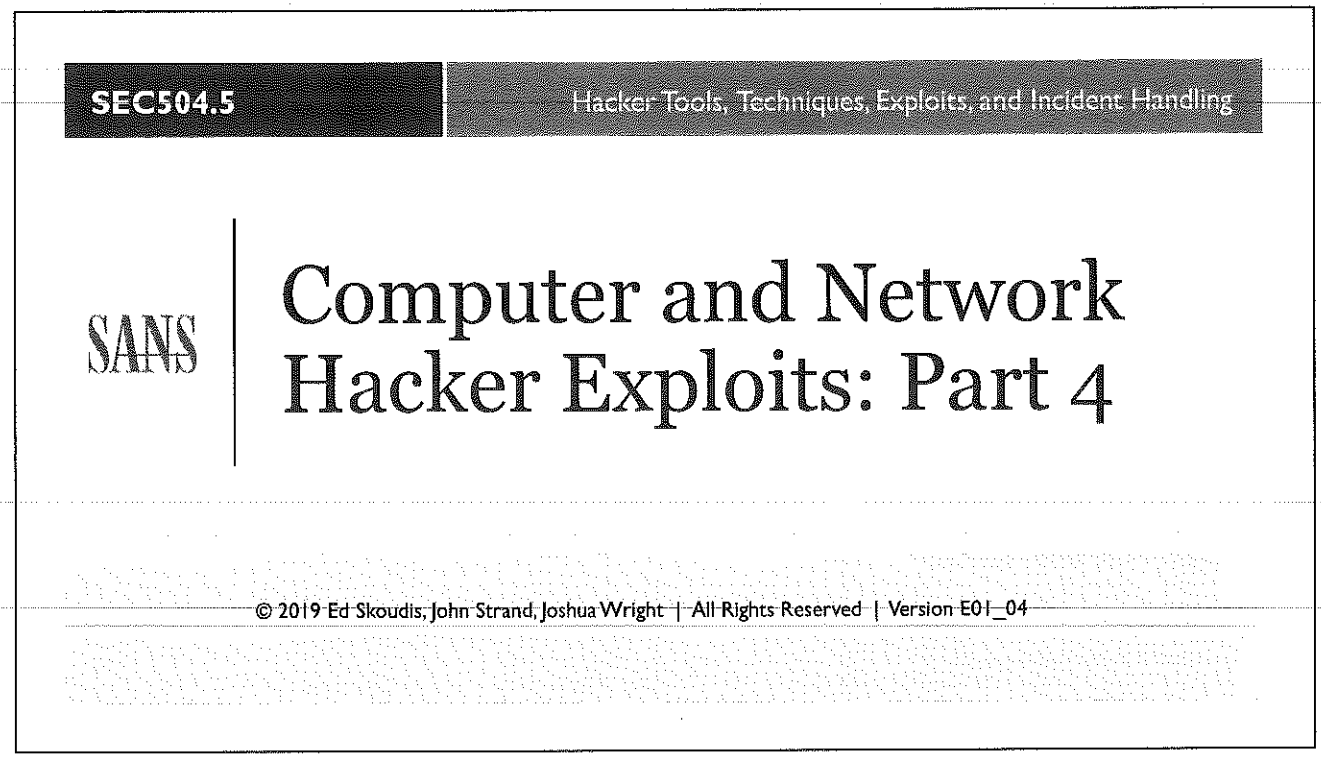
504.5
Computer and Network Hacker Exploits Part 4

SANS

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

504.5 Computer and Network Hacker Exploits Part 4

SANS



SEC504.5 Hacker Tools, Techniques, Exploits, and Incident Handling

SANS

Computer and Network Hacker Exploits: Part 4

Hello and welcome to book 5 of Hacker Tools, Techniques, Exploits, and Incident Handling.

Let's continue our journey.

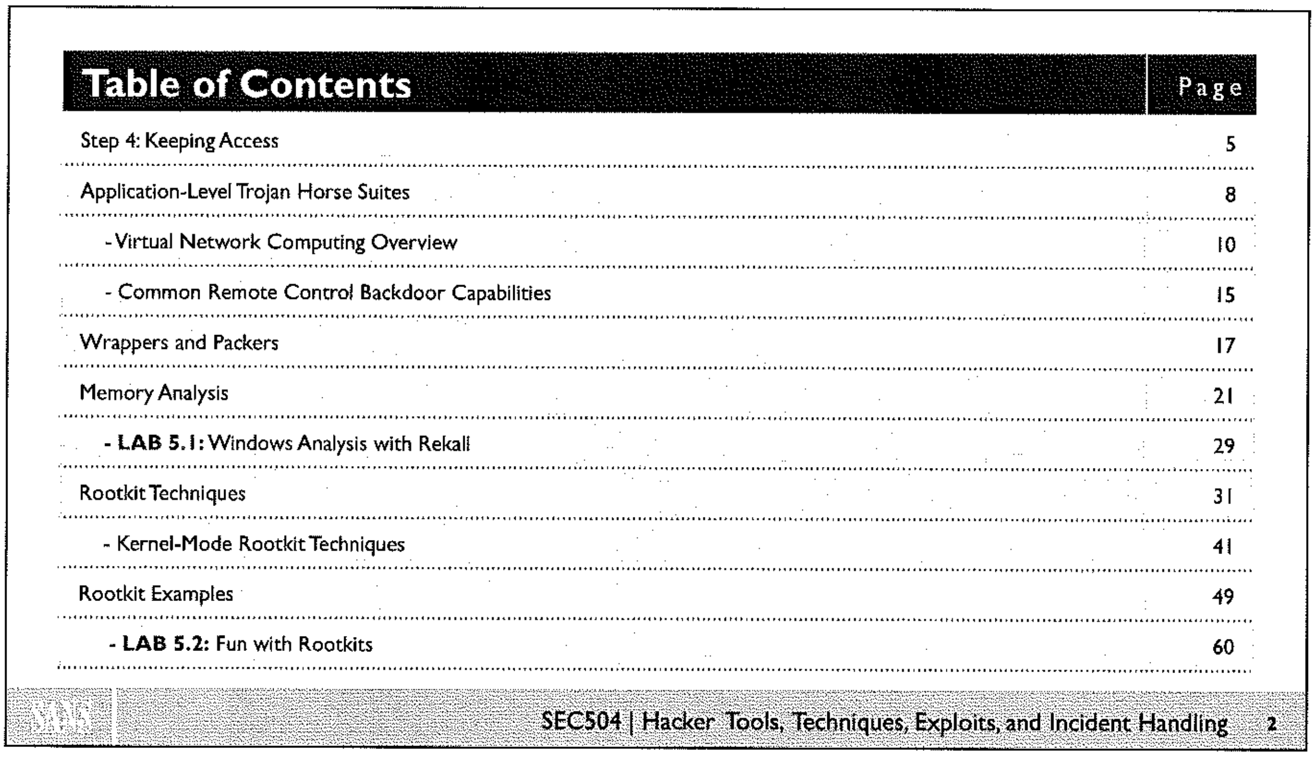


Table of Contents

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 2

This table of contents can be used for future reference.

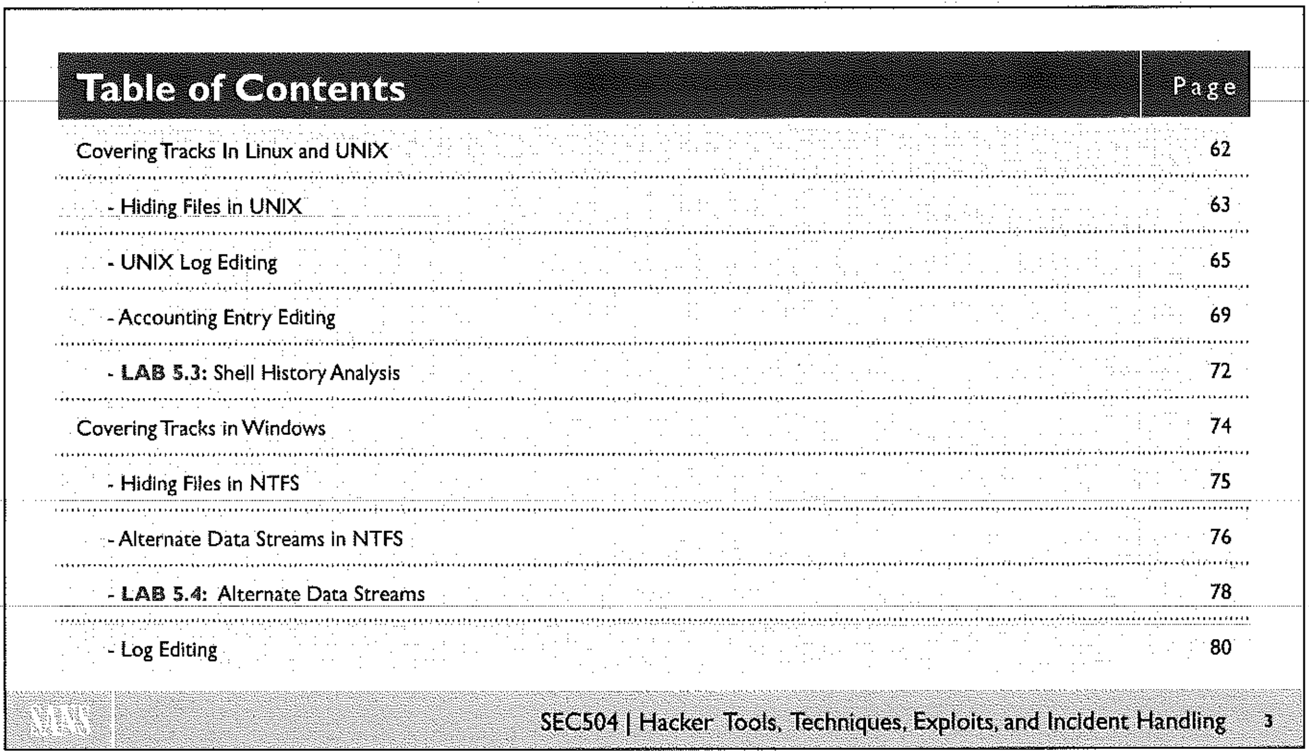


Table of Contents

This table of contents can be used for future reference.

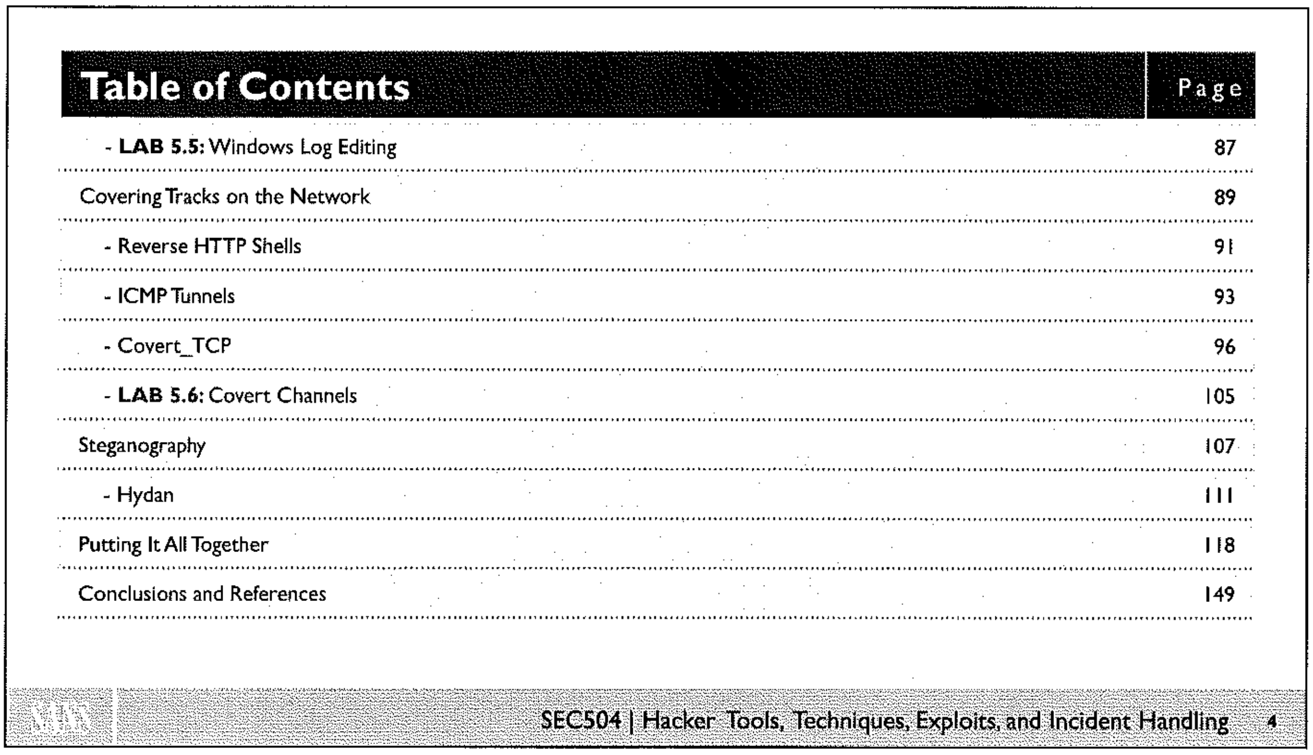
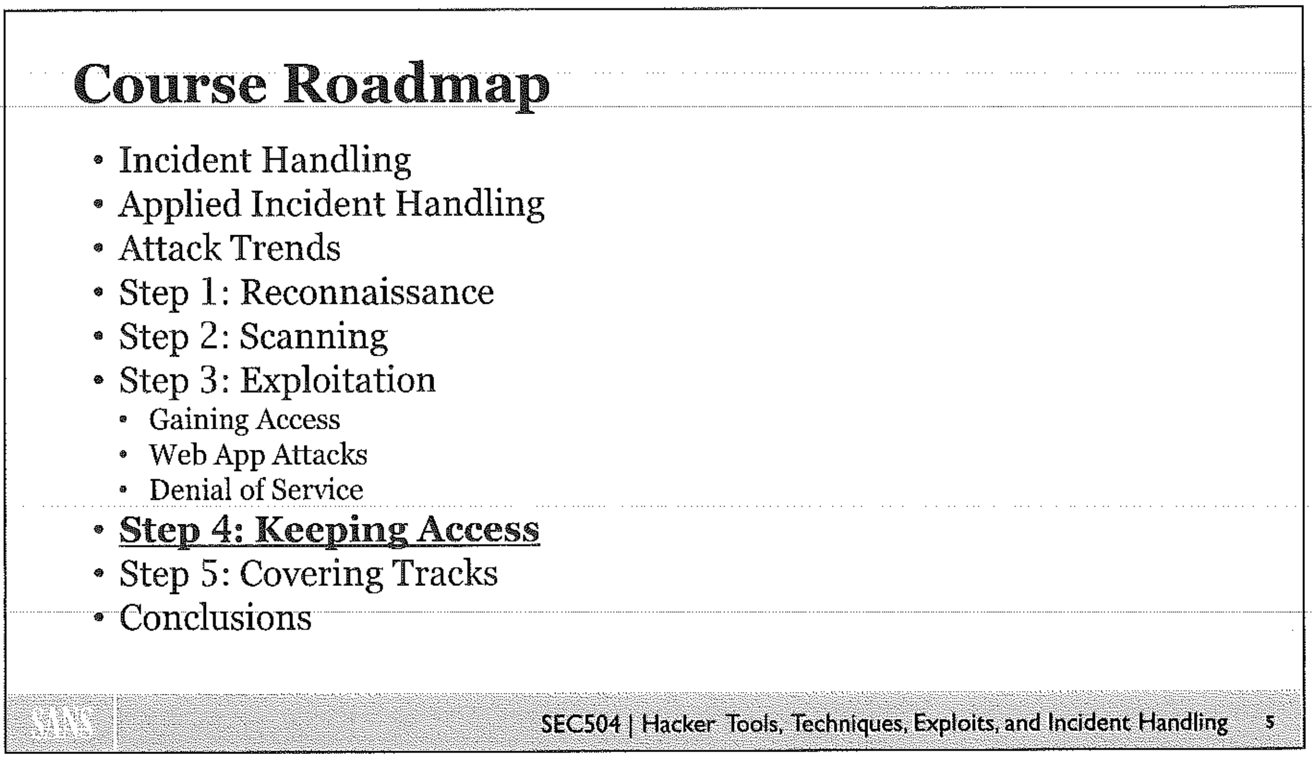


Table of Contents

This table of contents can be used for future reference.

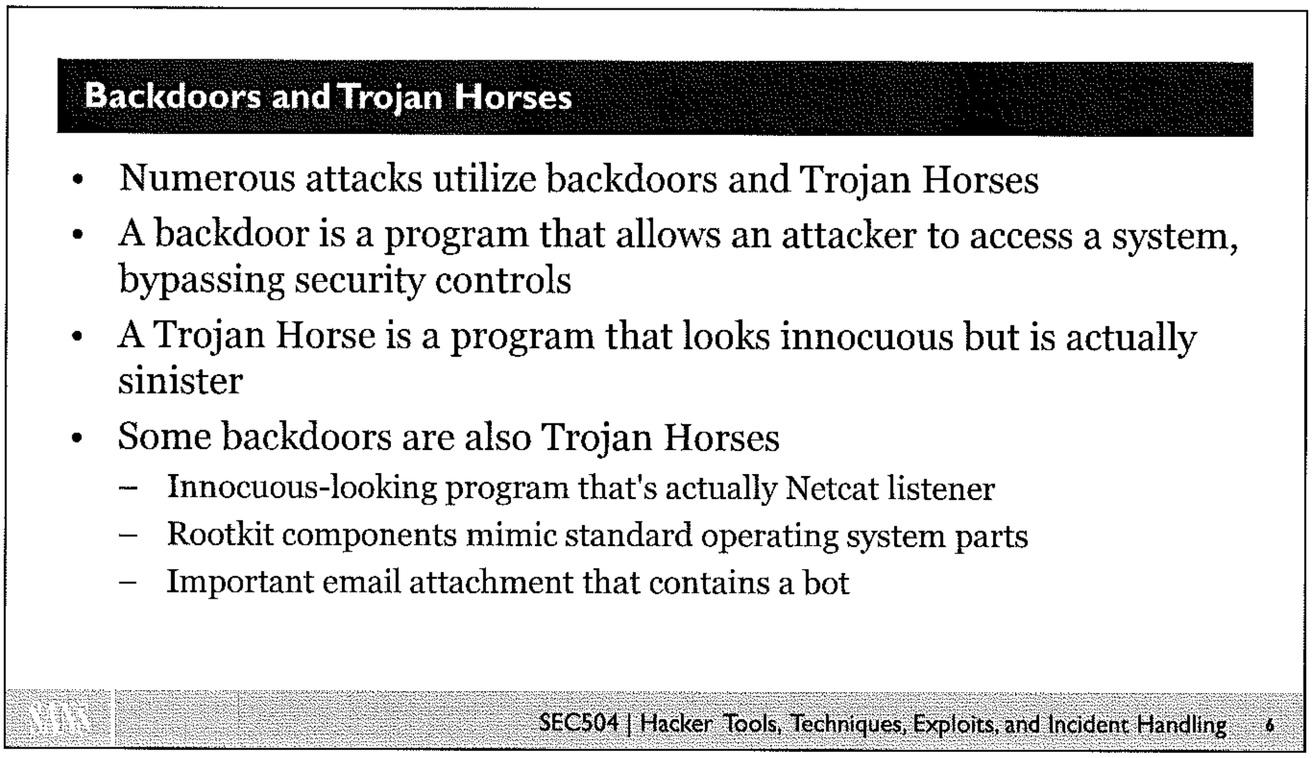


Course Roadmap

So far, we've discussed the first three steps of an attack. Let's review them quickly to help remember where we are in the overall picture of the attack:

- In Step 1, Reconnaissance, we discussed how attackers case the joint and try to find information about their target.
- In Step 2, Scanning, we talked about how an attacker probes the target looking for different ways to break in.
- In Step 3, Exploitation, we discussed how an attacker can take over a system and deny service from other users of the machine.

Now, we discuss Step 4, Keeping Access. Once attackers get into a system, they want to maintain that valued access so they can continue to access it time and time again.



Backdoors and Trojan Horses

- Numerous attacks utilize backdoors and Trojan Horses
- A backdoor is a program that allows an attacker to access a system, bypassing security controls
- A Trojan Horse is a program that looks innocuous but is actually sinister
- Some backdoors are also Trojan Horses
  - Innocuous-looking program that's actually Netcat listener
  - Rootkit components mimic standard operating system parts
  - Important email attachment that contains a bot

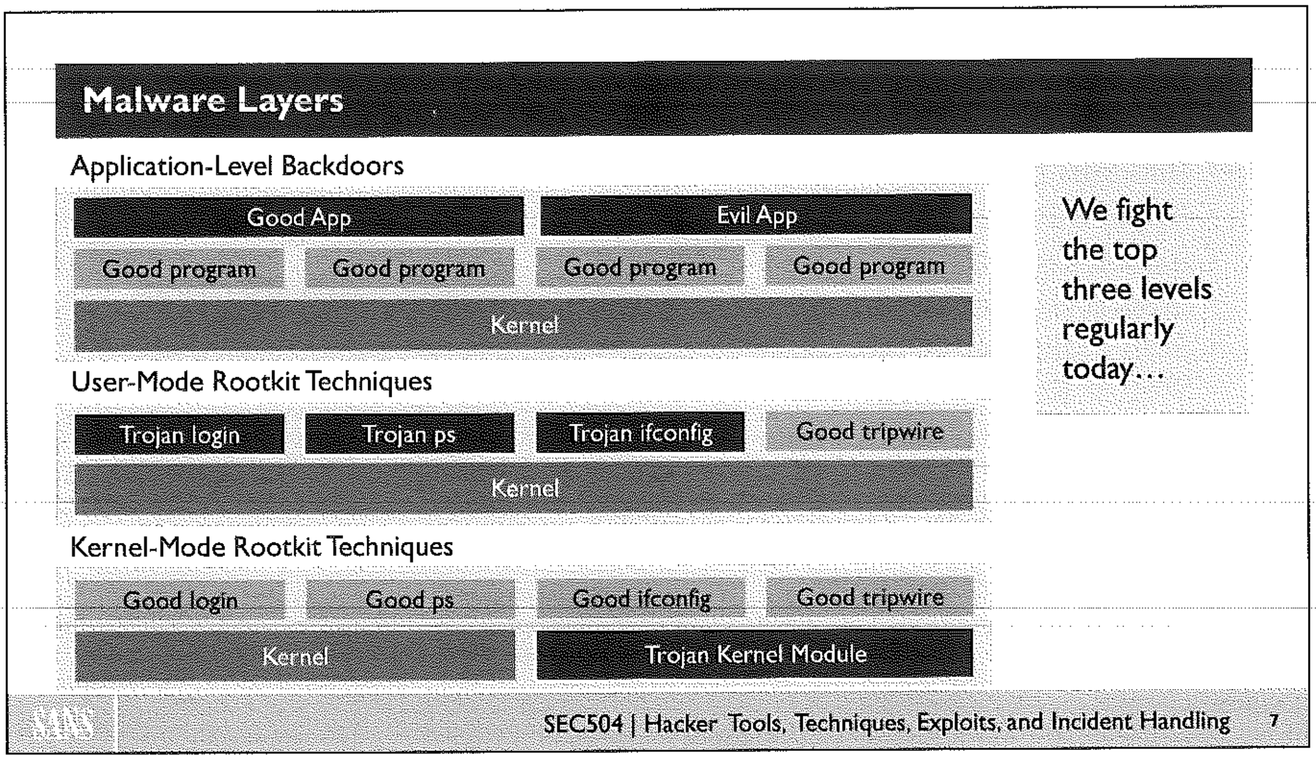
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 6

These are two techniques used by attackers to maintain their access into a system.

Numerous attackers, once they gain access, want to keep access and will use backdoors and Trojan Horses to accomplish this technique. A backdoor is a program that allows an attacker to bypass normal security controls on a system. The normal users of a system might have to type in a user ID and a password. A backdoor can allow an attacker to get around that so he doesn't have to provide a user ID and password.

A Trojan Horse is a separate concept from a backdoor. A Trojan Horse is a program that looks like it has some useful function but is actually sinister. It has some hidden capability used by the attacker.

Things get nasty and allow attackers to have a powerful technique when you meld the concept of a backdoor and Trojan Horse together. These are the so-called Trojan Horse backdoors. These are innocuous programs that might look useful, but in fact, have some capability to let the attacker get backdoor access into a system. Some examples here include a program that looks nice and useful but actually is a Netcat backdoor listener. Another type of backdoor Trojan Horse is a rootkit. We talk about these in more detail in a few slides. A rootkit alters your operating system so that, although it looks intact, it gives control to an attacker. Another example involves an important-looking email attachment from an apparently trusted source that is actually a bot installation program.



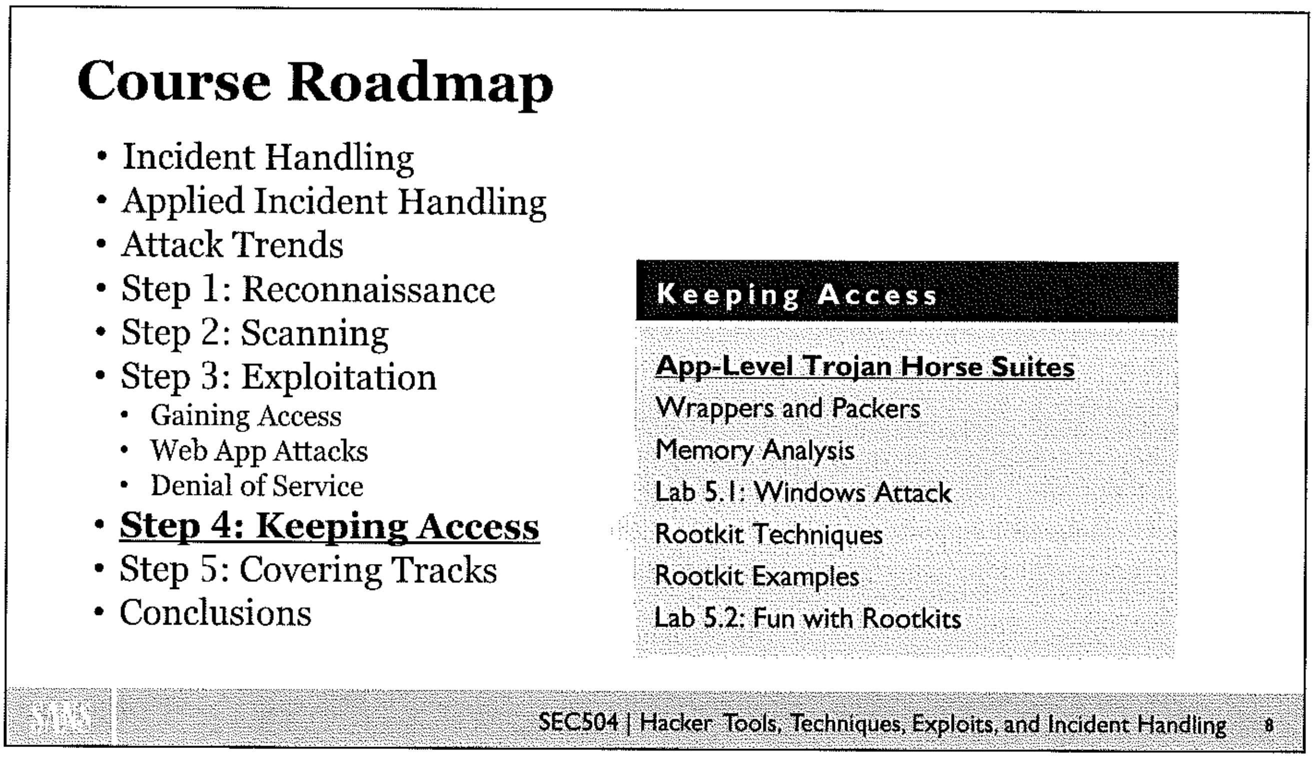
Malware Layers

We fight the top three levels regularly today…

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 7

Here we have a slide that displays where the various techniques we are going to discuss occur.

Please use it as a visual reference for the rest of this section.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- **Step 4: Keeping Access**
- Step 5: Covering Tracks
- Conclusions

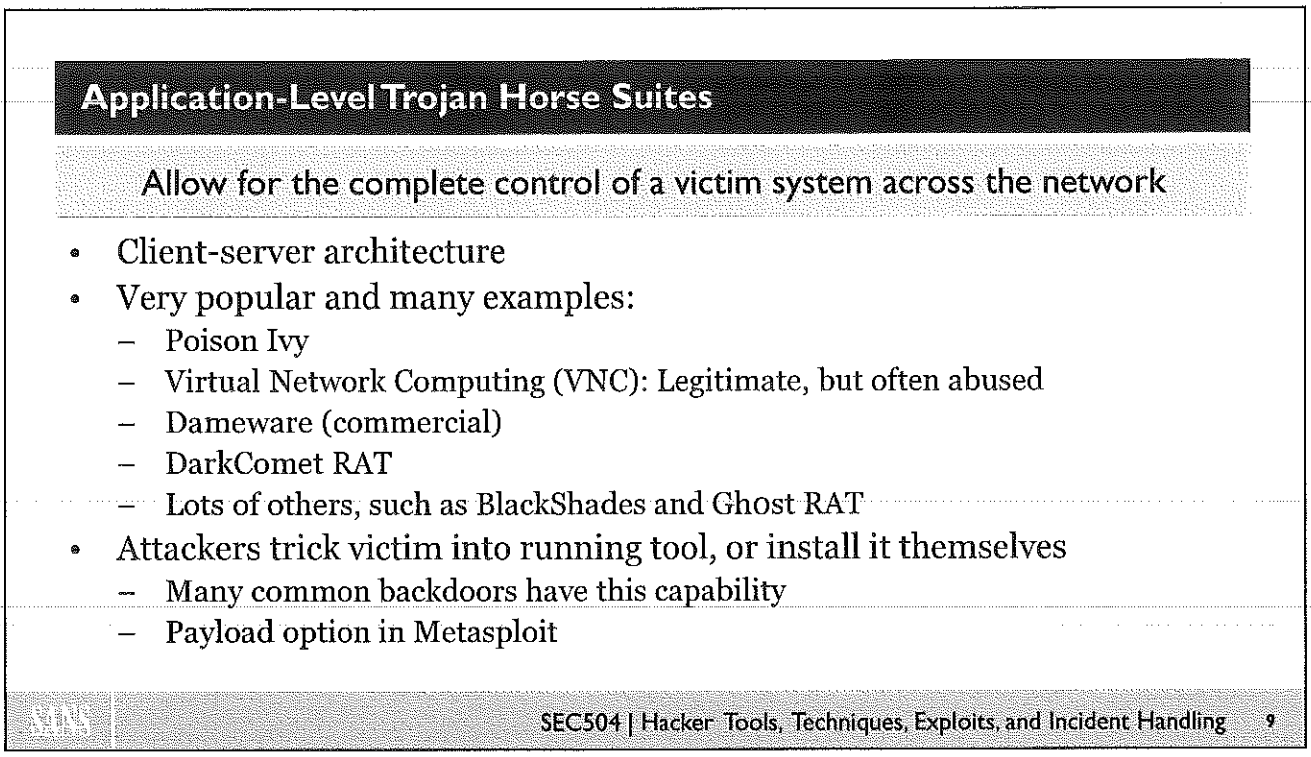
Keeping Access

App-Level Trojan Horse Suites
Wrappers and Packers
Memory Analysis
Lab 5.1: Windows Attack
Rootkit Techniques
Rootkit Examples
Lab 5.2: Fun with Rootkits

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 8

Now that we have explored some of the simple backdoor tools such as Netcat, we will explore some more powerful categories of application-level Trojan Horse backdoors.

We discuss application-level Trojan Horse backdoor suites, tools that give an attacker robust capabilities for full control of the target system across the network.



Application-Level Trojan Horse Suites

Allow for the complete control of a victim system across the network

- Client-server architecture
- Very popular and many examples:
  - Poison Ivy
  - Virtual Network Computing (VNC): Legitimate, but often abused
  - Dameware (commercial)
  - DarkComet RAT
  - Lots of others, such as BlackShades and Ghost RAT
- Attackers trick victim into running tool, or install it themselves
  - Many common backdoors have this capability
  - Payload option in Metasploit

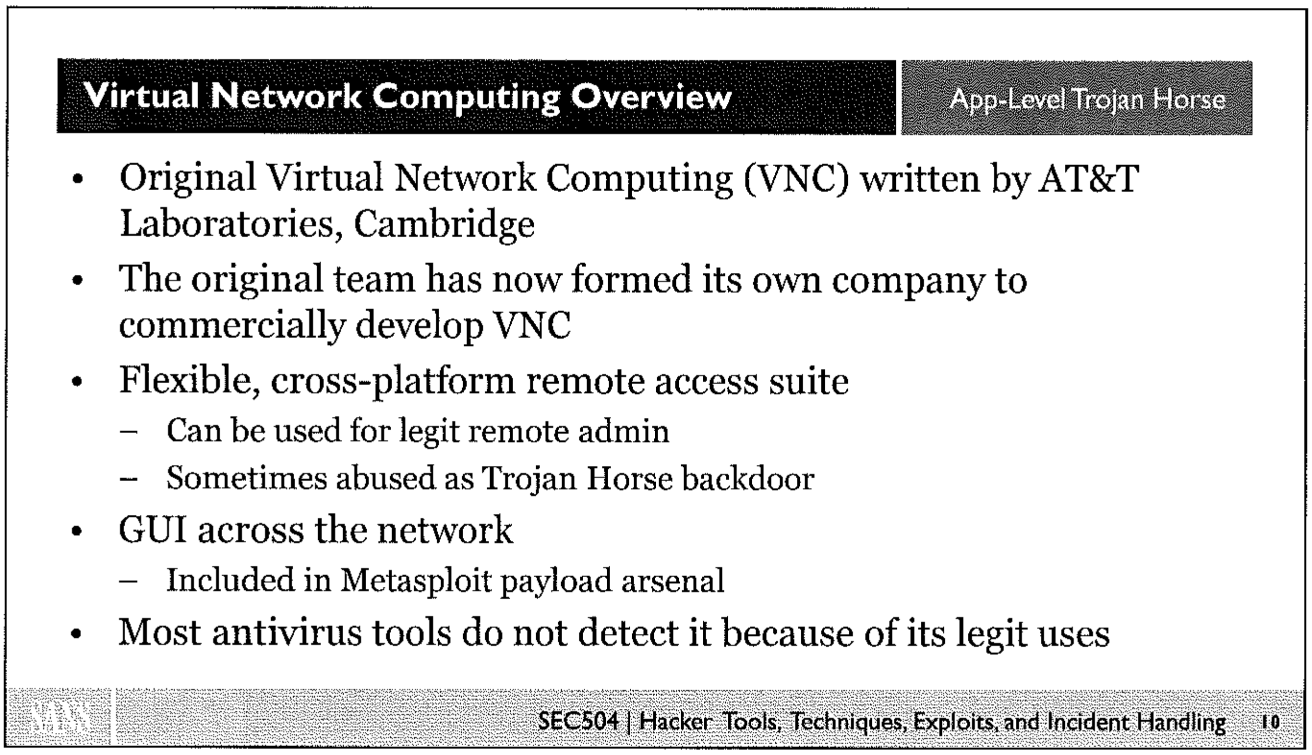
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 9

These tools allow an attacker to have complete control over a victim machine, and they have the following things in common:

- A server executable is installed on the victim's machine.
- The server is controlled from a client machine.
- The user interface allows the attacker to completely control the victim system.
- Most of these tools can be discovered with an antivirus tool. It's important to note, however, that the vast majority of antivirus tools will not detect VNC, the most popular remote control program used by good guys and attackers!

Many backdoors have the capability to install VNC because it is a great way to interact with a system in the same way a user would.

An interesting history of the use of the Poison Ivy RAT including recent campaign use is available at https://www.cyber.nj.gov/threat-profiles/trojan-variants/poison-ivy. Cases involving Gh0st RAT can also recently be seen, such as the case documented by James Quinn at https://www.alienvault.com/blogs/labs-research/the-odd-case-of-a-gh0strat-variant.



Virtual Network Computing Overview

App-Level Trojan Horse

- Original Virtual Network Computing (VNC) written by AT&T Laboratories, Cambridge
- The original team has now formed its own company to commercially develop VNC
- Flexible, cross-platform remote access suite
  - Can be used for legit remote admin
  - Sometimes abused as Trojan Horse backdoor
- GUI across the network
  - Included in Metasploit payload arsenal
- Most antivirus tools do not detect it because of its legit uses

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 10

VNC is free, popular, and quite feature rich! It uses TCP port 5900 to send a GUI across the network.

Several companies are using it for legitimate remote administration. VNC's default security is problematic. It does include a password but has been subject to monkey-in-the-middle and buffer overflow attacks in the past. VNC can be properly secured, however, especially when used in conjunction with SSH. Remember, SSH can be used to carry any TCP-based application traffic in a strongly authenticated, encrypted fashion. By setting up SSH port redirection for TCP port 5900, you can establish far more secure VNC sessions for use in administering your network. Whenever you use VNC for legitimate remote administration, we recommend that you always carry it across an SSH session using SSH protocol version 2 to benefit from the strong authentication and strong encryption SSH provides.

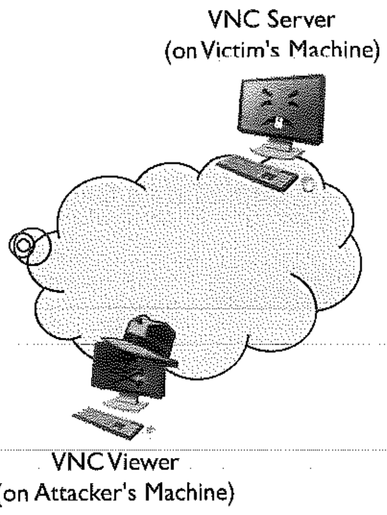
VNC is also included as a Metasploit payload, allowing for remote control of an exploited victim.

Most importantly, most antivirus tools do not detect VNC because it is such a widely used legitimate remote administration tool.

VNC is available at http://www.realvnc.com.

VNC Platforms

- Huge platform support with interoperability
  - Windows, Linux, macOS
  - Other UNIX platforms
  - Embedded device versions (non-Intel)
  - Client or server on each of these platforms
- Windows can control UNIX, and vice versa
- VNC is also a very useful Metasploit payload

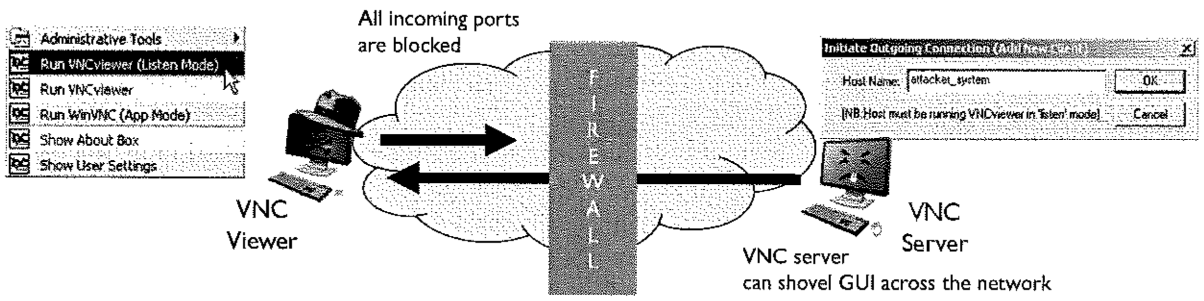


The tool works well on different platforms, including Windows, Linux, Solaris, and HP-UX 11. VNC is even available for embedded devices such as IoT platforms with non-Intel processor support (e.g. ARM, ARM64, etc.) It even works across platforms so that a Windows machine can manage a UNIX system, and vice versa. This flexibility makes VNC popular among both the good guys and the attackers. Also, VNC is a useful Metasploit payload, deployable on a target through the vast arsenal of exploits included in Metasploit.

VNC:Active Client and Listening Client

App-Level Trojan Horse

- The VNC client can run in two modes:
  - Active connection to server listening on a port (TCP 5900 by default)
  - Listening mode, waiting for server to send a connection to the client: "Shoveling" GUI
- When configured to listen, client uses TCP 5500 by default


Administrative Tools
Run VNCviewer (Listen Mode)
Run VNCviewer
Run WinVNC (App Mode)
Show About Box
Show User Settings
All incoming ports are blocked
FIREWALL
VNC Viewer
Initiate Outgoing Connection (Add New Client)
OK
Cancel
VNC Server
VNC server can shovel GUI across the network

Host Name: attacker_system

[NB.Host must be running VNCviewer in 'listen' mode]

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 12

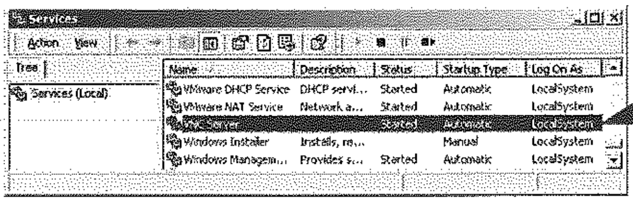
In addition, the VNC server can wait for a connection, just passively listening. By default, VNC servers listen on TCP port 5900. A VNC client can connect to that port and remotely control the GUI on the system running the VNC server. Also, VNC servers listen by default on TCP port 5800. When a browser connects to that port, VNC includes a little web server that will shoot to the browser a VNC viewer client implemented in Java. This viewer, running in the browser that connected to the VNC server, can then control the GUI of the VNC server machine, again over TCP 5900. Thus, management happens over TCP 5900; TCP 5800 just serves up a Java applet of a VNC viewer.

Alternatively, VNC can even actively send a connection to a waiting client. This latter option is shoveling the GUI across the network. The VNC viewer client listens on TCP 5500 (by default), and the VNC server initiates an outbound connection to the waiting client, as shown in the slide. With Netcat, we could shovel shell. With VNC, we can shovel GUI, pushing the GUI as an outgoing connection through a firewall.

WinVNC

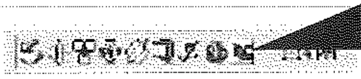
App-Level Trojan Horse

- Server can run in two modes:
  - App mode (shows up in tool tray)
  - Service mode (shows up in service list and in tool tray after reboot)
  - There is a config option to hide the tool tray icon



Name	Description	Status	Startup Type	Log On As
VMware DHCP Service	DHCP servi...	Started	Automatic	LocalSystem
VMware NAT Service	Network a...	Started	Automatic	LocalSystem
VNC Server		Started	Automatic	LocalSystem
Windows Installer	Installs, re...		Manual	LocalSystem
Windows Managem...	Provides s...	Started	Automatic	LocalSystem

Service mode shows up in the Service Control Panel and in the tool tray (after reboot).



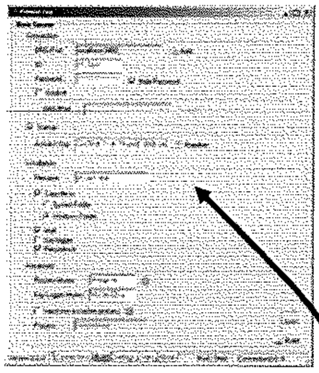
Application mode shows up in the tool tray.

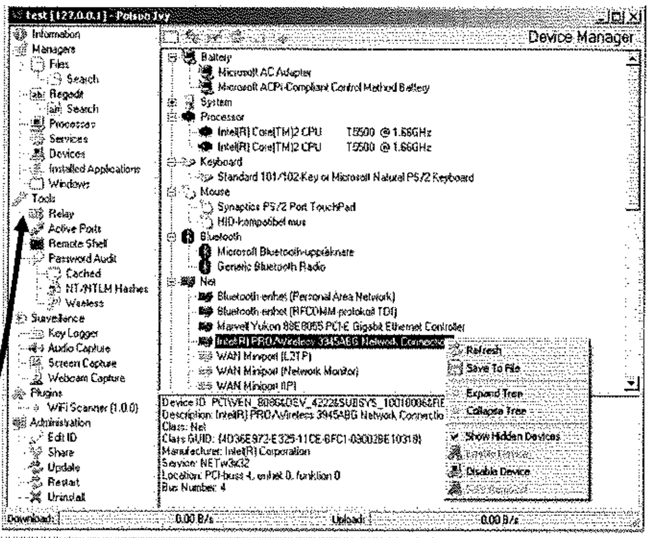
The server can be run in two modes: App mode and service mode. In app mode, a small VNC icon appears on the Windows GUI in the tool tray in the lower right-hand corner. In service mode, VNC shows up as a running service on the machine and in the tool tray after reboot.

Therefore, in the publicly released versions of WinVNC, the victim machine displays the tool tray icon, indicating to a user that the tool is installed. In recent VNC releases, there is a configuration option in the WinVNC server to omit the VNC icon from the tool tray display. That's a bit stealthier, but the VNC service and process will still be viewable on the system via the service listing and Task Manager, respectively.

Poison Ivy Configuration

App-Level Trojan Horse





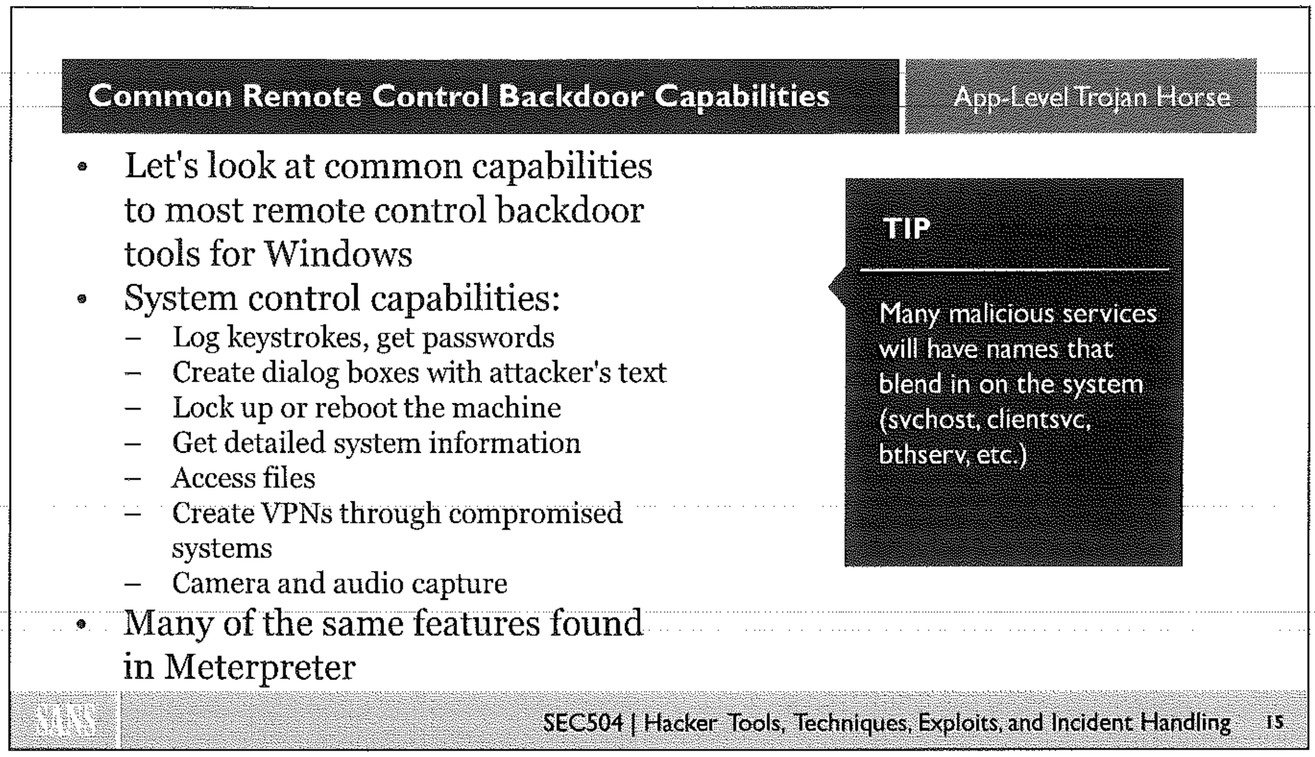
```
Battery
    Microsoft AC Adapter
    Microsoft ACPI-Compliant Control Method Battery
System
Processor
    Intel(R) Core(TM)2 CPU        T5500  @ 1.66GHz
    Intel(R) Core(TM)2 CPU        T5500  @ 1.66GHz
Keyboard
    Standard 101/102-Key or Microsoft Natural PS/2 Keyboard
Mouse
    Synaptics PS/2 Port TouchPad
    HID-kompatibel mus
Bluetooth
    Microsoft Bluetooth-oppräknare
    Generic Bluetooth Radio
Net
    Bluetooth-enhet (Personal Area Network)
    Bluetooth-enhet (RFCOMM-protokoll TDI)
    Marvell Yukon 88E8055 PCI-E Gigabit Ethernet Controller
    Intel(R) PRO/Wireless 3945ABG Network Connectio
    WAN Miniport (L2TP)
    WAN Miniport (Network Monitor)
    WAN Miniport (IP)
```

```
Device ID: PCI\VEN_8086&DEV_4222&SUBSYS_10010086&RE
Description: Intel(R) PRO/Wireless 3945ABG Network Connectio
Class: Net
Class GUID: {4D36E972-E325-11CE-BFC1-08002BE10318}
Manufacturer: Intel(R) Corporation
Service: NETw3x32
Location: PCI-buss 4, enhet 0, funktion 0
Bus Number: 4
```

1) First, config the server
2) Then, move server EXE to target
3) Then, use client GUI to control the target across the network

Now that we've seen VNC, let's look at Poison Ivy.

Like many of these remote control backdoor tools, Poison Ivy requires its user to first configure the server, setting up the communication method, filename, and various features that will be used. This configuration creates an executable suitable for installation of the server on the target. After the executable is run on the target, the attacker runs a client GUI to control the server across the network.



Common Remote Control Backdoor Capabilities

App-Level Trojan Horse

- Let's look at common capabilities to most remote control backdoor tools for Windows
- System control capabilities:
  - Log keystrokes, get passwords
  - Create dialog boxes with attacker's text
  - Lock up or reboot the machine
  - Get detailed system information
  - Access files
  - Create VPNs through compromised systems
  - Camera and audio capture
- Many of the same features found in Meterpreter

TIP

Many malicious services will have names that blend in on the system (svchost, clientsvc, bthserv, etc.)

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 15

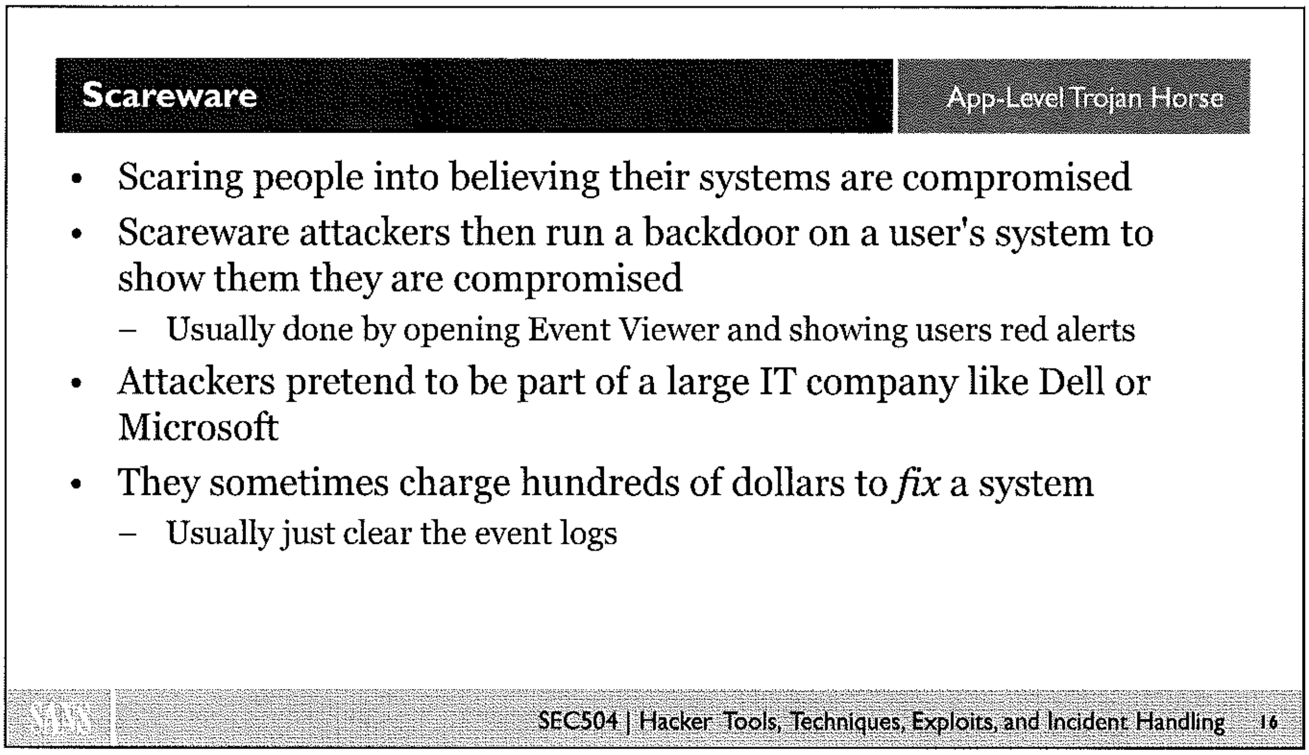
Next, let's zoom into the features that are common to most of these remote control backdoor tools that run on Windows, including Poison Ivy, Gh0st RAT, and the latest tools in this genre.

With a keystroke logger, the Trojan Horse backdoor can write all of the user's keystrokes to the filesystem so that the attacker can later look at the contents of the file. One particularly nasty use for this capability is gathering passwords of the user. Suppose the victim is utilizing a cryptographic program with a passphrase to protect a private key. The victim might utilize a very long, extremely secure passphrase, made up of a mixture of alphanumeric and special characters, which is used to protect the private key stored locally on a hard drive. If the attacker can get the victim to install a keystroke logging backdoor, the attacker can use the keystroke logger to grab the unguessable passphrase.

The system control capabilities of many remote control backdoors also include the ability to create dialog boxes with text of the attacker's choice. At the attacker's direction, a message pops up on the victim's screen saying something that the attacker wants to say. This feature can be used for social engineering, whereby the attacker uses dialog boxes to tell the user to take some specific action. For example, the attacker can tell the user to make sure that he mounts a certain network share or surf to a given site on the World Wide Web. Most users will do whatever their computer tells them to do.

In addition, the attacker can also lock up or reboot the machine and get detailed information about the victim's machine, such as the hard drive size, the processor speed, and the amount of available memory.

Many backdoors also have names that can be difficult to identify as malicious on a target system.



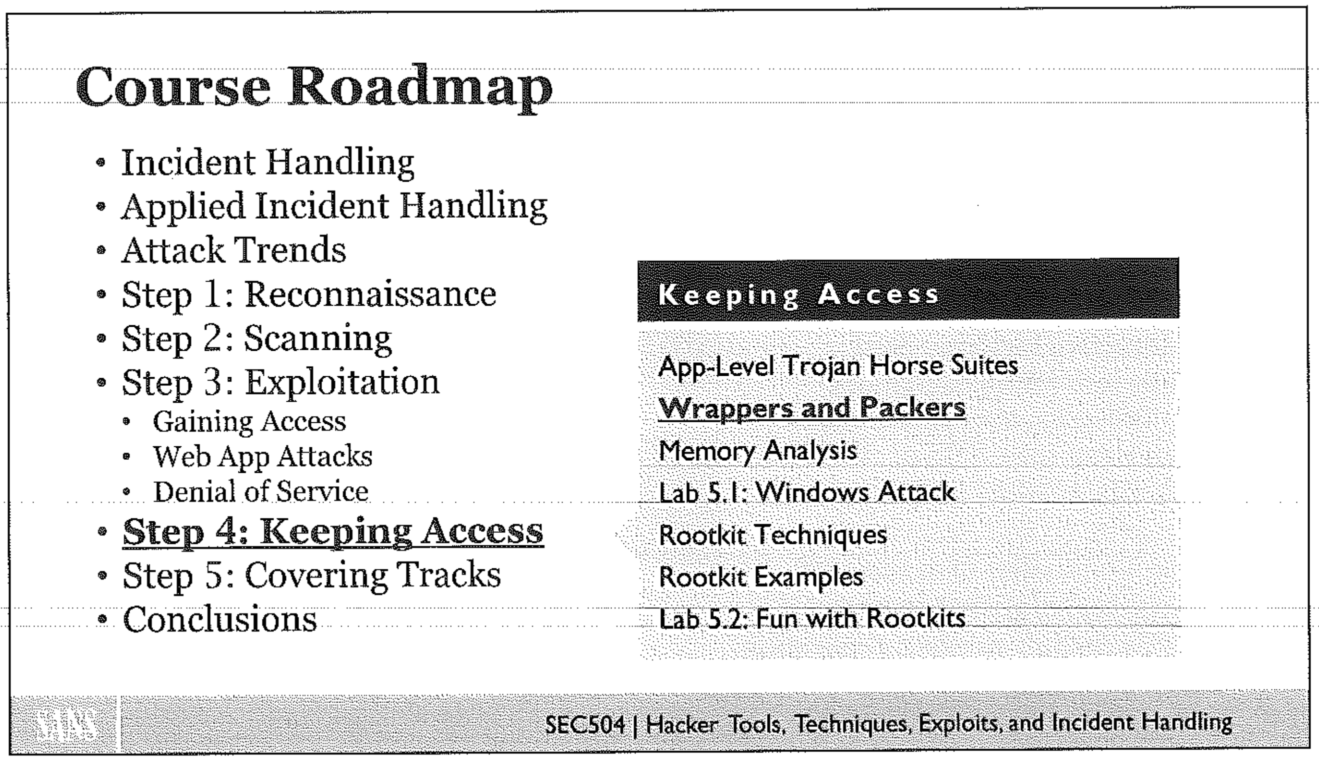
Scareware

App-Level Trojan Horse

- Scaring people into believing their systems are compromised
- Scareware attackers then run a backdoor on a user's system to show them they are compromised
  - Usually done by opening Event Viewer and showing users red alerts
- Attackers pretend to be part of a large IT company like Dell or Microsoft
- They sometimes charge hundreds of dollars to *fix* a system
  - Usually just clear the event logs

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 16

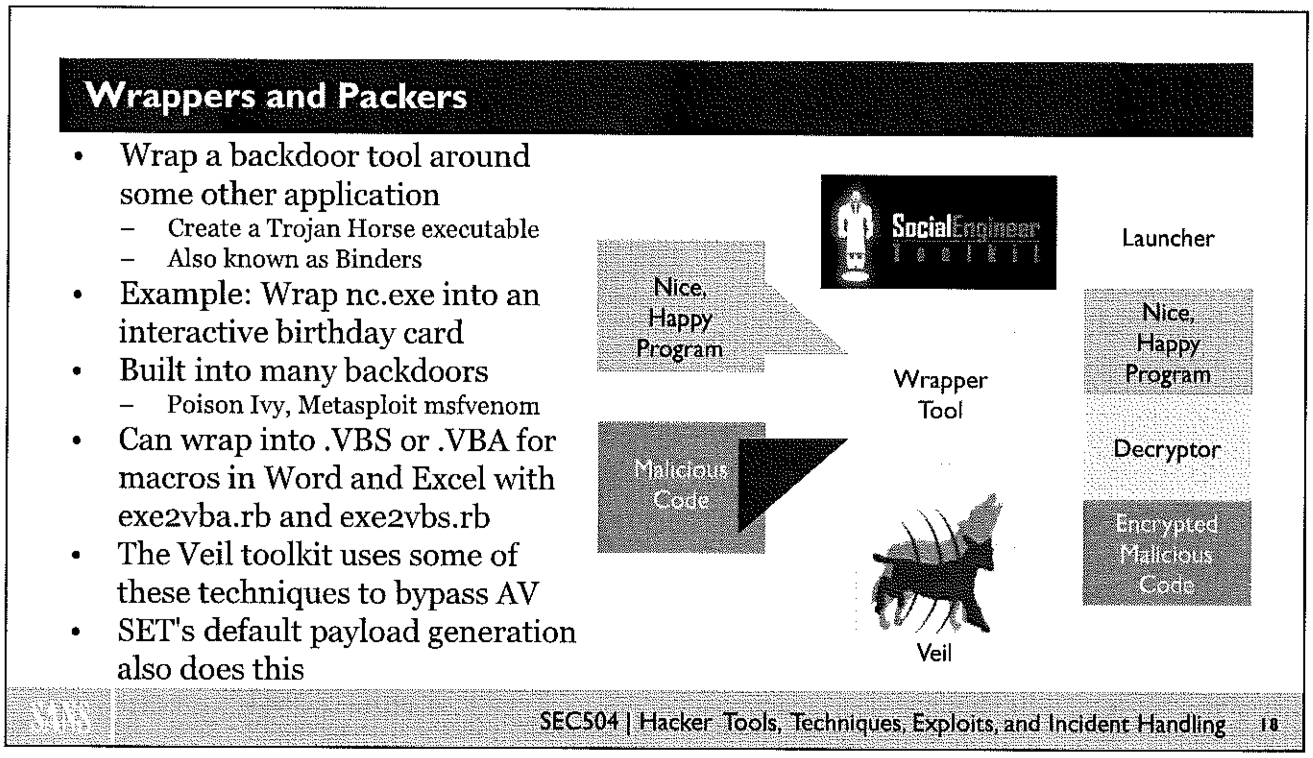
There is a new class of attacks called scareware. Unfortunately, this type of attack may be legal in some places. Still, it is far from moral. It also can be used for unauthorized people to gain access to your network. It's kind of like social engineering. How this class of attack works is by a person pretending to be an agent of a large IT company (usually Microsoft), who then actively calls a victim and tells them their computer is infected. They then show them event logs or processes, deliberately misleading the victim into believing they are compromised. Then, they run a backdoor on the victim computer so they can "fix" the issues. Many times, the actions they take are worthless to the computer. For doing this on a victim computer, they often charge hundreds of dollars.



Course Roadmap

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling

So, how do attackers attach these backdoors to normal-looking applications? They use wrappers, which is the next topic.



Wrappers and Packers

- Wrap a backdoor tool around some other application
  - Create a Trojan Horse executable
  - Also known as Binders
- Example: Wrap nc.exe into an interactive birthday card
- Built into many backdoors
  - Poison Ivy, Metasploit msfvenom
- Can wrap into .VBS or .VBA for macros in Word and Excel with exe2vba.rb and exe2vbs.rb
- The Veil toolkit uses some of these techniques to bypass AV
- SET's default payload generation also does this

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 18

Most users won't run a program called "nc.exe" if somebody sends them an email. Unfortunately, some will. For those that don't, an attacker can use a tool called a "wrapper" to integrate a backdoor program into any other program. Wrappers are essentially a way to develop Trojan Horse backdoors without any programming skills. Wrappers are also called "EXE Binders," or simply "Binders."

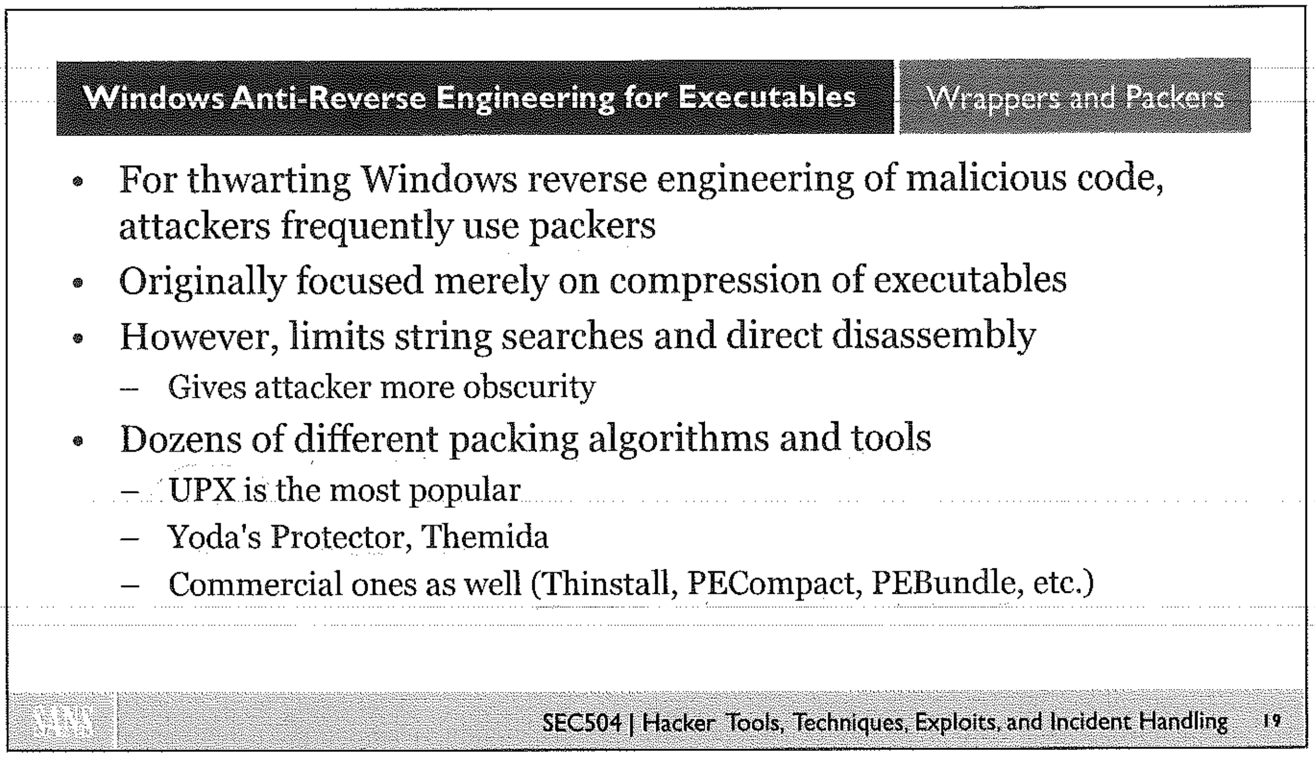
Wrappers take two inputs and have one output. The two inputs are programs that the wrapper will meld together into the single output executable. An attacker will take one executable program, such as a game or perhaps a word processing program, and wrap nc.exe into that program. The resulting output executable can be given a name like the original host program. Attackers frequently take innocuous-looking programs and wrap backdoor Trojan Horse tools into them. For example, an attacker might take a dancing birthday card executable or a pornographic video and wrap a Trojan Horse backdoor into it. The attacker will then send this resulting package to hundreds of users, hoping that one will execute it. When an unsuspecting user executes the wrapped program, the backdoor is installed first. Then the wrapped program is run. The victim sees only the latter action, not knowing that her system now has a backdoor running on it. Several wrapping programs are available, including SaranWrap and EliteWrap. An attacker can wrap any type of backdoor into another application using one of these tools. For example, Netcat, Poison Ivy, or other tools can be wrapped using any one of these wrappers.

Metasploit also has tools that allow an attacker to convert .exe files into .vba and/or .vbs. This allows him to insert malware into .doc and .xls file formats.

There is also an excellent AV bypass tool called "Veil." Veil utilizes some of these techniques to bypass AV engines. In fact, many of the Veil and wrapper techniques are also found in the Social Engineering Toolkit (SET).

Veil can be found at https://github.com/Veil-Framework/Veil-Evasion.

SET can be found at https://www.trustedsec.com/2010/09/social-engineer-toolkit-0-7-1-minor-update/.



Windows Anti-Reverse Engineering for Executables | Wrappers and Packers

- For thwarting Windows reverse engineering of malicious code, attackers frequently use packers
- Originally focused merely on compression of executables
- However, limits string searches and direct disassembly
  - Gives attacker more obscurity
- Dozens of different packing algorithms and tools
  - UPX is the most popular
  - Yoda's Protector, Themida
  - Commercial ones as well (Thinstall, PECompact, PEBundle, etc.)

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 19

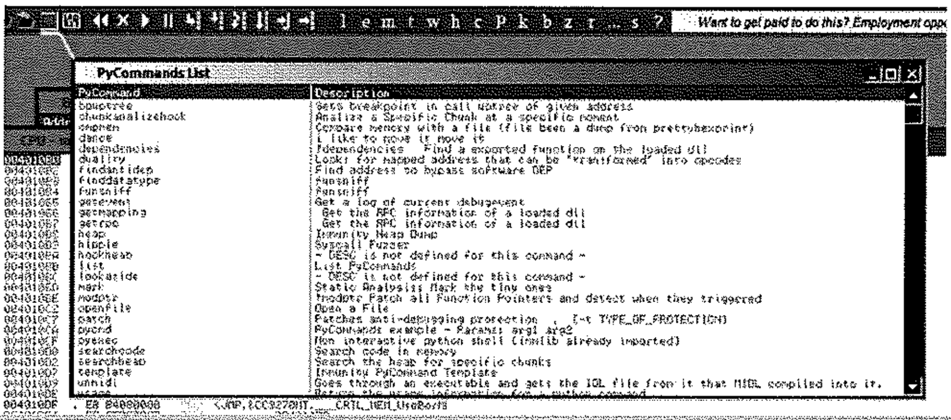
A number of ideas for thwarting reverse engineering of executables are based on a benign idea: Compressing a bloated executable to make it smaller for distribution, a process known as "packing." But attackers rapidly realized the advantages of packing executables to make them difficult to analyze. A packed executable, for example, does not reveal as many interesting strings, nor can it be directly disassembled. Until the code has been decompressed, there is just gibberish inside.

There are dozens of solid packing tools available, including the free powerful UPX packer (available at https://upx.github.io) and commercial tools like PECompact (https://bitsum.com/portfolio/pecompact/). The Yoda's Protector (https://sourceforge.net/projects/yodap/) and Themida (https://www.oreans.com/themida.php) tools are also quite popular. Many dozen different packing tools are widely used by spyware organizations and the computer underground today.

Defense: Reversing Windows Executables

Wrappers and Packers

- To thwart these, researchers use unpackers or use a plugin for a debugger
- Immunity Debugger: Very popular free Windows debugger
- NSA Ghidra reverse engineering toolkit



PyCommands List

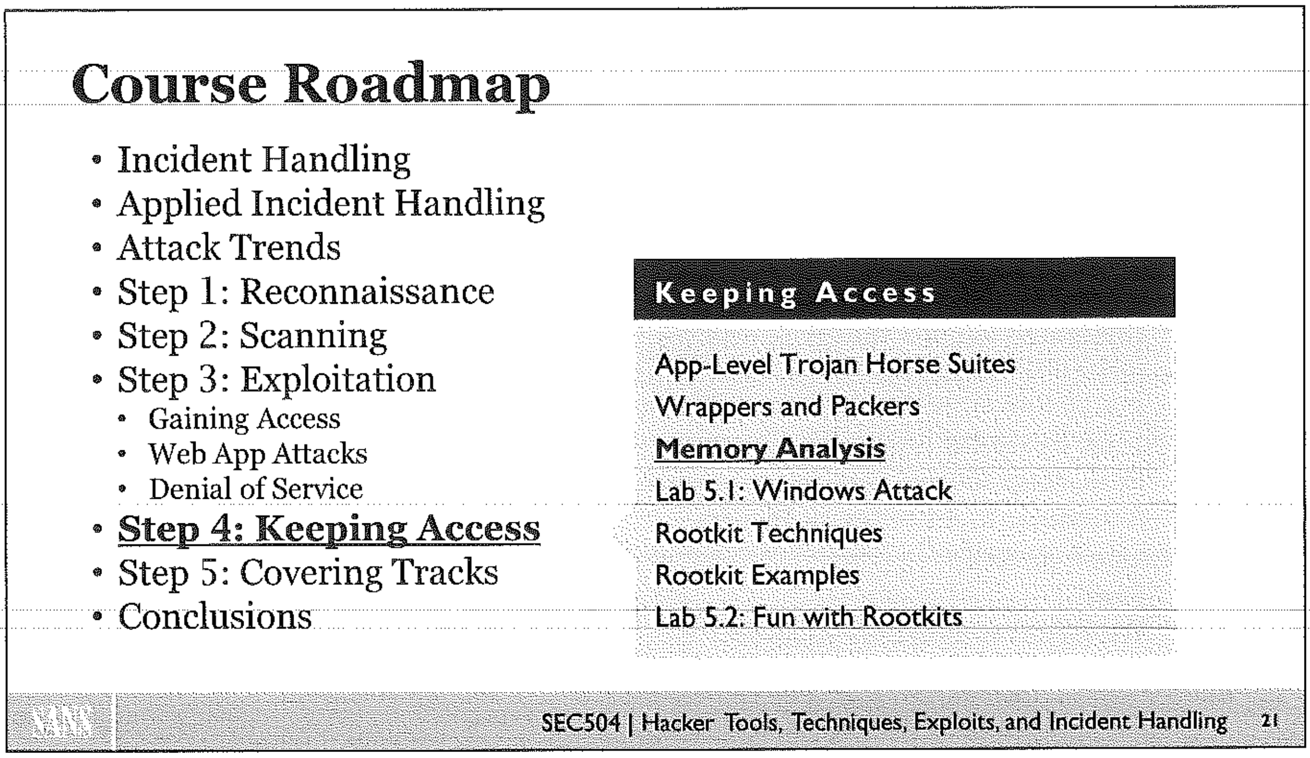
PyCommand	Description
bpupdtree	Sets breakpoint in call number of given address
chunkanalizehook	Analize a Specific Chunk at a specific moment
cmpmem	Compare memory with a file (file been a dump from prettyhexprint)
dance	i like to move it move it
dependencies	!dependencies Find a exported function on the loaded dll
duality	Looks for mapped address that can be "transformed" into opcodes
findantidep	Find address to bypass software DEP
finddatatype	funsniff
funsniff	funsniff
getevent	Get a log of current debugevent
getmapping	Get the RPC information of a loaded dll
getrpc	Get the RPC information of a loaded dll
heap	Immunity Heap Dump
hippie	Syscall Fuzzer
hookheap	~ DESC is not defined for this command ~
list	List PyCommands
lookaside	~ DESC is not defined for this command ~
mark	Static Analysis: Mark the tiny ones
modptr	!modptr Patch all Function Pointers and detect when they triggered
openfile	Open a File
patch	Patches anti-debugging protection , (-t TYPE_OF_PROTECTION)
pycmd	PyCommands example - Params: arg1 arg2
pyexec	Non interactive python shell (immlib already imported)
searchcode	Search code in memory
searchheap	Search the heap for specific chunks
template	Immunity PyCommand Template
unmidl	Goes through an executable and gets the IDL file from it that MIDL compiled into it.

```
0040100F  . E8 04000000    <JMP.&CC3270MT.___CRTL_MEM_UseBorMM
```

Immunity Debugger is an outstanding debugger for helping reverse engineering malware and exploit development. It supports Python scripts and also boasts a GUI and command-line interface. The Immunity Debugger is available at https://www.immunityinc.com/products/debugger/.

Another recent tool that is growing in popularity is the US National Security Agency (NSA) Ghidra tool. Ghidra is a debugger like the Immunity Debugger, but offers some advanced features and capabilities that parallel commercial tools while sporting a modern user interface. Ghidra is available at https://www.nsa.gov/resources/everyone/ghidra.

As an incident handler, it can become critical to have the skills to be able to step through code to see exactly what it is doing in memory, in real-time. Tools like the debugger above can greatly simplify this process. It should be noted that these tools are not tools you would want to attempt to learn in the middle of an incident. Rather, you should seek out training and online tutorials before you get to an incident so you are comfortable using these tools in a stressful situation.



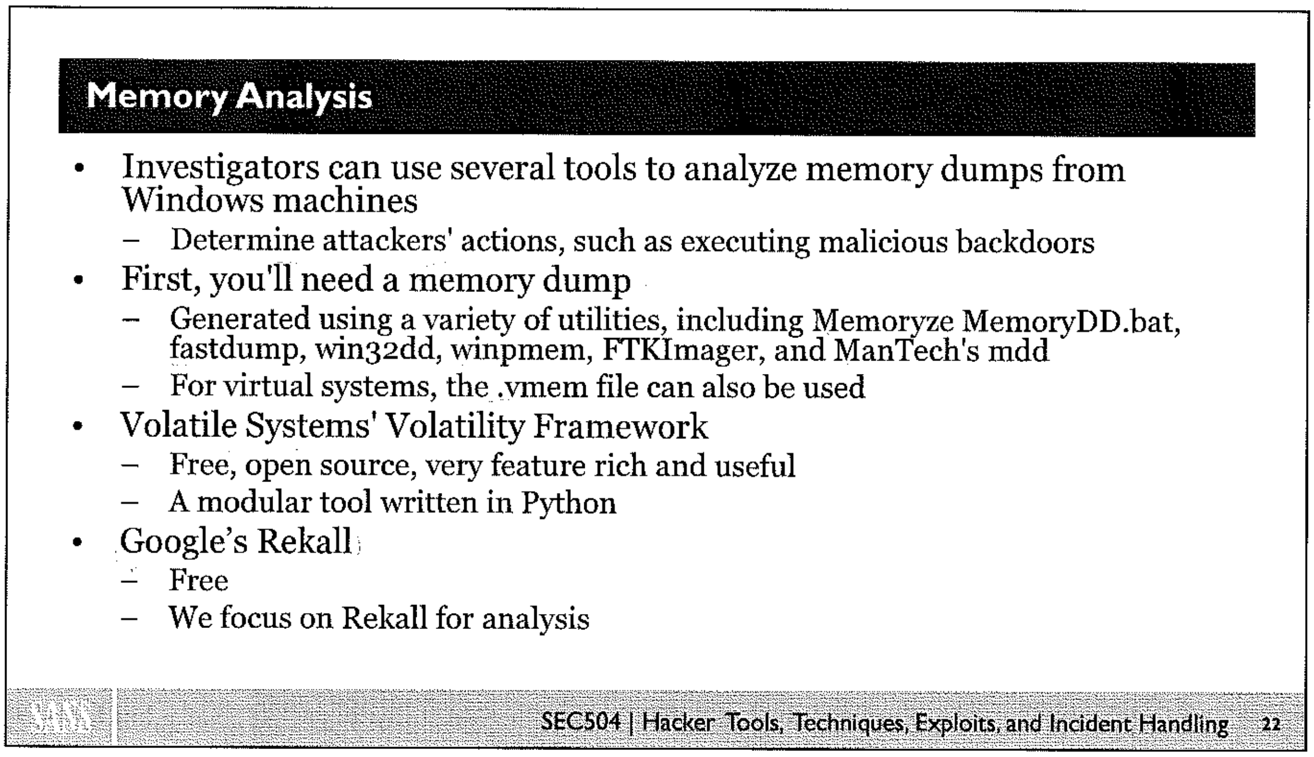
Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- **Step 4: Keeping Access**
- Step 5: Covering Tracks
- Conclusions

Keeping Access

App-Level Trojan Horse Suites
Wrappers and Packers
Memory Analysis
Lab 5.1: Windows Attack
Rootkit Techniques
Rootkit Examples
Lab 5.2: Fun with Rootkits

Now that we've seen several different application-level Trojan Horse backdoor techniques, let's look at some mechanisms that incident handlers can use for analyzing a system that has been attacked using such tools. In particular, we discuss some of the tools available for memory analysis of Windows machines, and then zoom in on Rekall, performing a hands-on lab with it.



Memory Analysis

- Investigators can use several tools to analyze memory dumps from Windows machines
  - Determine attackers' actions, such as executing malicious backdoors
- First, you'll need a memory dump
  - Generated using a variety of utilities, including Memoryze MemoryDD.bat, fastdump, win32dd, winpmem, FTKImager, and ManTech's mdd
  - For virtual systems, the .vmem file can also be used
- Volatile Systems' Volatility Framework
  - Free, open source, very feature rich and useful
  - A modular tool written in Python
- Google's Rekall
  - Free
  - We focus on Rekall for analysis

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 22

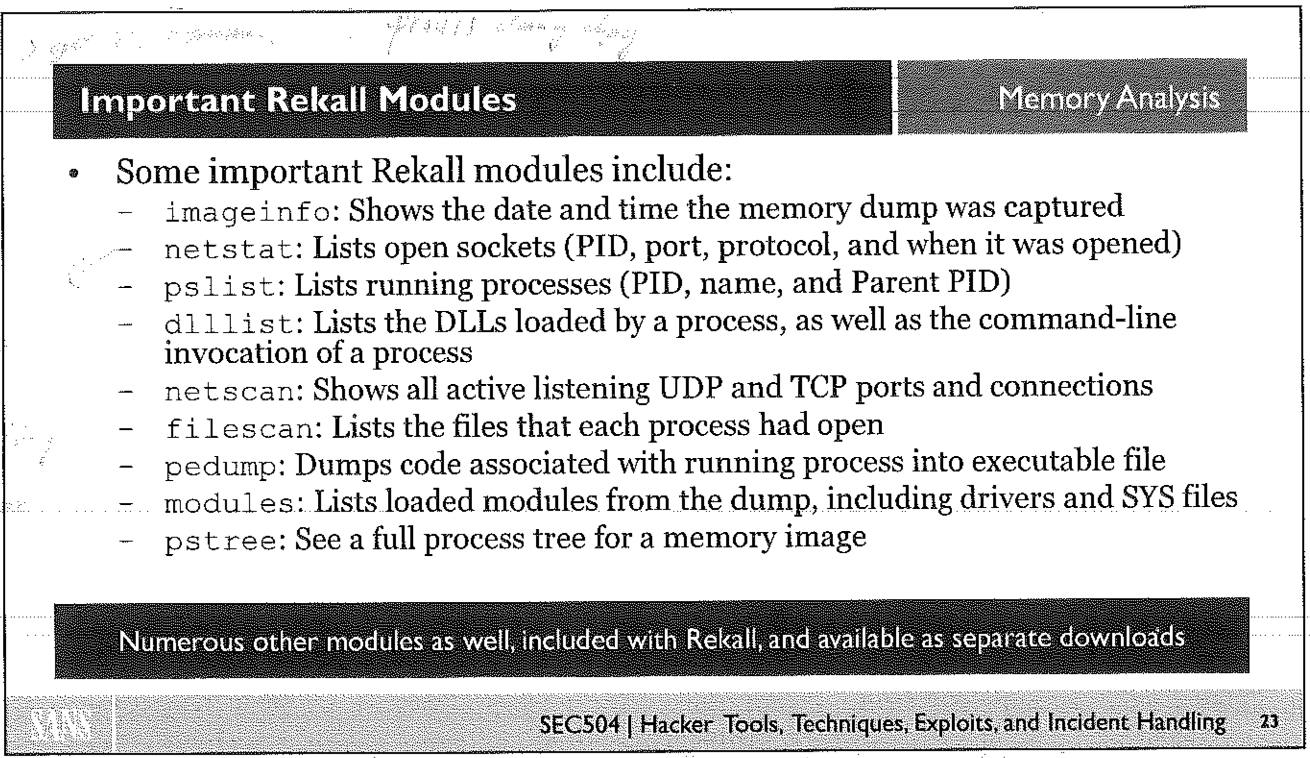
Trojan Horse backdoor tools leave an enormous amount of information in memory for an incident handler to analyze. The handler can dump memory from the machine using a large number of tools, including the MemoryDD.bat script that is part of Mandiant's free Memoryze suite. Alternatively, HBGary offers a free tool called "fastdump" for memory capture. Matthieu Suiche distributes a free program called "win32dd," and Mantech offers the free mdd tool. Some of these tools are older, but it is a good idea to keep them around just in case you need them.

If you are working with virtual systems, the VMware .vmem file for a suspended guest (or comparable for non-VMware platforms) can also be used for analysis. The primary advantage of using a .vmem file is that it doesn't have to be extracted from the physical memory of a target system, it already exists on the filesystem!

After a memory dump is created, you can move it off a machine using Netcat file transfer or copying it to an SMB file share. Then the memory capture can be analyzed using various tools.

In the past few years, numerous high-quality tools have been released for analyzing memory dumps from Windows systems. Volatile Systems offers the free, open-source Volatility Framework, an excellent tool that can pull an enormous amount of information from Windows dumps, including network connections, running processes, loaded drivers, etc. Volatility, written in Python, is a modular and extendable framework, for which other developers have written modules to pull out different kinds of information from memory dumps. The Volatility Framework is available at https://www.volatilityfoundation.org.

Google's Rekall framework is also outstanding. It has many of the same features as the Volatility Framework, available at http://www.rekall-forensic.com. For this class, we focus on using Rekall.



Important Rekall Modules

Memory Analysis

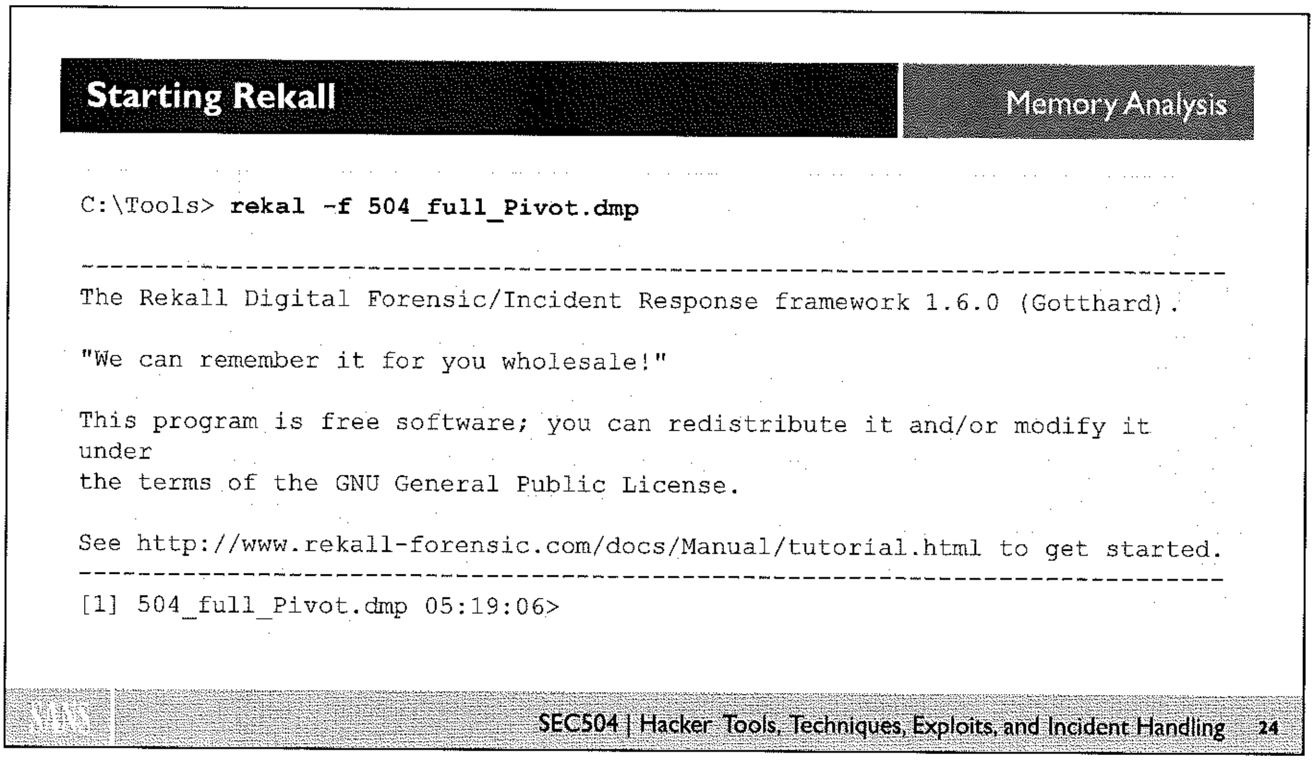
- Some important Rekall modules include:
  - `imageinfo`: Shows the date and time the memory dump was captured
  - `netstat`: Lists open sockets (PID, port, protocol, and when it was opened)
  - `pslist`: Lists running processes (PID, name, and Parent PID)
  - `dlllist`: Lists the DLLs loaded by a process, as well as the command-line invocation of a process
  - `netscan`: Shows all active listening UDP and TCP ports and connections
  - `filescan`: Lists the files that each process had open
  - `pedump`: Dumps code associated with running process into executable file
  - `modules`: Lists loaded modules from the dump, including drivers and SYS files
  - `pstree`: See a full process tree for a memory image

Numerous other modules as well, included with Rekall, and available as separate downloads

Rekall consists of numerous modules, each of which scours a Windows memory dump looking for specific artifacts. Some of the most useful ones include:

- **imageinfo**: This module shows the date and time that the memory dump was captured.
- **netstat**: This module lists open network sockets, showing the processID using the socket, the port it uses, the protocol associated with the communication, and the date and time when the socket was opened.
- **pslist**: This module shows a list of processes, including the PID, name, and ParentProcessID.
- **dlllist**: This module shows a list of DLLs loaded by a given process. It also shows the command-line invocation of each process, a highly useful capability when the attacker is using in-depth Windows command-line skills.
- **netscan**: This module shows all active listening UDP and TCP ports and connections.
- **filescan**: This module shows a list of all files that each process has open.
- **pedump**: This module dumps the executable code associated with a given process into an EXE file so that it can be analyzed separately. This feature is very helpful if the analyst doesn't have a copy of the filesystem from which the memory dump was taken.
- **modules**: This Rekall module shows the device drivers loaded by the Windows machine from which the dump was created. It also reveals related SYS files.
- **pstree:** This module gives a full breakdown of process parentage.

Numerous other modules are available for download at http://web.rekall-innovations.com/docs/Manual/Plugins/Windows/Modules.html.



Starting Rekall

Memory Analysis

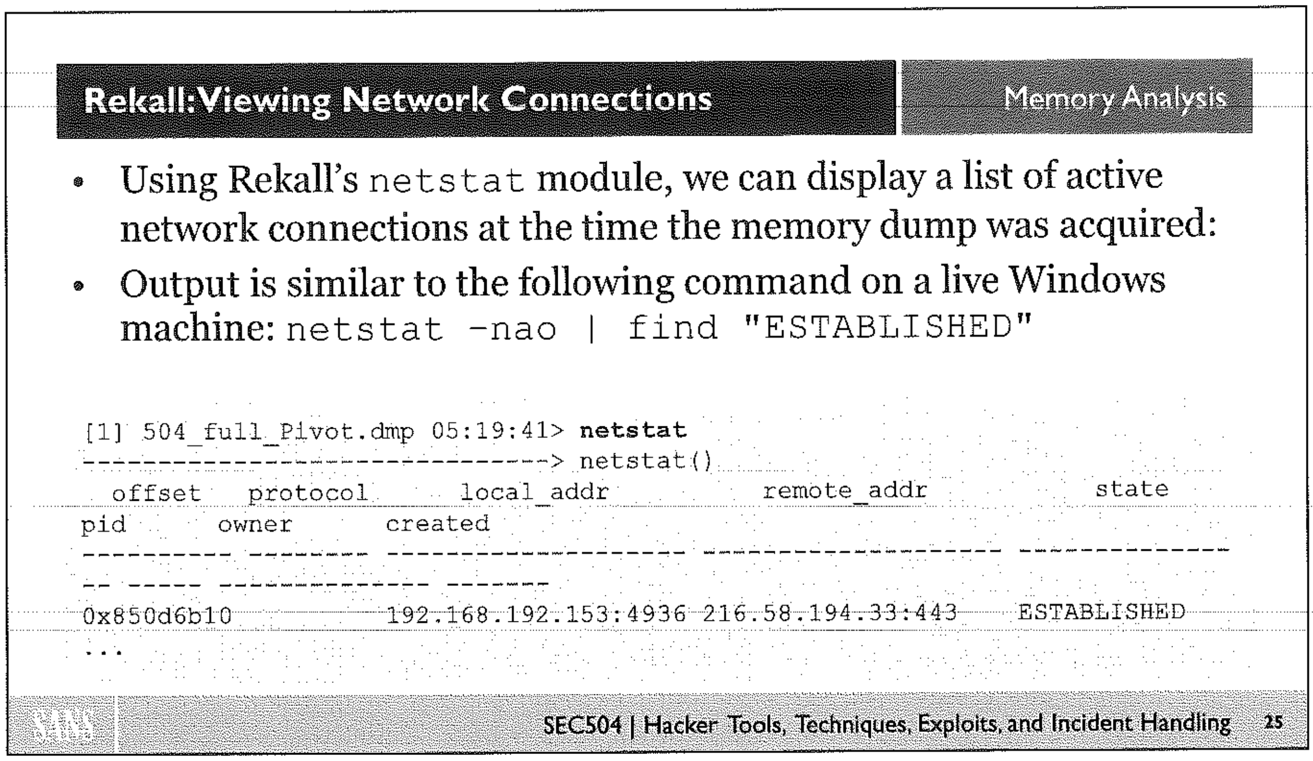
```
C:\Tools> rekal -f 504_full_Pivot.dmp

----------------------------------------------------------------------------
The Rekall Digital Forensic/Incident Response framework 1.6.0 (Gotthard).

"We can remember it for you wholesale!"

This program is free software; you can redistribute it and/or modify it
under
the terms of the GNU General Public License.

See http://www.rekall-forensic.com/docs/Manual/tutorial.html to get started.
----------------------------------------------------------------------------
[1] 504_full_Pivot.dmp 05:19:06>
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 24

For this lab, we will be using the Windows VM.

To start Rekall, simply run the following commands:

```
C:\> cd \Tools

C:\> rekal -f 504_full_Pivot.dmp
```



Rekall: Viewing Network Connections

Memory Analysis

- Using Rekall's `netstat` module, we can display a list of active network connections at the time the memory dump was acquired:
- Output is similar to the following command on a live Windows machine: `netstat -nao | find "ESTABLISHED"`

```
[1] 504_full_Pivot.dmp 05:19:41> netstat
-------------------------------> netstat()
  offset   protocol     local_addr           remote_addr          state
pid      owner      created
---------- -------- -------------------- -------------------- --------------
-- ----- -------------- -------
0x850d6b10          192.168.192.153:4936 216.58.194.33:443    ESTABLISHED
...
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 25

Let's explore some of Rekall's modules in more detail to help you prepare to analyze a Windows memory dump in the next lab.

Rekall uses many of Microsoft's native commands for pulling data from a memory dump, and these same commands can be executed on a live Windows machine. Incident handlers should master the associated Windows commands for live systems so that they can analyze ephemeral artifacts in the field. We should also have knowledge of the roughly equivalent command in Rekall, so we can recover similar evidence back in the lab from captured memory dump files.

For example, to pull a list of network connections from a memory image file, we can run:

```
[1] memimage.dump hh:mm:ss>  netstat
```

*****PLEASE NOTE, THE FIRST TIME YOU RUN THIS THERE WILL BE ERRORS; IT WORKS FINE. IT IS JUST COMPLAINING ABOUT THE LACK OF INTERNET CONNECTIVITY******

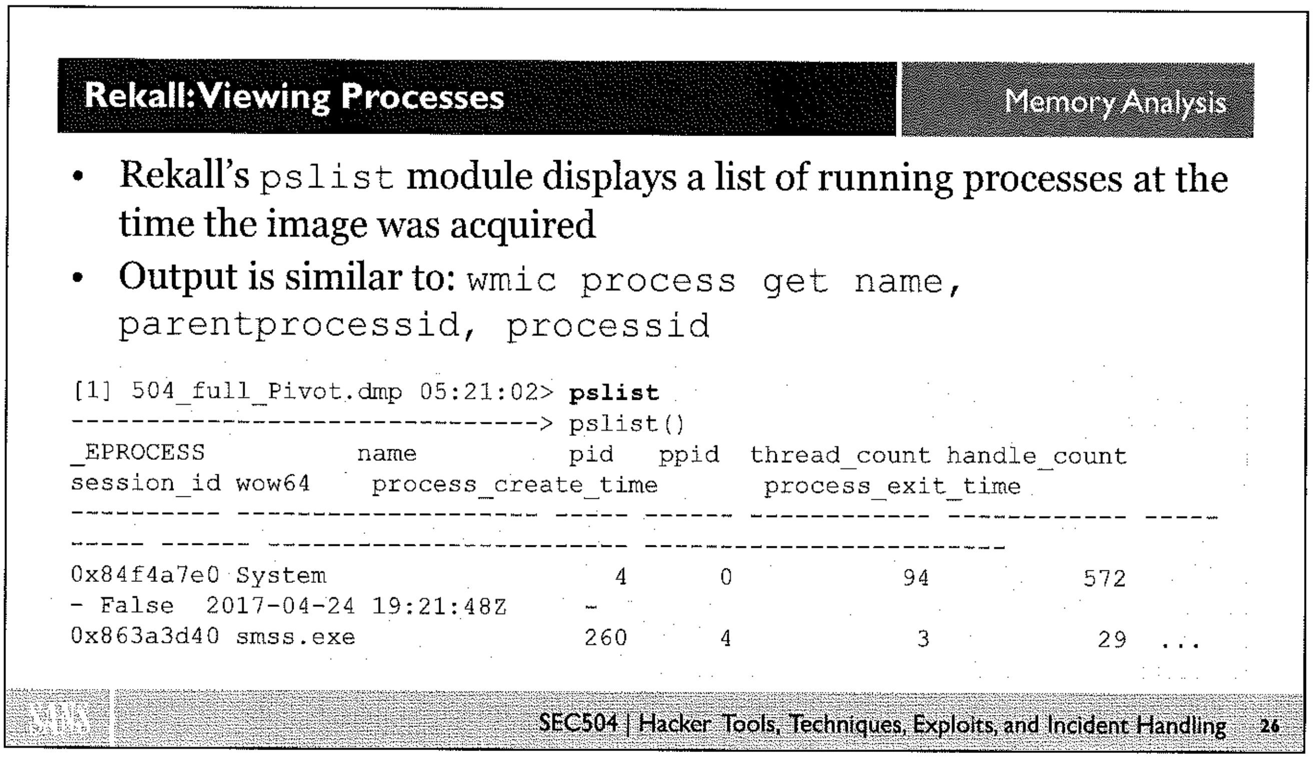
We can see in the output the local IP address and port number, as well as the remote address and port number. Note that this output includes the PID of the process responsible for each connection, just like netstat's "-o" option.

After you have a good list of the connections, you can then get the PID from netscan

```
[1] memimage.dump hh:mm:ss>  netscan
```

In fact, the command that provides similar output on a live Windows machine is the following:

```
C:\> netstat -nao | find "ESTABLISHED"
```



Rekall: Viewing Processes

Memory Analysis

- Rekall's `pslist` module displays a list of running processes at the time the image was acquired
- Output is similar to: `wmic process get name, parentprocessid, processid`

```
[1] 504_full_Pivot.dmp 05:21:02> pslist
-------------------------------> pslist()
_EPROCESS            name           pid   ppid  thread_count handle_count
session_id wow64      process_create_time        process_exit_time
---------- -------------------- ----- ------ ------------ ------------ -----
----- ------ ------------------------ ------------------------
0x84f4a7e0 System                   4      0           94          572
- False   2017-04-24 19:21:48Z     -
0x863a3d40 smss.exe               260      4            3           29  ...
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 26

Next, to display a list of processes that were running at the time the memory dump was created, we can run the following:

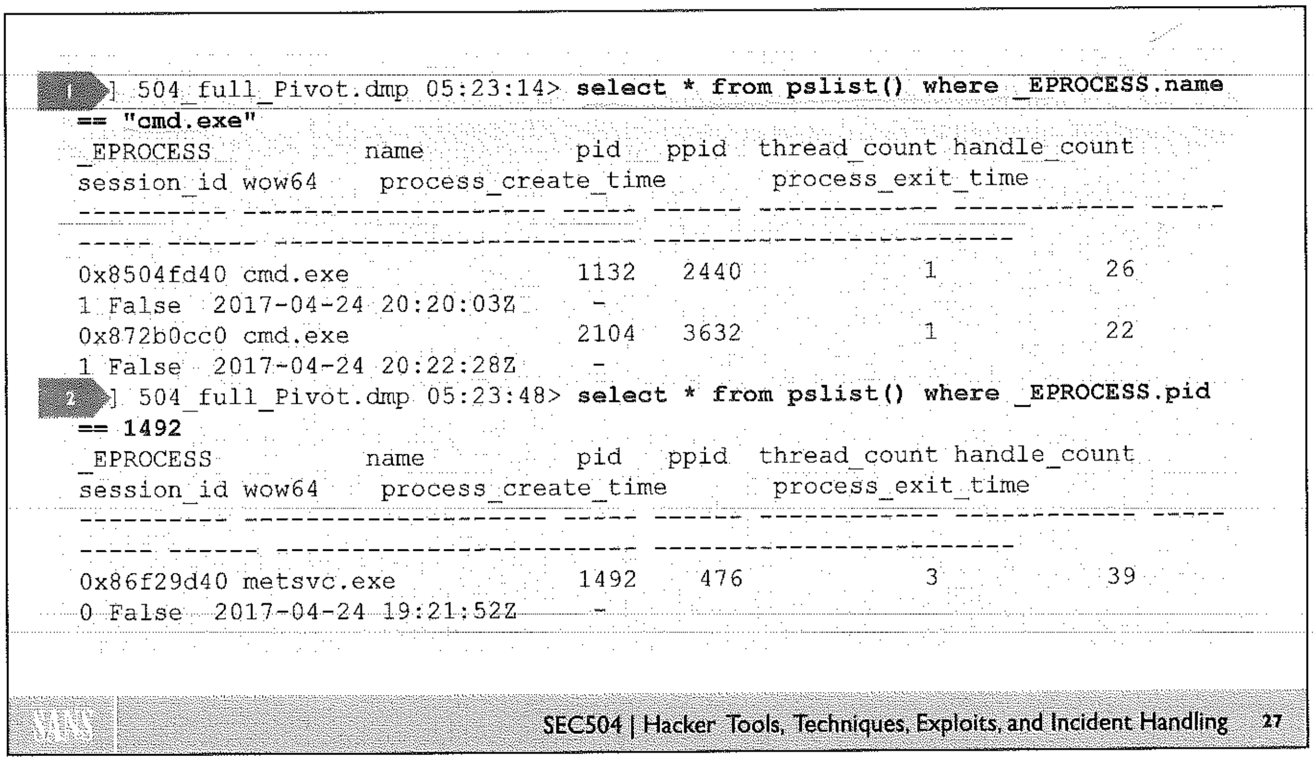
```
[1] memimage.dd hh:mm:ss>  pslist
```

This output includes the PID, as well as the Parent PID (PPID) and the time each process was started.

Showing both the PID and PPID for each running process helps an analyst determine the process tree, seeing which processes started other processes to help target an investigation on an initial point of attack or infection.

To get similar information from a live Windows machine, you can run the following:

```
C:\> wmic process get name, parentprocessid, processid
```


```
1 ] 504_full_Pivot.dmp 05:23:14> select * from pslist() where _EPROCESS.name
== "cmd.exe"
_EPROCESS          name                pid   ppid  thread_count handle_count
session_id wow64    process_create_time       process_exit_time
---------- -------------------- ----- ------ ------------ ------------ -----
----- ------ ------------------------ ------------------------
0x8504fd40 cmd.exe              1132   2440             1           26
1 False  2017-04-24 20:20:03Z     -
0x872b0cc0 cmd.exe              2104   3632             1           22
1 False  2017-04-24 20:22:28Z     -
2 ] 504_full_Pivot.dmp 05:23:48> select * from pslist() where _EPROCESS.pid
== 1492
_EPROCESS          name                pid   ppid  thread_count handle_count
session_id wow64    process_create_time       process_exit_time
---------- -------------------- ----- ------ ------------ ------------ -----
----- ------ ------------------------ ------------------------
0x86f29d40 metsvc.exe           1492    476             3           39
0 False  2017-04-24 19:21:52Z     -
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 27

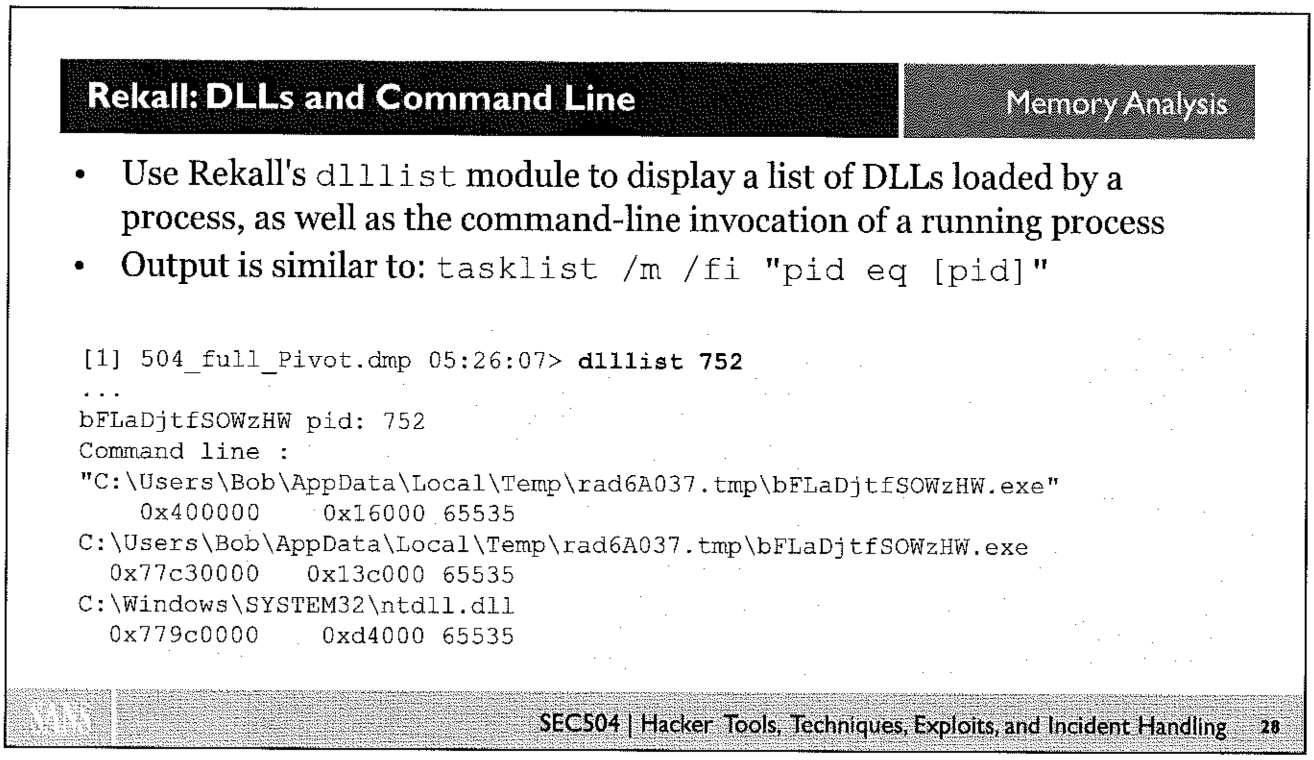
You can filter your results using SQL-like syntax:

The below command works well for filtering based on a string like cmd.exe, Please note the "" around cmd.exe

```
1.    [1] 504_full_Pivot.dmp 12:14:28> select * from pslist() where
_EPROCESS.name == "cmd.exe"
```

The below command works for filtering on numbers. Please note the lack of "" around the process ID

```
2.    [1] 504_full_Pivot.dmp 12:15:05> select * from pslist() where
_EPROCESS.pid == 1492
```



Rekall: DLLs and Command Line

Memory Analysis

- Use Rekall's `dlllist` module to display a list of DLLs loaded by a process, as well as the command-line invocation of a running process
- Output is similar to: `tasklist /m /fi "pid eq [pid]"`

```
[1] 504_full_Pivot.dmp 05:26:07> dlllist 752
...
bFLaDjtfSOWzHW pid: 752
Command line :
"C:\Users\Bob\AppData\Local\Temp\rad6A037.tmp\bFLaDjtfSOWzHW.exe"
    0x400000     0x16000 65535
C:\Users\Bob\AppData\Local\Temp\rad6A037.tmp\bFLaDjtfSOWzHW.exe
  0x77c30000    0x13c000 65535
C:\Windows\SYSTEM32\ntdll.dll
  0x779c0000     0xd4000 65535
```

Other helpful features of Rekall are included in its dlllist module, invoked with the following command:

```
[1] memimage.dd hh:mm:ss>  dlllist [pid_num]
```

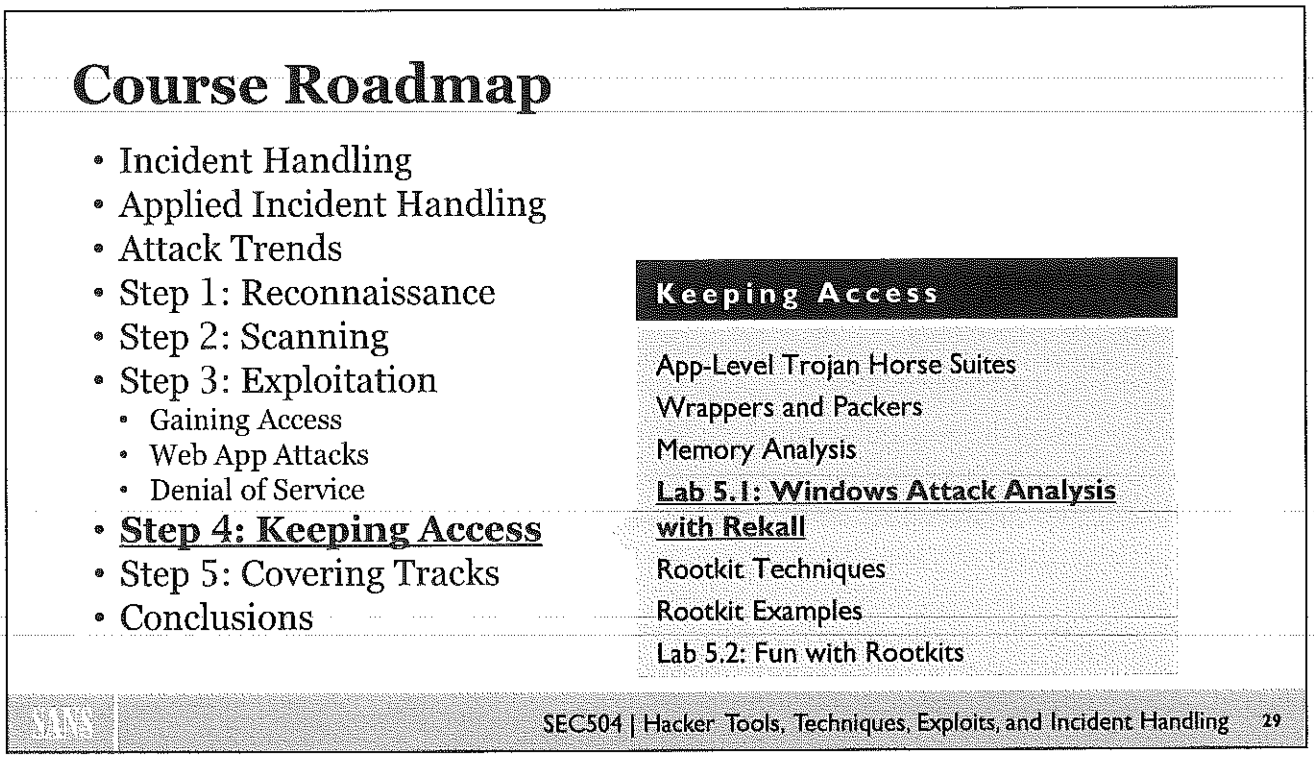
Here, we have provided the dlllist module a specific option (-p [pid]) to indicate which process we want to analyze. This invocation tells Rekall to determine the command-line invocation of the given process, plus a list of all DLLs that it has loaded.

To get similar information from a live Windows machine, we can run the WMIC and tasklist commands as follows:

For the command-line invocation of a process, run the following:
```
C:\> wmic process where processid=[pid] get commandline
```

For the list of DLLs loaded by every running process, we can use the following:
```
C:\> tasklist /m
```

For the list of DLLs loaded by a specific process, we can execute the following:
```
C:\> tasklist /m /fi "pid eq [pid]"
```



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- **Step 4: Keeping Access**
- Step 5: Covering Tracks
- Conclusions

Keeping Access

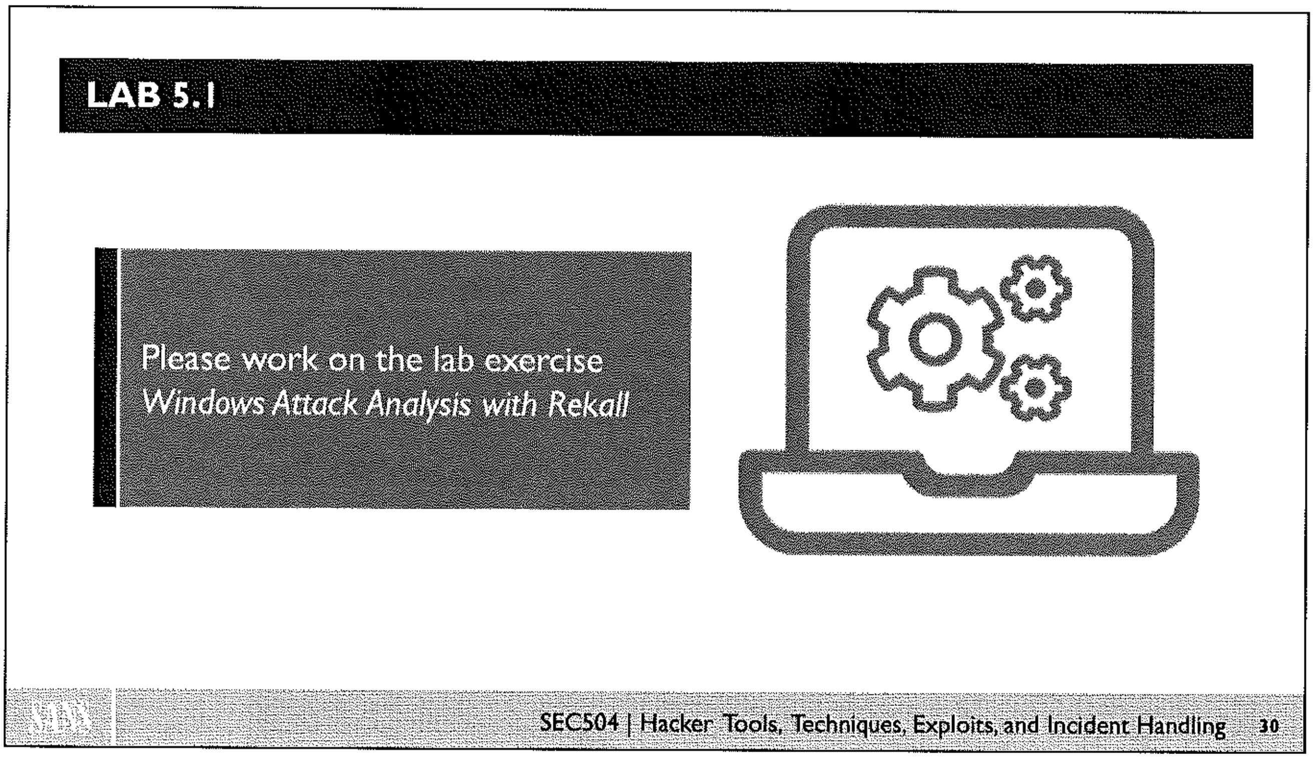
App-Level Trojan Horse Suites
Wrappers and Packers
Memory Analysis
Lab 5.1: Windows Attack Analysis with Rekall
Rootkit Techniques
Rootkit Examples
Lab 5.2: Fun with Rootkits

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 29

Next, let's perform a lab in which we analyze a memory dump from a victim Windows machine that suffered an attack via application-level malware.

We use the Rekall tool to look at the memory dump from a Windows machine.

As we perform this analysis from a captured memory image, please try to keep in mind the equivalent commands and analogous analytic steps you'd use on a live Windows machine via the Windows command line. Although the lab is focused on using Rekall for analysis of memory dumps, it is also designed to remind you on a regular basis of the commands you can run to perform similar steps on a live Windows machine.

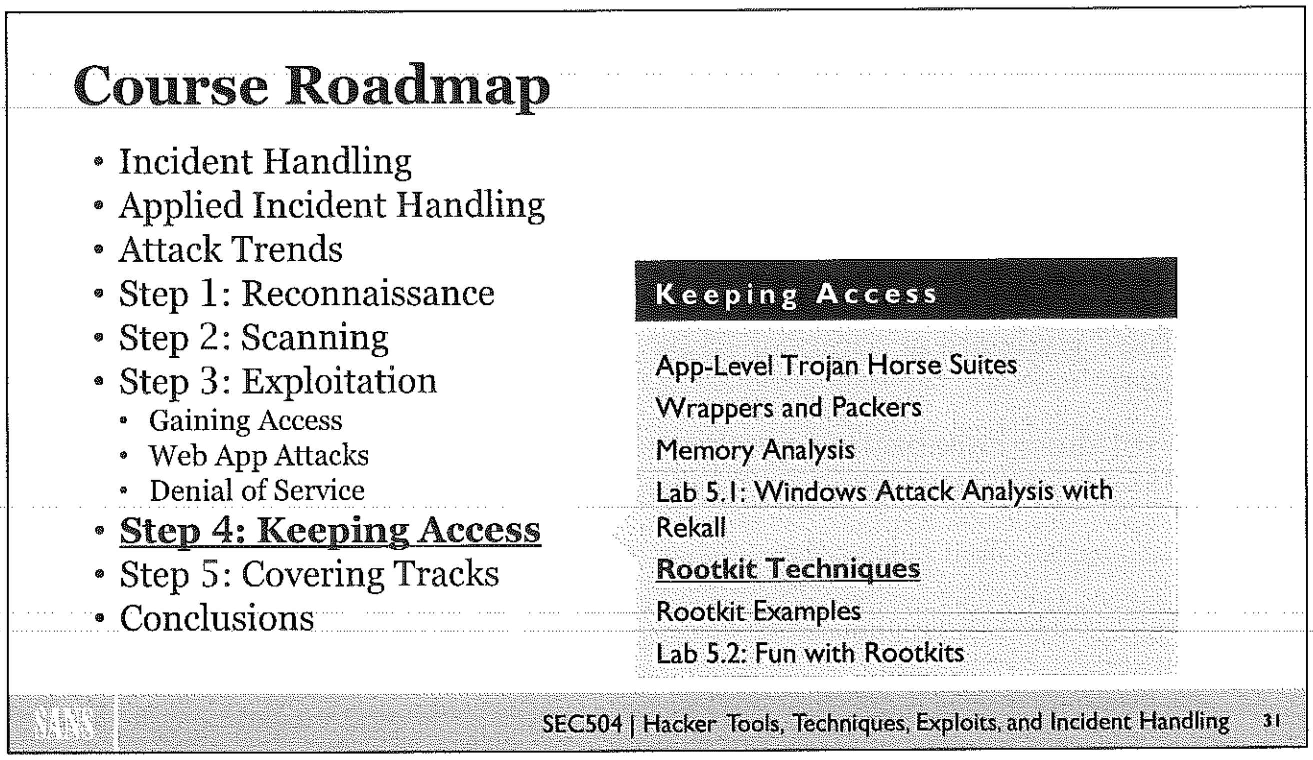


LAB 5.1

Please work on the lab exercise *Windows Attack Analysis with Rekall*

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 30

This page intentionally left blank.



Course Roadmap

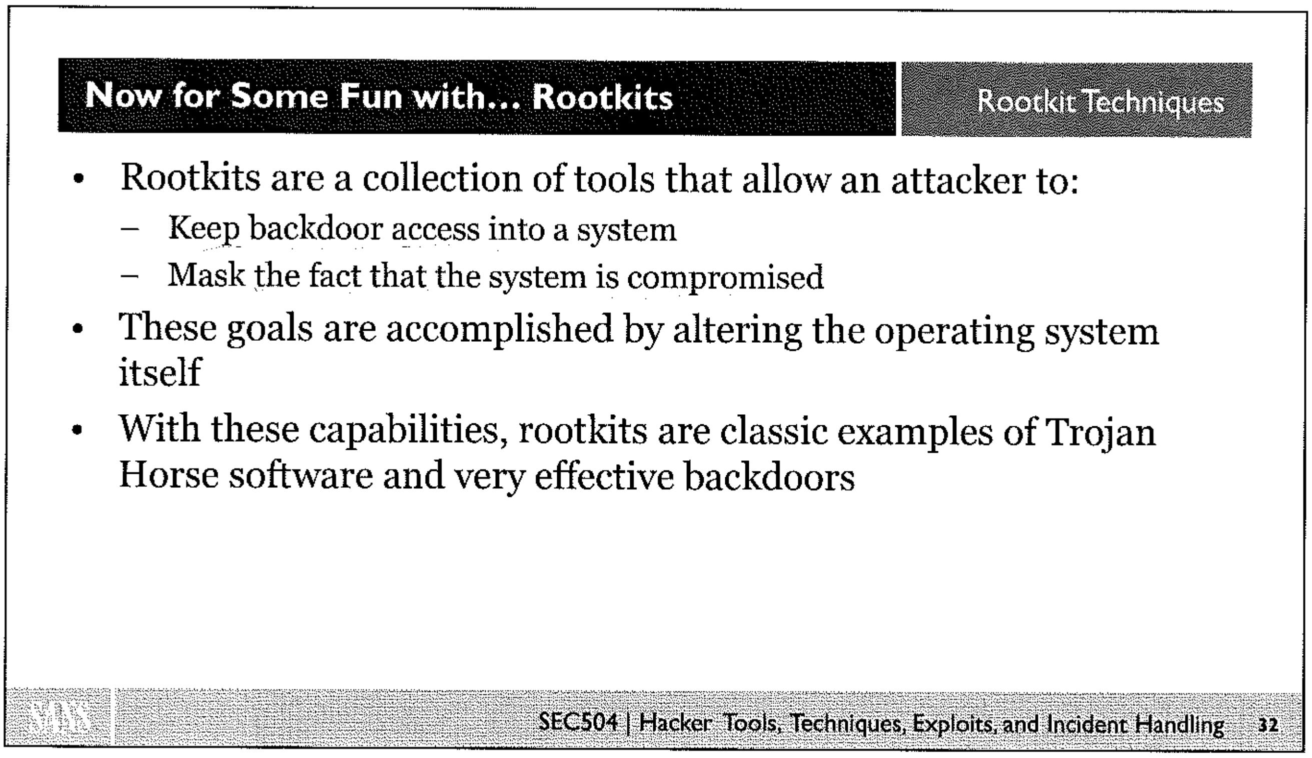
- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- **Step 4: Keeping Access**
- Step 5: Covering Tracks
- Conclusions

Keeping Access

App-Level Trojan Horse Suites
Wrappers and Packers
Memory Analysis
Lab 5.1: Windows Attack Analysis with Rekall
Rootkit Techniques
Rootkit Examples
Lab 5.2: Fun with Rootkits

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 31

Now we explore a more insidious type of tool, a rootkit. A rootkit modifies the existing programs on a system to create Trojan Horse backdoors.



Now for Some Fun with... Rootkits

Rootkit Techniques

- Rootkits are a collection of tools that allow an attacker to:
  - Keep backdoor access into a system
  - Mask the fact that the system is compromised
- These goals are accomplished by altering the operating system itself
- With these capabilities, rootkits are classic examples of Trojan Horse software and very effective backdoors

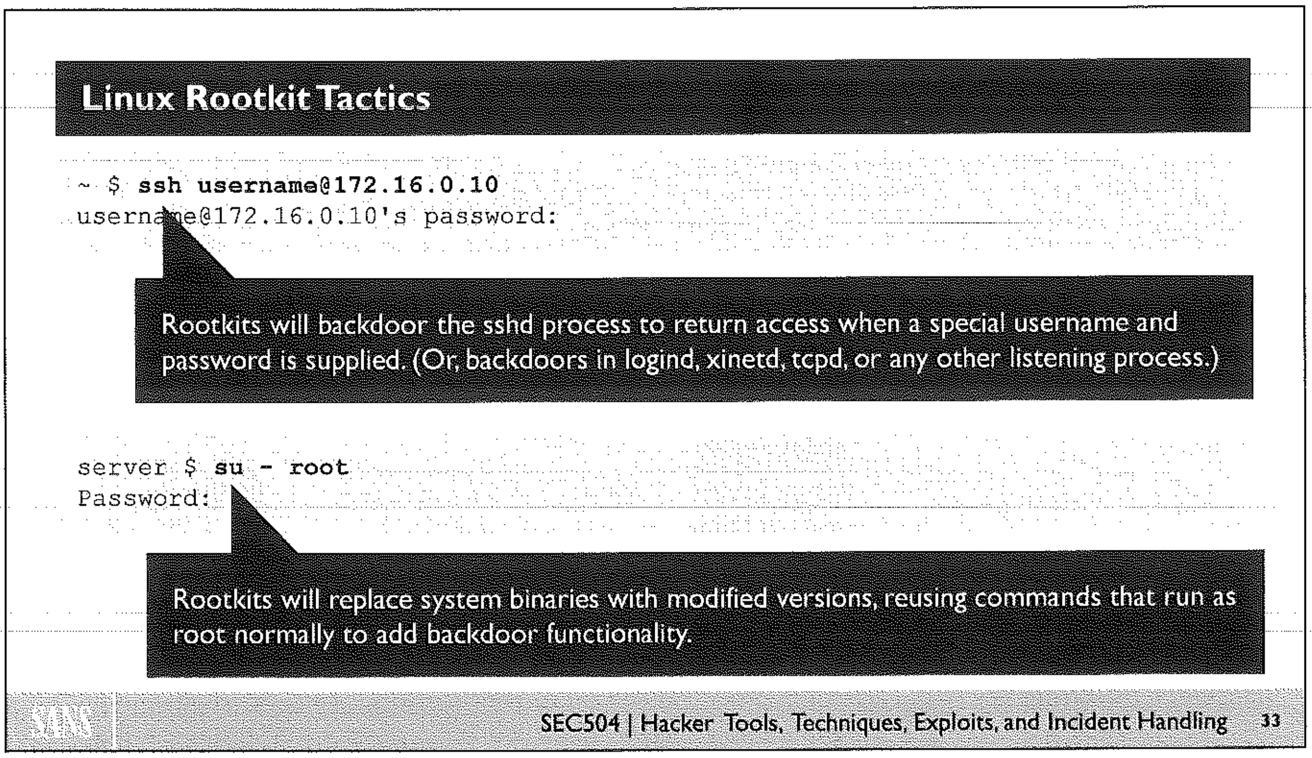
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 32

Contrary to what their name implies, rootkits do not allow an attacker to gain root access. Rootkits depend on the attackers already having root access, which was likely gotten with a root exploit (such as a buffer overflow or other type of attack).

Although rootkits do not let an attacker gain root access, they do allow attackers to maintain root access once they've gotten it. Rootkits let an attacker place a backdoor onto the system to maintain control of the machine. Some rootkits also include capabilities for gathering information from the local network through sniffing. One of the most significant areas in rootkit tools involves masking the attacker's presence on the system. Rootkits hide logins, programs, files, and processes from a system administrator.

To accomplish these goals, rootkits alter the existing operating system on the victim machine. Rather than adding a new application to the system like we saw with application-level Trojan Horse backdoors, rootkits alter the existing programs on the machine. Because they modify existing programs, rootkits are classic examples of Trojan Horse backdoors.

Although their name carries the UNIX word "root," rootkits have been released for not only UNIX and Linux operating systems, but also Windows machines.



Linux Rootkit Tactics

```
~ $ ssh username@172.16.0.10
username@172.16.0.10's password:
```

Rootkits will backdoor the sshd process to return access when a special username and password is supplied. (Or, backdoors in logind, xinetd, tcpd, or any other listening process.)

```
server $ su - root
Password:
```

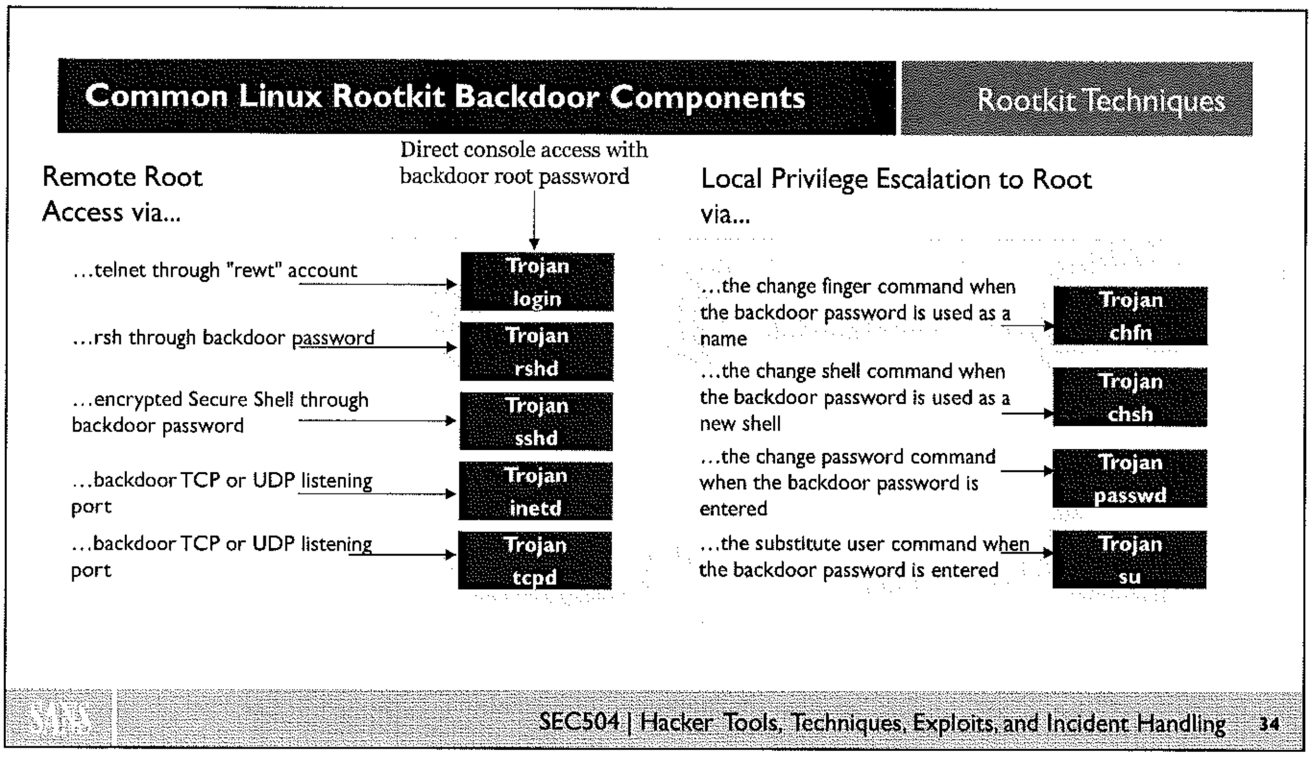
Rootkits will replace system binaries with modified versions, reusing commands that run as root normally to add backdoor functionality.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 33

After compromising a Linux system, attackers will apply several tactics to preserve their access to the system through the deployment of rootkits.

For example, the attacker may replace the login, rshd, sshd, inetd, and tcpd services to include a backdoor password. If the attacker connects to any one of these processes from across the network and provides the password, the attacker is instantly given remote root access, despite any local account controls (e.g. defying any password or authentication restrictions on the system).

In addition, various local commands are modified to include local root-level backdoors. The chfn, chsh, passwd, and su commands are overwritten with new versions. If a non-root user runs any of these programs with a command argument that is the backdoor password, that user is instantly elevated to root access. Think of these programs as little teleporters to instantly get root.



Common Linux Rootkit Backdoor Components

Rootkit Techniques

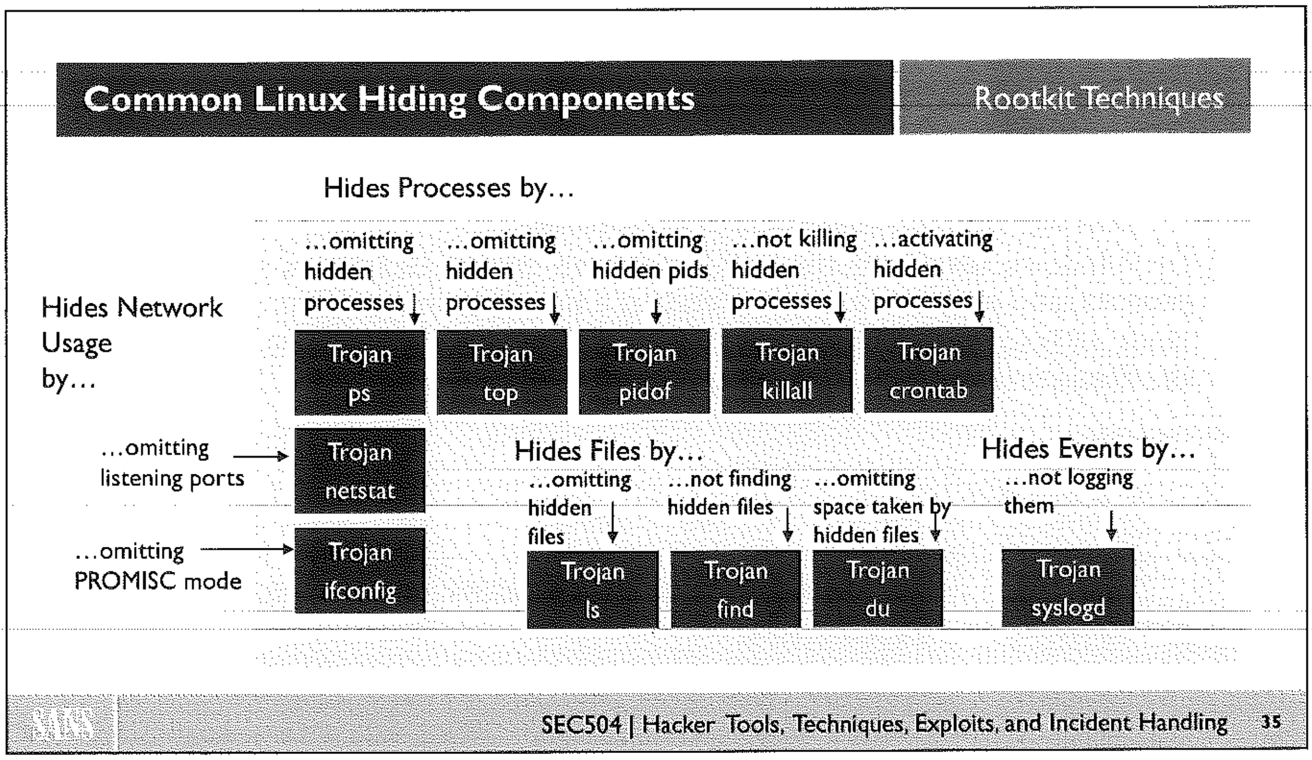
Direct console access with backdoor root password
Remote Root Access via...
...telnet through "rewt" account
Trojan login
...rsh through backdoor password
Trojan rshd
...encrypted Secure Shell through backdoor password
Trojan sshd
...backdoor TCP or UDP listening port
Trojan inetd
...backdoor TCP or UDP listening port
Trojan tcpd
Local Privilege Escalation to Root via...
...the change finger command when the backdoor password is used as a name
Trojan chfn
...the change shell command when the backdoor password is used as a new shell
Trojan chsh
...the change password command when the backdoor password is entered
Trojan passwd
...the substitute user command when the backdoor password is entered
Trojan su

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 34

In addition to modifying the login program and ifconfig, rootkits will replace several other critical files on the system. Each of these modifications is designed to help the attacker hide on the machine.

This slide shows the various backdoor components included in common Linux rootkits. The login, rshd, sshd, inetd, and tcpd services are all modified to include a backdoor password. If the attacker connects to any one of these processes from across the network and provides the password, the attacker is instantly given remote root access.

In addition, various local commands are modified to include local root-level backdoors. The chfn, chsh, passwd, and su commands are overwritten with new versions. If a non-root user runs any of these programs with a command argument that is the backdoor password, that user is instantly elevated to root access. Think of these programs as little teleporters to instantly get root.



Common Linux Hiding Components

Rootkit Techniques

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 35

This slide shows numerous programs that are overwritten by various rootkits to hide the attacker's presence.

In essence, there are four categories of hiding tools: Process hiding, network hiding, file hiding, and event hiding.

For process hiding, common Linux rootkits include a replacement or redirection for ps, top, and pidof. These tools will not show the attacker's processes running on the box. In addition, many rootkits replace killall so that the attacker's processes cannot be killed using this command. Finally, the crontab program is often altered so that it starts the attacker's processes upon system boot or at a specific time, without any lines in the cron configuration files associated with these programs.

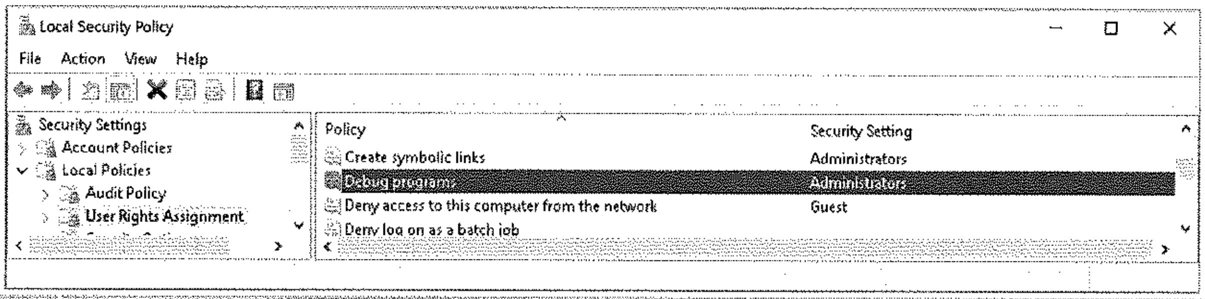
Files are hidden by changing the ls and find commands so that they do not display the attacker's files. The du command is changed so that it omits the attacker's file from its disk usage calculation.

Finally, the attackers modify syslogd so that it will not record log events associated with the attacker's machine and/or accounts on the victim box.

Windows User-Mode Rootkit: DLL Inj. and API Hooking

Rootkit Techniques

- EXEs and DLLs are commonly used methods for packaging code in Windows
  - EXEs run, and utilize shared DLLs to get stuff done
- On Windows, users with the Debug right can inject a DLL into a running process (and start it running by creating a thread in the target process)
- Hook APIs to change programs' views of running processes, open ports, and the filesystem



Local Security Policy

File Action View Help

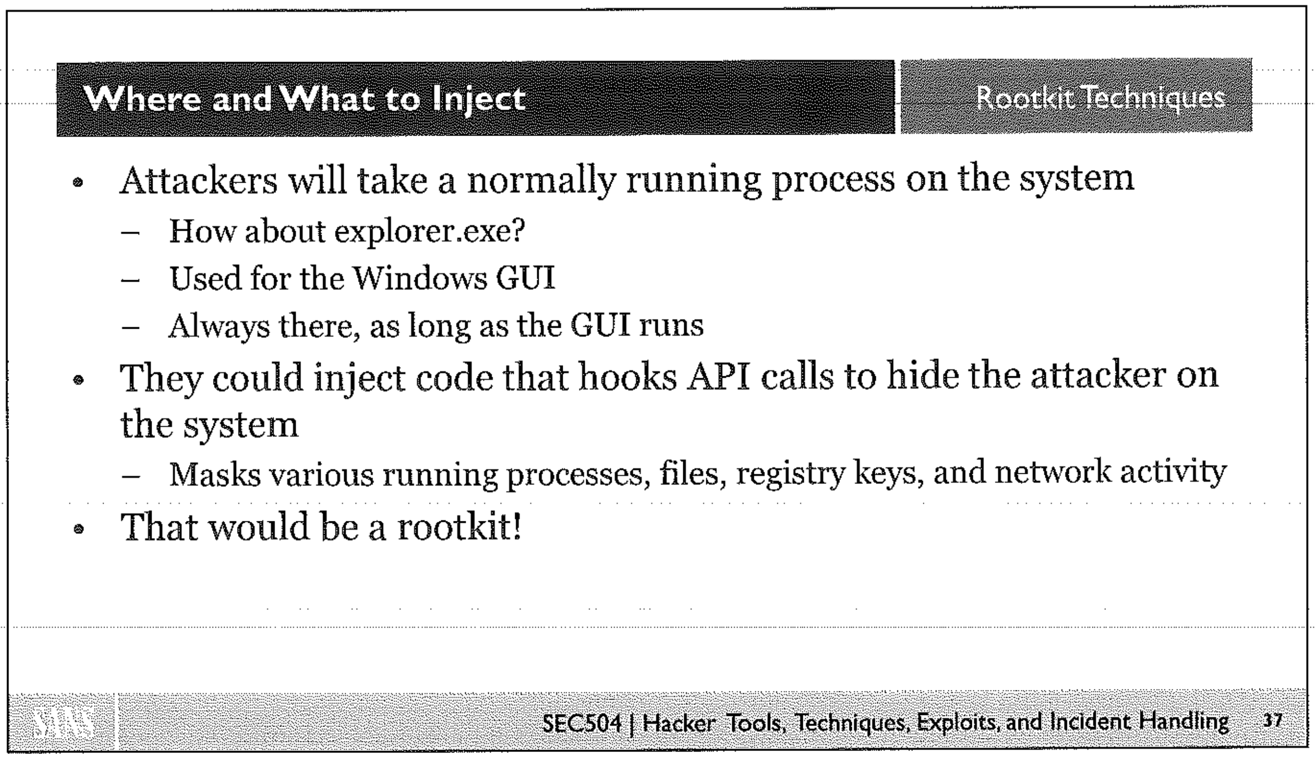
Security Settings
- Account Policies
- Local Policies
  - Audit Policy
  - User Rights Assignment

Policy	Security Setting
Create symbolic links	Administrators
Debug programs	Administrators
Deny access to this computer from the network	Guest
Deny log on as a batch job	

To understand Windows rootkits, we first have to analyze the concept of DLL injection on Windows. An EXE program loads the various DLLs it requires and relies on them to take actions on the system. Attackers use a technique called "DLL injection" to force an unsuspecting running EXE process to accept a DLL that it never requested. Very rudely, an attacker injects code in the form of a DLL directly into the victim EXE process's memory space. DLL injection requires several steps to be taken by the attacker, including:

- **Allocating space in the victim process for the DLL code to occupy**: Microsoft has included a built-in API in Windows to accomplish this task, called "VirtualAllocEx."
- **Allocating space in the victim process for the parameters required by the DLL to be injected**: This step, too, can be done using the built-in Windows VirtualAllocEx function call.
- **Writing the name and code of the DLL into the memory space of the victim process**: Again, Windows includes an API with a function for doing this step too. The WriteProcessMemory function call can be used to write arbitrary data into the memory of a running process.
- **Creating a thread in the victim process to actually run the newly injected DLL**: As you might have guessed by now, Windows includes an API with this capability too. Microsoft has made this entire process much easier with these various API calls. The CreateRemoteThread starts an execution thread in another process, which will run any code already in that process, including a newly injected DLL.
- **Freeing up resources in the victim process after execution is completed**: If the attacker is extra polite, he or she can even free up the resources consumed by this technique after the victim thread or process finishes running, using the VirtualFreeEx function.
- **Overwriting API calls**: This technique, called "API Hooking," lets an attacker undermine any running process in its interactions with Windows itself. By changing various calls associated with getting a list of running processes, looking at open ports, viewing the registry, and interacting with the filesystem, the attacker can hide.

You can see which accounts on your local Windows system have Debug privileges by going to Start → Run... and typing **secpol.msc**. Then, navigate to Security Settings → Local Policies → User Rights Assignments. Then, look at Debug programs. You'll see, by default, that this right is given to everyone in the admin group. On my production environment, I have removed this right, because developers run debuggers in a dev environment, and sometimes in a Quality Assurance (QA) environment, but never in a production environment in most organizations.



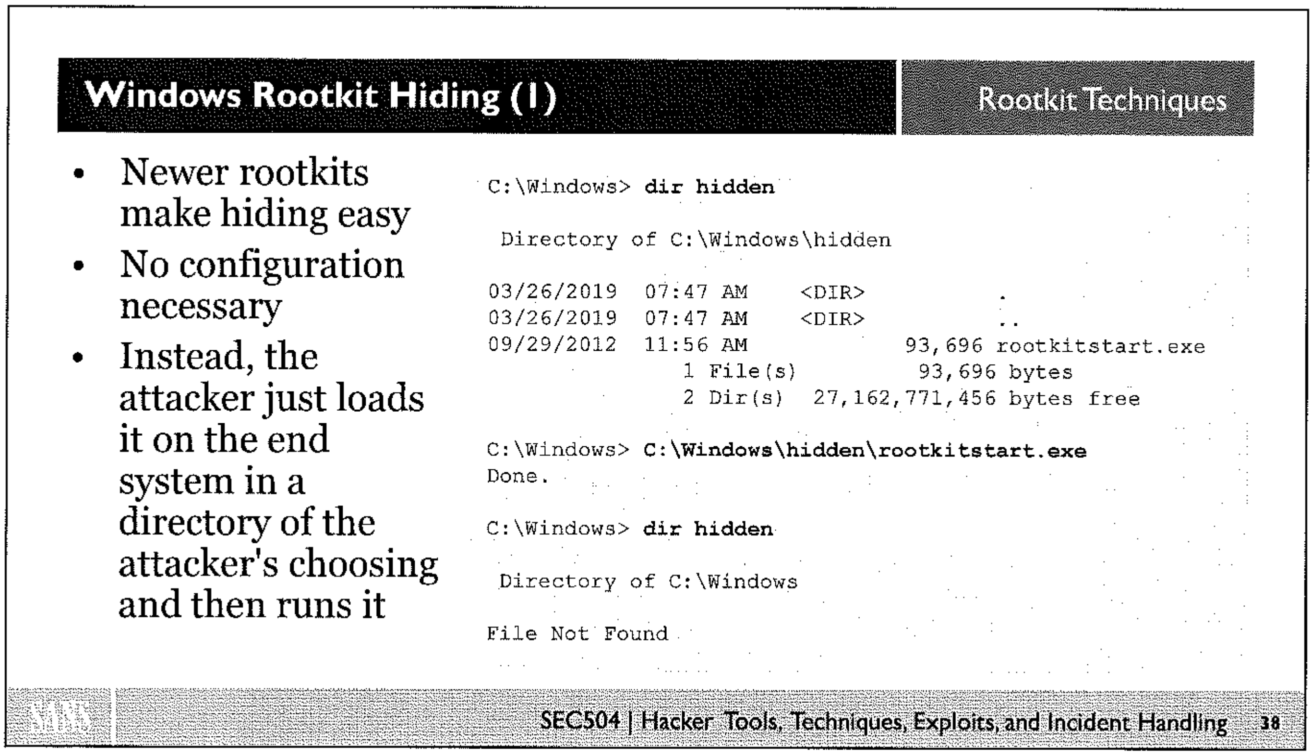
Where and What to Inject

Rootkit Techniques

- Attackers will take a normally running process on the system
  - How about explorer.exe?
  - Used for the Windows GUI
  - Always there, as long as the GUI runs
- They could inject code that hooks API calls to hide the attacker on the system
  - Masks various running processes, files, registry keys, and network activity
- That would be a rootkit!

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 37

So, attackers can inject code into any running process. Which processes lend themselves to rootkit-style attack? One particularly interesting target is the explorer.exe process. This process implements the Windows GUI that you probably stare into, day in and day out. It's always running, as long as the Windows machine is displaying a GUI. This process displays information to the user, so an attacker could undermine it to mask the attacker's presence. The bad guy could inject code into the explorer.exe process to hide the attacker's running processes, files, registry keys, and network activity. In other words, an attacker could implement a rootkit!



Windows Rootkit Hiding (1)

Rootkit Techniques

- Newer rootkits make hiding easy
- No configuration necessary
- Instead, the attacker just loads it on the end system in a directory of the attacker's choosing and then runs it

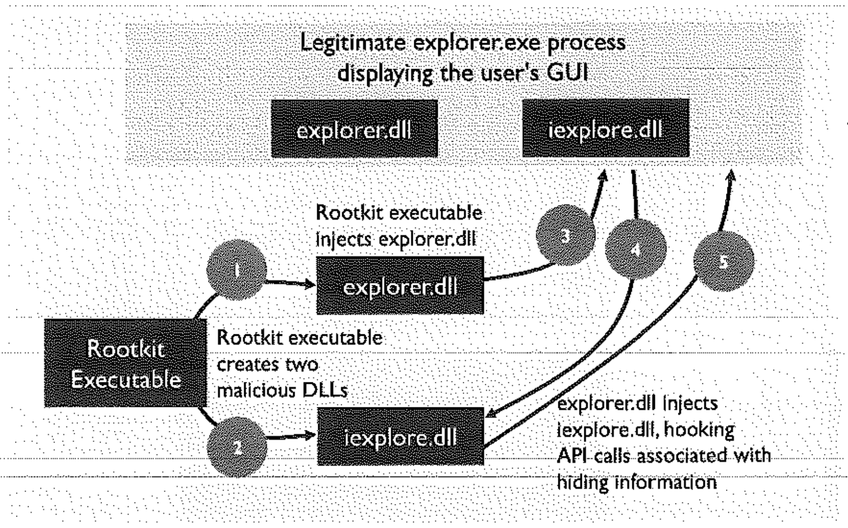
```
C:\Windows> dir hidden

 Directory of C:\Windows\hidden

03/26/2019  07:47 AM    <DIR>          .
03/26/2019  07:47 AM    <DIR>          ..
09/29/2012  11:56 AM            93,696 rootkitstart.exe
               1 File(s)         93,696 bytes
               2 Dir(s)  27,162,771,456 bytes free

C:\Windows> C:\Windows\hidden\rootkitstart.exe
Done.

C:\Windows> dir hidden

 Directory of C:\Windows

File Not Found
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 38

The attacker just places the rootkit executable on the target machine in a given directory and runs it with admin privileges. Any files in that directory are hidden. Any processes associated with executables from that registry are hidden. Any registry keys created by processes run out of that directory are hidden. Of course, TCP and UDP ports listened on by something from that directory are hidden as well.

Rootkit Hooking in Action

Rootkit Techniques



Legitimate explorer.exe process displaying the user's GUI
explorer.dll
iexplore.dll
Rootkit executable injects explorer.dll
1
2
3
4
5
explorer.dll
Rootkit Executable
Rootkit executable creates two malicious DLLs
iexplore.dll
explorer.dll injects iexplore.dll, hooking API calls associated with hiding information

NOTE

These names may be confusing. That's intentional. There are no legitimate explorer.dll or iexplore.dll files on Windows, but they seem plausible enough to avoid suspicion.

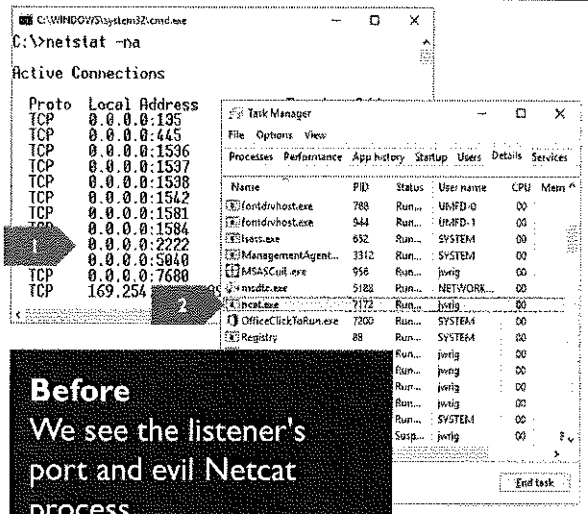
Let's look at the process a Windows rootkit may use to hide its presence on a system. When the rootkit runs on the victim machine, the rootkit executable first makes a copy of itself in the system32 directory. Then, in Steps 1 and 2, it creates two other files in the same directory: iexplore.dll and explorer.dll.

With names like that, these files sure look like they belong on the machine, don't they? They look kind of like some files you might think are associated with the legitimate programs Internet Explorer (iexplore.exe) and Windows Explorer (explorer.exe). But pay careful attention to the file suffixes here; the rootkit creates iexplore.dll and explorer.dll. On a stock Windows machine, there aren't any files named "iexplore" and "explore" with a DLL suffix. That's pretty tricky.

After writing these DLLs in the system32 directory, the rootkit executable injects the explorer.dll into running processes named "explorer.exe," in Step 3. The explorer.exe process is the legitimate running program that displays the Windows GUI to the user. Once inside the legitimate explorer.exe process, the malicious explorer.dll then does API hooking. It grabs the code inside iexplore.dll in Step 4. To finish the process, in Step 5, the explorer.dll then injects iexplore.dll into the explorer.exe process, overwriting function calls associated with displaying processes, files, registry keys, and connections. When a standard Windows tool, such as the task manager, file viewer, registry editor, or netstat command, is executed, the malicious API code injected into the legitimate Windows Explorer filters the hidden stuff from the output. In this way, the attacker's nefarious deeds are hidden on the machine.

Windows Rootkit Hiding (2)

Rootkit Techniques



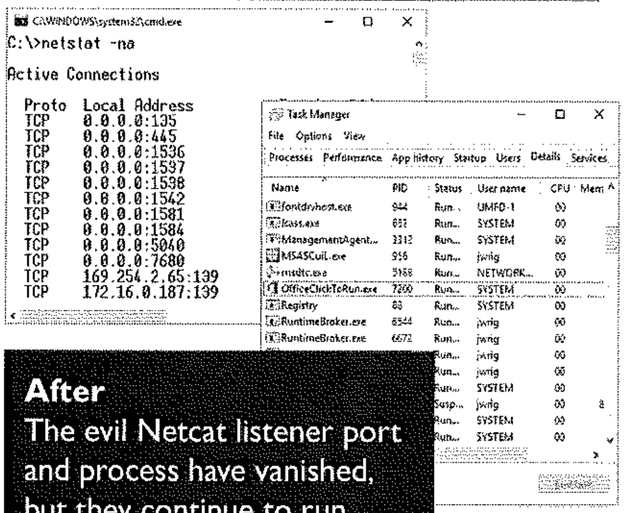
C:\WINDOWS\system32\cmd.exe

```
C:\>netstat -na

Active Connections

  Proto  Local Address
  TCP    0.0.0.0:135
  TCP    0.0.0.0:445
  TCP    0.0.0.0:1536
  TCP    0.0.0.0:1537
  TCP    0.0.0.0:1538
  TCP    0.0.0.0:1542
  TCP    0.0.0.0:1581
         0.0.0.0:1584
         0.0.0.0:2222
         0.0.0.0:5040
  TCP    0.0.0.0:7680
  TCP    169.254
```

1
2
Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Mem
fontdrvhost.exe	788	Run...	UMFD-0	00	
fontdrvhost.exe	944	Run...	UMFD-1	00	
lsass.exe	652	Run...	SYSTEM	00	
ManagementAgent...	3312	Run...	SYSTEM	00	
MSASCuiL.exe	956	Run...	jwrig	00	
msdtc.exe	5188	Run...	NETWORK...	00	
ncat.exe	7172	Run...	jwrig	00	
OfficeClickToRun.exe	7200	Run...	SYSTEM	00	
Registry	88	Run...	SYSTEM	00	
		Run...	jwrig	00	
		Run...	jwrig	00	
		Run...	jwrig	00	
		Run...	jwrig	00	
		Run...	SYSTEM	00	
		Susp...	jwrig	00	

Before
We see the listener's port and evil Netcat process



C:\WINDOWS\system32\cmd.exe

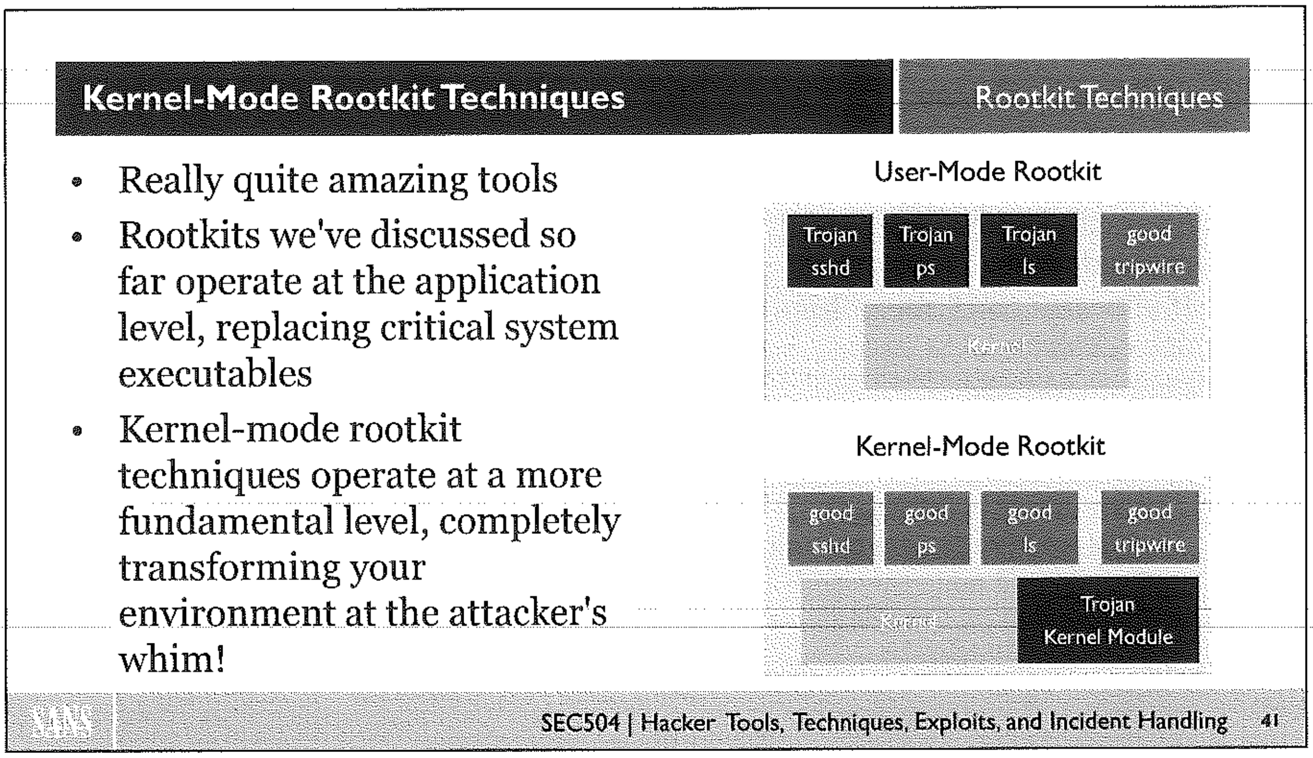
```
C:\>netstat -na

Active Connections

  Proto  Local Address
  TCP    0.0.0.0:135
  TCP    0.0.0.0:445
  TCP    0.0.0.0:1536
  TCP    0.0.0.0:1537
  TCP    0.0.0.0:1538
  TCP    0.0.0.0:1542
  TCP    0.0.0.0:1581
  TCP    0.0.0.0:1584
  TCP    0.0.0.0:5040
  TCP    0.0.0.0:7680
  TCP    169.254.2.65:139
  TCP    172.16.0.187:139
```

Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Mem
fontdrvhost.exe	944	Run...	UMFD-1	00	
lsass.exe	652	Run...	SYSTEM	00	
ManagementAgent...	3312	Run...	SYSTEM	00	
MSASCuiL.exe	956	Run...	jwrig	00	
msdtc.exe	5188	Run...	NETWORK...	00	
OfficeClickToRun.exe	7200	Run...	SYSTEM	00	
Registry	88	Run...	SYSTEM	00	
RuntimeBroker.exe	6544	Run...	jwrig	00	
RuntimeBroker.exe	6672	Run...	jwrig	00	
		Run...	jwrig	00	
		Run...	jwrig	00	
		Run...	SYSTEM	00	
		Susp...	jwrig	00	8
		Run...	SYSTEM	00	
		Run...	SYSTEM	00	

After
The evil Netcat listener port and process have vanished, but they continue to run

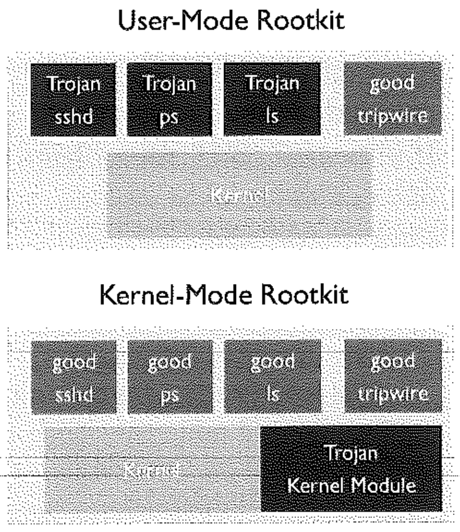
On the left side of this figure, you can see what a normal Netcat listener (named ncat.exe) looks like as a running process with a listening port on the box before the rootkit was installed. On the right side, you can see what the same system looks like when the rootkit is installed, configured to hide the evilnc.exe process and the port that it is listening on, TCP 2222.



Kernel-Mode Rootkit Techniques

Rootkit Techniques

- Really quite amazing tools
- Rootkits we've discussed so far operate at the application level, replacing critical system executables
- Kernel-mode rootkit techniques operate at a more fundamental level, completely transforming your environment at the attacker's whim!



SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 41

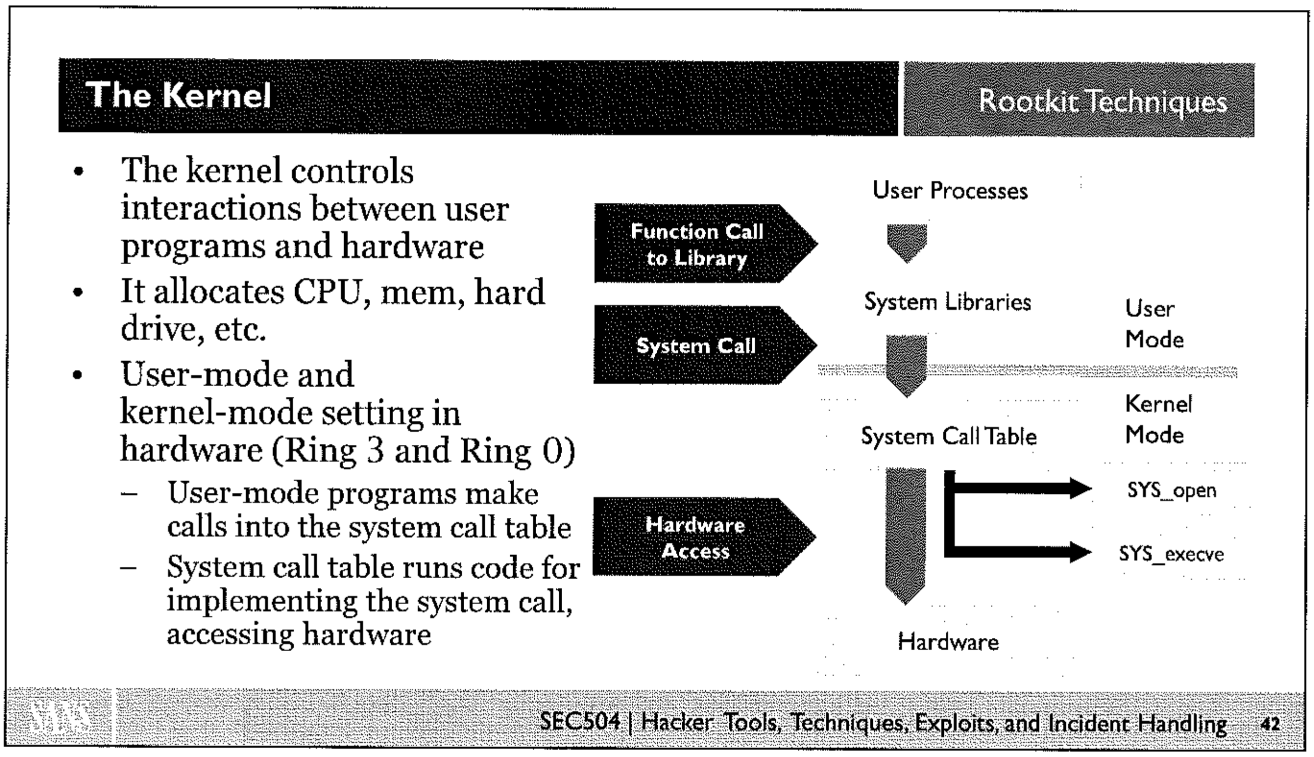
Kernel-mode rootkit techniques represent the latest evolution of rootkit-like tools. Many modern tools trace their history back to some ideas that were originally included in a tool named "sumfuq" for SunOS 4.1.X, and itf.c for Linux. So yes, these ideas have been around for a long time!

Because they run at the kernel mode, kernel-mode rootkit techniques have much more power over the system than rootkits that live at the process/program/application level. Detection is far more difficult because detection programs themselves run at the process/program/application level and rely on the kernel to function.

For example, a good Tripwire routine running on a system can detect a user-mode rootkit Trojan Horse program by opening the programs and looking at them. Keep in mind, however, that to open a program file, the filesystem integrity checker has to make calls into the system kernel. All access to the hard drive or any other hardware components of the machine must use the kernel.

With a kernel-mode rootkit installed, the attacker doesn't have to modify the individual application programs. When the kernel is modified to lie to administrators, all files would be intact, while the actual kernel can give backdoor access and hide the attacker.

By operating at the kernel, the attacker can essentially create an alternate universe within your computer that looks intact and happy. Really, though, beyond this appearance, your system could be laced with Trojan Horse programs.



The Kernel

Rootkit Techniques

- The kernel controls interactions between user programs and hardware
- It allocates CPU, mem, hard drive, etc.
- User-mode and kernel-mode setting in hardware (Ring 3 and Ring 0)
  - User-mode programs make calls into the system call table
  - System call table runs code for implementing the system call, accessing hardware

User Processes
Function Call to Library
System Libraries
System Call
User Mode
Kernel Mode
System Call Table
Hardware Access
SYS_open
SYS_execve
Hardware

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 42

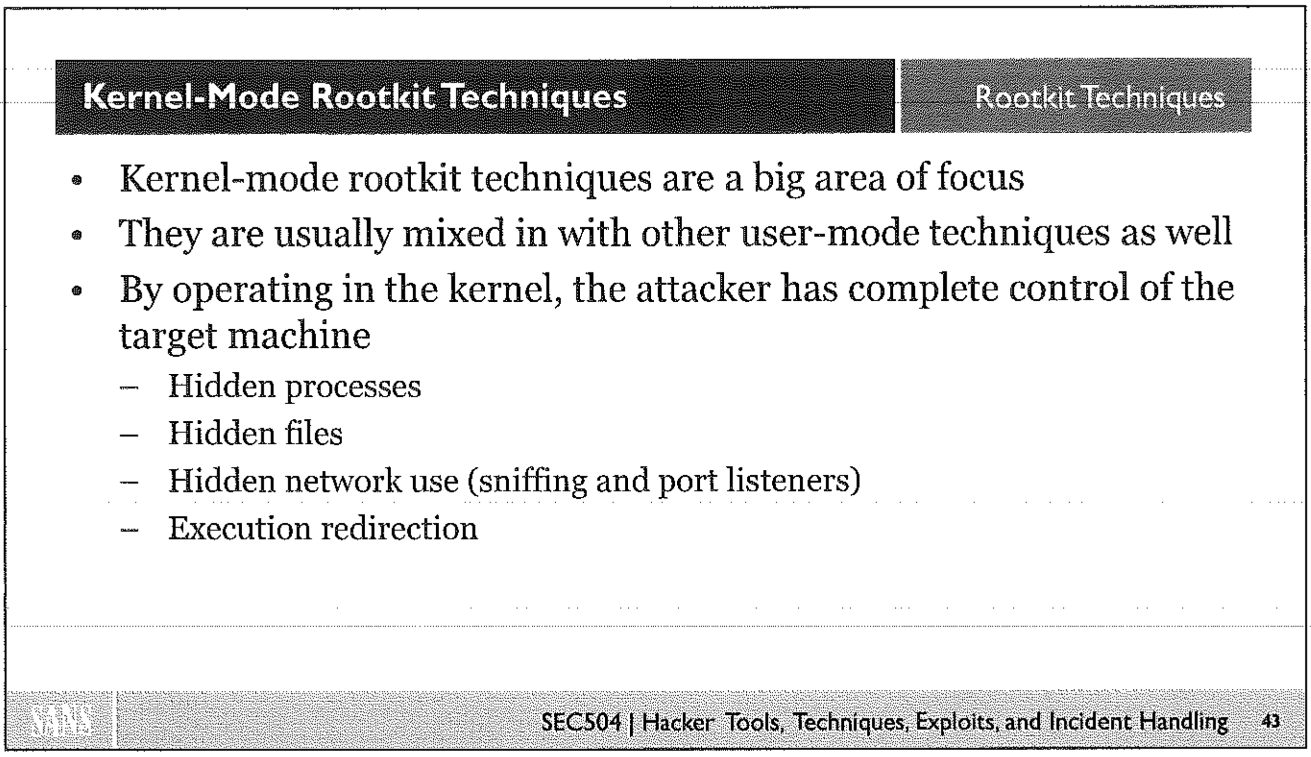
In most operating systems, including UNIX and Windows, the kernel is special software that controls various extremely important elements of the machine. The kernel sits between individual running programs and the hardware itself. Many kernels, including those found in UNIX and Windows systems, include the following core features: Process and thread control, inter-process communication control, memory control, filesystem control, other hardware control, and interrupt control.

As it runs, the kernel relies on hardware-level protections implemented in the system's CPU. By using hardware-level protection, the kernel tries to safeguard its own critical data structures from accidental or deliberate manipulation by user-level processes on the machine. Most CPUs include hardware features to let software on the system run at different levels of privilege. The memory space and other elements of highly sensitive software (like the kernel) cannot be accessed by code running at a less important level (such as user processes). On x86-compatible CPUs, these different sensitivity levels are called "Rings" and range from Ring 0, the most sensitive level, to Ring 3, the least sensitive level. As it runs different tasks, the CPU switches between these different levels depending on the sensitivity of the particular software currently executing.

For the Linux and Windows operating systems, only Rings 0 and 3 are used, while the other options supported by x86 CPUs (that is, Rings 1 and 2) are not utilized. The kernel itself, in both Linux and Windows, runs in Ring 0. User-mode processes run in Ring 3, and, under most conditions, are not able to access kernel space directly.

To interact with the kernel, user-mode processes rely on a concept termed "system calls." These system calls include functions for executing a program or opening a file. Now, most user-mode processes don't activate these system calls directly. Instead, the operating system includes a system library full of code that actually invokes the system call when it is required. So, a running user-mode process calls a system library to take some action. The system library, in turn, activates a system call in the kernel. To activate a system call, the system library sends an interrupt to the CPU, essentially tapping the CPU on the shoulder, telling it that it needs to change to Ring 0 and handle a system call using kernel-mode code.

The system call table is actually an array maintained by the kernel that maps individual system call names and numbers into the corresponding code inside the kernel needed to handle each system call. The system call table is just a collection of pointers to various chunks of the kernel that implement the actual system calls.



Kernel-Mode Rootkit Techniques

Rootkit Techniques

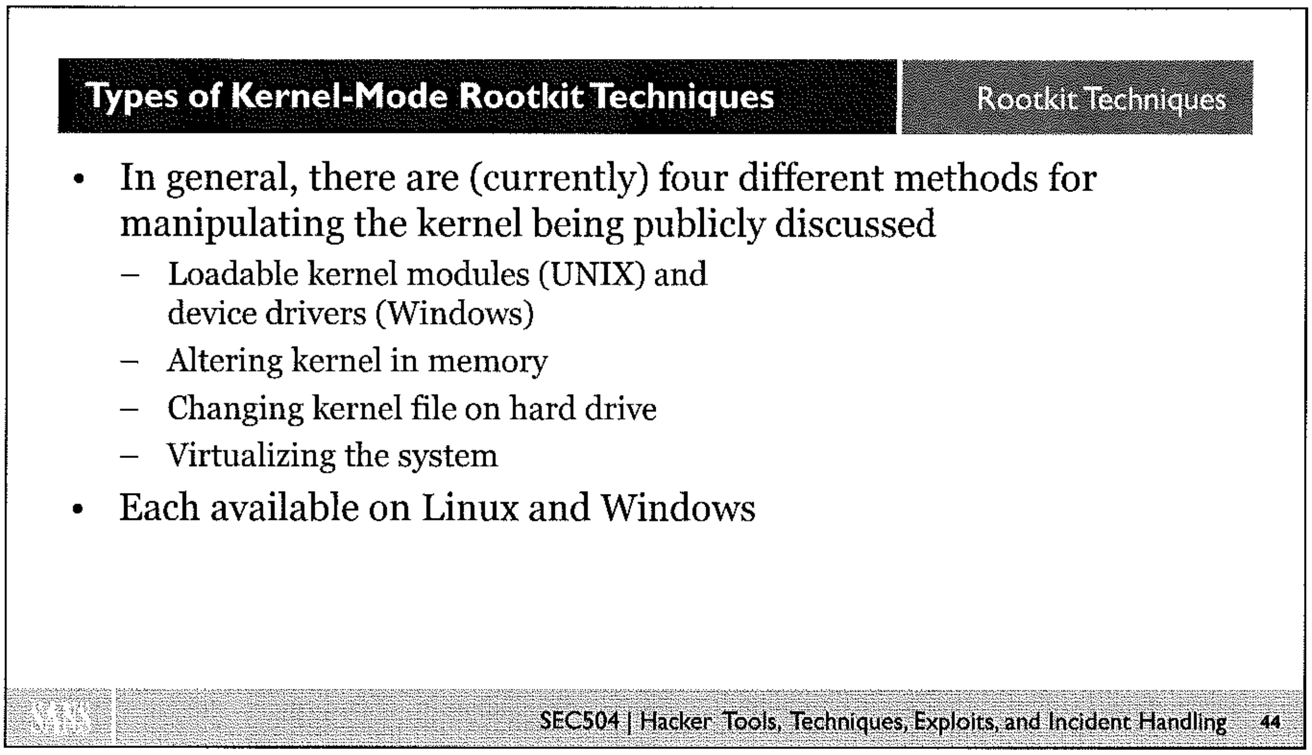
- Kernel-mode rootkit techniques are a big area of focus
- They are usually mixed in with other user-mode techniques as well
- By operating in the kernel, the attacker has complete control of the target machine
  - Hidden processes
  - Hidden files
  - Hidden network use (sniffing and port listeners)
  - Execution redirection

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 43

Kernel-mode rootkit techniques continue to get a lot of attention from attackers. They allow the attacker to completely undermine the system. They require root permissions to install. However, once they are on the system, they let an attacker hide all of his/her nefarious activities. Most kernel-mode rootkit techniques let an attacker:

- Hide processes so backdoors don't show up in a process list.
- Hide files so the attacker's malicious software doesn't show up on the filesystem.
- Hide network usage, including masking any TCP or UDP ports used by an attacker's backdoor, and hide promiscuous mode to disguise sniffers.
- Execute redirection so that when a user runs a program, the kernel will substitute an evil program in its place.

The real focus area lately in kernel-mode rootkit techniques is how to get them implemented. People have been doing loadable kernel modules for kernel rootkits since 1999!



Types of Kernel-Mode Rootkit Techniques

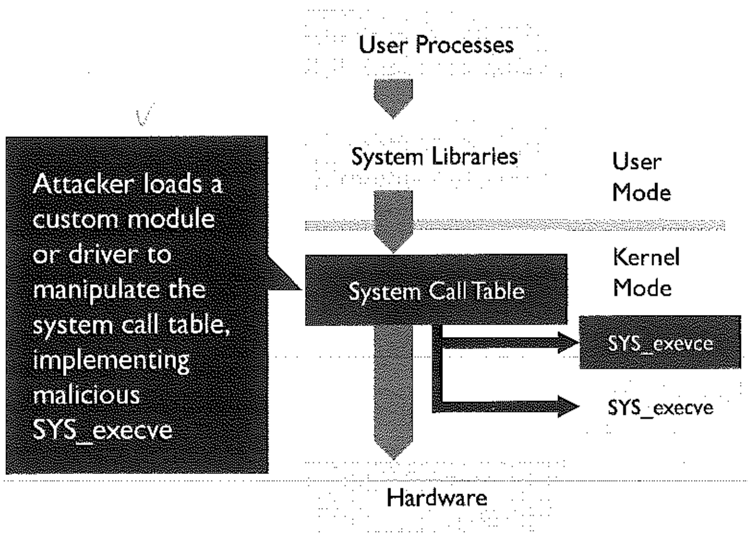
- In general, there are (currently) four different methods for manipulating the kernel being publicly discussed
  - Loadable kernel modules (UNIX) and device drivers (Windows)
  - Altering kernel in memory
  - Changing kernel file on hard drive
  - Virtualizing the system
- Each available on Linux and Windows

The real focus area lately in kernel-mode rootkit techniques is how to get them implemented. People have been doing loadable kernel modules and evil device drivers for kernel rootkits since at least 1997. This technique remains the most popular today. This is because there are a large number of developers and books on device drivers. Many of the same techniques are used for both evil and good purposes.

In the future, we might even see attackers virtualizing systems to implement rootkits or running programs directly in kernel mode. Although these latter two techniques haven't yet caught on, they are a possibility and have been discussed publicly. More importantly, these advanced techniques are getting more and more automated and easy for attackers to use.

Method 1: Loadable Kernel Modules, Drivers

- Can augment kernel capabilities with
  - Loadable modules (Linux)
  - Device drivers (Windows)
  - Overwriting existing kernel functionality
- Windows requires mandatory device driver signing
  - Could steal signing keys (Stuxnet)
  - Or use method 2, altering the kernel in memory

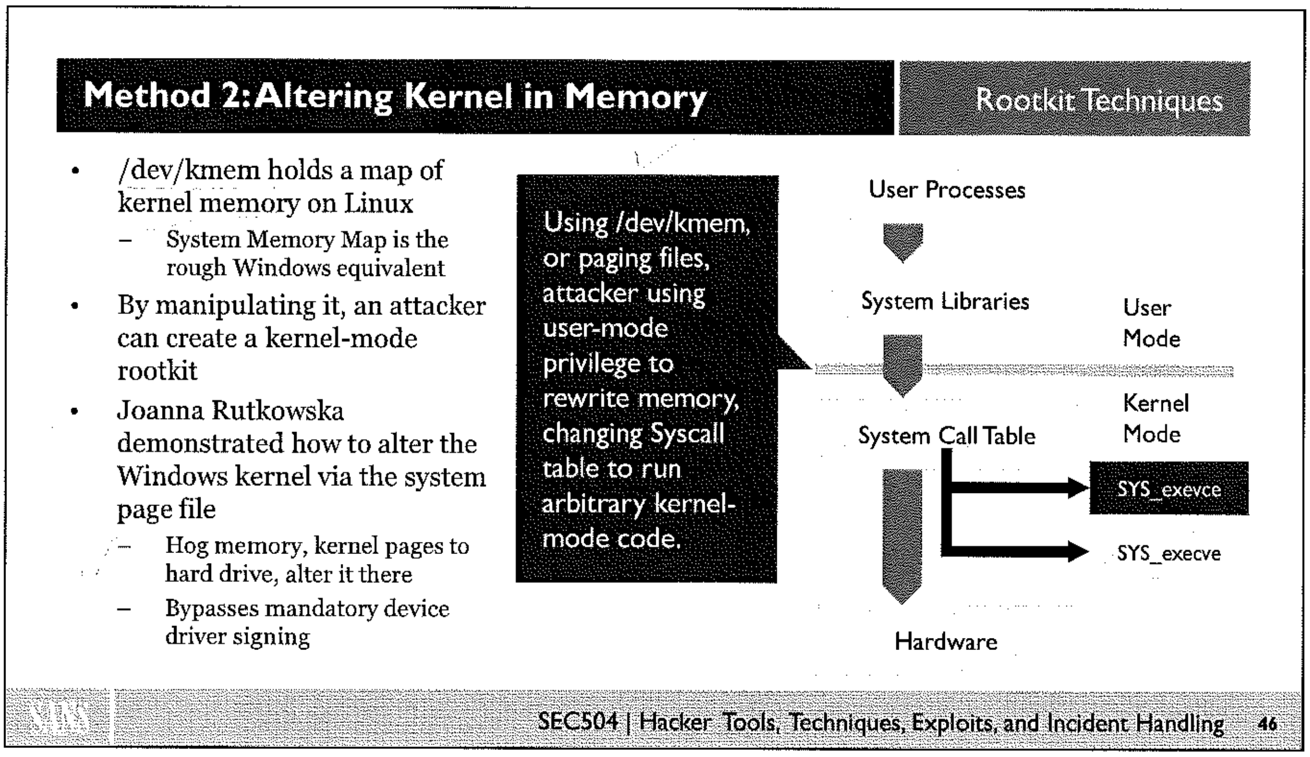

Attacker loads a custom module or driver to manipulate the system call table, implementing malicious SYS_execve
User Processes
System Libraries
User Mode
Kernel Mode
System Call Table
SYS_exevce
SYS_execve
Hardware

A primary method for invading the Linux kernel to implement a kernel-mode rootkit involves creating an evil loadable kernel module that manipulates the existing kernel. Today, it remains the most popular technique for implementing kernel-mode rootkit techniques on Linux systems.

Remember, loadable kernel modules are a legitimate feature of the Linux kernel, sometimes used to add support for new hardware or otherwise insert code into the kernel to support new features. Loadable kernel modules run in kernel mode and can augment or even replace existing kernel features, all without a system reboot. Because of the convenience of this feature for injecting new code into the kernel, it's one of the easiest methods for implementing kernel-mode rootkit techniques on systems that support kernel modules (such as Linux and Solaris). To abuse this capability for implementing rootkits, some malicious loadable kernel modules change the way that various system calls are handled by the kernel.

Similarly, by creating malicious device drivers, an attacker can undermine the Windows kernel. Device drivers run at kernel mode and have been used to implement Windows kernel-mode rootkit techniques by altering the system call table.

Starting with Windows Vista (and following with Windows 7, Windows Server 2008, and Windows 8 and 10), Microsoft required mandatory device driver signing for Windows kernel components. Before this, Windows allowed a user with admin privileges to load unsigned code into the kernel (or code signed by a key that Windows doesn't trust). Thus, once an attacker compromised admin privileges, he or she could accept unsigned code and load a kernel-mode rootkit. But, with Vista, Win7/8/10, and Windows Server 2008, Windows has mandatory device driver signing. There are ways to subvert this process, however. One way is to steal legitimate private keys issued by Microsoft to a legitimate software company and use them to sign malware. This technique was used in the Stuxnet worm, which relied on stolen signing keys issued to two legitimate companies. Alternatively, mandatory device driver signing could be bypassed using method 2 for altering the kernel: Manipulating memory.



Method 2: Altering Kernel in Memory

Rootkit Techniques

- /dev/kmem holds a map of kernel memory on Linux
  - System Memory Map is the rough Windows equivalent
- By manipulating it, an attacker can create a kernel-mode rootkit
- Joanna Rutkowska demonstrated how to alter the Windows kernel via the system page file
  - Hog memory, kernel pages to hard drive, alter it there
  - Bypasses mandatory device driver signing

Using /dev/kmem, or paging files, attacker using user-mode privilege to rewrite memory, changing Syscall table to run arbitrary kernel-mode code.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 46

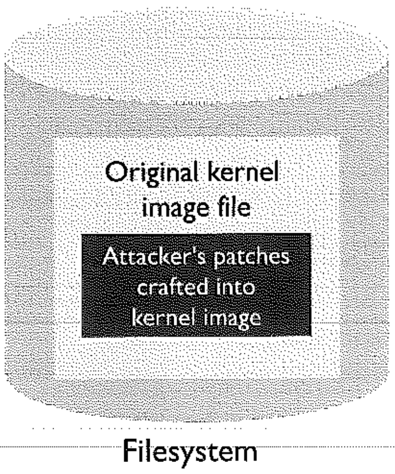
Although modifying a running kernel using loadable kernel modules and device drivers is a widespread and effective technique, it's not the only game in town for implementing kernel-mode rootkit techniques. Suppose the target machine was built without kernel module support. When compiling a custom kernel for a Linux machine, an administrator can choose whether to add loadable kernel module support or omit it from the resulting kernel. Without module support in the kernel, the administrator will have to build all kernel-mode functionality right into the core kernel itself. Such kernels cannot be abused with evil loadable kernel modules because the hooks necessary for loading such modules into the kernel (stuff like the /proc/ksyms file) are left out. For information about building a kernel that doesn't require or support modules, you can refer to various free internet guides.

Some libraries for the core operating systems today have to exist in the same location for service pack/language builds. Attackers can pull these locations using tools like debuggers or msfpescan/msfelfscan to find these locations. Yes, this also impacts systems with ASLR.

Similar ideas have been implemented on Windows, utilizing the System Memory Map object. In particular, the FU tool implemented this functionality on Windows years ago. Joanna Rutkowska has also demonstrated how to change the Windows kernel. These are neat proof-of-concept examples that displayed the first incarnations of these attacks. The attacker writes a user-mode program that hogs memory, forcing the kernel to page some of its functionality to the system page file. Then, with system privileges in user mode, she changes the page file elements that contain kernel code. Any system-level process on Windows can read and write the page file. With the page file alterations in place, she frees up memory. The kernel then loads the evil code back into kernel space. This method bypasses the mandatory device driver signing functionality of the Windows kernel by changing the kernel in the Windows virtual memory system.

Method 3: Changing Kernel File on Hard Drive

- Instead of altering live kernel in memory, attackers could overwrite kernel file on the hard drive
- On Linux, the file is vmlinuz
  - *Phrack Magazine*, "Static Kernel Patching" by jbtzhm
- On Windows, kernel functionality is in ntoskrnl.exe and win32k.sys files
  - Attacker must foil NTLDR integrity checks of these files
  - Disable them or make them lie (alter their code or in memory on startup)

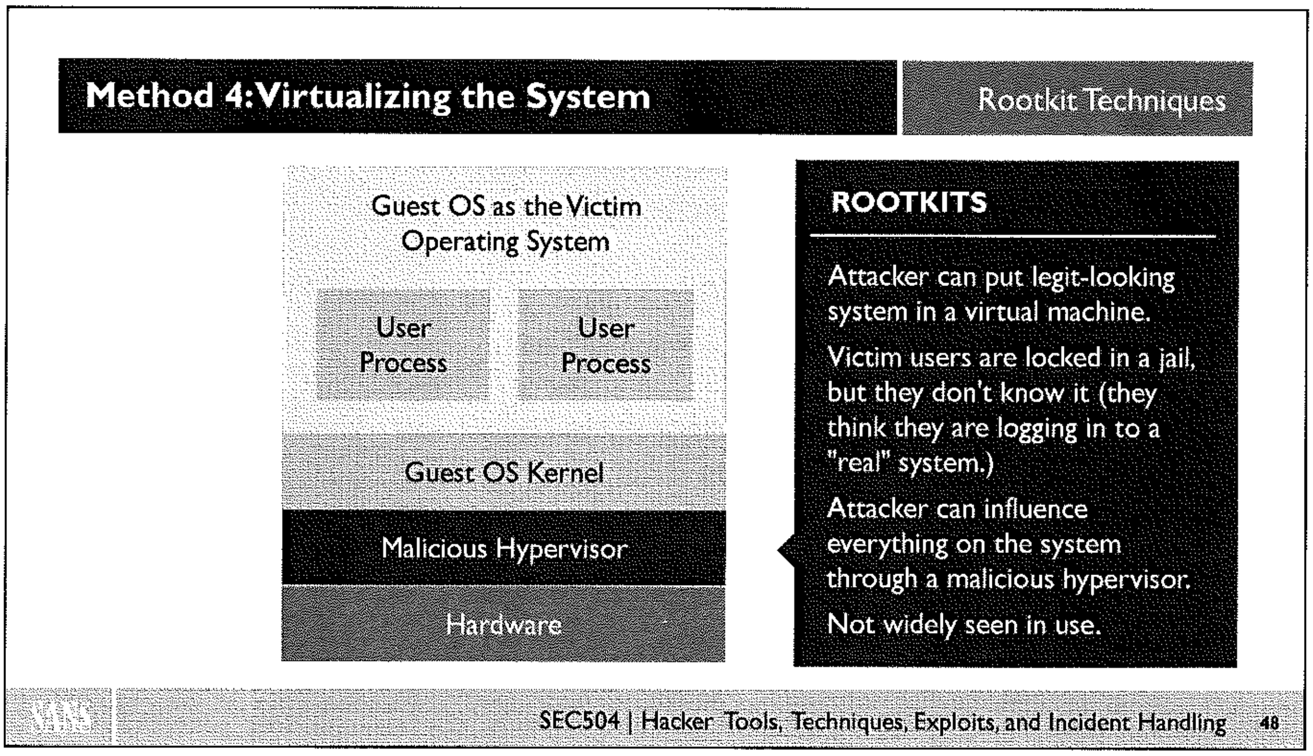

Original kernel image file
Attacker's patches crafted into kernel image
Filesystem

With root-level permissions on the box, the attacker could just replace or patch the kernel image file on the hard drive itself. That way, upon the next reboot of the system, the attacker's evil kernel would be reloaded into the system instead of the original wholesome kernel.

Because the kernel image on the hard drive is just a file (readable and writable by root-level accounts), there's no need for the attacker to jump from user mode to kernel mode to make changes to this file. User-mode-to-kernel-mode transitions (such as those that occur through system calls, insmod, and /dev/kmem) are required only to interact with a running kernel, but aren't necessary to change the kernel image file on the hard drive. By just exercising rootly privileges, the attacker can overwrite the kernel image file on the hard drive and get the new, evil kernel loaded into memory at the next reboot.

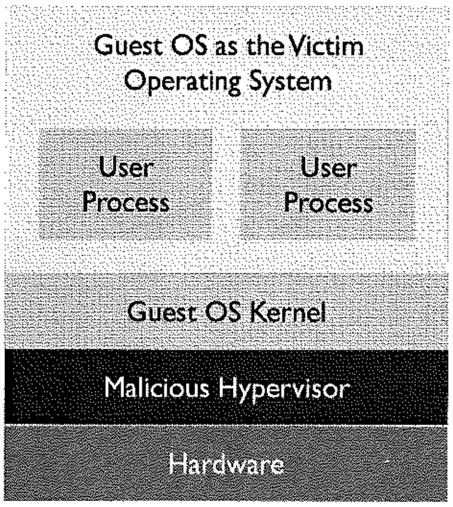
In the Linux filesystem, the kernel image is stored in a file called "vmlinuz," typically located in the /boot directory. An informative article on patching this image is "Static Kernel Patching" by jbtzhm, published in *Phrack Magazine* issue 60 at http://phrack.org/issues/60/8.html#article.

On Windows, an attacker would alter Ntoskrnl.exe or win32k.sys with modified software that provides a backdoor and hides an attacker's presence on the machine. Now, an attacker cannot alter the Ntoskrnl.exe file by itself because the integrity of this file is checked each time the system boots. During the boot process, a program called "NTLDR" verifies the integrity of Ntoskrnl.exe before the kernel is loaded into memory. If the Ntoskrnl.exe file has been altered, the NTLDR program displays a fearsome blue screen of death message, indicating that the kernel itself is corrupt. To get around this difficulty, the bad guys manipulate both the NTLDR and the Ntoskrnl.exe files. Using a small patch to overwrite a few machine language instructions inside NTLDR so that it skips its integrity check, the attackers can then freely alter Ntoskrnl.exe at will.



Method 4: Virtualizing the System

Rootkit Techniques



ROOTKITS

Attacker can put legit-looking system in a virtual machine.

Victim users are locked in a jail, but they don't know it (they think they are logging in to a "real" system.)

Attacker can influence everything on the system through a malicious hypervisor.

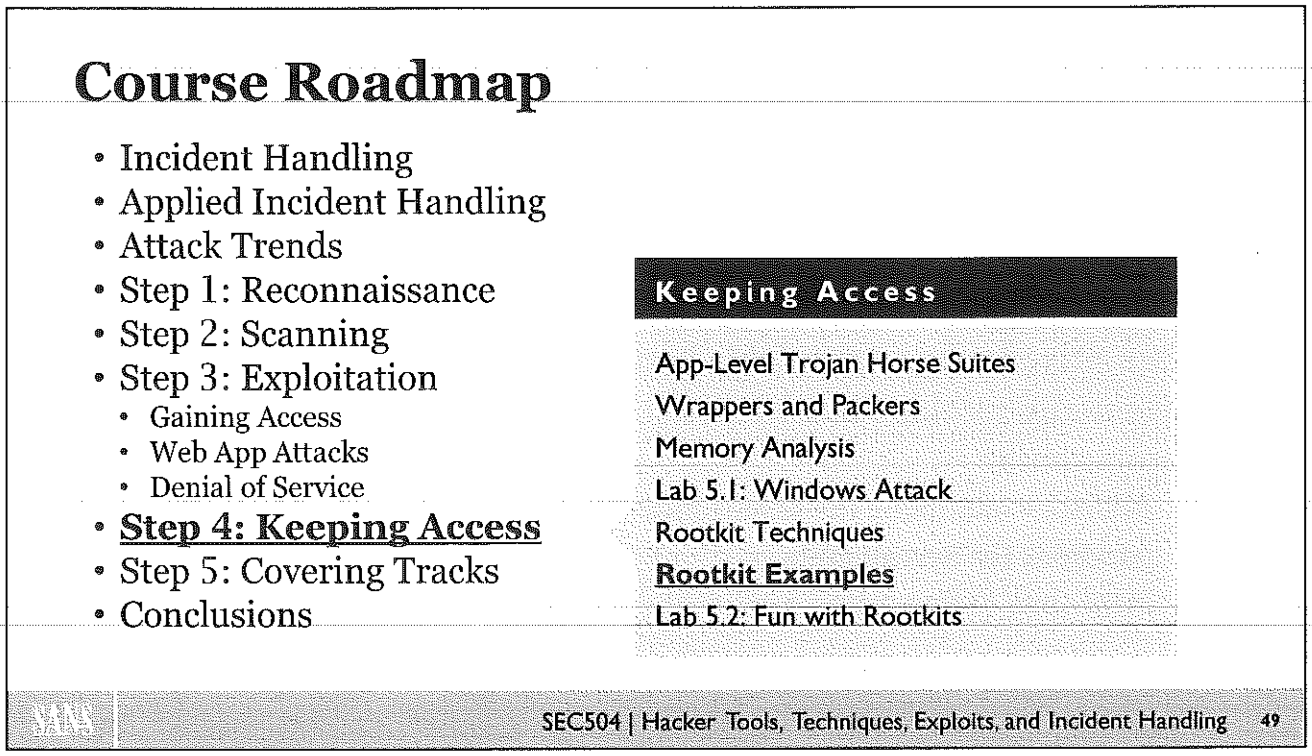
Not widely seen in use.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 48

Another kernel-mode rootkit alternative involves the bad guy breaking into the machine with root privileges, and making the existing operating system a guest inside a virtual machine environment. Then, after starting this virtual machine containing the original system, the attacker runs underneath with his or her own hypervisor, controlling the system underneath the kernel of the guest operating system. All users and administrators logging in to the machine would be unwittingly accessing the guest instance, and not the "real" underlying operating system, controlled by the attacker. The attacker, meanwhile, could run all sorts of nasty code underneath in the hypervisor, which the users inside the virtual instance would not be able to notice. In essence, this attack works like a reverse honeypot. Instead of trapping attackers inside a jail without their knowing it, which normal honeypots do, this type of attack traps system administrators and users in a jail.

By deploying a virtual machine on a victim system, attackers turn the whole system into their playground, confining normal users and administrators into a small virtual prison tucked away in a corner of the system. The real concern here, of course, is that the users and administrators have no idea that they are in a prison. The virtual machine becomes a cone of silence wrapped around legitimate users and administrators. With the virtual machine going about its business, the system looks normal to them. Their normal kernel is running, all of their files are still on the hard drive, and programs run the same way they did before the attack occurred. The victims are blissfully ignorant of their virtual machine–induced cage.

Historically speaking, in 2006, researchers at the University of Michigan and at Microsoft Corporation released the first research paper on this technique, along with a description of a proof-of-concept tool called "SubVert," a virtual machine–based rootkit. This tool implements two rootkits: One with a Windows XP host running Virtual PC and another with a Linux Gentoo host running VMware. Going further, Joanna Rutkowska then furthered this research and released a paper on her experiments with hardware-based virtualization technologies offered in AMD's Pacific chips, which extend the x86 instruction set to offer direct processor hardware support for virtualization. Joanna's work implements a hypervisor underneath Windows. She called her handiwork "The Blue Pill." Dino Dai Zovi performed similar research on Intel's VT-x technology, its hardware-based virtualization extensions of the x86 instruction set, as well. The resulting Vitriol rootkit is implemented as a hypervisor underneath macOS running on processors supporting the VT-x instruction set.



Course Roadmap

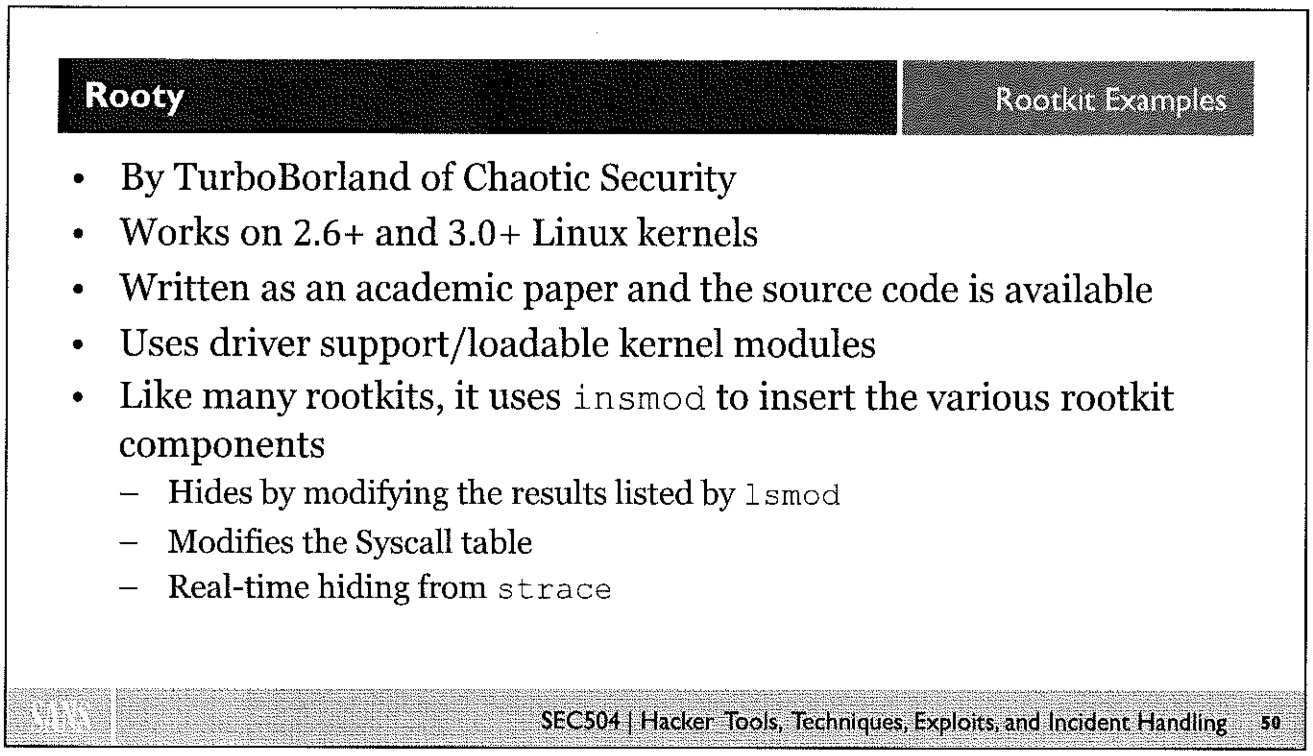
- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- **Step 4: Keeping Access**
- Step 5: Covering Tracks
- Conclusions

Keeping Access

App-Level Trojan Horse Suites
Wrappers and Packers
Memory Analysis
Lab 5.1: Windows Attack
Rootkit Techniques
Rootkit Examples
Lab 5.2: Fun with Rootkits

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 49

Now we explore some rootkit examples to reinforce the concepts in this module.



Rooty

Rootkit Examples

- By TurboBorland of Chaotic Security
- Works on 2.6+ and 3.0+ Linux kernels
- Written as an academic paper and the source code is available
- Uses driver support/loadable kernel modules
- Like many rootkits, it uses `insmod` to insert the various rootkit components
  - Hides by modifying the results listed by `lsmod`
  - Modifies the Syscall table
  - Real-time hiding from `strace`

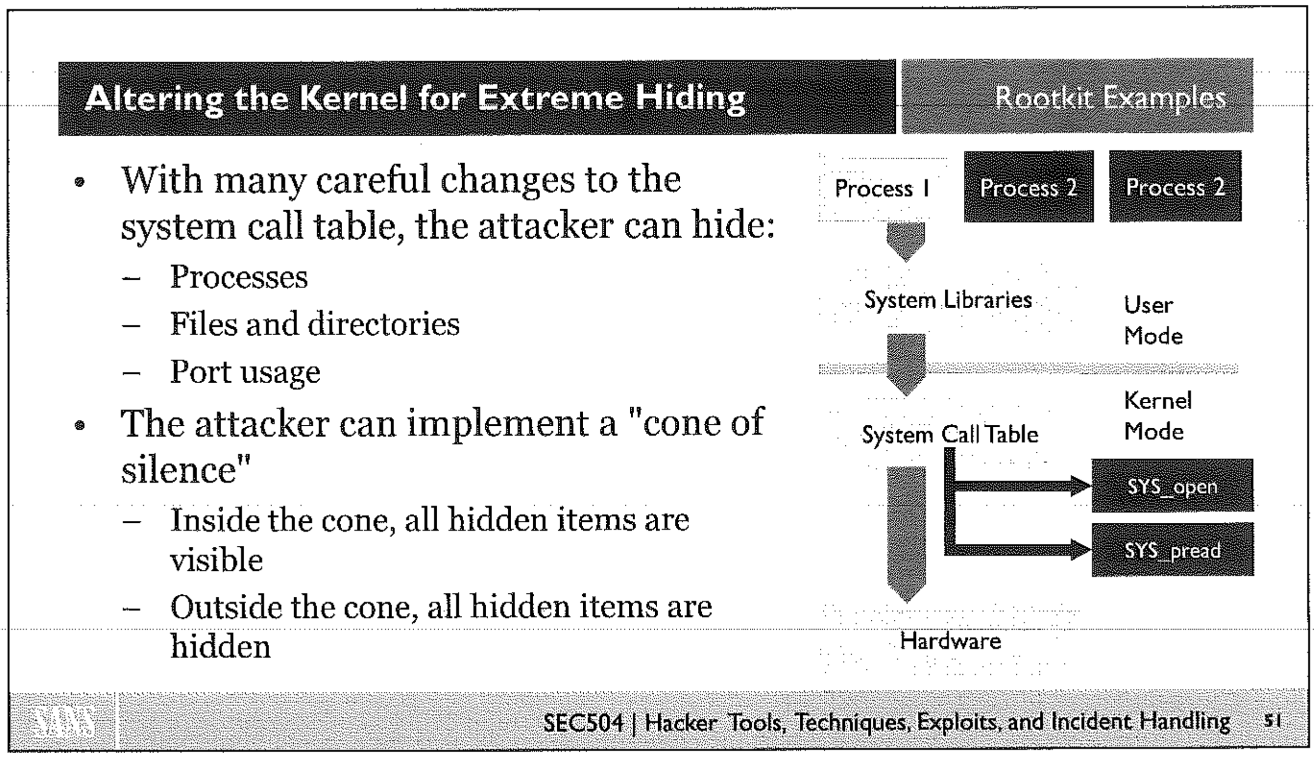
SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 50

Another great rootkit is Rooty by TurboBorland. This kernel-mode rootkit supports 2.6+ and 3.0+ Linux kernels. Another cool feature of this rootkit is the ability to automatically determine whether a target system is 32 or 64 bits and automatically install for the proper version. It does this by reviewing unistd.h and comparing the system calls in this file with the known system call addresses for 32-bit and 64-bit systems.

Like most Linux rootkits, Rooty installs itself as a driver or loadable kernel module. It does this through the use of the insmod command, which is a common way to install drivers on Linux systems.

It also hides by modifying the results listed by lsmod (which lists loaded kernel modules) and from systrace.

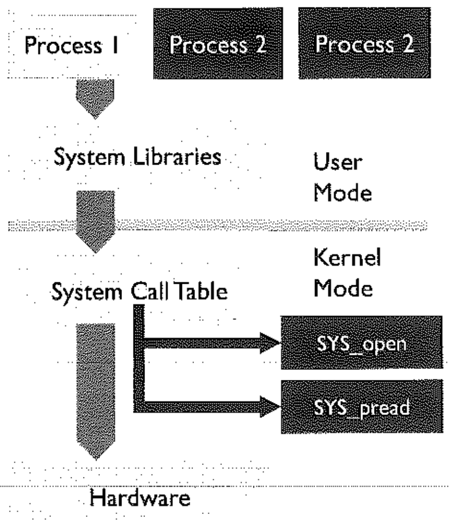
You can get it here: https://github.com/d3sre/rootkit/blob/master/rooty.c



Altering the Kernel for Extreme Hiding

Rootkit Examples

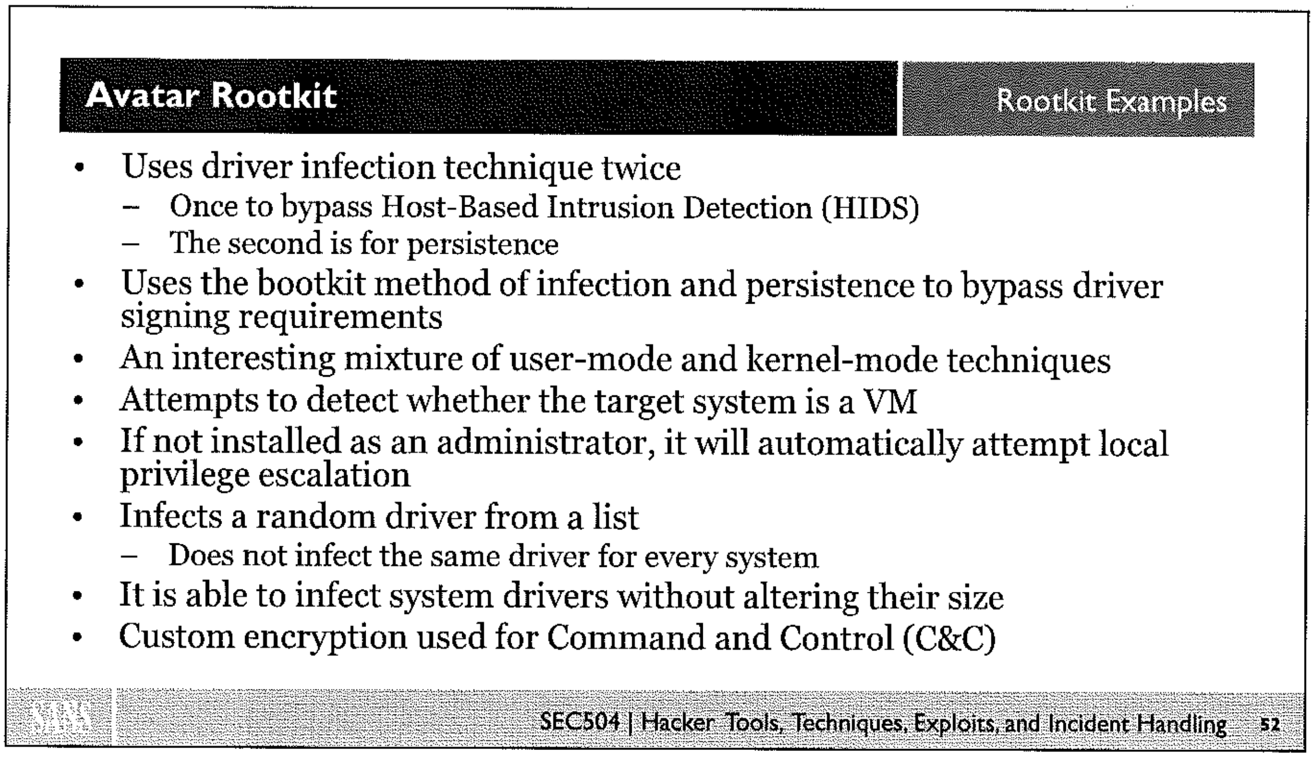
- With many careful changes to the system call table, the attacker can hide:
  - Processes
  - Files and directories
  - Port usage
- The attacker can implement a "cone of silence"
  - Inside the cone, all hidden items are visible
  - Outside the cone, all hidden items are hidden



SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 51

Rooty manipulates the system table in numerous ways, redirecting system calls associated with executing programs and opening files to create a "cone of silence" hiding the attacker. This cone of silence essentially carves user mode into two worlds: A visible environment and a cloaked environment. From inside the cone of silence, where the attacker lives, everything on the system is viewable, hidden items and visible items alike. Outside the cone of silence, where users and administrators dwell, all hidden items are completely invisible.

The Rooty rootkit keeps the two worlds separate by carefully manipulating the system call table to hide things from visible processes while allowing invisible processes to see. Now, that's a highly effective paradigm for interacting with a kernel-mode rootkit.



Avatar Rootkit

Rootkit Examples

- Uses driver infection technique twice
  - Once to bypass Host-Based Intrusion Detection (HIDS)
  - The second is for persistence
- Uses the bootkit method of infection and persistence to bypass driver signing requirements
- An interesting mixture of user-mode and kernel-mode techniques
- Attempts to detect whether the target system is a VM
- If not installed as an administrator, it will automatically attempt local privilege escalation
- Infects a random driver from a list
  - Does not infect the same driver for every system
- It is able to infect system drivers without altering their size
- Custom encryption used for Command and Control (C&C)

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 52

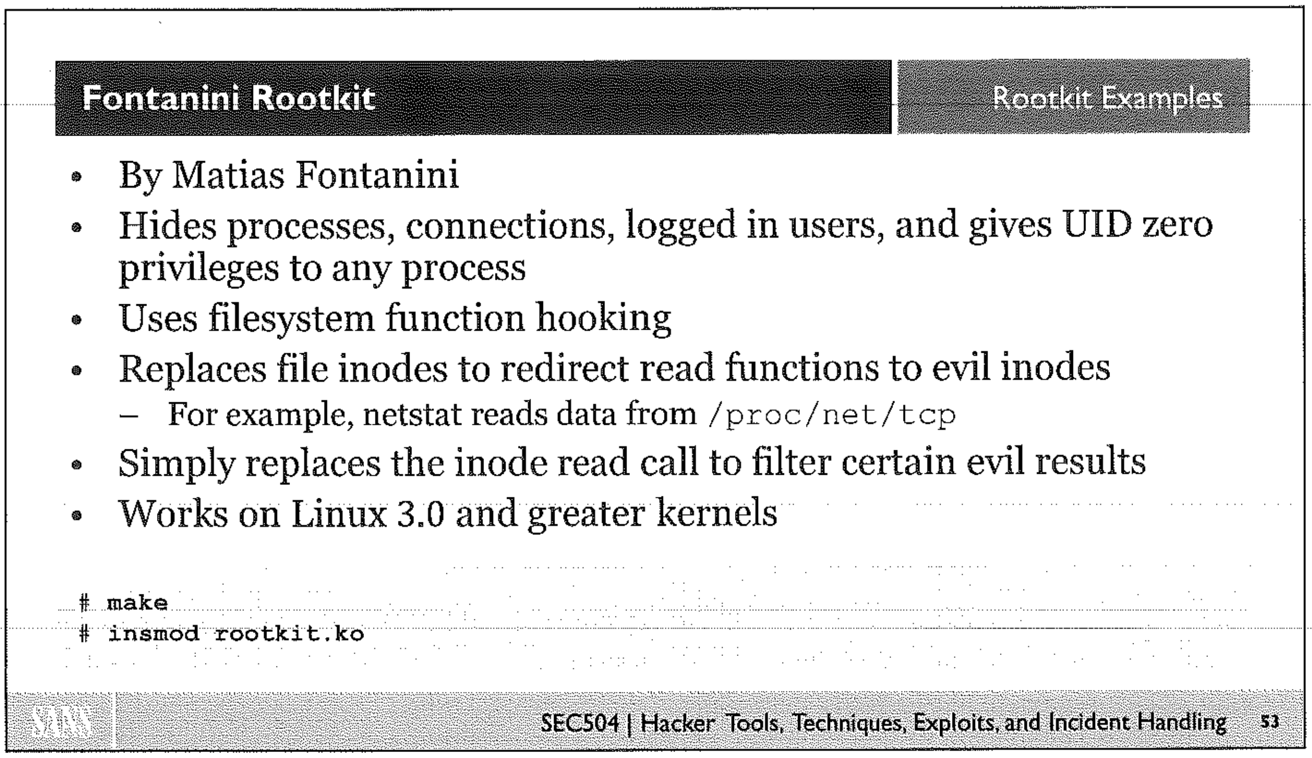
The Avatar rootkit takes a number of very interesting techniques for infection and persistence. First, it uses two different driver infection techniques when it is enabled. It drops one driver to bypass Host-Based Intrusion Detection. Next, it drops a second driver for persistence. It also uses bootkit techniques to bypass Windows mandatory driver signing. This technique is similar to the type of technique used in tools like Kon-boot. The vector is to bypass controls before they are completely enabled as part of the Windows boot process.

If the rootkit is not installed with administrator privileges, it will attempt a local privilege escalation exploit. The privilege escalation exploit is straight out of Metasploit, so it can be easily modified and updated as more exploits are released.

However, what driver does it infect? Turns out, it is not a consistent driver. Rather, it picks which driver to infect at random from a list of available drivers. Oh! And the driver size stays the same!

Finally, as part of Command and Control, it uses custom RSA encryption, so simple network-level detection is tough.

You can find it here:
http://www.welivesecurity.com/2013/05/01/mysterious-avatar-rootkit-with-api-sdk-and-yahoo-groups-for-cc-communication/


Fontanini Rootkit

- By Matias Fontanini
- Hides processes, connections, logged in users, and gives UID zero privileges to any process
- Uses filesystem function hooking
- Replaces file inodes to redirect read functions to evil inodes
  - For example, netstat reads data from `/proc/net/tcp`
- Simply replaces the inode read call to filter certain evil results
- Works on Linux 3.0 and greater kernels

```
# make
# insmod rootkit.ko
```

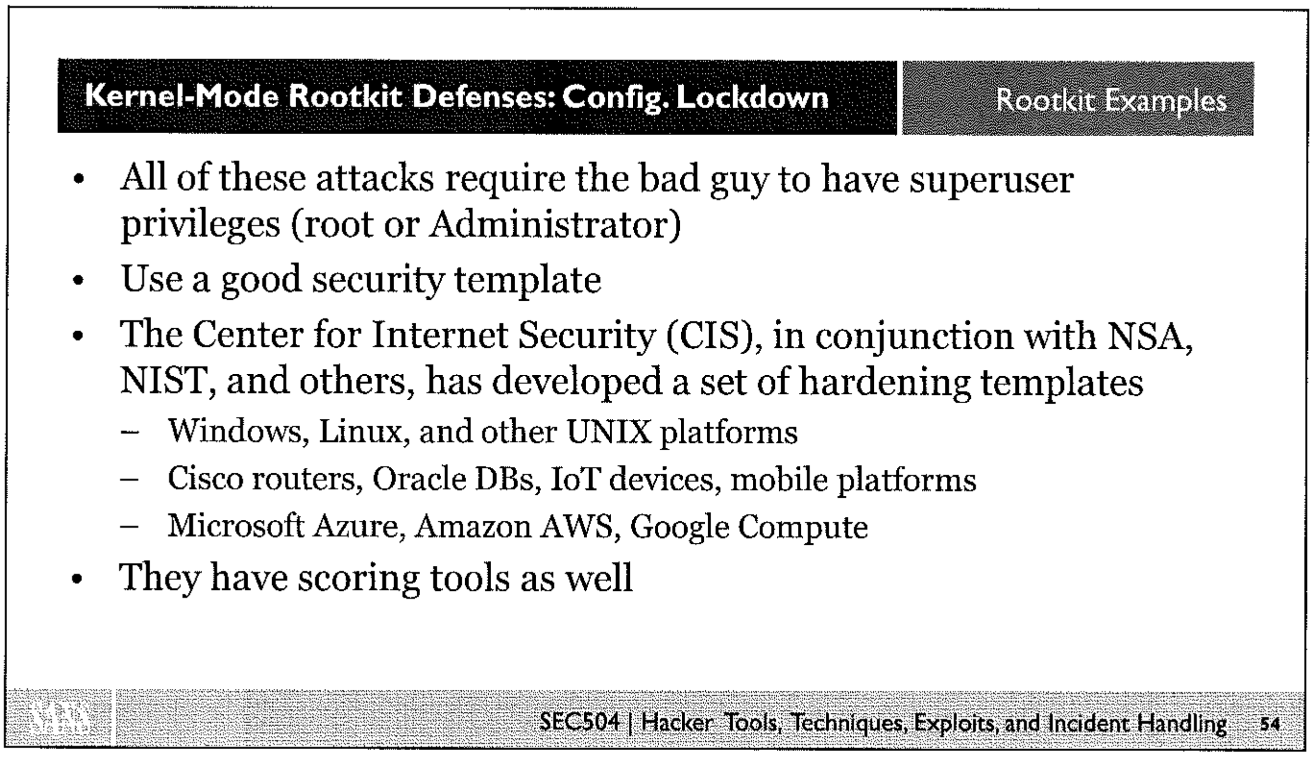
A cool proof of concept rootkit is the Fontanini rootkit by Matias Fontanini.

This rootkit replaces the *read* function in filesystem function hooking. For example, when a program like netstat runs, it pulls data from other device files like /proc/net/tcp. But, with the Fontanini rootkit in use, it will filter out the results relating to the rootkit user, processes, and network connections. It does this by updating the inode or file location pointer information for the file or program.

It compiles and works on Linux kernels newer than 3.0.

Installation is easy. Simply run the following two commands to install it.

```
# make
# insmod rootkit.ko
```



Kernel-Mode Rootkit Defenses: Config. Lockdown

Rootkit Examples

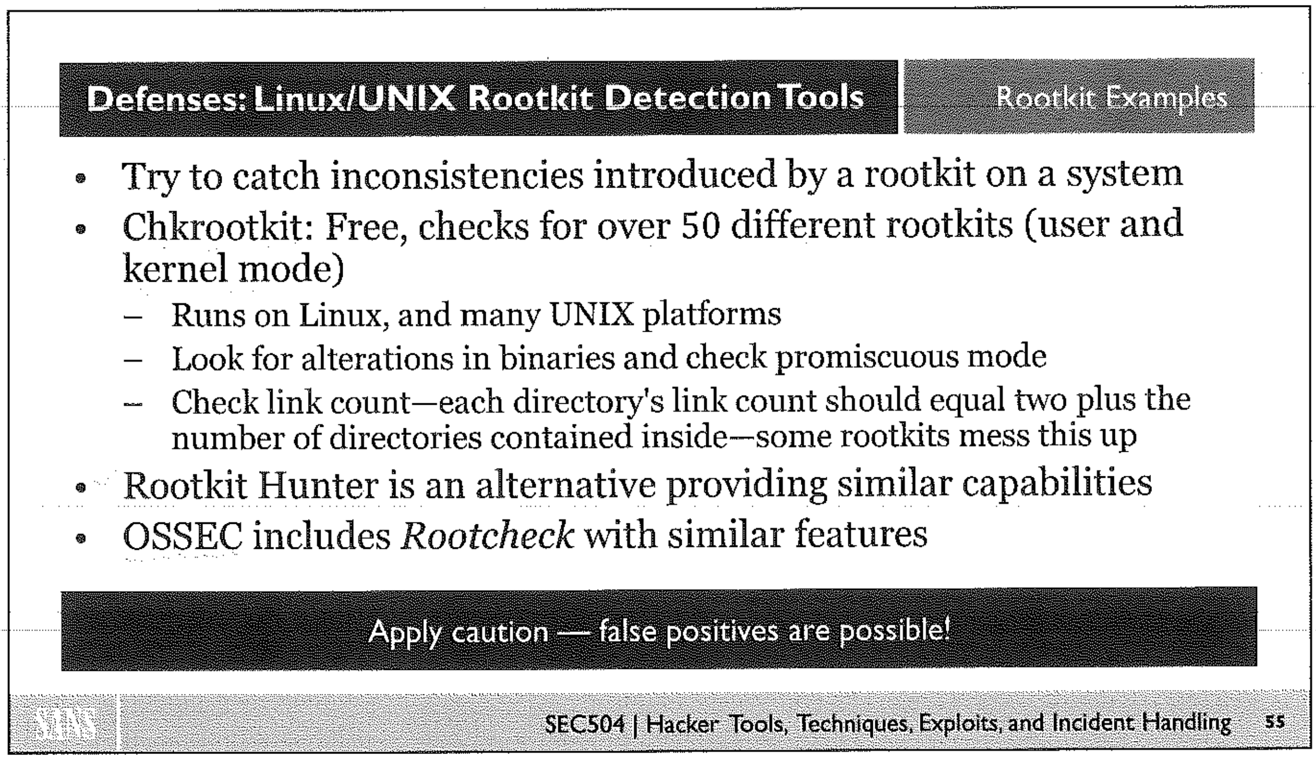
- All of these attacks require the bad guy to have superuser privileges (root or Administrator)
- Use a good security template
- The Center for Internet Security (CIS), in conjunction with NSA, NIST, and others, has developed a set of hardening templates
  - Windows, Linux, and other UNIX platforms
  - Cisco routers, Oracle DBs, IoT devices, mobile platforms
  - Microsoft Azure, Amazon AWS, Google Compute
- They have scoring tools as well

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 54

Hardening your systems to prevent superuser compromise is a critical first step. Microsoft ships Windows with a variety of security template files for workstations, servers, and domain controllers. However, these built-in security templates tend to be either way too weak so that any attacker can slice through them, or so strong that they render the system unusable in a real-world environment. What the world needs is a reasonable security template that isn't too weak, or too strong, but is just right for most environments.

The Center for Internet Security (CIS), the National Security Agency, and the SANS Institute, together with a variety of other organizations, embarked on creating just such a template. They spent several months devising various standards that would meet the most pressing needs of all of these organizations. Finally, they achieved consensus and released several system hardening templates, available at http://www.cisecurity.org. These templates apply to Windows, Linux, Solaris, HP-UX, Cisco routers, and Oracle Databases, as well as numerous other types of platforms (Microsoft Exchange Server, Microsoft SQL Server, Apache, BIND, Novell eDirectory, etc.). They serve as an excellent starting and reference point for your security configuration. You can tweak them to make them stronger or loosen their restrictions for your environment.

The Center for Internet Security has also released free scoring tools, so you can check to see how well your security settings match a given template, such as the Win2K Pro Gold Template. You run scoring tools on a local system to compare your security stance to a baseline template, giving you a summary score between 0 and 10. The higher your score, the more closely you match the template used for comparison.



Defenses: Linux/UNIX Rootkit Detection Tools

Rootkit Examples

- Try to catch inconsistencies introduced by a rootkit on a system
- Chkrootkit: Free, checks for over 50 different rootkits (user and kernel mode)
  - Runs on Linux, and many UNIX platforms
  - Look for alterations in binaries and check promiscuous mode
  - Check link count—each directory's link count should equal two plus the number of directories contained inside—some rootkits mess this up
- Rootkit Hunter is an alternative providing similar capabilities
- OSSEC includes *Rootcheck* with similar features

Apply caution — false positives are possible!

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 55

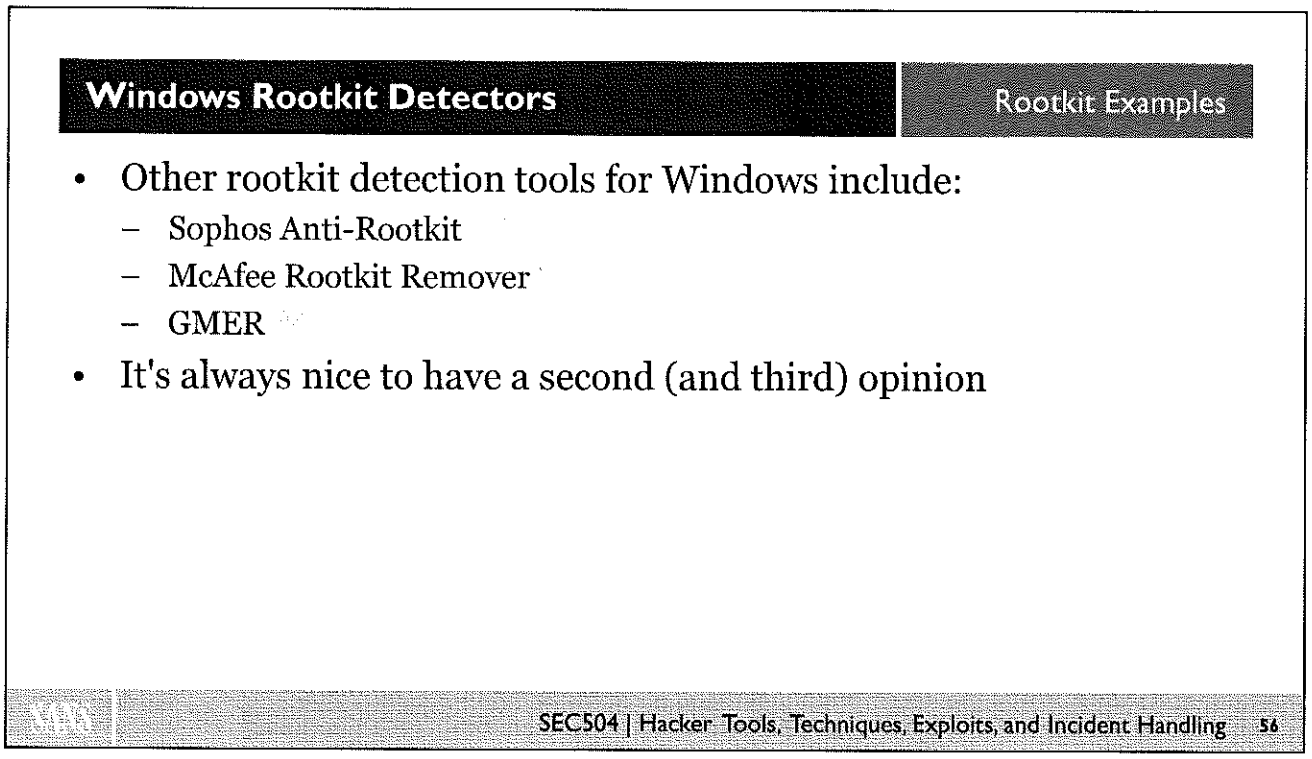
By looking for various system anomalies introduced by kernel-mode rootkit techniques, the free Chkrootkit tool (www.chkrootkit.org) can detect Adore, SucKIT, and several other kernel-mode rootkit techniques. For you fans of *The Matrix* movie, Chkrootkit is actually looking for glitches in the Matrix. As you might recall, in the movie, glitches in the Matrix occur when the bad guys start changing things, creating a déjà vu in the movie. Similarly, with a kernel-mode rootkit, an inconsistency in the system's appearance could be an indication that something foul has been installed. The scripts included in Chkrootkit perform tests that can be used to catch the kernel in a lie about the existence of certain files and directories, network interface promiscuous mode, and other issues that kernel-mode rootkit techniques generally fib about.

One of the ways that Chkrootkit finds kernel-mode rootkit techniques is by looking for inconsistencies in the directory structure when a file or directory is hidden. Each directory in the filesystem has a link count, which indicates the number of other directories that a given directory is connected to in the filesystem structure. For each directory, this link count should be two more than the number of subdirectories in the directory. That way, the directory would have one link for each subdirectory, plus one for the parent directory (..) and one for itself (.). Many kernel-mode rootkit techniques hide files and directories without manipulating the link count of the parent directory. Chkrootkit combs through the entire directory structure, counting the number of subdirectories that it can see inside each directory and comparing it to the link count. If it finds a discrepancy, Chkrootkit prints a message indicating that there might very well be directories that are hidden by a kernel-mode rootkit.

Rootkit Hunter is a similar tool that is also immensely useful in investigations, available at http://rkhunter.sourceforge.net.

OSSEC, a general-purpose system-monitoring and analysis tool, includes a feature called "Rootcheck" with similar rootkit detection capabilities. OSSEC is available at www.ossec.net.

Watch out for false positives from these tools. They are very possible.



Windows Rootkit Detectors

Rootkit Examples

- Other rootkit detection tools for Windows include:
  - Sophos Anti-Rootkit
  - McAfee Rootkit Remover
  - GMER
- It's always nice to have a second (and third) opinion

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 56

Other Windows rootkit detectors include:

- Sophos Anti-Rootkit – https://www.sophos.com/en-us/products/free-tools/sophos-anti-rootkit.aspx
- McAfee Rootkit Remover – https://www.mcafee.com/enterprise/en-us/downloads/free-tools/rootkitremover.html
- GMER – http://www.gmer.net/

These tools all ask the system a series of questions, looking for discrepancies in the answers that could be a sign of a rootkit. Several of the tools have been released by antivirus vendors.

An incident handler should consider carrying several of these tools on a USB to get multiple opinions of whether a rootkit might be present on a machine under investigation.

Defenses: File Integrity Checking Tools

- Look for changes to critical system files
- File integrity checking tools help
  - Although a well-designed kernel-mode rootkit can trick the file integrity checker using execution redirection
  - Still, if the attacker makes any unmasked changes, you'll spot him
- Tripwire is a classic
  - Also looks for registry modifications
- OSSEC is also very good
  - Runs on Linux, UNIX, macOS, and Windows

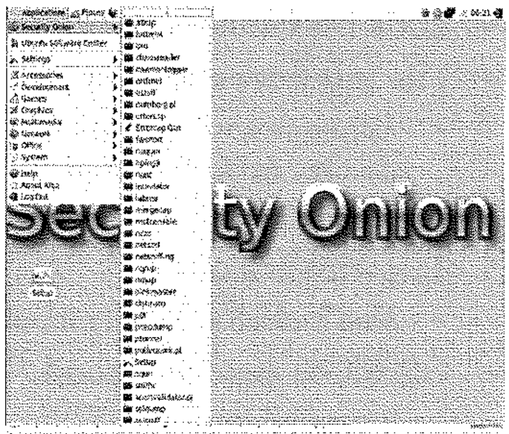
Although they can be tricked by very thorough kernel-mode rootkit techniques, you should still use file integrity checking tools, such as Tripwire (commercial at https://www.tripwire.com), OSSEC (free at www.ossec.net), AIDE (a free and open-source alternative to Tripwire, available at https://aide.github.io/), and the related programs. As we've discussed, a thoroughly bad guy will configure the manipulated kernel with execution redirection and other alterations that lie to the file integrity checker about all file changes on the system. If the attackers very carefully cover all of their tracks, they can fool a file integrity checker.

However, a less careful attacker might forget to configure the kernel-mode rootkit to hide alterations to one or two sensitive system files. Even a single mistake in the file-hiding configuration of the kernel-mode rootkit by the bad guys could expose them to detection by your file integrity checker. Therefore, file integrity checking tools remain very valuable, even though a kernel-mode rootkit can foil them if the attacker is very careful. I'd rather not depend solely on the attackers' mistakes to discover their treachery, but you better believe I'll be sure to take thorough advantage of their errors. Deploying file integrity checking tools on all of my sensitive systems lets me prepare for such circumstances.

Network Intelligence/Forensics

Rootkit Examples

- A lot of malware tries hard to hide on the end system (morphing and kernel hacking)
- Still, its use and propagation have definite patterns that can be observed on the network
  - Strange communication pairs (scans, client → client, server → server, server → client?)
  - Real Intelligence/Threat Analytics (RITA)
  - Network-level intelligence and forensics can help detect such behavior early
- Nifty autodetection and throttling via network-based IPS (NetWitness, FireEye, Sourcefire, TippingPoint, ForeScout, etc.)

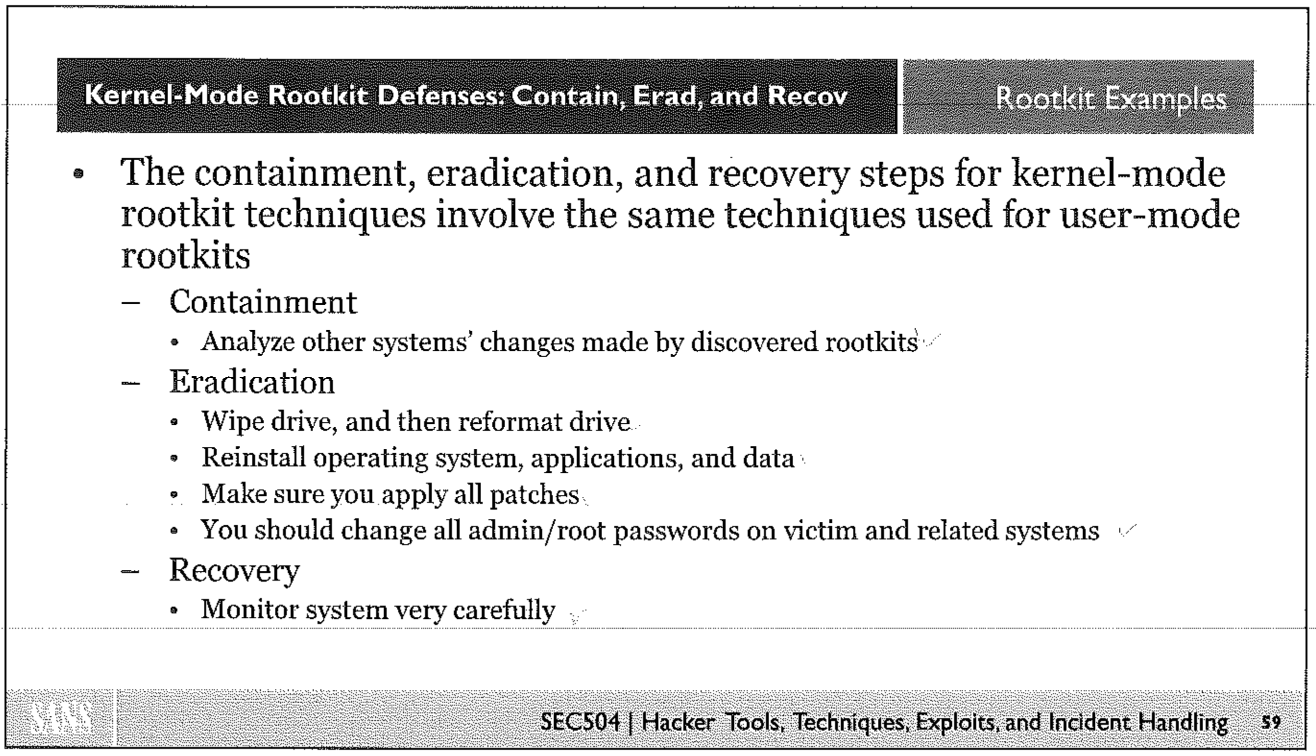


Get the Security Onion at https://securityonion.net/

Malware is getting better and better at hiding itself on an end system, using the kernel-mode rootkit techniques and other attacks we've discussed. So, with the bad stuff pretty well *stealthified* on an end system, what can we do to detect it? Increasingly, people are turning to network-based detection and defensive tools in the form of network forensics applications that pull data from lots of systems and correlate events to determine what's actually going on.

By looking for unusual network behavior caused by malware infections, security personnel have a better shot at detecting the pathogenic code. For example, under normal circumstances, clients communicate with servers, but typically not with each other. If they do communicate with each other, it's typically a small number of packets. Similarly, server-to-server communication is typically relatively small compared to client-server communication. By looking for significant and sudden deviations from these patterns, a tool might be able to identify malicious code spreading on a network. Worm propagation patterns, in particular, are easy to spot on a network, as an eruption of unusual communication pairs fires up. The Security Onion is quite possibly the single best open-source network forensic distribution.

Some tools can even autodetect and throttle such behavior when they spot these worm patterns. These Network-Based Intrusion Prevention Systems offer some interesting capabilities for detection and defense.



Kernel-Mode Rootkit Defenses: Contain, Erad, and Recov

Rootkit Examples

- The containment, eradication, and recovery steps for kernel-mode rootkit techniques involve the same techniques used for user-mode rootkits
  - Containment
    - Analyze other systems' changes made by discovered rootkits
  - Eradication
    - Wipe drive, and then reformat drive
    - Reinstall operating system, applications, and data
    - Make sure you apply all patches
    - You should change all admin/root passwords on victim and related systems
  - Recovery
    - Monitor system very carefully

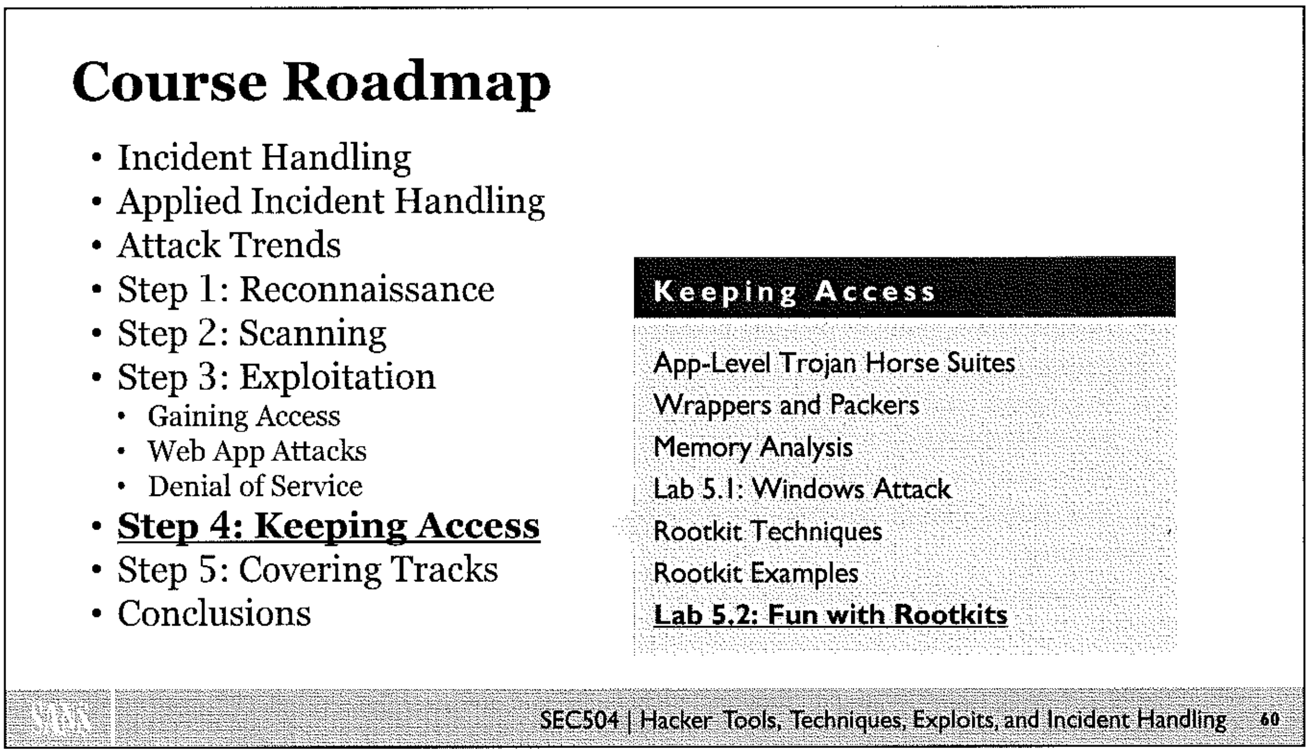
SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 59

The containment, eradication, and recovery steps associated with a kernel-mode rootkit apply the same techniques we discussed with user-mode rootkits.

When a kernel-mode rootkit is detected, you should completely wipe and reformat the drive, reinstall all operating system components and applications, and then thoroughly patch the machine.

Restore data from a recent backup.

You might need to do a sanity check on the data to make sure it was not corrupted as well.


Course Roadmap

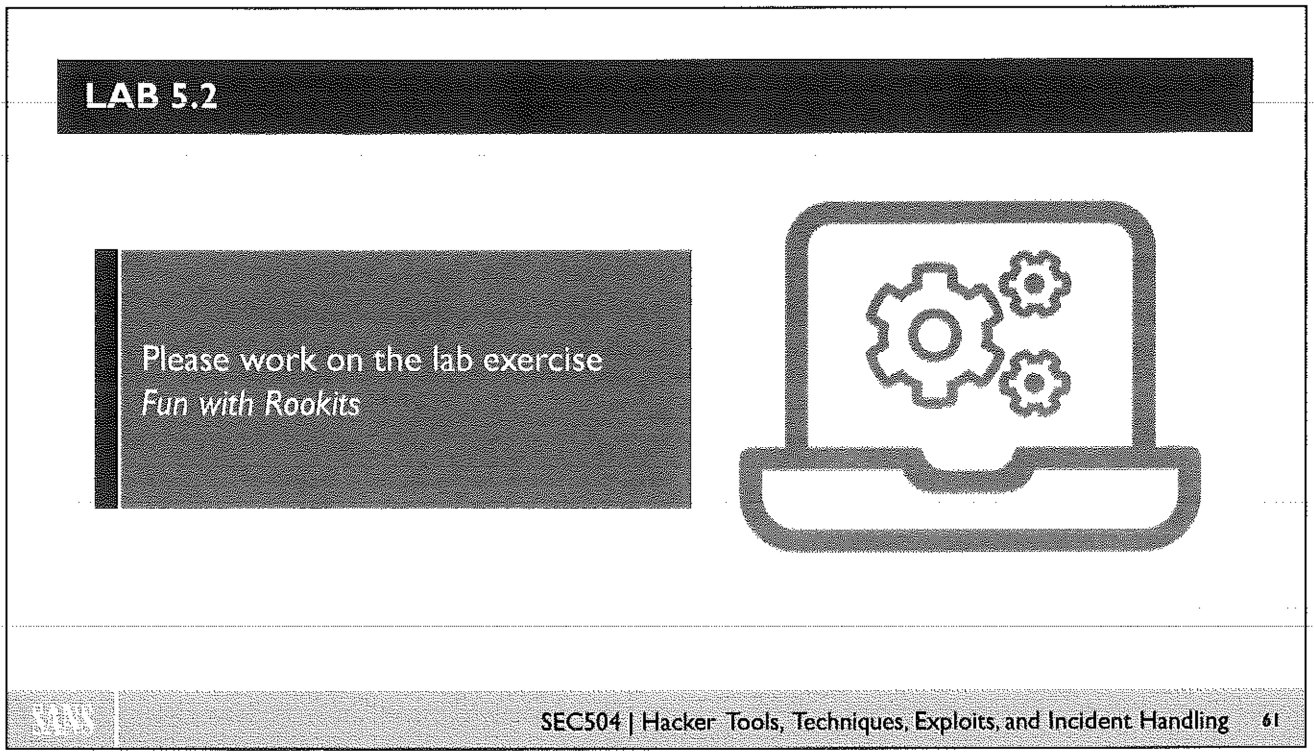
- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- **Step 4: Keeping Access**
- Step 5: Covering Tracks
- Conclusions

Keeping Access

App-Level Trojan Horse Suites
Wrappers and Packers
Memory Analysis
Lab 5.1: Windows Attack
Rootkit Techniques
Rootkit Examples
Lab 5.2: Fun with Rootkits

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 60

Now let's have some fun with rootkits.

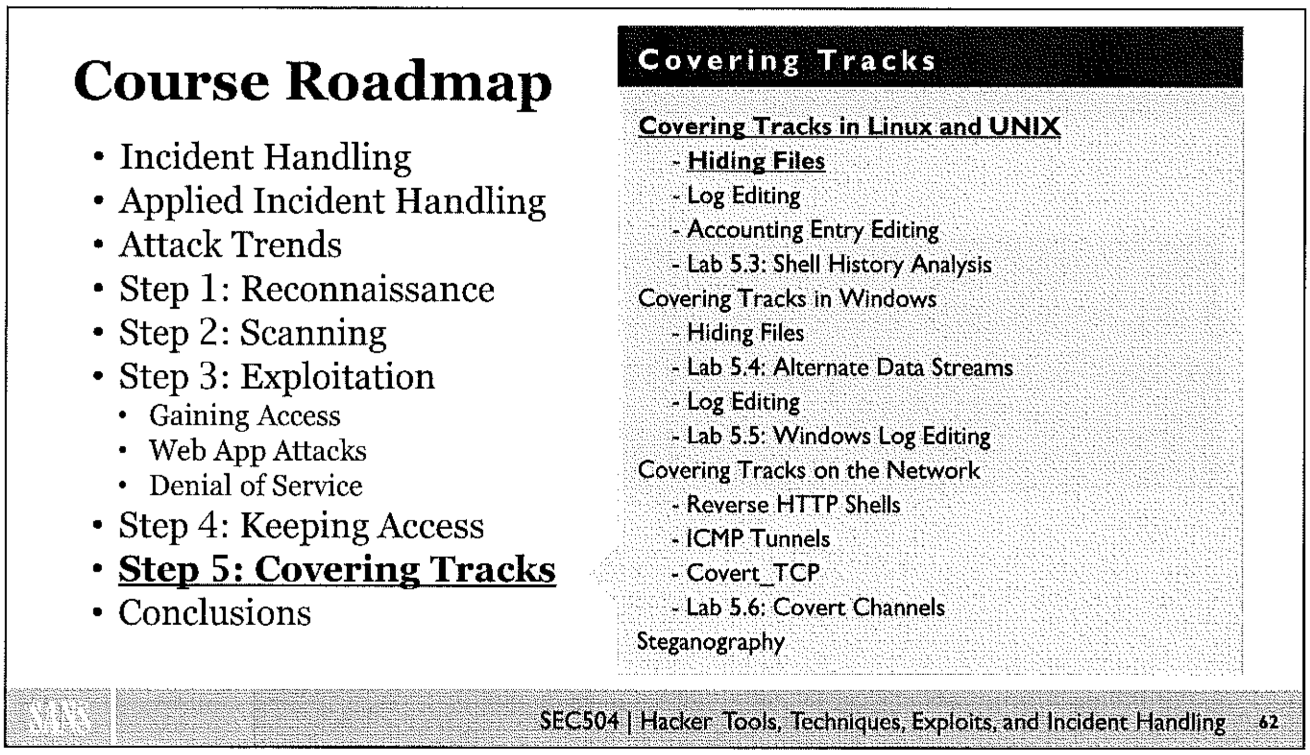


LAB 5.2

Please work on the lab exercise
Fun with Rookits

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 61

This page intentionally left blank.



Course Roadmap

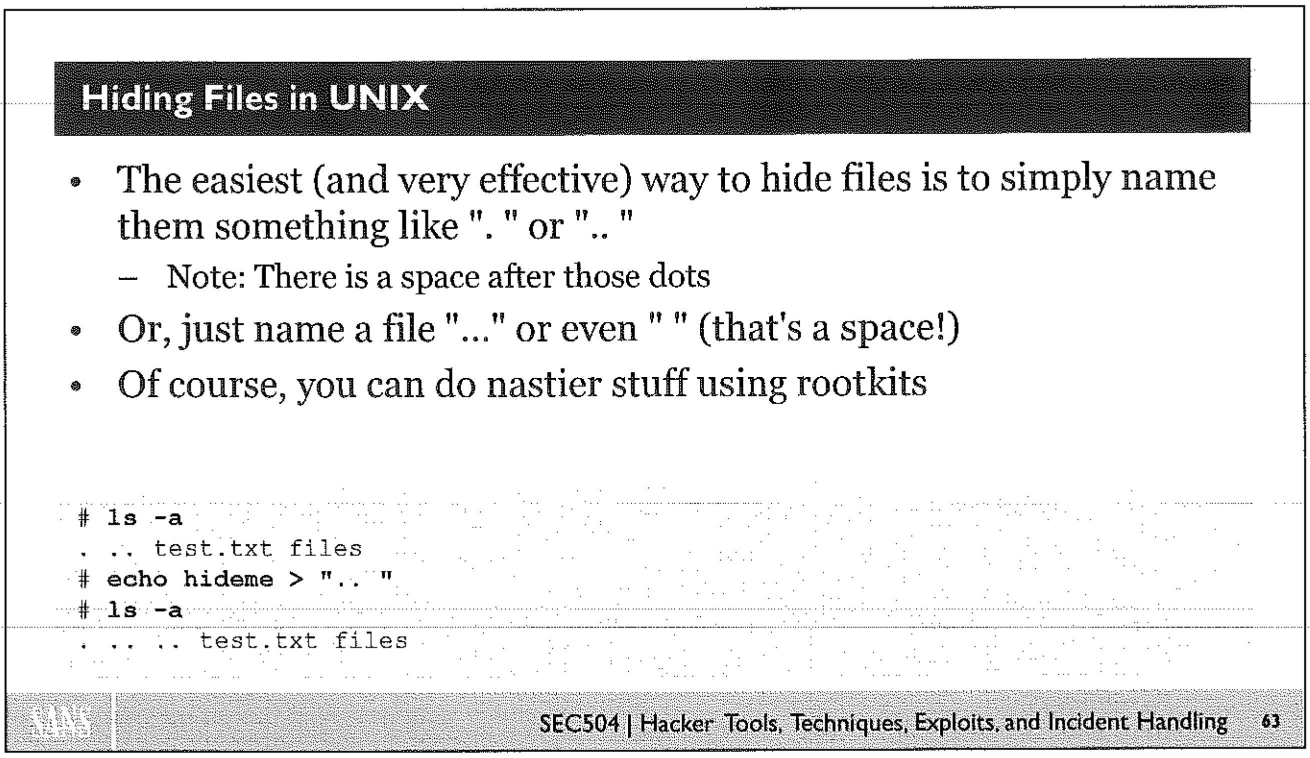
- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 62

This slide shows the details of the techniques we discuss for covering the tracks.

We start by talking about how attackers cover their tracks in UNIX. Then, we talk about covering tracks in Windows. We then address how attackers cover their tracks on the network using various covert listeners on the victim machine. We finish this section by discussing steganography, a technique used to hide data.



Hiding Files in UNIX

- The easiest (and very effective) way to hide files is to simply name them something like ". " or ".. "
  - Note: There is a space after those dots
- Or, just name a file "..." or even " " (that's a space!)
- Of course, you can do nastier stuff using rootkits

```
# ls -a
. .. test.txt files
# echo hideme > ".. "
# ls -a
. .. .. test.txt files
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 63

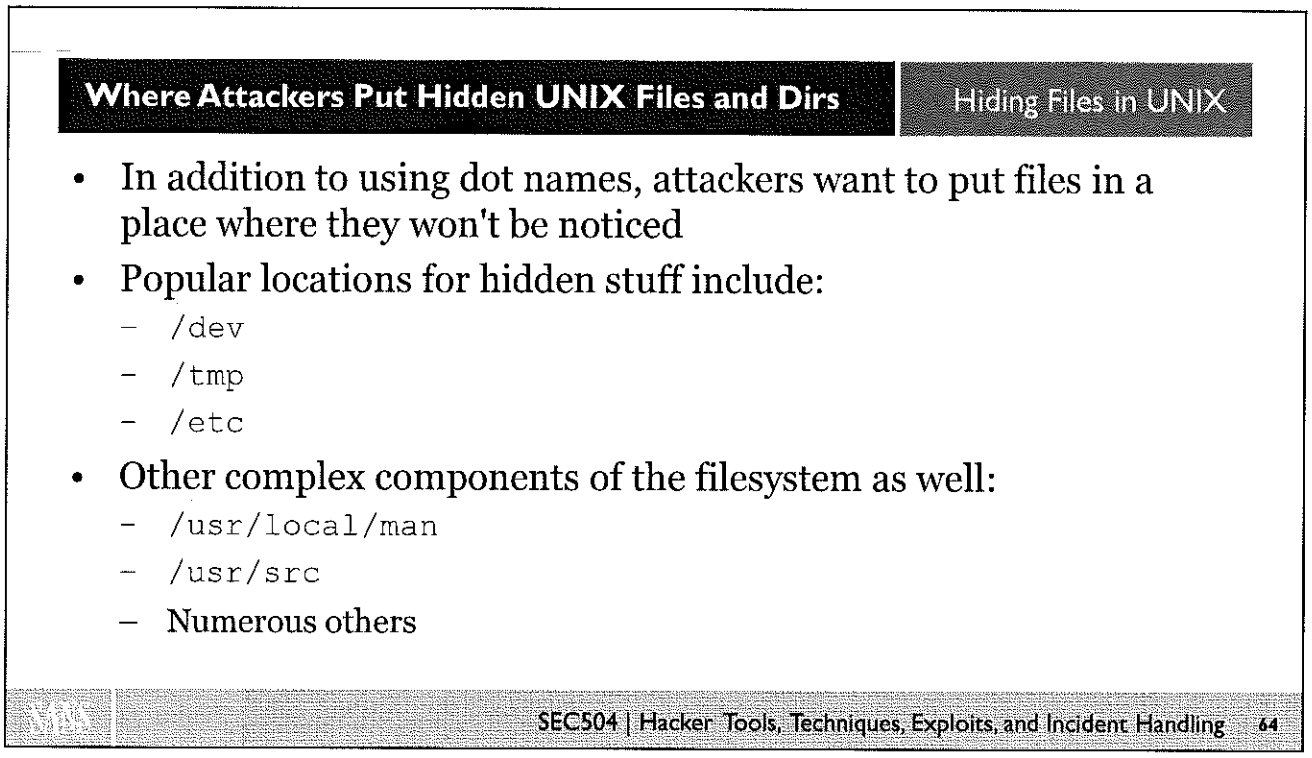
One of the most common methods for hiding files on a UNIX system is to simply give the file a name that starts with a dot. Sure, an attacker can use one of the rootkits or kernel-mode rootkit techniques to hide files on a system. However, if the attacker does not yet have root privileges, he or she might still need to hide files without the use of a rootkit.

On a UNIX system, each directory contains at least two other directories. One of these directories is called ".". This name refers to the current directory itself. That's why if you type, "cd .", you change to the current directory.

The other directory that is found in every single directory on UNIX systems is called "..". This name refers to the parent directory in the filesystem.

Attackers will disguise files and directories by naming them dot-space, dot-dot-space, dot-dot-dot, or even just space.

This isn't terribly sophisticated, but it works well in a pinch! Just name a file dot-dot-space, and many administrators won't even notice the file. Essentially, the file is camouflaged so that an unobservant user or administrator will not notice it as the redundant dots flash by on the screen.



Where Attackers Put Hidden UNIX Files and Dirs

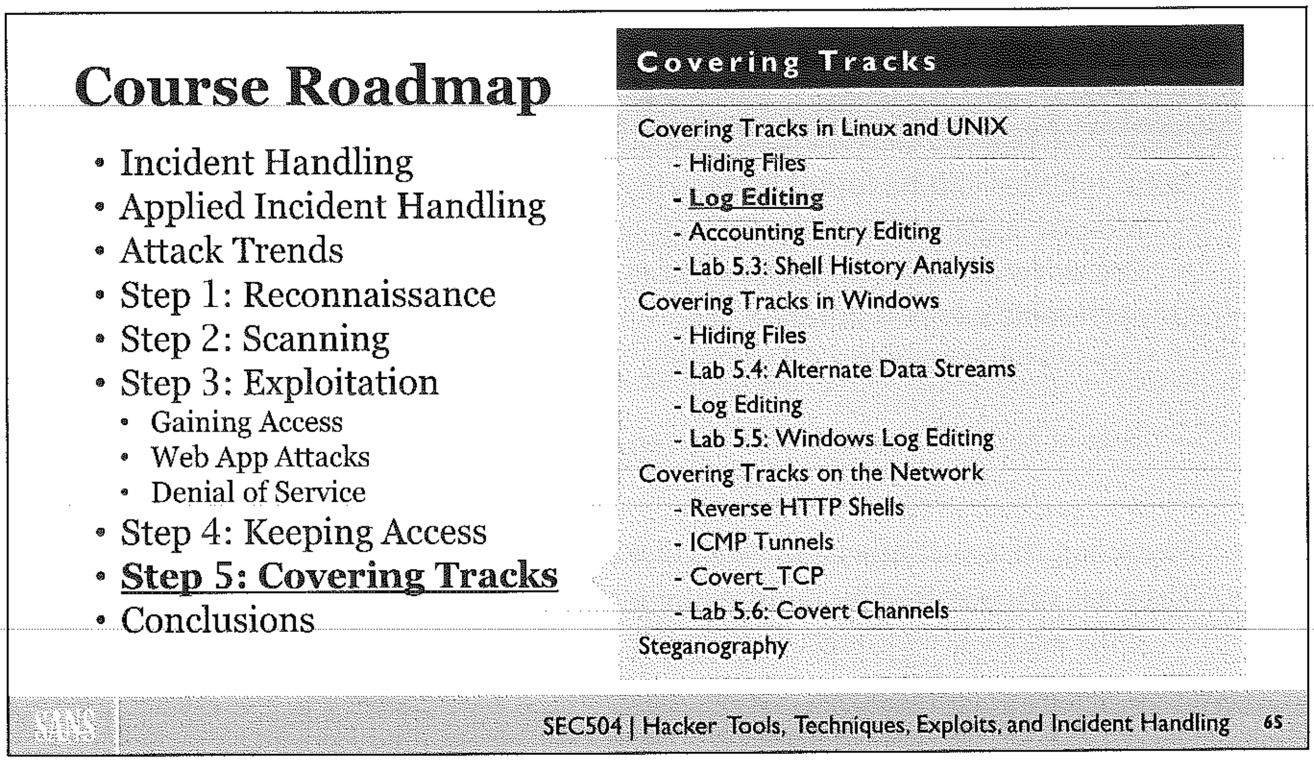
Hiding Files in UNIX

- In addition to using dot names, attackers want to put files in a place where they won't be noticed
- Popular locations for hidden stuff include:
  - `/dev`
  - `/tmp`
  - `/etc`
- Other complex components of the filesystem as well:
  - `/usr/local/man`
  - `/usr/src`
  - Numerous others

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 64

Attackers frequently hide data inside /dev, /tmp, and /etc. The /dev directory contains information about devices on the system, such as chunks of your hard drive and references to terminals. It's full of thousands of items and is, therefore, a good place to sneak additional files into. The /tmp directory often contains strangely named files created by various applications to temporarily store data, and it therefore makes a good hiding place. On many UNIX variants, the /tmp file is emptied during a reboot. Therefore, an attacker would have to restore data that is hidden in /tmp. The /etc directory, in my opinion, is a bad place to store hidden files because it holds the configuration of the machine. We carefully monitor /etc, as we hope your system administrators do as well. Still, attackers make this bad choice of storing stuff in /etc/, and I'm happy to have them make such a detectable mistake!

Another set of locations used to store the attacker's wares involves complex components of the filesystem. Users often don't understand these areas of the filesystem, and administrators often don't scour these areas either. Two increasingly popular places to hide things are /usr/local/man (the man pages), and /usr/src (the default location of source code in Linux). An attacker could likewise choose numerous other places in the filesystem.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- **Log Editing**
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

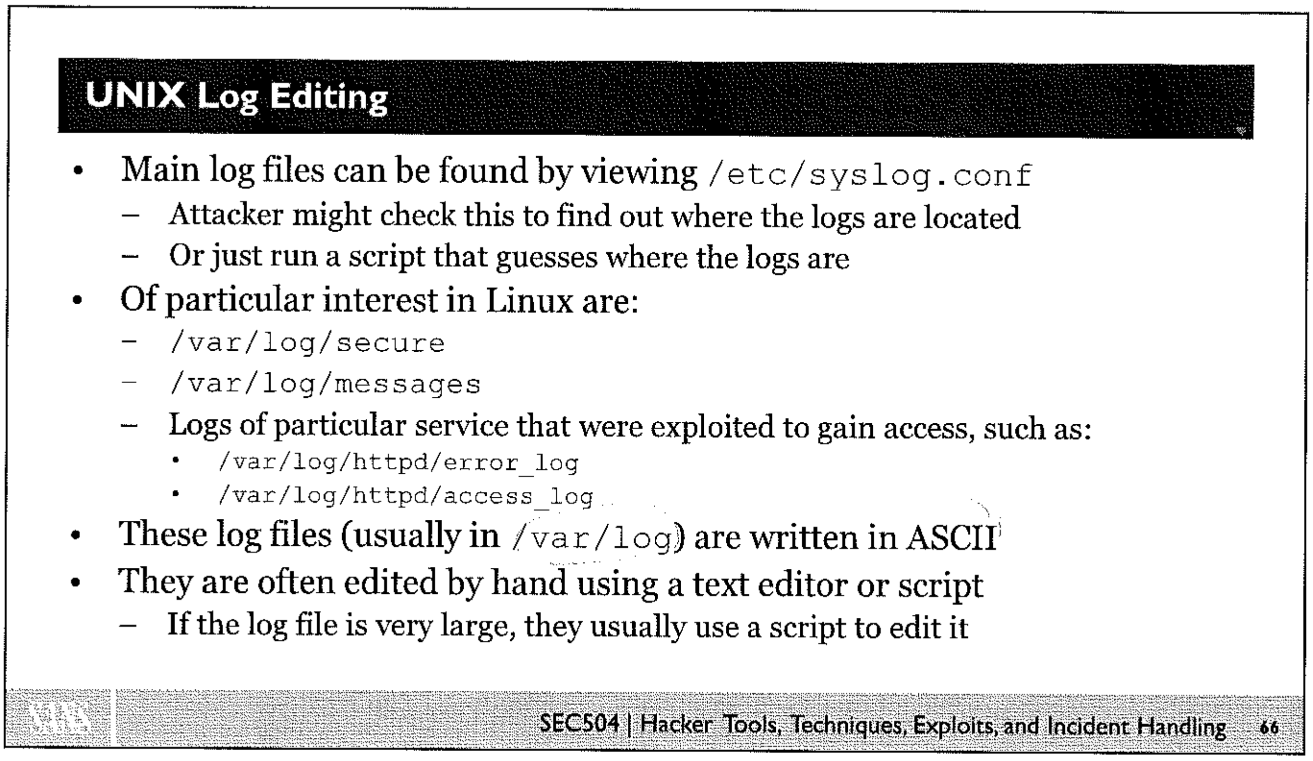
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 65

Let's go back to our roadmap and talk about how attackers can edit the log files on a UNIX system.



UNIX Log Editing

- Main log files can be found by viewing `/etc/syslog.conf`
  - Attacker might check this to find out where the logs are located
  - Or just run a script that guesses where the logs are
- Of particular interest in Linux are:
  - `/var/log/secure`
  - `/var/log/messages`
  - Logs of particular service that were exploited to gain access, such as:
    - `/var/log/httpd/error_log`
    - `/var/log/httpd/access_log`
- These log files (usually in `/var/log`) are written in ASCII
- They are often edited by hand using a text editor or script
  - If the log file is very large, they usually use a script to edit it

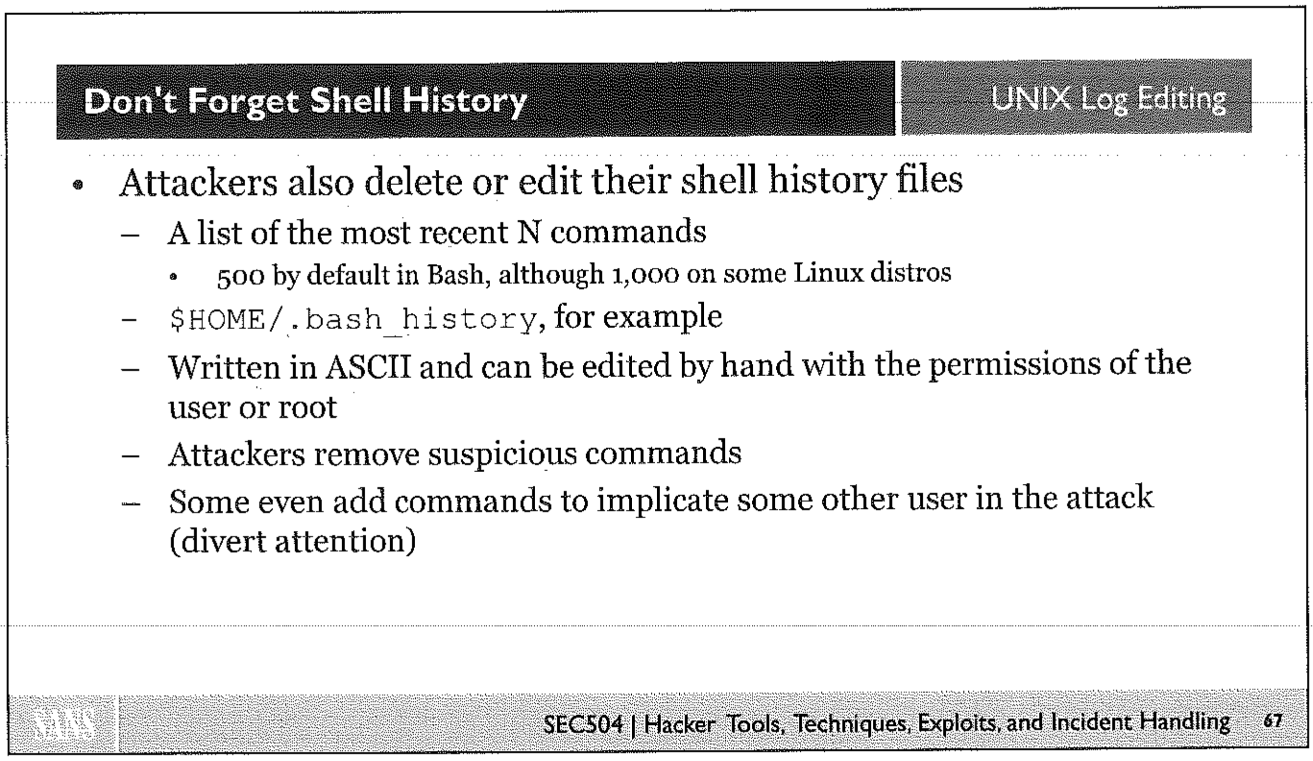
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 66

After an attacker takes over a system, he usually wants to alter the system logs to erase the entries associated with the techniques he used to gain access to the system. Some attackers will just clean the logs out entirely, deleting everything from the log file. Such a technique is quite noticeable by the administrators, however.

Therefore, more sophisticated attackers will delete selected entries from the log files. Only entries associated with the attacker's gaining access, such as incorrect logins or process crashing, will be removed.

On a UNIX system, the syslog process stores the logs for the machine. The configuration for the system logger is found in the /etc/syslog.conf file. When a careful attacker takes over a system, he will look at this file to see where the system is configured to store its logs. These careful attackers will then modify the log files by hand. An attacker that is not very careful, such as a script kiddie, will likely just run some script that guesses a default value where the logs might be stored. Oftentimes these log-cleaning scripts malfunction because they are run on an improper version of UNIX, or the logs are stored in a non-default location.

By default, several flavors of UNIX store their system logs in the /var/log directory. Certain applications, such as a web server (httpd), store their logs in their own directories. These particular directories do vary on different types of UNIX systems. With root privileges, an attacker can edit the log files (usually in /var/log) directly, unless you take special precautions to prevent the alteration, such as encrypting them. Almost all of the logs in /var/log (or its equivalent) are written in straight ASCII, so they can be edited using any editing tool, such as vi or emacs. The attacker will comb through the log file, finding specific entries to delete and remove them.



Don't Forget Shell History

UNIX Log Editing

- Attackers also delete or edit their shell history files
  - A list of the most recent N commands
    - 500 by default in Bash, although 1,000 on some Linux distros
  - `$HOME/.bash_history`, for example
  - Written in ASCII and can be edited by hand with the permissions of the user or root
  - Attackers remove suspicious commands
  - Some even add commands to implicate some other user in the attack (divert attention)

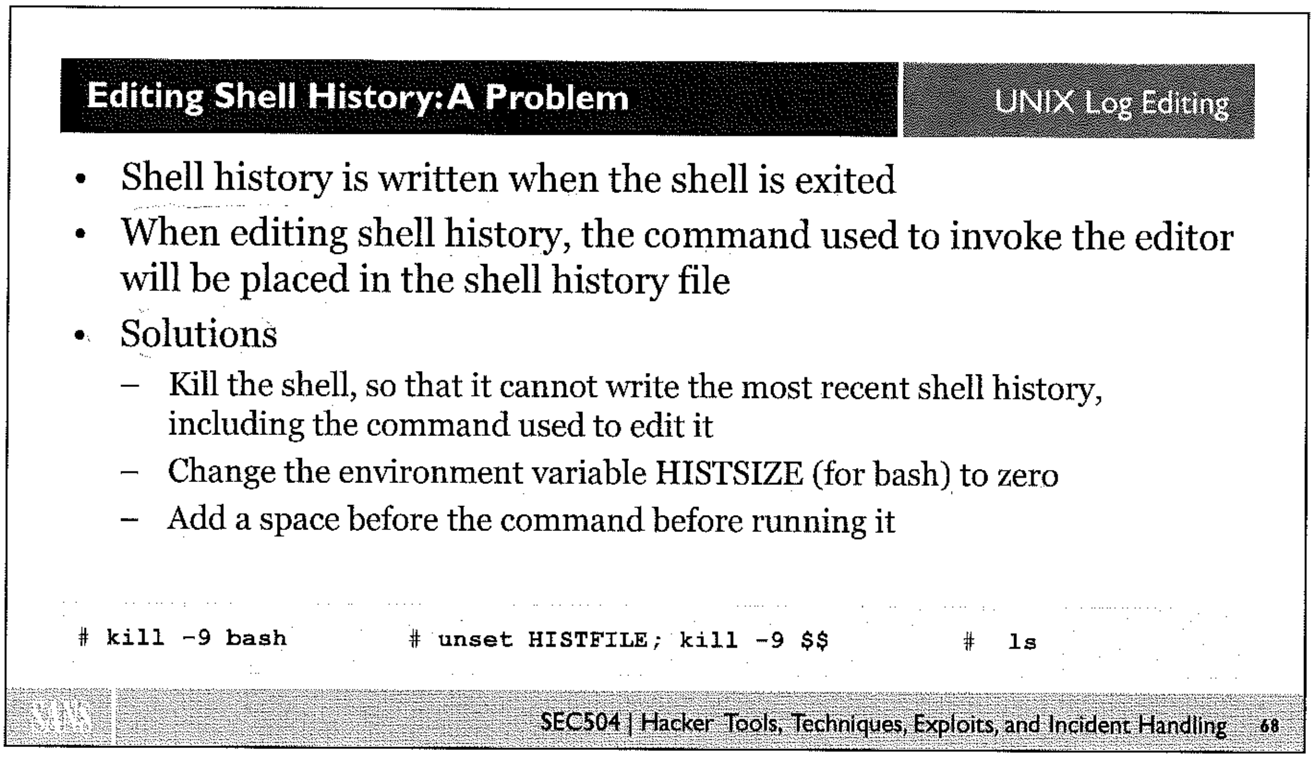
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 67

Whenever you type a command in a UNIX shell, the shell has the option (if it is configured appropriately) of recording each command. By default, the Bash shell included in Linux stores the most recent commands typed in. The default history file size is 500 commands in Bash, although some Linux distros increase this to 1,000. An investigator can, therefore, look in the Bash history file for a user to see what that user has typed recently. Bad guys don't want the investigators to see what happened, so they will often edit the Bash history file, which is written in plain ASCII.

A particularly nasty attacker might plant false commands into another user's history file to divert attention during an investigation.

[illegible]

[illegible]



Editing Shell History: A Problem

UNIX Log Editing

- Shell history is written when the shell is exited
- When editing shell history, the command used to invoke the editor will be placed in the shell history file
- Solutions
  - Kill the shell, so that it cannot write the most recent shell history, including the command used to edit it
  - Change the environment variable HISTSIZE (for bash) to zero
  - Add a space before the command before running it

```
# kill -9 bash          # unset HISTFILE; kill -9 $$          #  ls
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 68

It's important to remember that shell history is written when the shell is exited. Therefore, you won't see your most recent commands in the shell history; they are stored in RAM until the shell is exited. This has significant impact when editing shell history. In particular, the command used to invoke the editor will be placed in the shell history file, so an investigator can see something like "vi .bash_history." That's bad news for the attacker.

To deal with this problem in an unsuccessful way, the attacker could edit the file, exit the shell, start another shell, and edit the history file again to remove it.
It will, however, be added again! What we have here is a chicken and the egg problem.

There are two widely used solutions for the attacker, including:

1. Killing the shell, so that it cannot write the most recent shell history, including the command used to edit it.

   ```
   # kill -9 [pid]
   ```

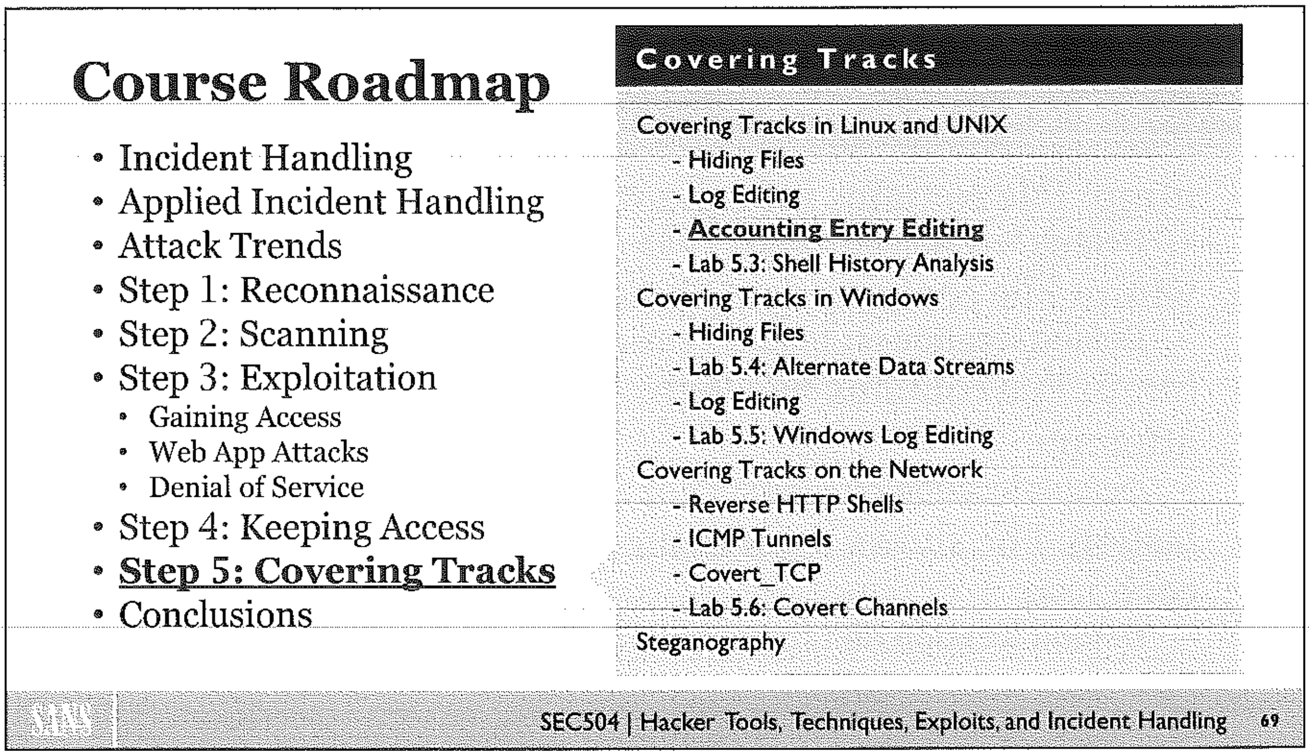
   Alternatively, the attacker could simply kill all bash shells by running the command:

   ```
   # killall -9 bash
   ```

2. Changing the environment variable HISTFILE:

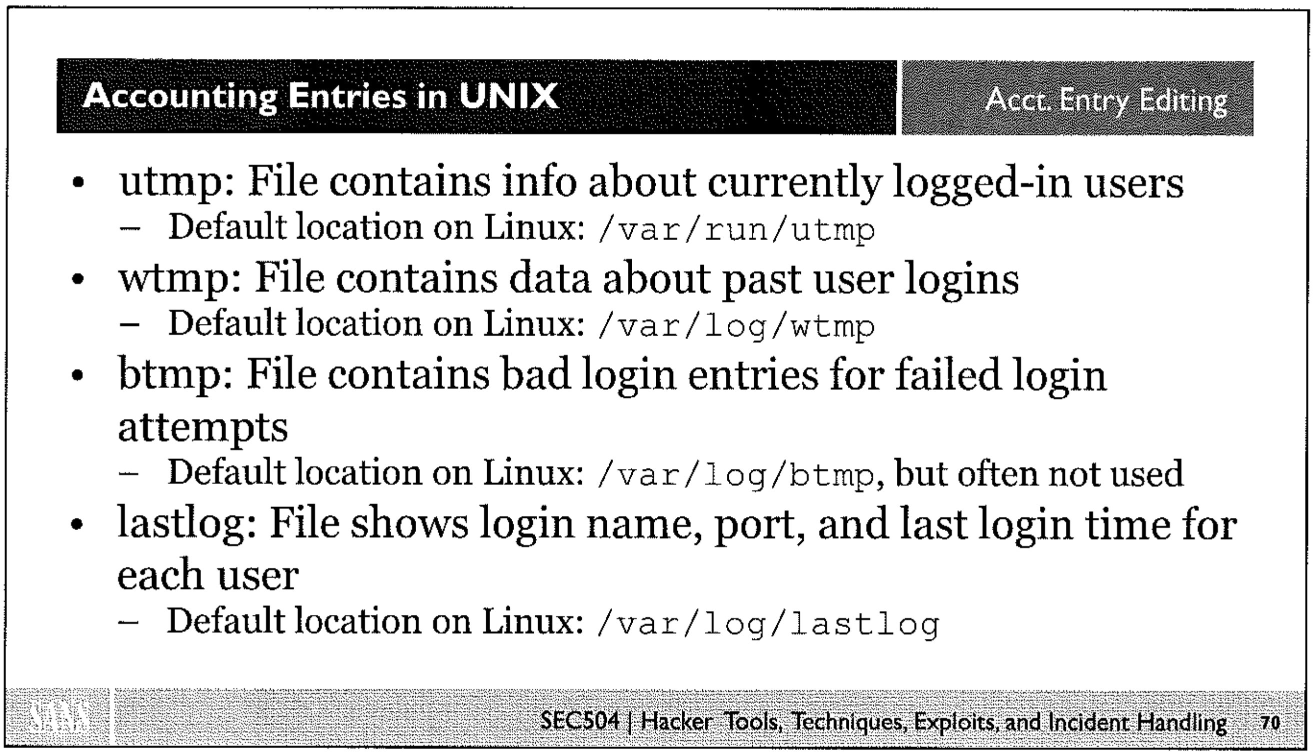
   `# unset HISTFILE` then `kill -9 $$`

3. You can also add a space before the command in bash to not log specific commands.



Course Roadmap

We now discuss how attackers edit accounting entries on UNIX systems.



Accounting Entries in UNIX

Acct. Entry Editing

- utmp: File contains info about currently logged-in users
  - Default location on Linux: `/var/run/utmp`
- wtmp: File contains data about past user logins
  - Default location on Linux: `/var/log/wtmp`
- btmp: File contains bad login entries for failed login attempts
  - Default location on Linux: `/var/log/btmp`, but often not used
- lastlog: File shows login name, port, and last login time for each user
  - Default location on Linux: `/var/log/lastlog`

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 70

UNIX systems have four files better known as the accounting entries.

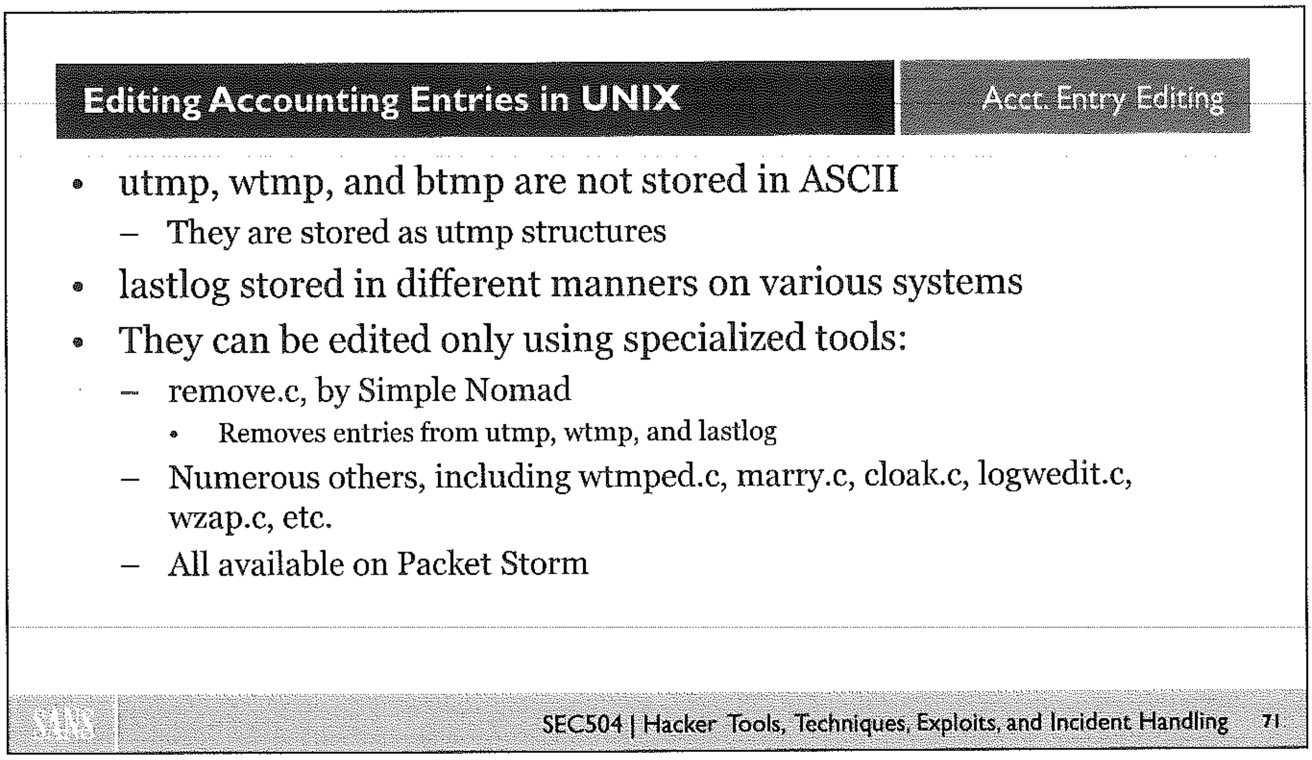
The utmp file stores information about all users currently logged in to the system. This file is consulted by the "who" command to print a list of users with actively logged in sessions on the system.

The wtmp file stores information about all users who have ever logged in to the machine.

The btmp file stores information about bad login attempts (i.e., failures to properly authenticate). This functionality is often configured to be off because most sysadmins don't want to leave a file sitting around with bad user ID attempts in it because they might contain passwords accidentally typed by users at the "login:" prompt. Users often type in their password as a user ID, **and vice versa**.

The lastlog file shows information associated with the most recent login time and date for each user. This file is consulted by the login program when each user logs in to the system to display the last login date and time for the user.

Attackers want to modify these files so system administrators can't tell what they're up to.



Editing Accounting Entries in UNIX

Acct. Entry Editing

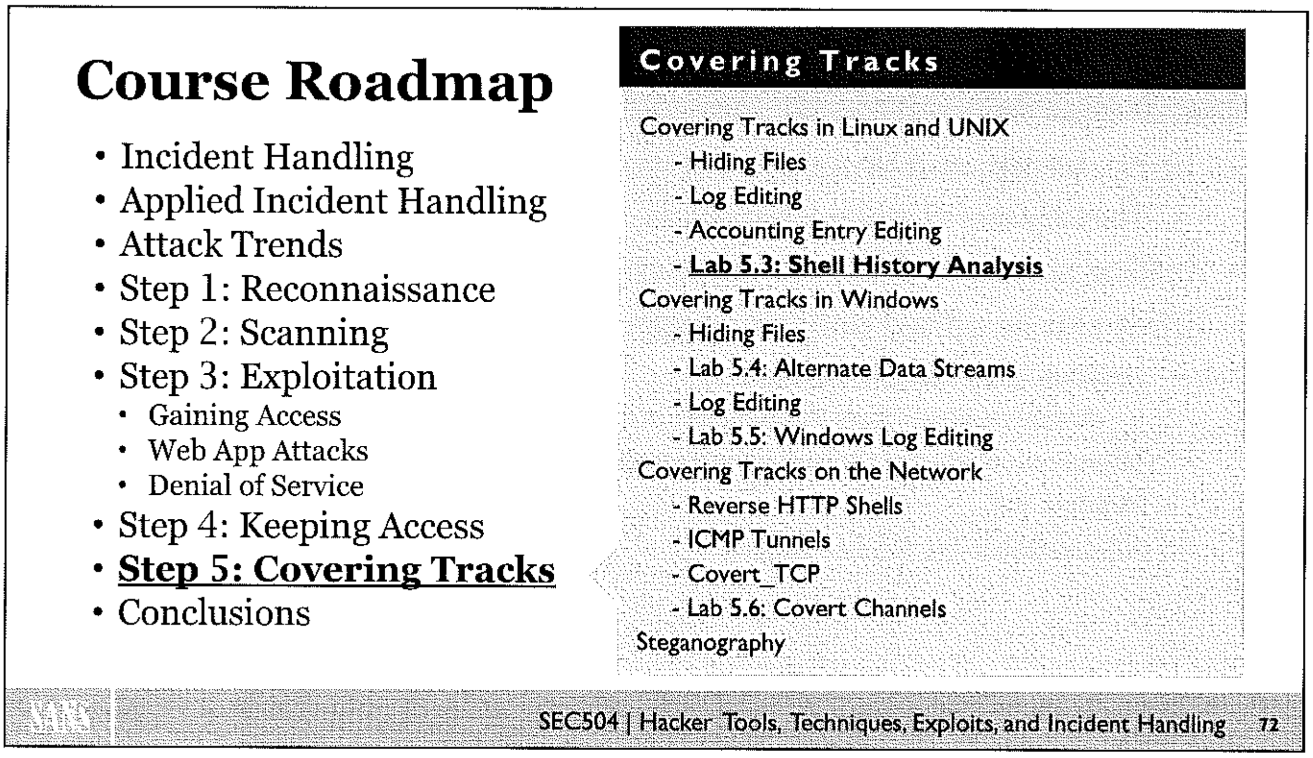
- utmp, wtmp, and btmp are not stored in ASCII
  - They are stored as utmp structures
- lastlog stored in different manners on various systems
- They can be edited only using specialized tools:
  - remove.c, by Simple Nomad
    - Removes entries from utmp, wtmp, and lastlog
  - Numerous others, including wtmped.c, marry.c, cloak.c, logwedit.c, wzap.c, etc.
  - All available on Packet Storm

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 71

An attacker cannot simply edit the utmp, wtmp, btmp, and lastlog files by hand. These files are written in a special format (a "utmp" structure) and must be edited with a tool that can read and write this format. If an attacker attempts to edit these files without a tool that understands this format, the files will become corrupted.

To edit the accounting files, an attacker can choose from several tools, including "remove," "marry," and others. These tools allow for the alteration of the wtmp, utmp, btmp, and lastlog files by recreating the appropriate binary format information so the files will not appear corrupted.

Many different log editing and wiping tools are available on the Packet Storm site at www.packetstormsecurity.org/UNIX/penetration/log-wipers.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **<u>Step 5: Covering Tracks</u>**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- **Lab 5.3: Shell History Analysis**

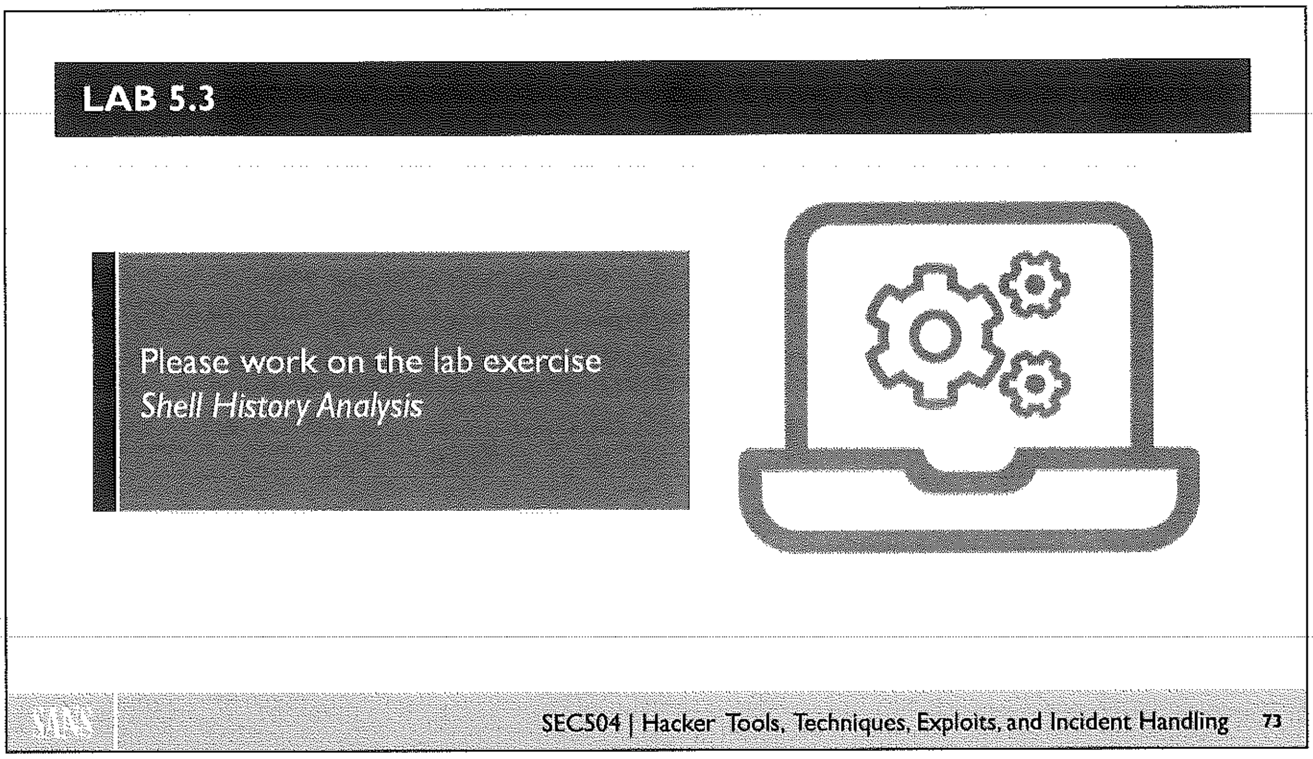
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 72

Next, let's work on a lab in which we look at attackers' tracks in a shell history file.



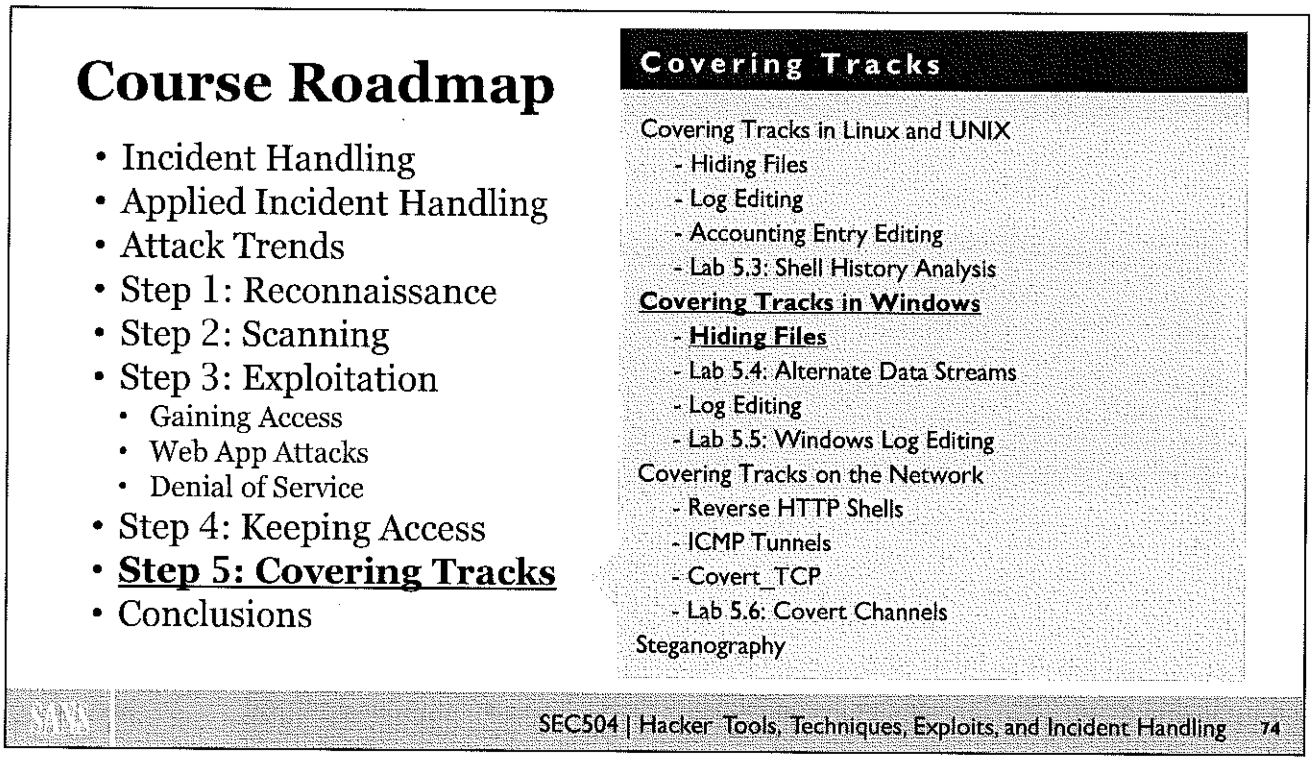
LAB 5.3

Please work on the lab exercise
Shell History Analysis

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 73

This page intentionally left blank.

[illegible]



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

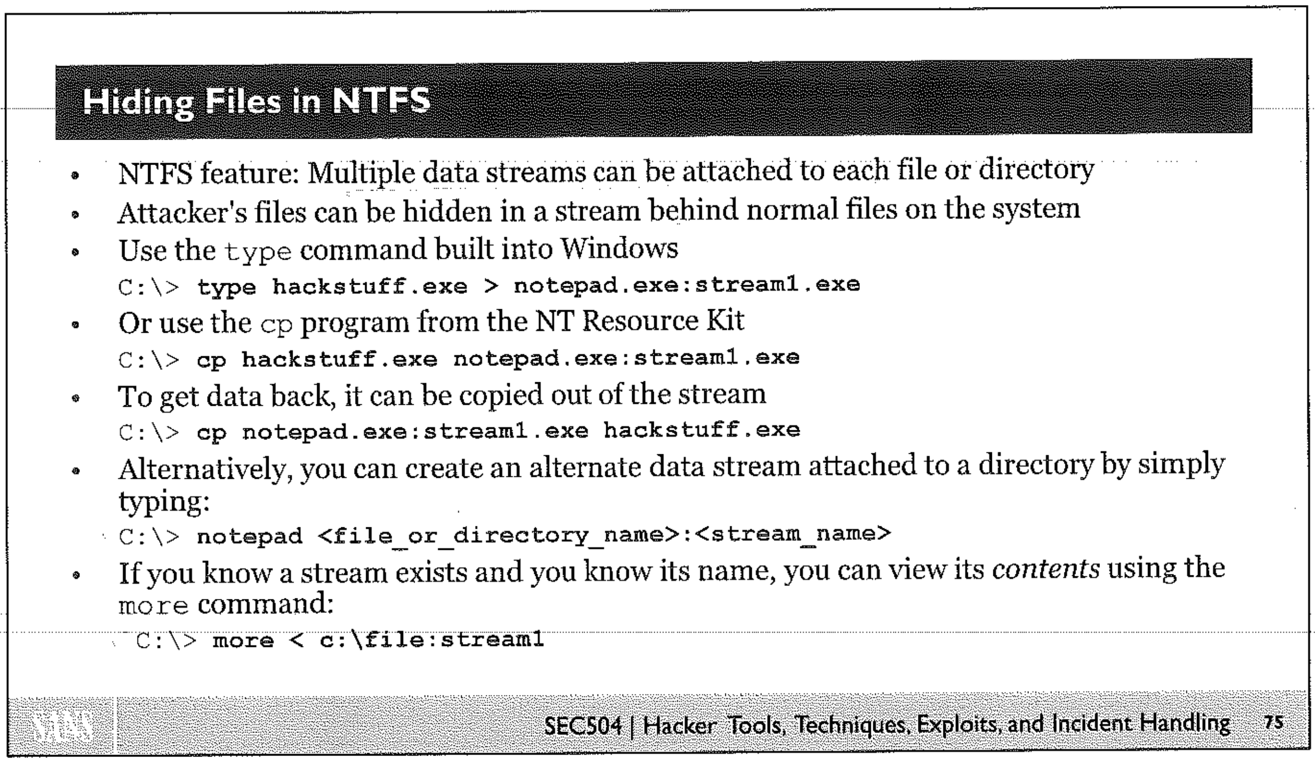
Covering Tracks in Windows
- **Hiding Files**
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 74

Now that we have seen how attackers hide their tracks on a UNIX system, we're back to our roadmap. Let's explore how they hide files and edit logs on a Windows machine.



Hiding Files in NTFS

- NTFS feature: Multiple data streams can be attached to each file or directory
- Attacker's files can be hidden in a stream behind normal files on the system
- Use the `type` command built into Windows

```
C:\> type hackstuff.exe > notepad.exe:stream1.exe
```

- Or use the `cp` program from the NT Resource Kit

```
C:\> cp hackstuff.exe notepad.exe:stream1.exe
```

- To get data back, it can be copied out of the stream

```
C:\> cp notepad.exe:stream1.exe hackstuff.exe
```

- Alternatively, you can create an alternate data stream attached to a directory by simply typing:

```
C:\> notepad <file_or_directory_name>:<stream_name>
```

- If you know a stream exists and you know its name, you can view its *contents* using the `more` command:

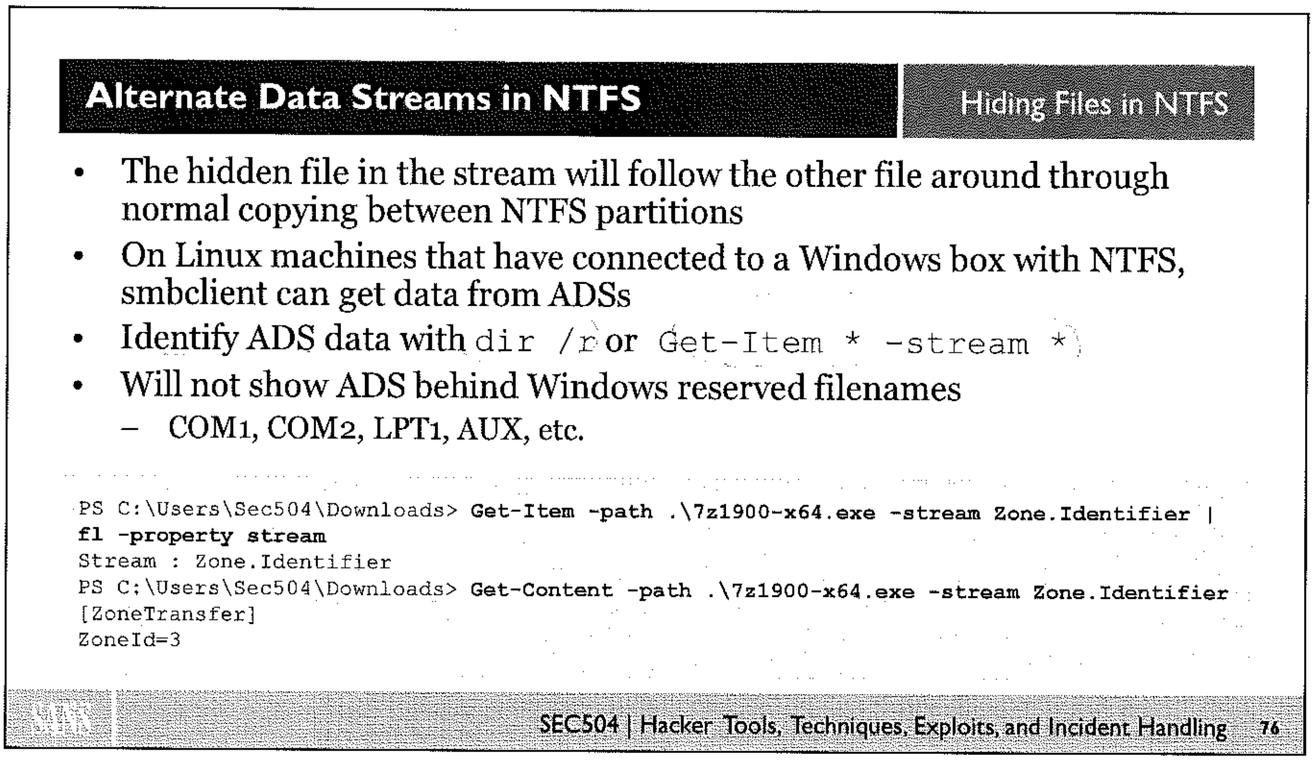
```
C:\> more < c:\file:stream1
```

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 75

The Windows NT File System (NTFS) supports a feature known as "file streaming." File streaming applies only to NTFS partitions; it does not apply to FAT partitions. Each file on an NTFS partition acts kind of like a chest of drawers. The name of the chest is the name of the file itself. Underneath this name, there can be arbitrary numbers of data streams associated with the file's name. Each data stream acts as a drawer into which a user can place data. All streams are still associated with the name of the original file. The first stream associated with a file is the contents of the file itself, the thing you see when you look at it in the Windows Explorer. When you look at a file in the Windows Explorer, you see the filename followed by the size of the first stream.

An attacker can create additional streams associated with any file or directory name on the system. The attacker can then use these streams to hide his or her sensitive information, such as attack tools or sniffer logs. Any file or directory on an NTFS partition can be used to hide such information, such as Notepad or Word. To create and interact with file streams, the type command built into Windows can be used. Alternatively, the cp program may be used from the Windows NT Resource Kit. The attacker simply copies his or her information to be hidden into the filename, followed by a colon, followed by the stream name assigned by the attacker. This stream acts like another drawer in the chest of drawers.

The Windows "more" command can be used to view the contents of a stream, but you'll need to know the stream's location and name to invoke the "more" command to view its data.



Alternate Data Streams in NTFS

Hiding Files in NTFS

- The hidden file in the stream will follow the other file around through normal copying between NTFS partitions
- On Linux machines that have connected to a Windows box with NTFS, smbclient can get data from ADSs
- Identify ADS data with `dir /r` or `Get-Item * -stream *`
- Will not show ADS behind Windows reserved filenames
  - COM1, COM2, LPT1, AUX, etc.

```
PS C:\Users\Sec504\Downloads> Get-Item -path .\7z1900-x64.exe -stream Zone.Identifier |
fl -property stream
Stream : Zone.Identifier
PS C:\Users\Sec504\Downloads> Get-Content -path .\7z1900-x64.exe -stream Zone.Identifier
[ZoneTransfer]
ZoneId=3
```

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 76

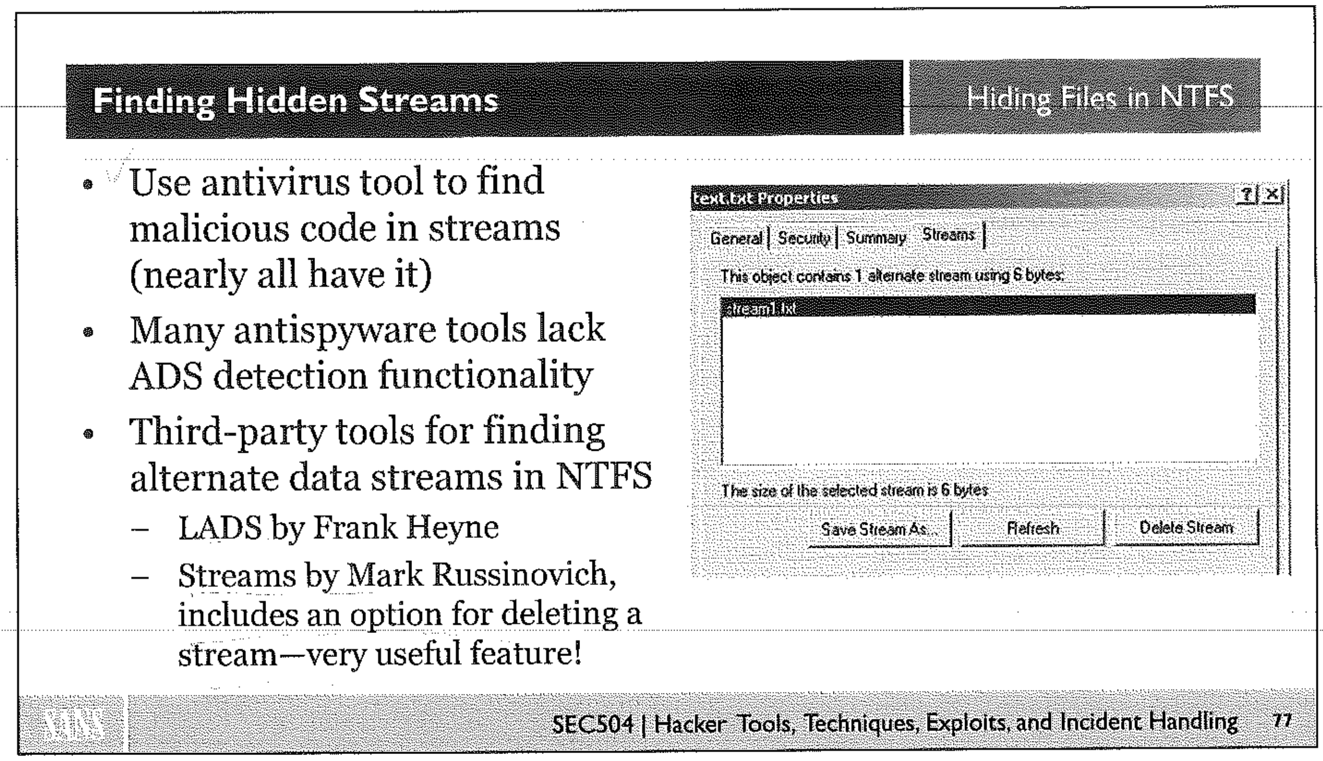
When the data is hidden in the stream, the Windows Explorer and dir command still show only the name and size of the first stream of data (the top drawer). The name of subsequent streams and the size of subsequent streams are not listed in Windows Explorer or standard dir or the Get-ChildItem cmdlet.

When you move a file that has many streams associated with it, all of the streams move on the system. For example, if you copy a file between partitions or move it across a Windows network share, Windows will send all of the streams. Therefore, if you have a small file, such as 6 KB, and it has a large stream associated with it, say 1 GB, you'll move all of the contents of the file even though it appears to be only 6 KB long. Therefore, if you move such a file via Windows file sharing, it will take quite a lot of time as it transfers over more than 1 GB of data.

You can identify the presence of an alternate data stream in Windows using the `cmd.exe dir /r` command, or using the `Get-Item` PowerShell cmdlet with the `-property stream` option, as shown on this page. In this example, the `7z1900-x64.exe` file contains the `Zone.Identifier` stream, used to record the downloaded file's internet zone. Of course, attackers can plant much more nefarious data in ADSs as well.

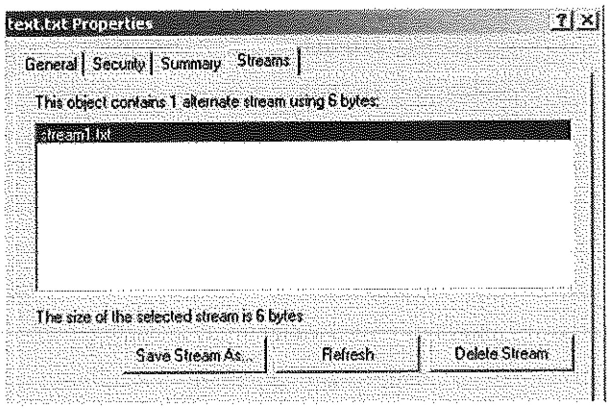
Interestingly, the Linux smbclient program can also read data from alternate data streams from a Windows share, but the attacker needs to know the stream name to be able to refer to it and pull out the data.

It's important to note that Windows offers no built-in capability for deleting a stream, however. To delete a stream using built-in Windows functionality, you could move the file with the stream to a FAT partition (such as a floppy disk) and then move it back. The stream will then be dropped.



Finding Hidden Streams

- Use antivirus tool to find malicious code in streams (nearly all have it)
- Many antispyware tools lack ADS detection functionality
- Third-party tools for finding alternate data streams in NTFS
  - LADS by Frank Heyne
  - Streams by Mark Russinovich, includes an option for deleting a stream—very useful feature!



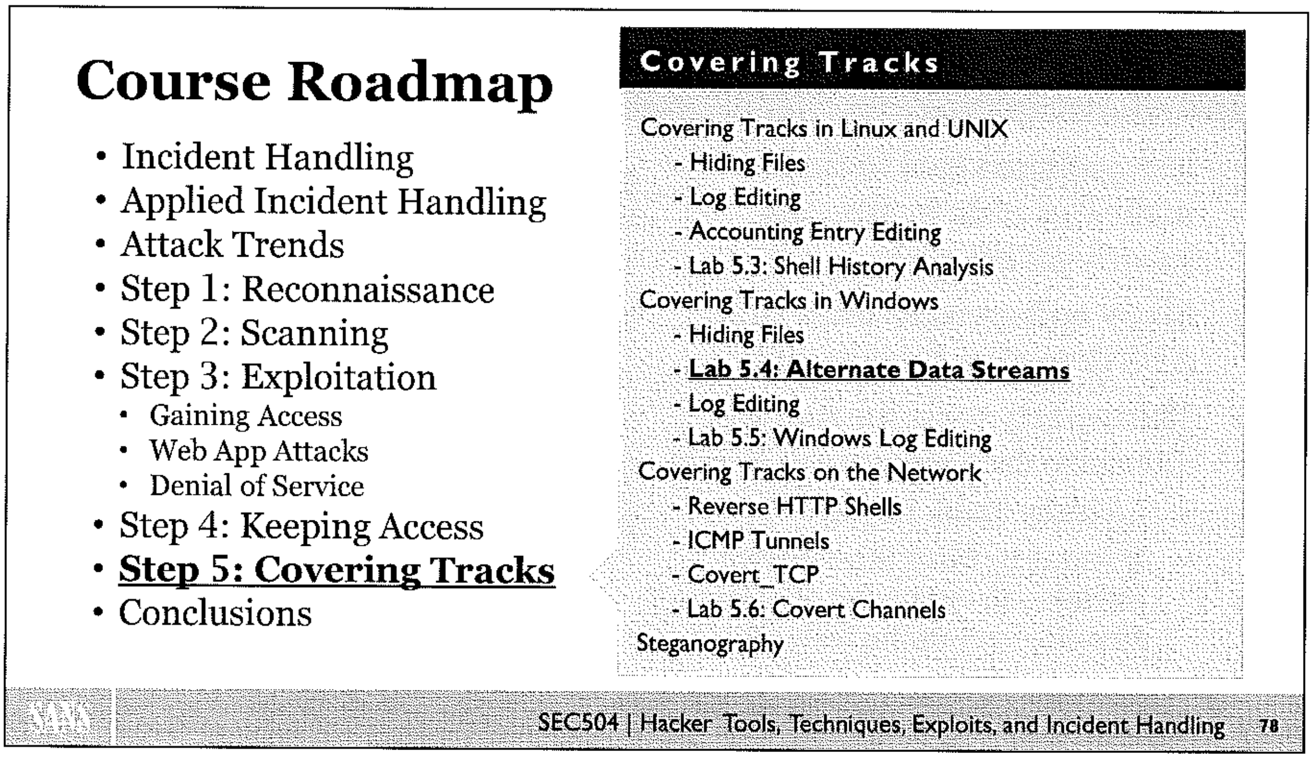
This object contains 1 alternate stream using 6 bytes:

stream1.txt

The size of the selected stream is 6 bytes

To detect malicious software hidden in a file stream, you must make sure to use an updated antivirus tool. Over the last couple of years, most major antivirus manufacturers have included the ability to search through file streams to find malicious software. Prior to that, antivirus tools did not look into additional streams. Today, many antispyware tools do not look into ADSs during their real-time or on-demand scans. Thus, spyware can stay undetected on a system by loading into an ADS and running from inside it.

LADS is a tool dedicated to finding alternate data streams in NTFS. It is very well written, and we strongly recommend that you get a copy and become familiar with it for future forensic use. Another useful tool is *"streams"* by Mark Russinovich, at https://docs.microsoft.com/en-us/sysinternals/. The `streams` program includes a very handy option for deleting a stream without impacting the host file. Finally, Ryan Means wrote a shell extension utility that adds stream-viewing capabilities to Windows in the Properties window for each file, as shown in the screenshot on the slide.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

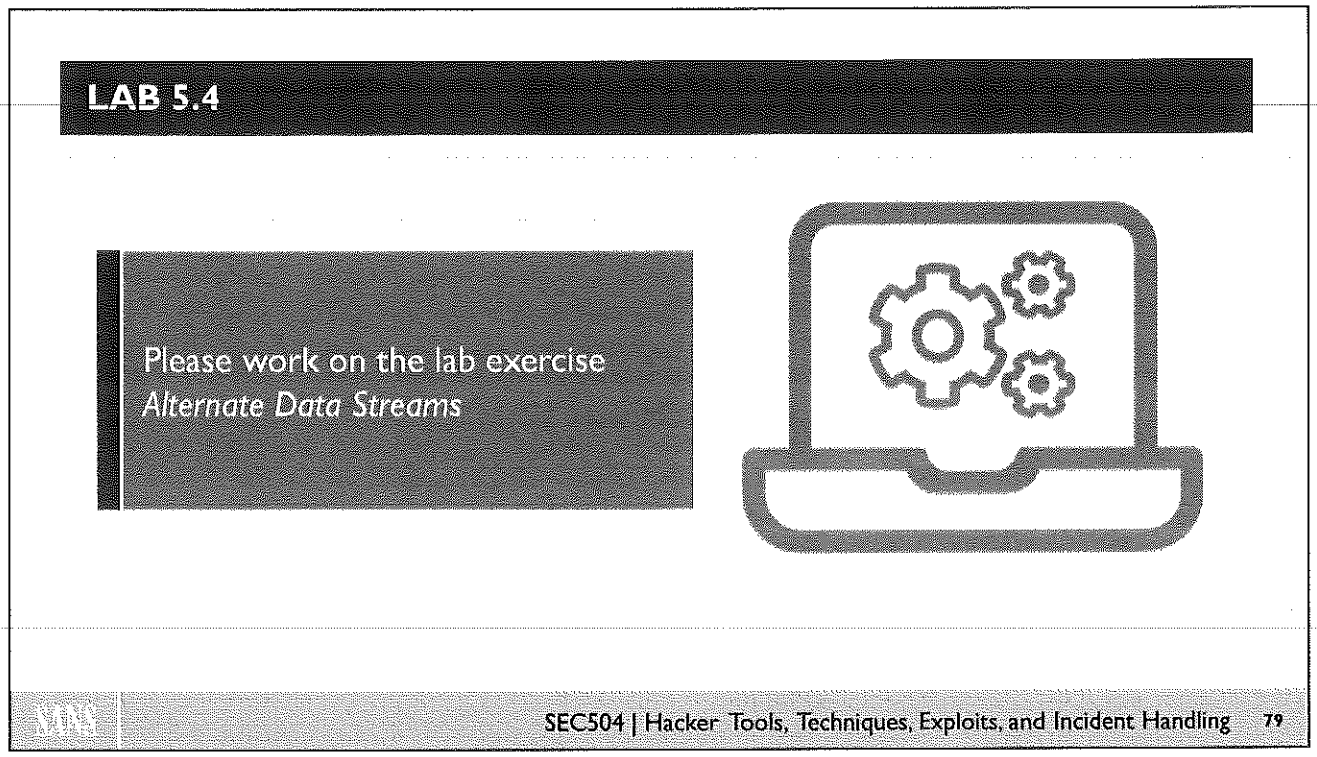
Covering Tracks in Windows
- Hiding Files
- **Lab 5.4: Alternate Data Streams**
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 78

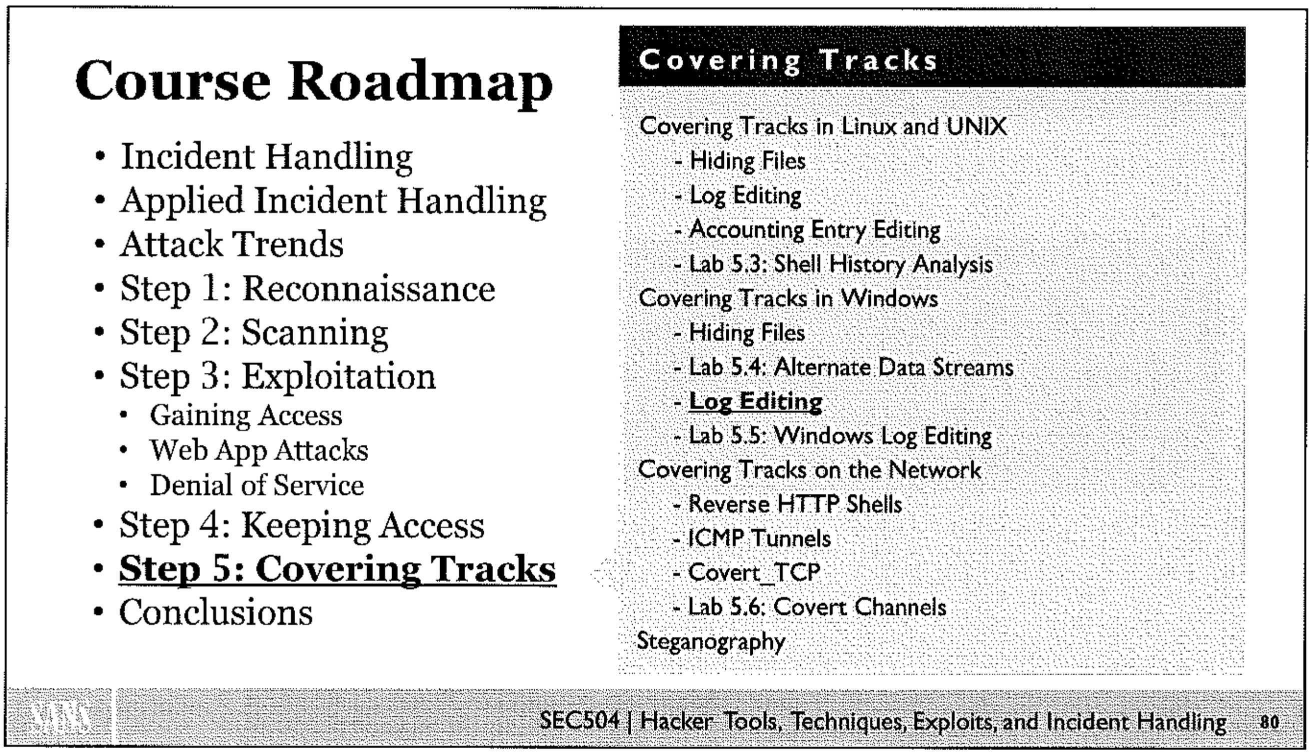
Next, let's experiment with alternate data streams on Windows by conducting a lab featuring them.



LAB 5.4

Please work on the lab exercise
Alternate Data Streams

This page intentionally left blank.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

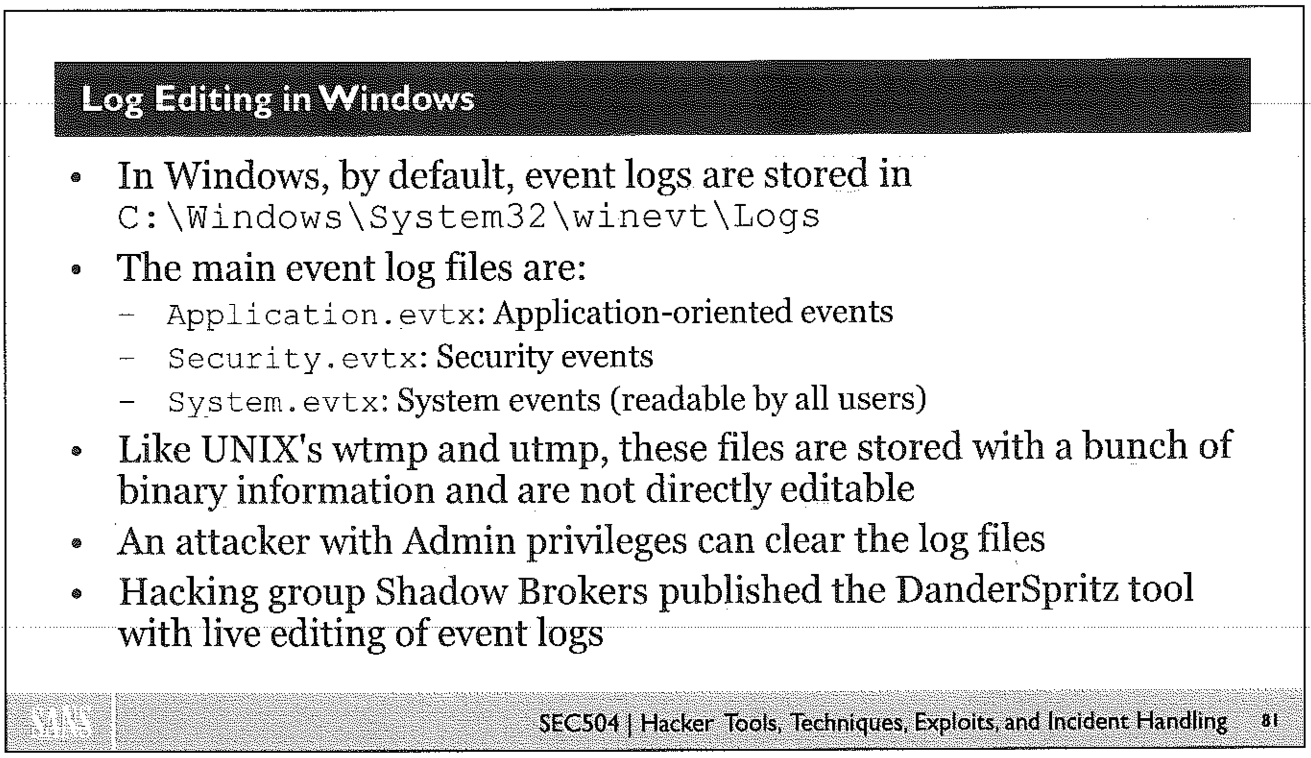
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- **Log Editing**
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 80

Looking at our roadmap, let's now explore how log editing works on Windows.



Log Editing in Windows

- In Windows, by default, event logs are stored in `C:\Windows\System32\winevt\Logs`
- The main event log files are:
  - `Application.evtx`: Application-oriented events
  - `Security.evtx`: Security events
  - `System.evtx`: System events (readable by all users)
- Like UNIX's wtmp and utmp, these files are stored with a bunch of binary information and are not directly editable
- An attacker with Admin privileges can clear the log files
- Hacking group Shadow Brokers published the DanderSpritz tool with live editing of event logs

In Windows, system logs are generated by the Event Logger service. These files are not readable for all practical purposes.

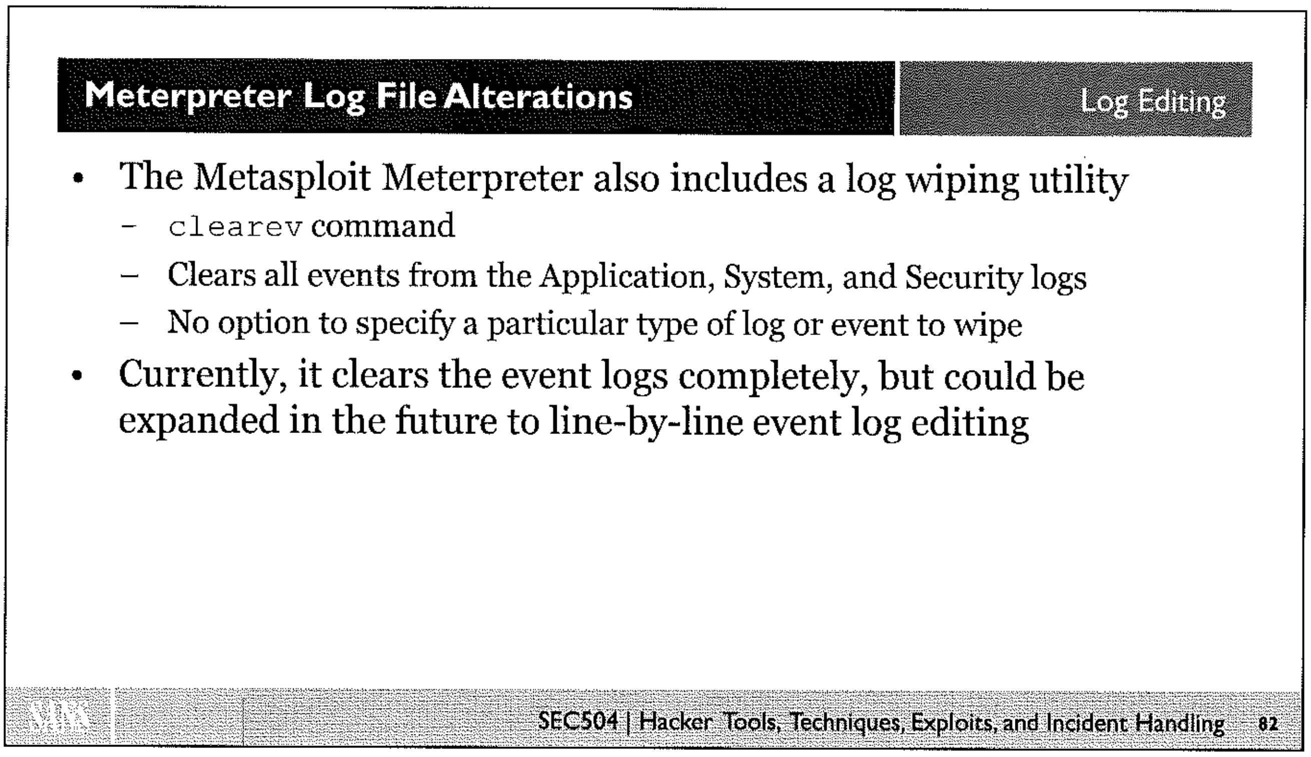
Each .LOG file is periodically rewritten into a .EVT format automatically, in the following files:

- system.evtx
- security.evtx
- application.evtx

Some versions of Windows have additional EVT files, but these three are the primary ones. These files are readable through the Event Viewer and are the main log files in Windows. They are write-protected and cannot be altered on a running system by conventional means.

One option to erase traces of activity is for a perpetrator to edit SECEVENT.EVT, but to be more confident that all traces of the perpetrator's activity are gone, the attacker would possibly want to edit the other log files as well. Note that *deleting* any .EVT file is no problem for anyone who has the proper right ("Manage Audit and Security Log") or permission (say, "Delete" for the \windows\system32\config directory that holds these logs). The logs can be cleared using the Event Viewer tool itself with administrative privileges. However, completely blowing away a log file is likely to be noticed by a system administrator, so a sophisticated attacker would rather do line-by-line editing of the log files.

In 2017, the Shadow Brokers hacking group published a tool called DanderSpritz (available at https://danderspritz.com/). Included in this suite of post-exploitation capabilities is a plugin called eventlogsedit. This was the first time a tool for editing event logs was observed in use in the wild. An excellent article written by Wouter Jansen on the capabilities of this tool, and forensic analysis capabilities to detect its use is available at https://blog.fox-it.com/2017/12/08/detection-and-recovery-of-nsas-covered-up-tracks/.



Meterpreter Log File Alterations

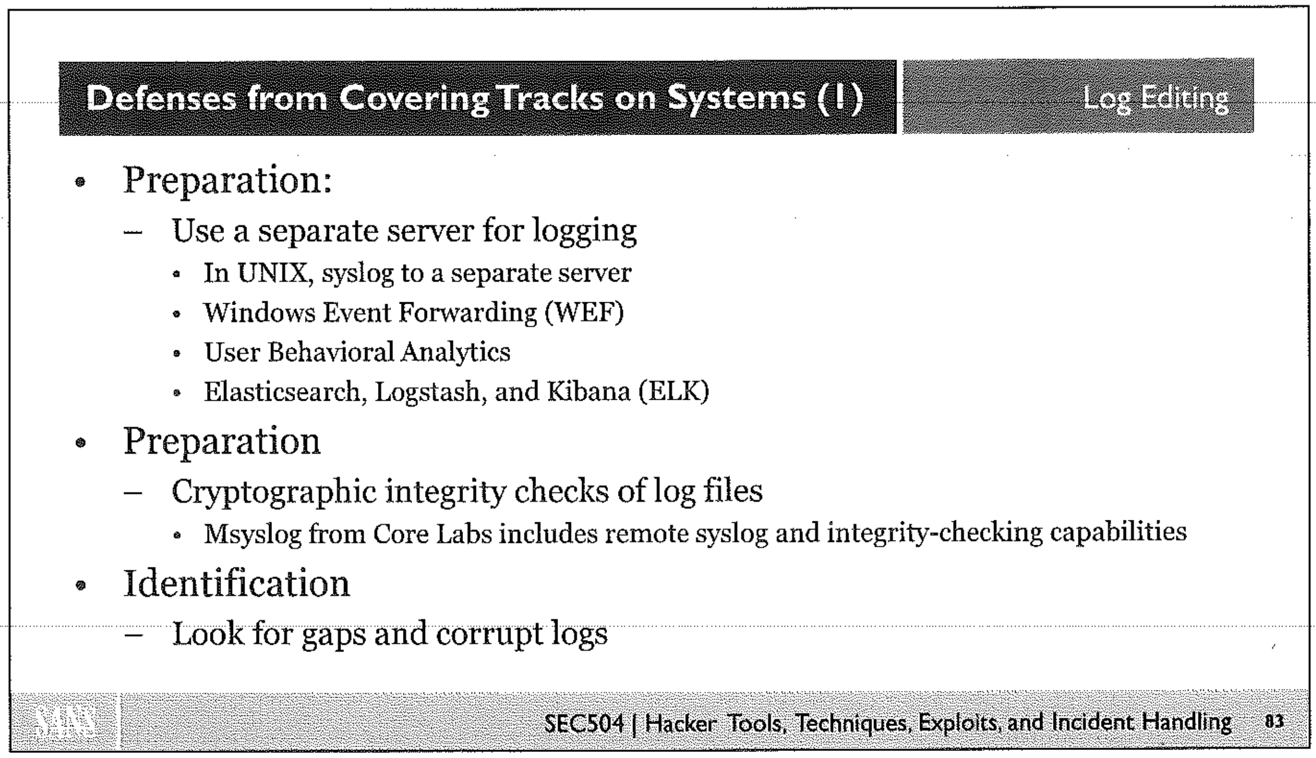
Log Editing

- The Metasploit Meterpreter also includes a log wiping utility
  - `clearev` command
  - Clears all events from the Application, System, and Security logs
  - No option to specify a particular type of log or event to wipe
- Currently, it clears the event logs completely, but could be expanded in the future to line-by-line event log editing

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 82

The Metasploit Meterpreter includes a feature to clear logs via the "clearev" command. This command, when invoked from within the Meterpreter of a compromised machine, clears the entire contents of the Application, System, and Security logs.

There is no option to specify a particular log or event type to clear. This feature currently is a one-shot log eraser and does not offer line-by-line editing of logs at this time.



Defenses from Covering Tracks on Systems (1)

Log Editing

- Preparation:
  - Use a separate server for logging
    - In UNIX, syslog to a separate server
    - Windows Event Forwarding (WEF)
    - User Behavioral Analytics
    - Elasticsearch, Logstash, and Kibana (ELK)
- Preparation
  - Cryptographic integrity checks of log files
    - Msyslog from Core Labs includes remote syslog and integrity-checking capabilities
- Identification
  - Look for gaps and corrupt logs

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 83

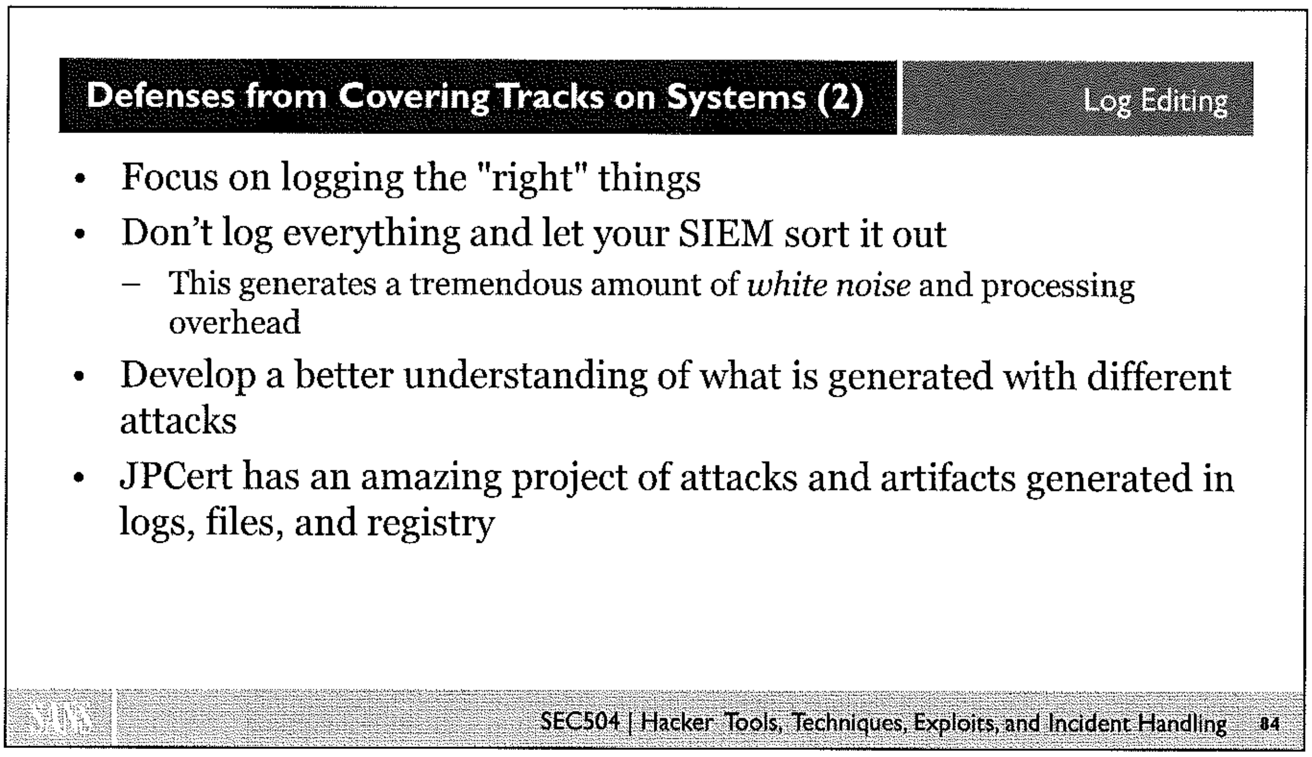
Clearly, we don't want attackers to alter our log files. How can we protect our system logs? Use of these log-defending techniques depends on the sensitivity of the server. Clearly, for internet-accessible machines with sensitive data, a great amount of care must be taken with the logs. For internal systems, logging might be less important. This is clearly a policy issue and depends on the needs of your organization. All of the defenses listed on the slide can be applied together. They are not mutually exclusive.

One of the most obvious and effective ways to defend your logs is to employ a separate logging server. Think about it: If an attacker takes over one of your machines and has root or administrative privileges on that system, he or she can easily find the system logs on that machine and alter them. However, if the system logs are sent to a remote system, the attacker will not be able to edit them on the machine that he or she has just taken over. Instead, after taking over one system, he or she will have to mount another successful attack against your logging server. Because your logging server can be dedicated to just gathering system logs, it can be very carefully secured. In UNIX, the syslog process can be easily configured to send its logs to a remote host.

For Windows systems, consider the deployment of the Windows Event Forwarding (WEF, https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection) tools. WEF leverages the built-in WinRM service to pull or push logs to a central repository where they can be aggregated for subsequent analysis.

There are also great tools to help with parsing and searching through logs. One of our favorites is ELK. This is a combination of tools that make it far easier to ingest, store, search, and display results for certain event types.

In addition to sending the logs to a remote logging server, you can also generate a cryptographic integrity check of the log files to protect them. The free msyslog tool from Core Labs includes remote system logging and cryptographic integrity checking capabilities. Msyslog is available at https://www.coresecurity.com/grid/index-open-source-tools.



Defenses from Covering Tracks on Systems (2)

Log Editing

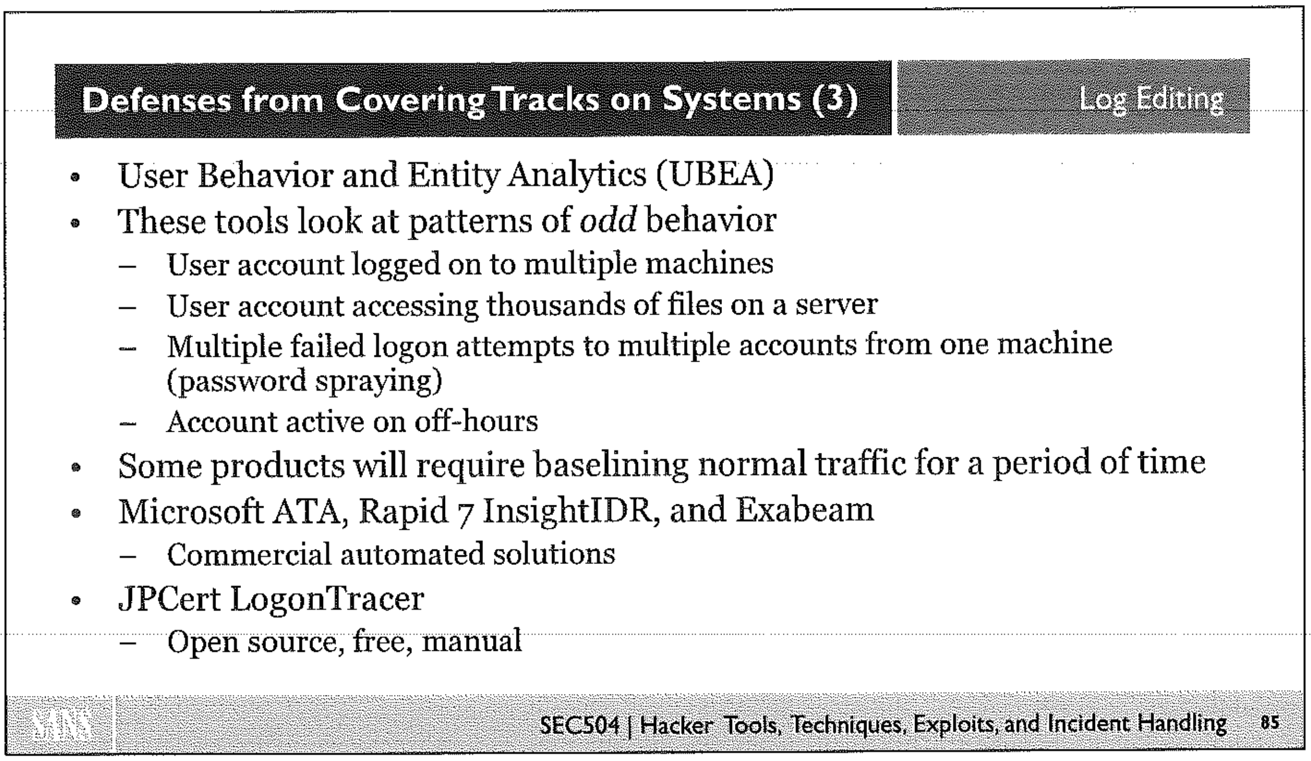
- Focus on logging the "right" things
- Don't log everything and let your SIEM sort it out
  - This generates a tremendous amount of *white noise* and processing overhead
- Develop a better understanding of what is generated with different attacks
- JPCert has an amazing project of attacks and artifacts generated in logs, files, and registry

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 84

We also need to break away from logging everything in the hopes that our SIEM will sort it all out for us. In most organizations, less than 10% of all the event logs that are captured have any direct correlation to an attack or to any type of rule or alert. We can do better.

We can tune the SIEM to look for events of interest. JPCert has done a great job logging which type of events are triggered when certain attack tools and techniques are used. They have a full list of the events and tools here:

https://jpcertcc.github.io/ToolAnalysisResultSheet



Defenses from Covering Tracks on Systems (3)

Log Editing

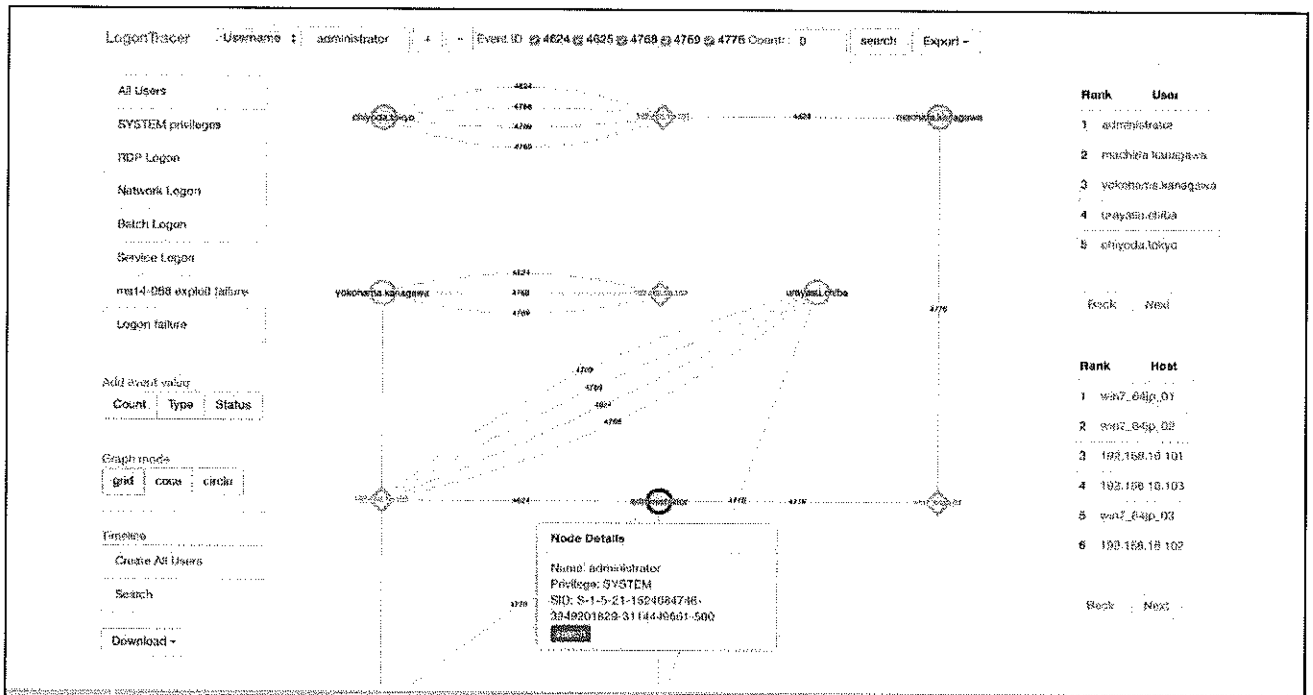
- User Behavior and Entity Analytics (UBEA)
- These tools look at patterns of *odd* behavior
  - User account logged on to multiple machines
  - User account accessing thousands of files on a server
  - Multiple failed logon attempts to multiple accounts from one machine (password spraying)
  - Account active on off-hours
- Some products will require baselining normal traffic for a period of time
- Microsoft ATA, Rapid 7 InsightIDR, and Exabeam
  - Commercial automated solutions
- JPCert LogonTracer
  - Open source, free, manual

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 85

Another approach is to stop looking for specific events to be generated, but rather looking at the behavior of a user or an IT asset as a whole. This is called behavioral analytics.

This is effective to look for activities like a user account logging on to multiple systems at the same time. Or, an account accessing thousands of files on a server. We discussed password spraying earlier in the course, and UBEA can be highly effective for detecting these types of attacks as well.

Some of these products will require a period of time to ingest and baseline the traffic of events to establish what is *normal*. There are a wide range of commercial and free tools to help with this process. The commercial tools have the advantage of automating and reporting, while other tools like LogonTracer can be used in one-off tactical situations for free.



Rank	User
1	administrator
2	machida.kanagawa
3	yokohama.kanagawa
4	urayasu.chiba
5	chiyoda.tokyo

Rank	Host
1	win7_64jp_01
2	win7_64jp_02
3	192.168.16.101
4	192.168.16.103
5	win7_64jp_03
6	192.168.16.102

Node Details

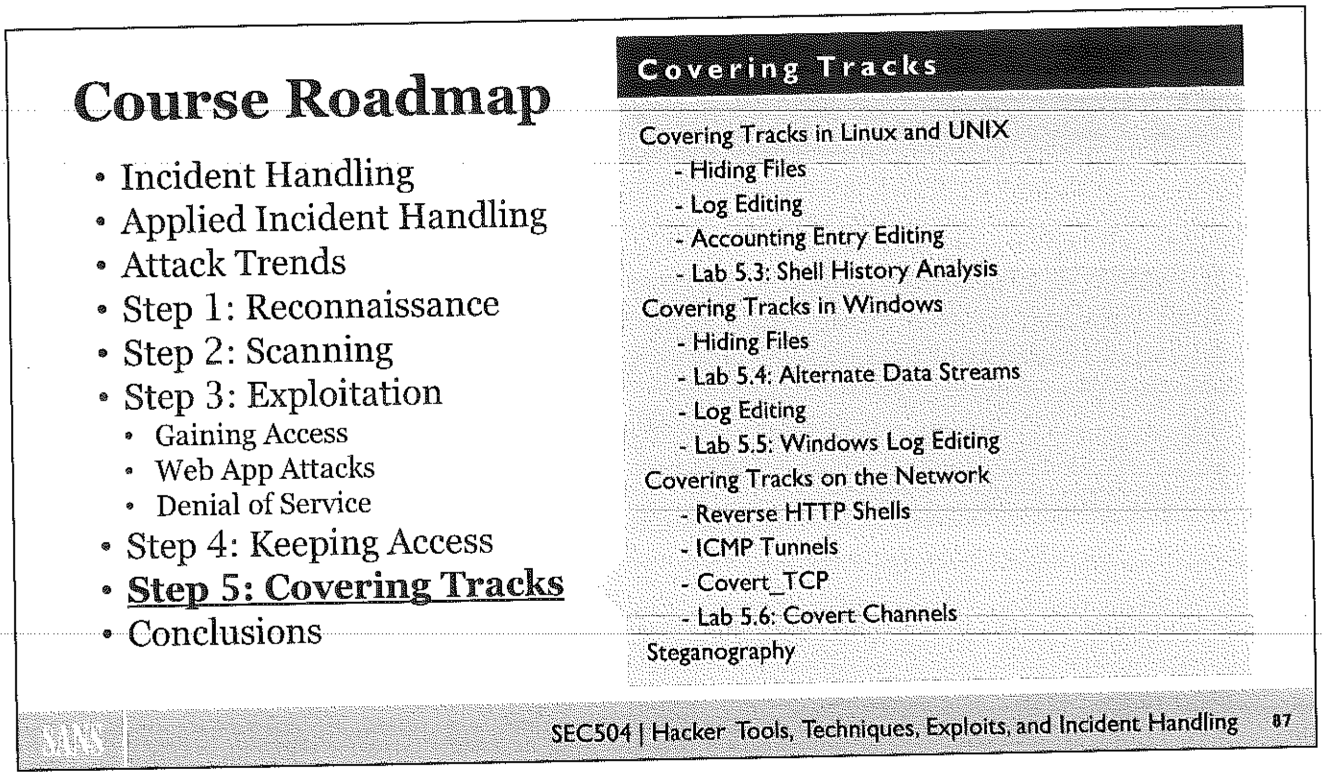
Name: administrator
Privilege: SYSTEM
SID: S-1-5-21-1524084746-3249201829-3114449661-500

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 86

LogonTracer is one of our favorite threat-hunting tools as it is very simplistic and has great data visualizations. At its heart, it takes event IDs 4624 (Successful logon), 4625 (Logon failure), 4768 (Kerberos Authentication TGT Request), 4769, (Kerberos Service Ticket Request), 4776 (NTLM Authentication), and 4672 (Assign special privileges) and ranks accounts and systems that use these events and visualizes the results.

To do this, it ingests and parses the evtx files from a Domain controller.

LogonTracer is freely available at https://github.com/JPCERTCC/LogonTracer/wiki.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

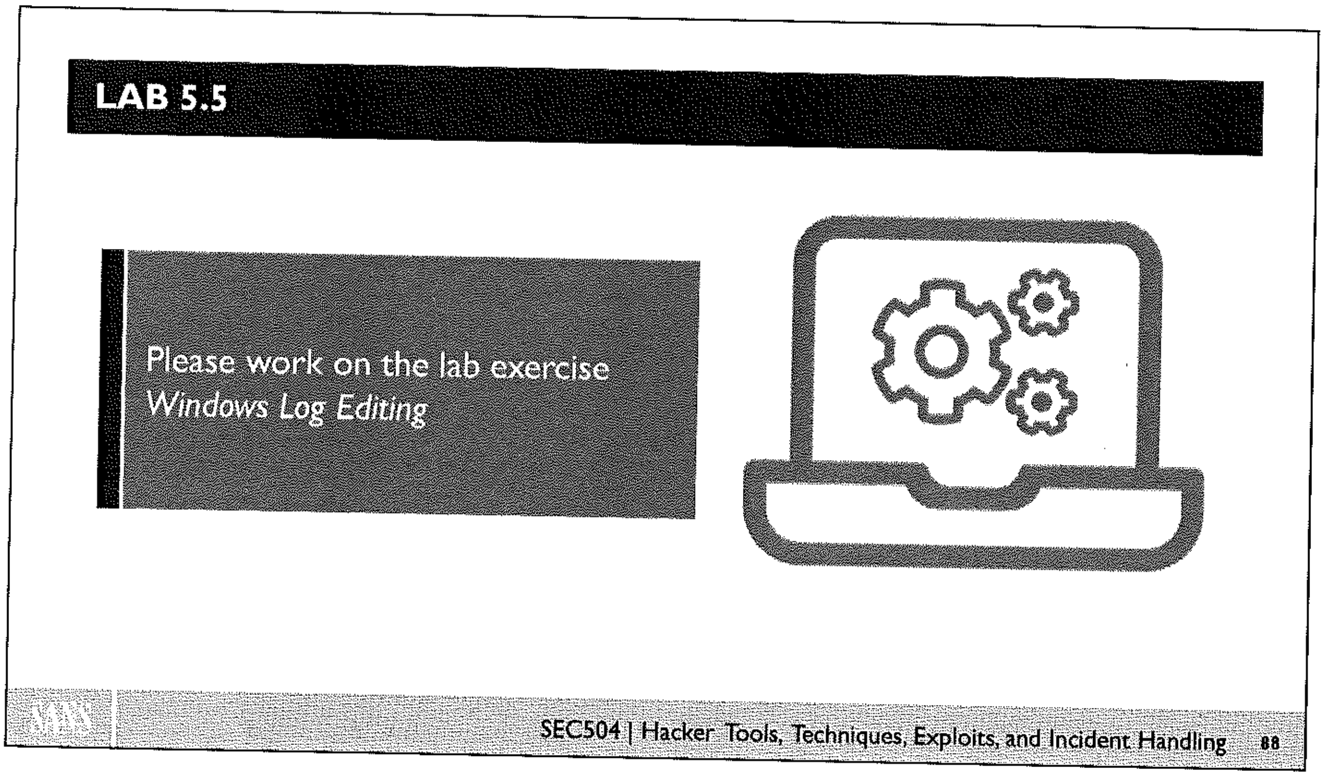
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 87

Next we will work on a lab exercise for Windows Log Editing.

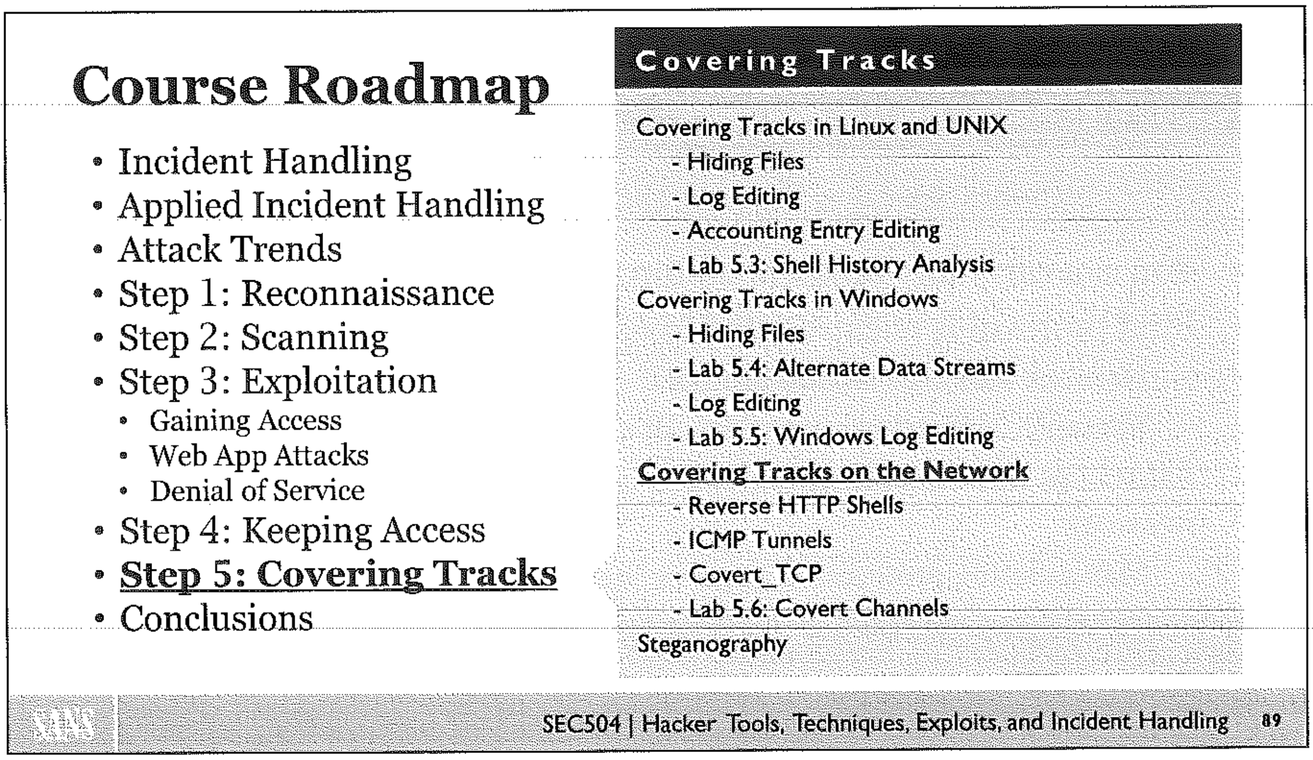


LAB 5.5

Please work on the lab exercise
Windows Log Editing

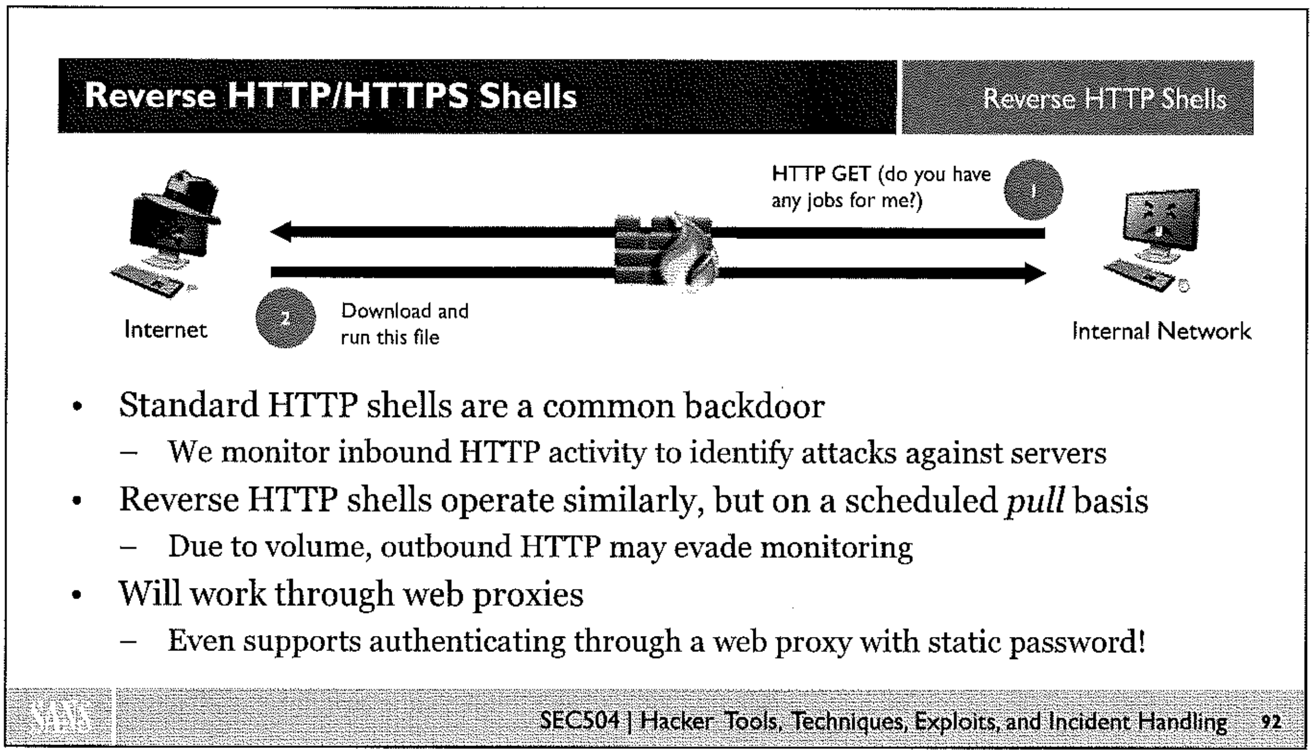
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 88

This page intentionally left blank.

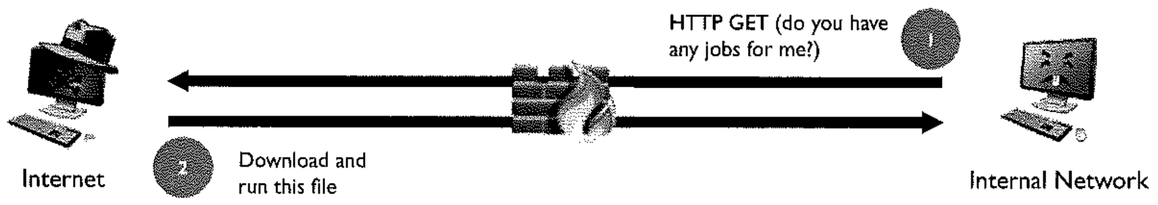


Course Roadmap

Now that we've seen how attackers hide their tracks on a UNIX and Windows system, let's explore how they cover their tracks on a network.



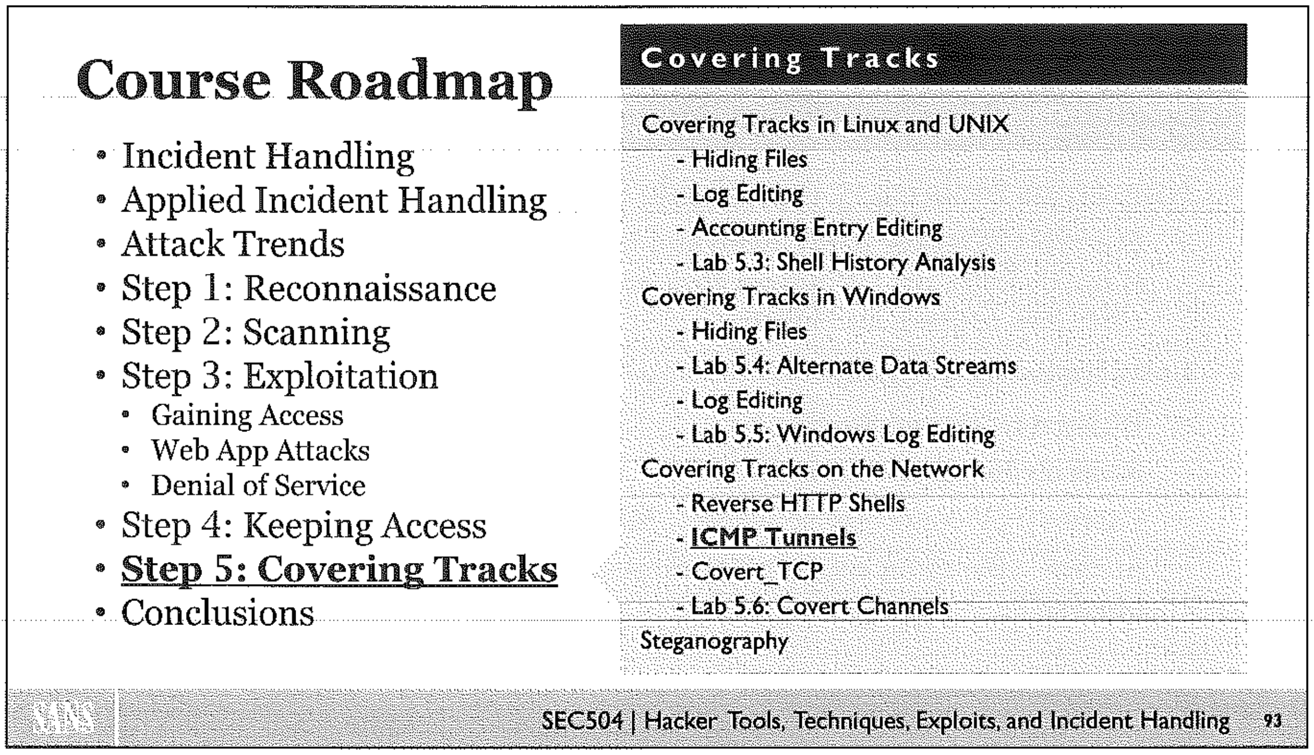
Reverse HTTP/HTTPS Shells


HTTP GET (do you have any jobs for me?)
1
Internet
2
Download and run this file
Internal Network

- Standard HTTP shells are a common backdoor
  - We monitor inbound HTTP activity to identify attacks against servers
- Reverse HTTP shells operate similarly, but on a scheduled *pull* basis
  - Due to volume, outbound HTTP may evade monitoring
- Will work through web proxies
  - Even supports authenticating through a web proxy with static password!

This slide shows pictorially how Reverse HTTP shells work (using HTTP or HTTPS, we'll use the short of *HTTP*, but understand that this is done over both insecure and secure HTTP channels). At predetermined intervals, the Reverse HTTP shell program on the internal system surfs the internet asking for commands from the attacker's external machine. The attacker types in commands at the external machine on the internet and sends the commands back to the victim machine as HTTP responses. These commands are then executed on the internal network host. The results are pushed out with the next web request.

Unfortunately, you are still not safe if you require HTTP authentication with static passwords to get out of your firewall. Reverse HTTP shells allow the attacker to program the system with a user ID and password that will be given to the outgoing web proxy firewall for authentication. Pretty slick!



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

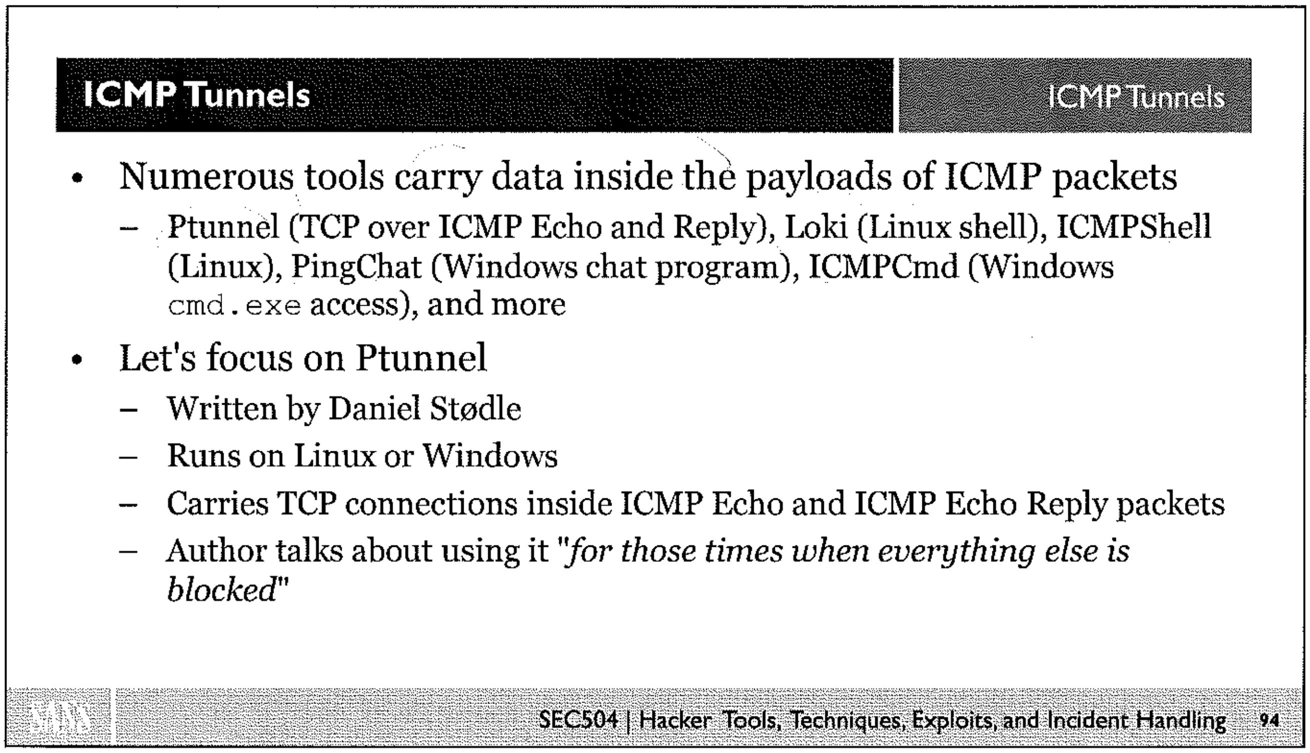
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- **ICMP Tunnels**
- Covert_TCP
- Lab 5.6: Covert Channels

Steganography

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 93

Next we'll look at a more devious covert network channel: ICMP tunneling.



ICMP Tunnels

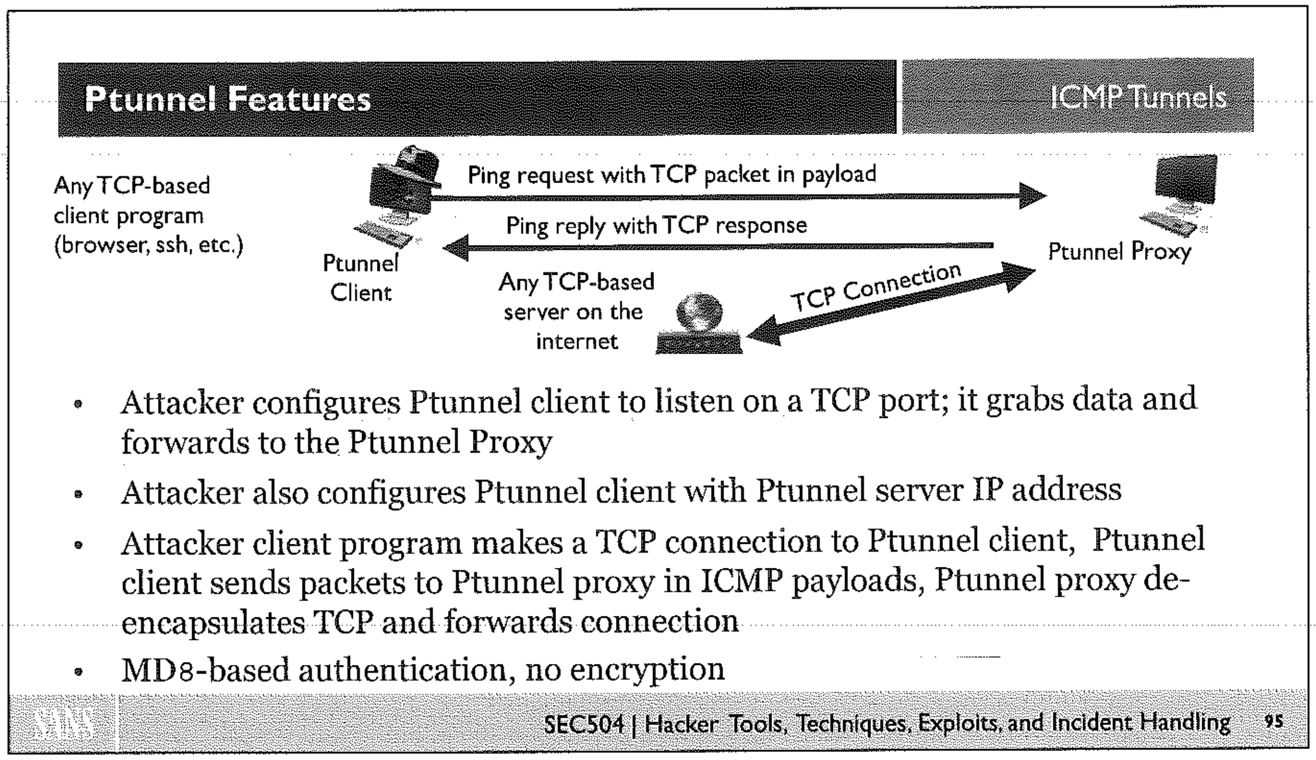
- Numerous tools carry data inside the payloads of ICMP packets
  - Ptunnel (TCP over ICMP Echo and Reply), Loki (Linux shell), ICMPShell (Linux), PingChat (Windows chat program), ICMPCmd (Windows `cmd.exe` access), and more
- Let's focus on Ptunnel
  - Written by Daniel Stødle
  - Runs on Linux or Windows
  - Carries TCP connections inside ICMP Echo and ICMP Echo Reply packets
  - Author talks about using it "*for those times when everything else is blocked*"

Instead of carrying data via HTTP traffic, some attackers opt for other protocols. Numerous tools are readily available to carry traffic via covert channels using ICMP packets. Many networks allow outbound ICMP Echo packets and their associated responses, making ping packets a useful way to tunnel traffic in a covert fashion.

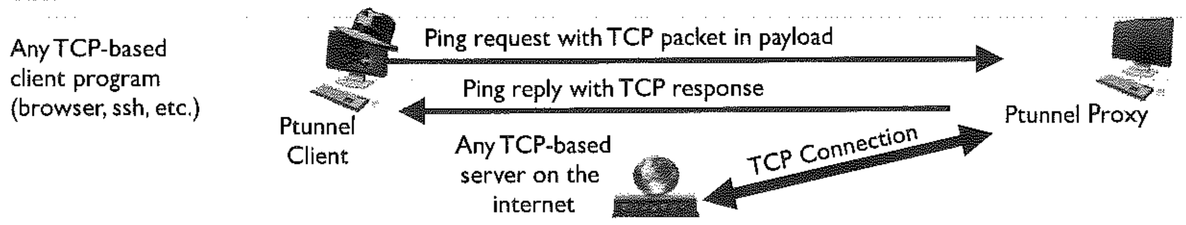
Some of the tools that offer this capability include Ptunnel (which carries TCP connections over ICMP Echo and Reply packets), Loki (which carries shell between its Linux client and Linux server software using ICMP Echo and Reply packets), ICMPShell (another Linux shell tool), PingChat (a Windows chat program that uses ICMP), and ICMPCmd (a Windows shell tool using ICMP).

Let's focus on Ptunnel because it is one of the most flexible tools in this genre. Written by Daniel Stødle and available at http://www.cs.uit.no/~daniels/PingTunnel/, this free tool runs on Linux or Windows, carrying TCP connections inside ICMP Echo and ICMP Echo Reply packets.

Its author talks about using it when you find yourself on a network that blocks out TCP and UDP packets but allows you to ping arbitrary hosts on the internet. Attackers can abuse this kind of tool to tunnel out sensitive information in the payload of ICMP packets.



Ptunnel Features

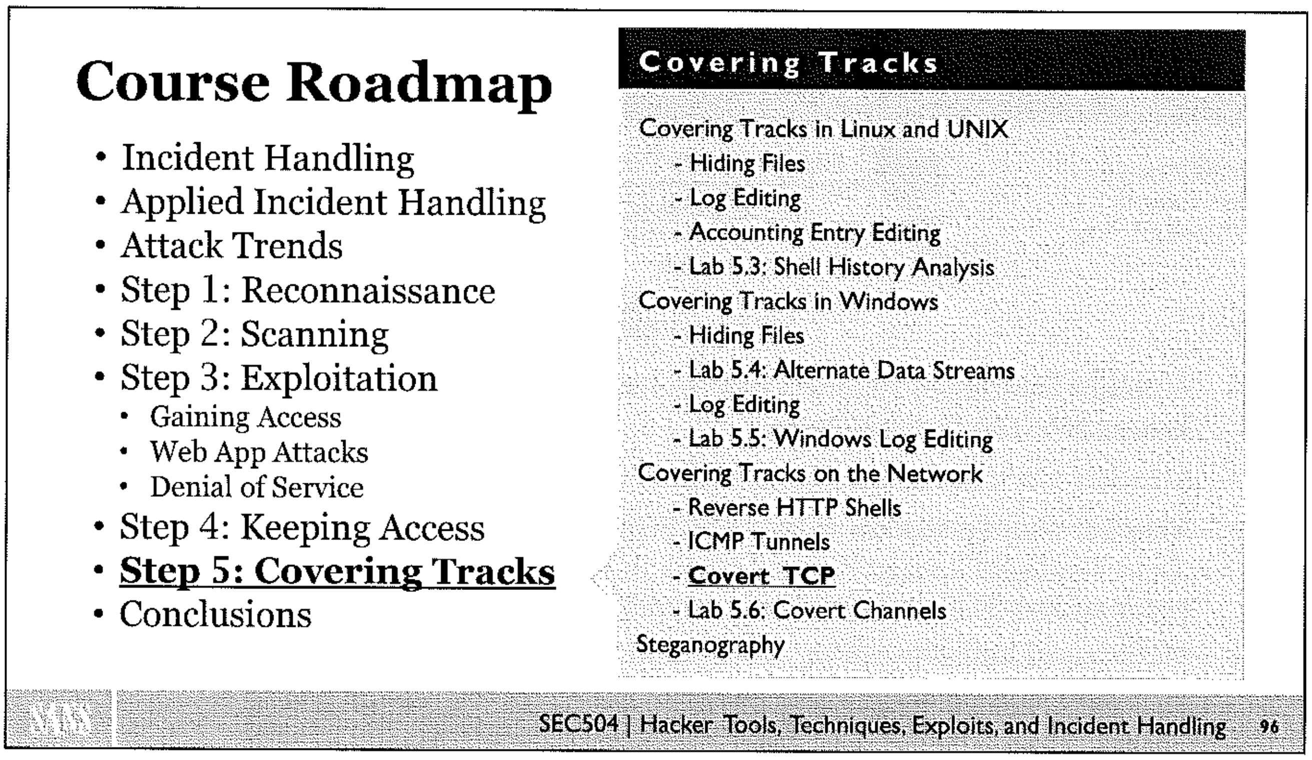

Any TCP-based client program (browser, ssh, etc.)
Ptunnel Client
Ping request with TCP packet in payload
Ping reply with TCP response
Ptunnel Proxy
Any TCP-based server on the internet
TCP Connection

- Attacker configures Ptunnel client to listen on a TCP port; it grabs data and forwards to the Ptunnel Proxy
- Attacker also configures Ptunnel client with Ptunnel server IP address
- Attacker client program makes a TCP connection to Ptunnel client, Ptunnel client sends packets to Ptunnel proxy in ICMP payloads, Ptunnel proxy de-encapsulates TCP and forwards connection
- MD5-based authentication, no encryption

Ptunnel consists of two components: The Ptunnel client and the Ptunnel proxy. The attacker configures the Ptunnel client to listen on a given TCP port on the localhost interface of the client machine. In addition, the attacker must configure the Ptunnel proxy, which runs on an external machine, accessible via ping packets from the Ptunnel client. Finally, the attacker configures the Ptunnel client with a given ultimate destination address. That destination machine can provide any TCP-based service, including HTTP or Secure Shell. Note that the Ptunnel client software is configured with this destination address, which it tells to the Ptunnel proxy for each packet that it sends.

The attacker then runs some TCP-based client program on the attacker's machine, directing it to connect to the localhost interface on the TCP port where the Ptunnel client is listening. The Ptunnel client takes the TCP packets, encapsulates them in ICMP Echo packets, and forwards the resulting packets to the Ptunnel proxy. From a network perspective, only ping packets (with the TCP packet as the payload) are being sent. The Ptunnel proxy then de-encapsulates the TCP packet and forwards it to its ultimate destination simply using TCP. Likewise, the Ptunnel proxy encapsulates any responses that come back from that destination into ICMP Echo Reply packets, forwarding them back to the client.

The Ptunnel proxy can be configured to authenticate the Ptunnel client, using an MD5-based challenge/response authentication algorithm. Currently, Ptunnel does not support encryption. However, if the application using the TCP-based connection encrypts the data (such as HTTPS or SSH), the attacker would have some degree of protection of the data.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

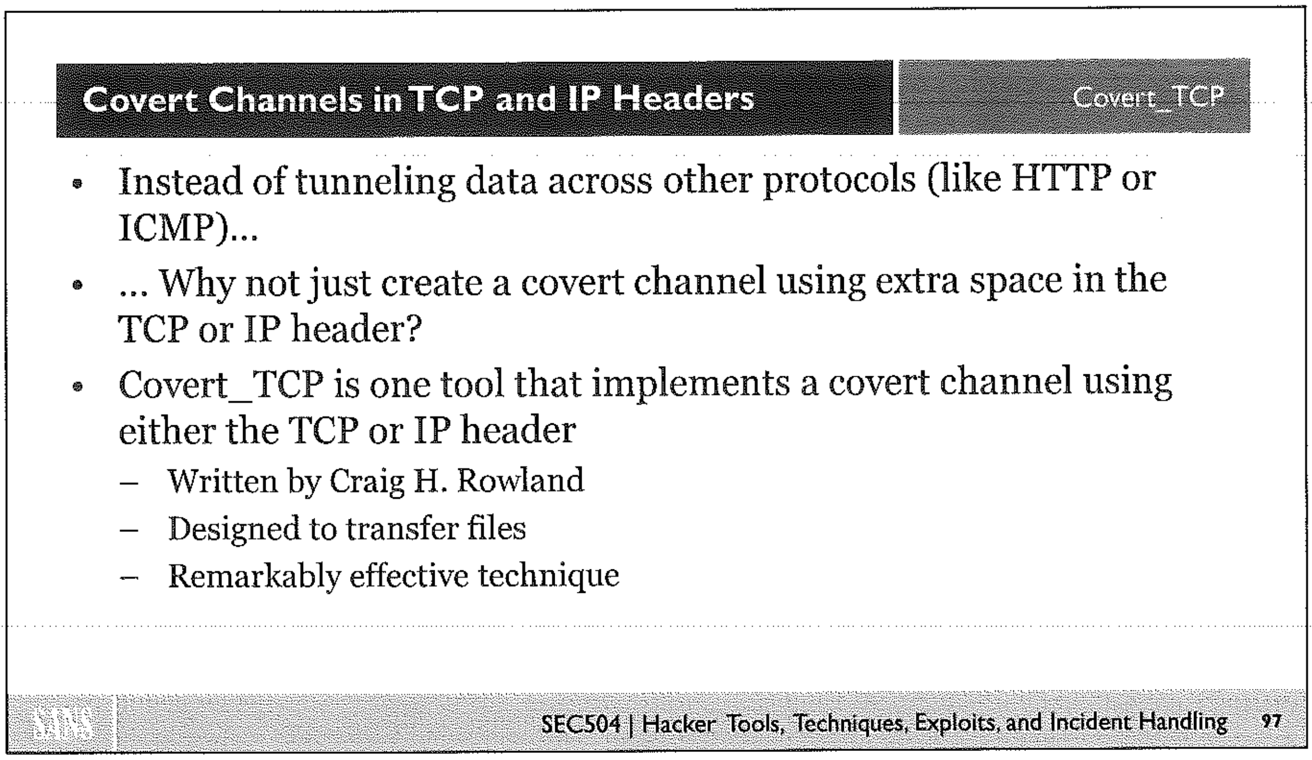
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- **Covert_TCP**
- Lab 5.6: Covert Channels

Steganography

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 96

We'll wrap up the discussion of covert network channels with a look at the features of Covert_TCP and related tools.



Covert Channels in TCP and IP Headers

Covert_TCP

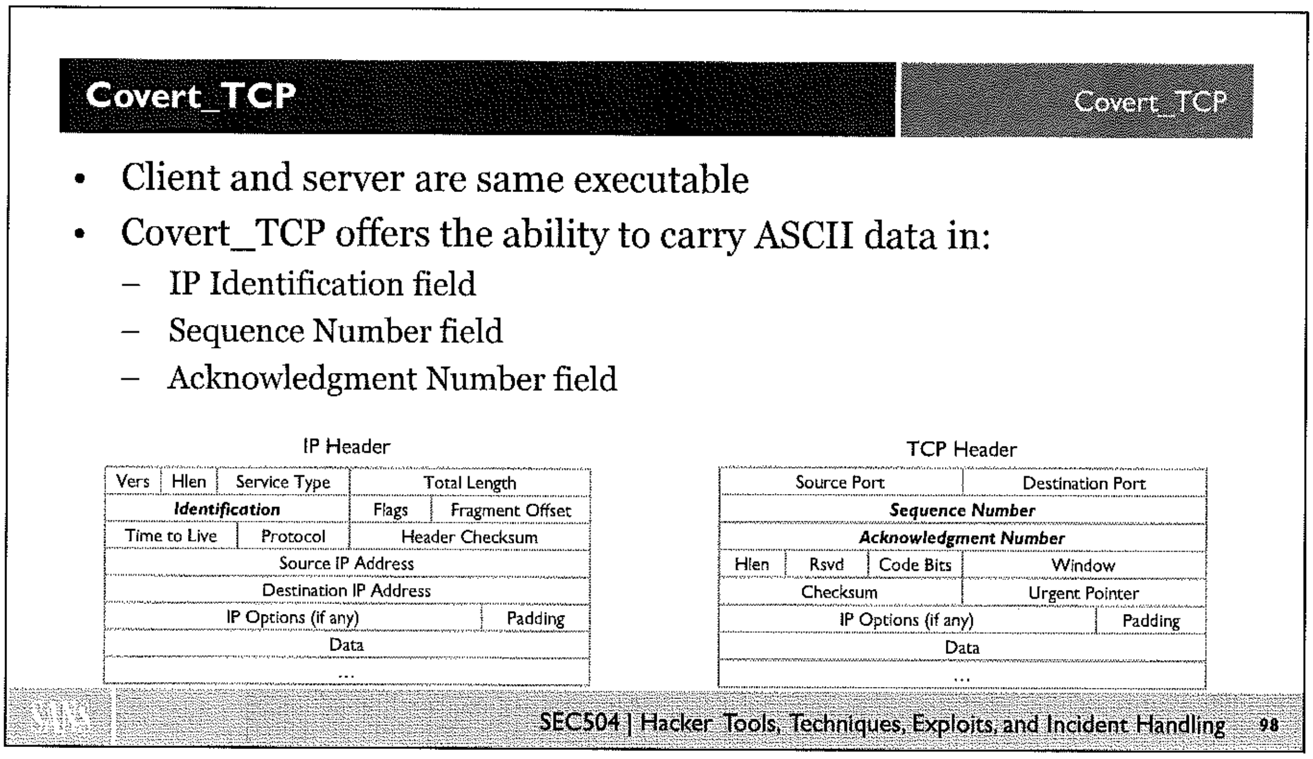
- Instead of tunneling data across other protocols (like HTTP or ICMP)...
- ... Why not just create a covert channel using extra space in the TCP or IP header?
- Covert_TCP is one tool that implements a covert channel using either the TCP or IP header
  - Written by Craig H. Rowland
  - Designed to transfer files
  - Remarkably effective technique

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 97

Although covert channels created by embedding one protocol entirely in a different protocol can be quite effective, covert channels can also be constructed by inserting data into unused or misused fields of protocol headers themselves. The TCP/IP protocol suite is particularly useful in carrying covert channels. Many of the fields in the TCP and IP headers have vast openings through which data can be sent.

One particularly interesting tool that illustrates exploiting TCP/IP headers to create covert channels is called "Covert_TCP." Written by Craig H. Rowland and included as part of his paper, "Covert Channels in the TCP/IP Protocol Suite," Covert_TCP shows how data can be secretly carried in TCP/IP headers by implementing a simple file transfer routine using the technique. Even though the tool was introduced many years ago, it illustrates important concepts in implementing covert channels.

Covert_TCP itself transfers only ASCII files between systems. However, the same concepts can be used to transport commands for a backdoor shell, or for any other movement of data across the network.



Covert_TCP

Covert_TCP

- Client and server are same executable
- Covert_TCP offers the ability to carry ASCII data in:
  - IP Identification field
  - Sequence Number field
  - Acknowledgment Number field

IP Header

Vers	Hlen	Service Type	Total Length
Identification		Flags	Fragment Offset
Time to Live	Protocol		Header Checksum
Source IP Address			
Destination IP Address			
IP Options (if any)			Padding
Data			
...			

TCP Header

Source Port			Destination Port
Sequence Number			
Acknowledgment Number			
Hlen	Rsvd	Code Bits	Window
Checksum			Urgent Pointer
IP Options (if any)			Padding
Data			
...			

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 98

Covert_TCP allows for transmitting information by entering ASCII data in the following TCP/IP header fields:

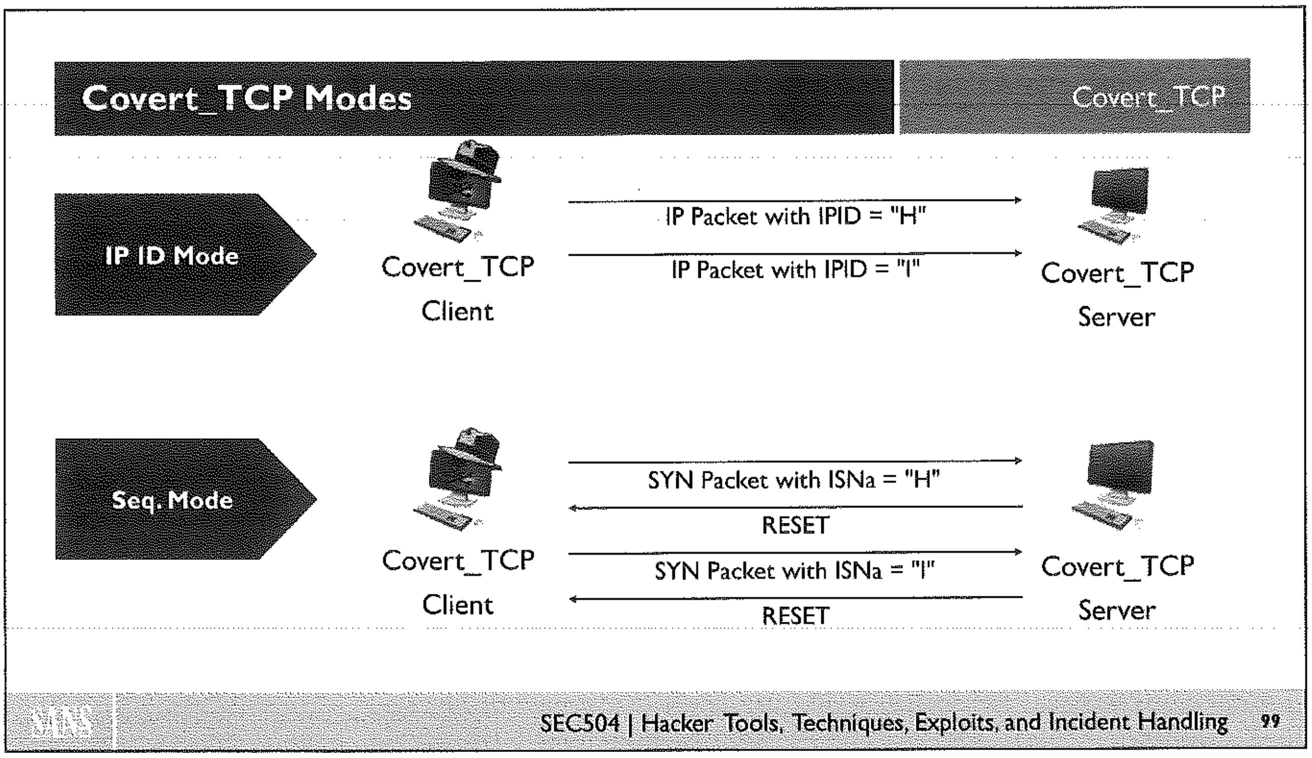
- IP identification
- TCP initial sequence number
- TCP acknowledgment number

The *IP identification* field is sometimes used in conjunction with packet fragmentation. When a system legitimately fragments a packet, it uses the *IP identification* field to create a unique number to differentiate between different streams of fragmented packets.

The *sequence number* field and the *acknowledgment number* field are used by TCP to order the packets as they get transmitted across the network.

An attacker can load information into these fields and use it to transmit the data across the network using covert TCP.

Of course, other components of the TCP and IP headers (or even UDP headers for that matter) could be used to transmit data, such as the *reserved, window, code bits, options,* or *padding fields*, but only three options are supported by Covert_TCP. These options were selected because they are often left unaltered as packets traverse a network. Even though only three different fields are supported in the tool, Covert_TCP is still remarkably effective in creating a covert channel.

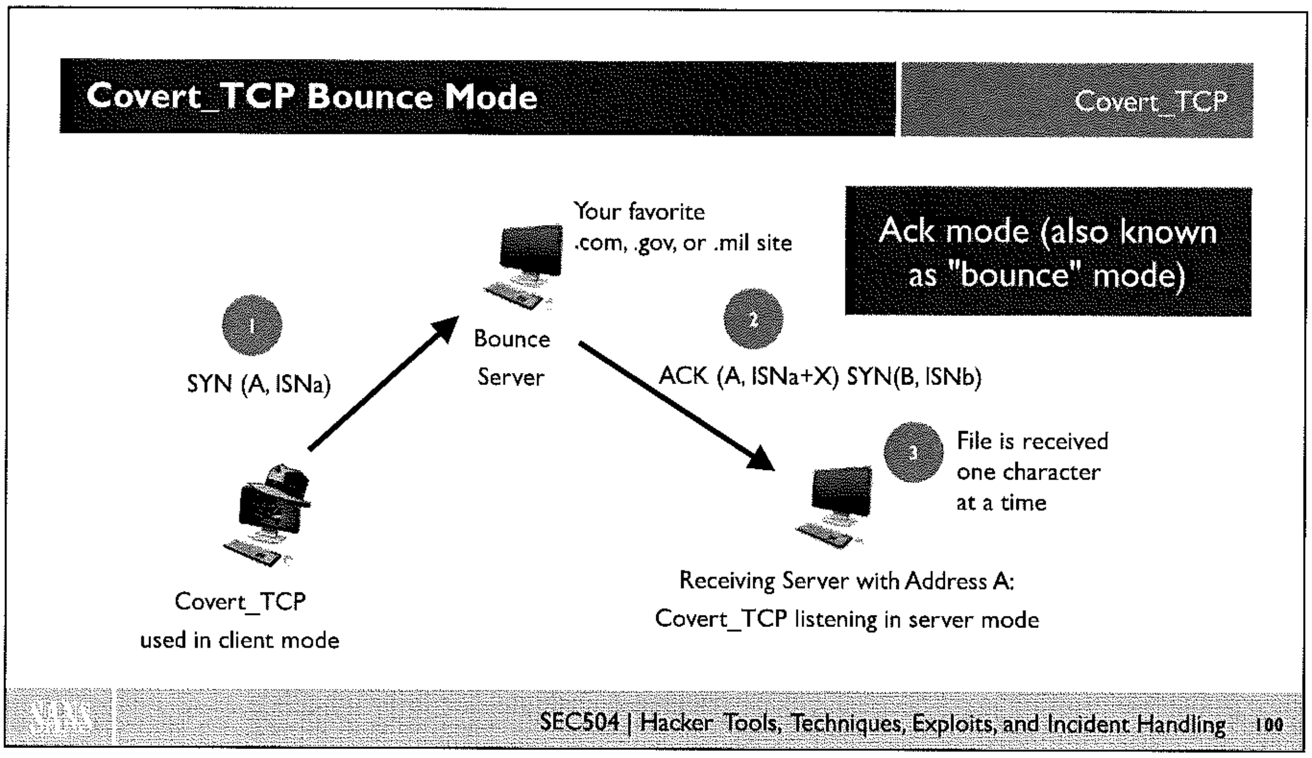


Covert_TCP Modes

IP ID Mode
Covert_TCP Client
IP Packet with IPID = "H"
IP Packet with IPID = "I"
Covert_TCP Server
Seq. Mode
Covert_TCP Client
SYN Packet with ISNa = "H"
RESET
SYN Packet with ISNa = "I"
RESET
Covert_TCP Server

The IP Identification mode is quite simple. ASCII data is simply dropped into that field of the IP header at the client and extracted at the server. A single character is carried in each packet. The tool uses the IP field in the IP header of a packet that is carrying a TCP segment so that it can be directed from a given TCP source port to a given TCP destination port.

The TCP initial sequence number mode is somewhat more complex. The first part of the TCP three-way handshake (the initial "SYN" packet) carries an initial sequence number (ISNA) set to represent the ASCII value of the first character in the file to be covertly transferred. The Covert_TCP server sends back a RESET packet because the intent of the communication is to deliver the character in the sequence number field, not to establish a connection. The client then sends another session initiation (again, the first part of the three-way handshake), containing another character in the ISNA field. Again, the server sends a RESET and the three-way handshake is not completed. Although not terribly efficient in transferring data, this Covert_TCP mode is still quite useful.



Covert_TCP Bounce Mode

Covert_TCP

Your favorite .com, .gov, or .mil site
Ack mode (also known as "bounce" mode)
1
SYN (A, ISNa)
Bounce Server
2
ACK (A, ISNa+X) SYN(B, ISNb)
3
File is received one character at a time
Receiving Server with Address A: Covert_TCP listening in server mode
Covert_TCP used in client mode

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 100

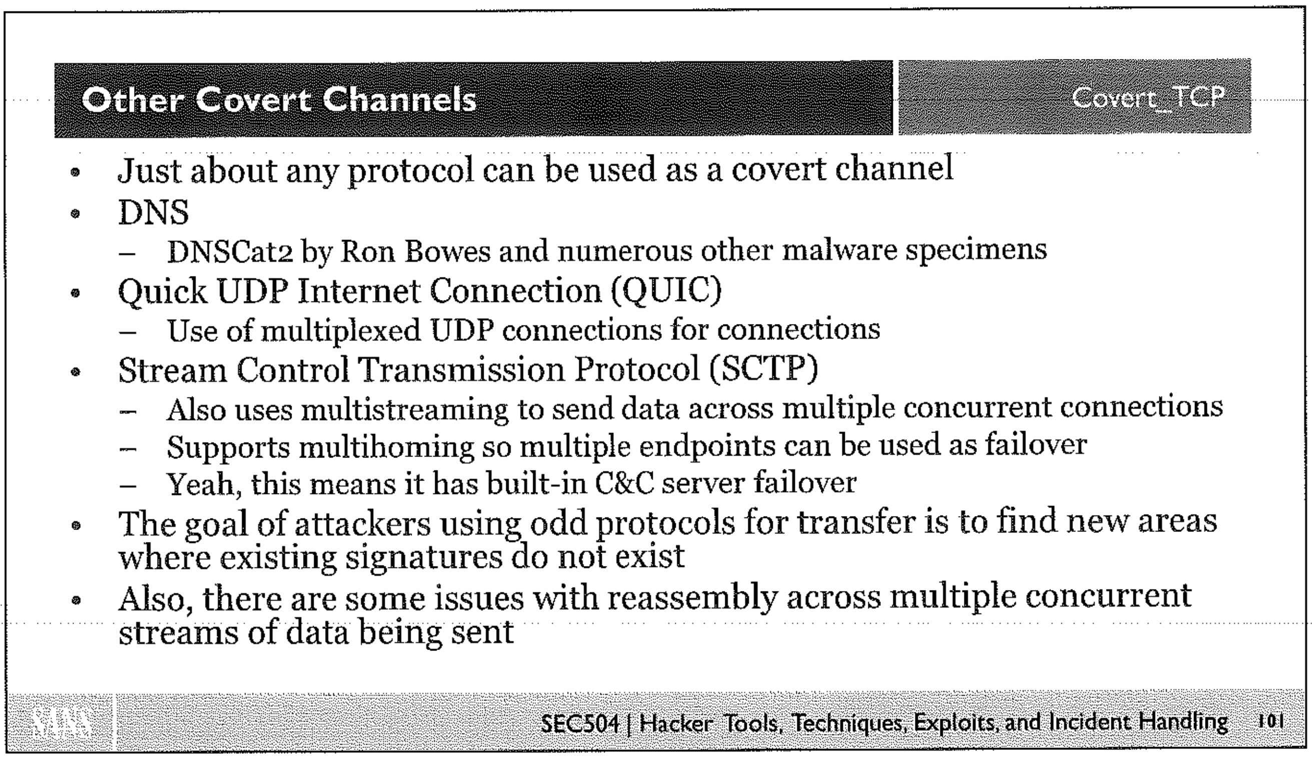
The most complex mode of operation for Covert_TCP is using the TCP acknowledgment sequence number, which applies in a so-called "bounce" operation. For scenarios where the Acknowledgment mode is used, three systems are involved: The server (receiver of the file), the client (the sender of the file), and the bounce server. In this mode, the attacker essentially sends data from the client, bouncing it off the bounce server using spoofing techniques, thereby transmitting it to the receiving server.

Step 0: The attacker establishes a Covert_TCP server on the receiving server, putting it in "ack" mode. The attacker selects a bounce server, which can be any accessible machine on the internet, potentially a high-profile internet commerce website, the DNS server of your favorite news source, a mail server from a university, or your friendly neighborhood three-letter government agency. No attacker software is required on the bounce server! All it needs is a TCP/IP stack and network connectivity. The attacker will send the file over a covert channel from the client system to the receiving system via the bounce server.

Step 1: The client generates TCP SYN packets with a spoofed source address of the receiving server and a destination address of the bounce server. The Initial Sequence Number of these packets (ISN_A) contains a representation of the ASCII character that needs to be transmitted. The packet is sent to the bounce server.

Step 2: The bounce server receives the packet. If the destination port (which is configurable by the attacker) on the bounce server is open, the bounce server will send a SYN/ACK response, thereby completing the second part of the three-way handshake. If the destination port on the bounce server is closed, the bounce server might send a RESET message. Regardless of whether the port is open or closed, the bounce server will send its response (a SYN/ACK or a RESET) to the apparent source of the message: The *receiving* server. That is how the bounce occurs—the client spoofs the address of the receiving server, duping the bounce server to forward the message. Of course, the SYN/ACK or RESET will have an ACK sequence number value incremented one more than the initial SYN. Based on this ACK number value, the receiver can determine the character that was sent.

Step 3: The receiving server gets the SYN/ACK or RESET, recovers the character from the sequence number field, and waits for more. The data is gathered from the sequence numbers and is written to a local file.

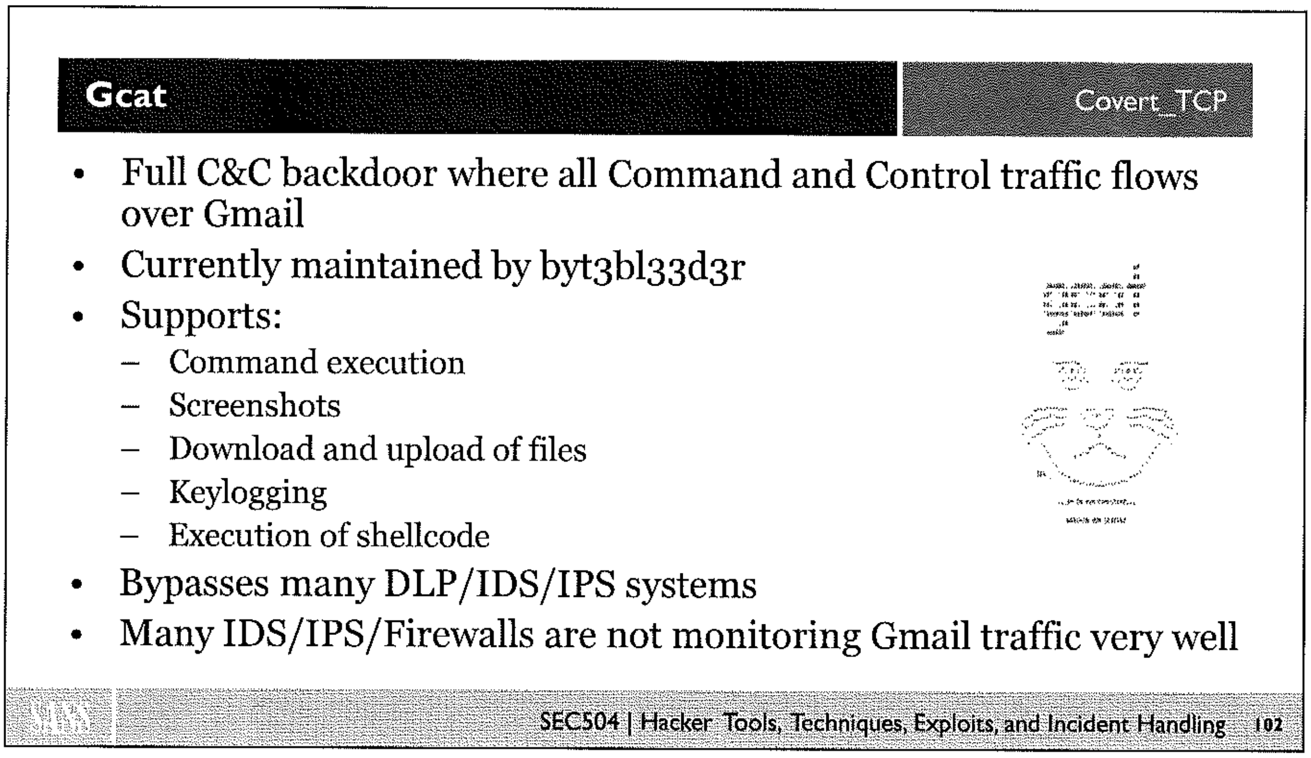


Other Covert Channels

- Just about any protocol can be used as a covert channel
- DNS
  - DNSCat2 by Ron Bowes and numerous other malware specimens
- Quick UDP Internet Connection (QUIC)
  - Use of multiplexed UDP connections for connections
- Stream Control Transmission Protocol (SCTP)
  - Also uses multistreaming to send data across multiple concurrent connections
  - Supports multihoming so multiple endpoints can be used as failover
  - Yeah, this means it has built-in C&C server failover
- The goal of attackers using odd protocols for transfer is to find new areas where existing signatures do not exist
- Also, there are some issues with reassembly across multiple concurrent streams of data being sent

One of our favorite protocols to use for Command and Control (C&C) for our assessments is DNS. There are a number of reasons for this. First, it is generally allowed outbound when internet whitelisting is in play. Second, it is very hard for IDS/IPS/Firewall vendors to create solid signatures for it. This is because it can be difficult to state exactly what should be in a DNS packet. Domain names are shifty and can be all over the place statically when taking into account multiple languages.

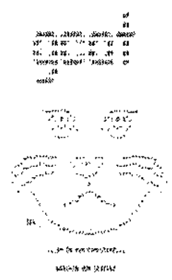
Although we in the security industry are having some issues with detecting Command and Control (C&C) traffic in existing protocols like DNS, attackers are actively using and abusing cutting-edge protocols like QUIC and SCTP. Protocols like these have a lot of advantages for attackers because they are multistreaming/multiplexed protocols. This means it is not as simple as reassembling a simple TCP stream between two hosts and two ports. Rather, the data can be sent across multiple ports in parallel. However, signatures eventually will be robust and written. The real advantage of these protocols is that they are relatively and/or severely underutilized. Because of these two issues, signatures can be slow to materialize because there are often more pressing signatures for other malware specimens that need to be written. Simply take these protocols as examples as to what attackers can do. Once these protocols are well defined and monitored by the security community, attackers will move on to the next new set of protocols.



Gcat

Covert_TCP

- Full C&C backdoor where all Command and Control traffic flows over Gmail
- Currently maintained by byt3bl33d3r
- Supports:
  - Command execution
  - Screenshots
  - Download and upload of files
  - Keylogging
  - Execution of shellcode
- Bypasses many DLP/IDS/IPS systems
- Many IDS/IPS/Firewalls are not monitoring Gmail traffic very well

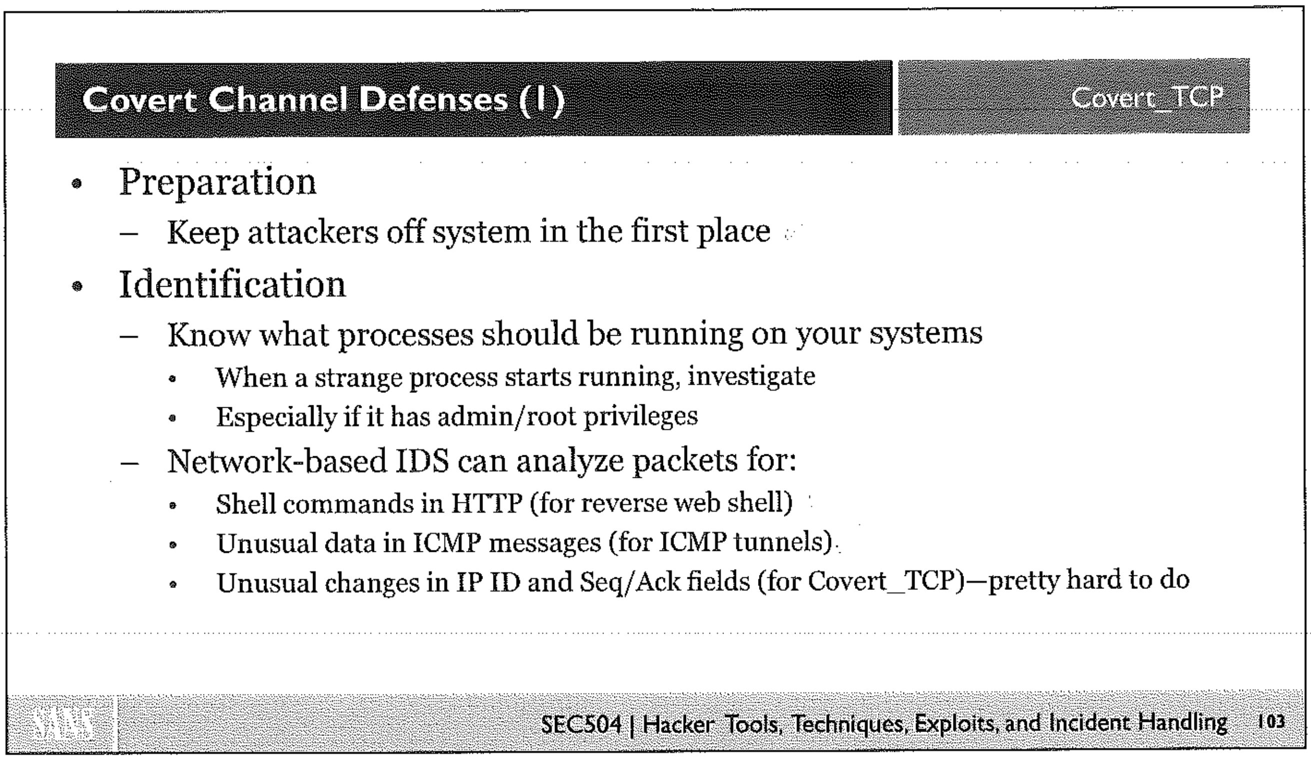


SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 102

Another tool we created at BHIS for testing and evaluating DLP and advanced firewall products is Gcat. Initially, it was a simple proof of concept we used to see whether many of the modern security products today were actively monitoring Gmail traffic. We quickly found out they are not, and this makes sense. How many times a day do you google for specific operating system commands? How many times do you email your friends and co-workers a command or the results of a command? Gcat was designed to take advantage of this blind spot in many organizations.

Currently, it is being maintained by byt3bl33d3r, who has added an impressive array of new features, and it can be found at:

https://github.com/byt3bl33d3r/gcat

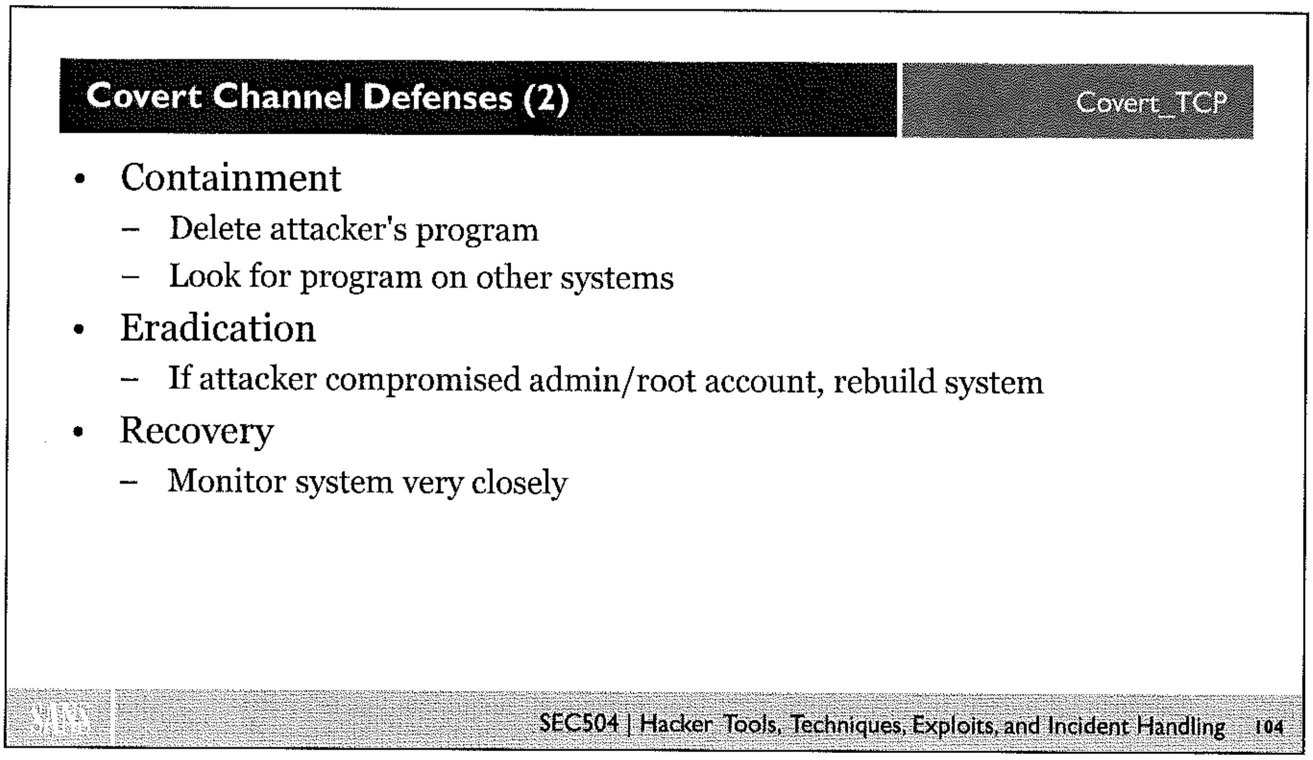


Covert Channel Defenses (1)

Covert_TCP

- Preparation
  - Keep attackers off system in the first place
- Identification
  - Know what processes should be running on your systems
    - When a strange process starts running, investigate
    - Especially if it has admin/root privileges
  - Network-based IDS can analyze packets for:
    - Shell commands in HTTP (for reverse web shell)
    - Unusual data in ICMP messages (for ICMP tunnels)
    - Unusual changes in IP ID and Seq/Ack fields (for Covert_TCP)—pretty hard to do

The defenses for Covert_TCP are the same as what we saw for defending against reverse web shell and Ptunnel. Keep the attackers off your systems by applying the principle of least privileges and stopping unknown processes from running on your machines. Also, employ network-based intrusion detection systems to look for anomalous behavior in the traffic.



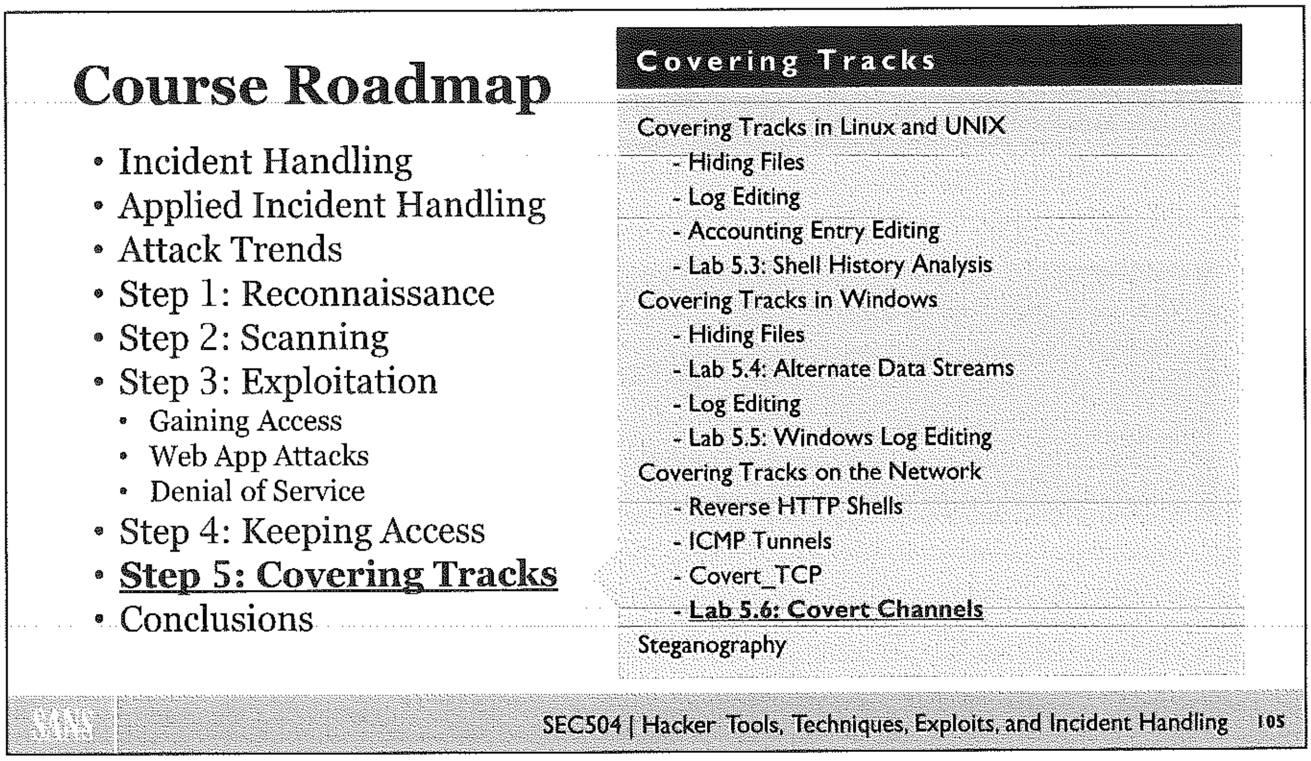
Covert Channel Defenses (2)

Covert_TCP

- Containment
  - Delete attacker's program
  - Look for program on other systems
- Eradication
  - If attacker compromised admin/root account, rebuild system
- Recovery
  - Monitor system very closely

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 104

If an attacker is using covert channels on your machines, you should scour other related systems looking for the same program. The system should be rebuilt if the attacker compromised root or admin-level accounts on the box.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

Covering Tracks

Covering Tracks in Linux and UNIX
- Hiding Files
- Log Editing
- Accounting Entry Editing
- Lab 5.3: Shell History Analysis

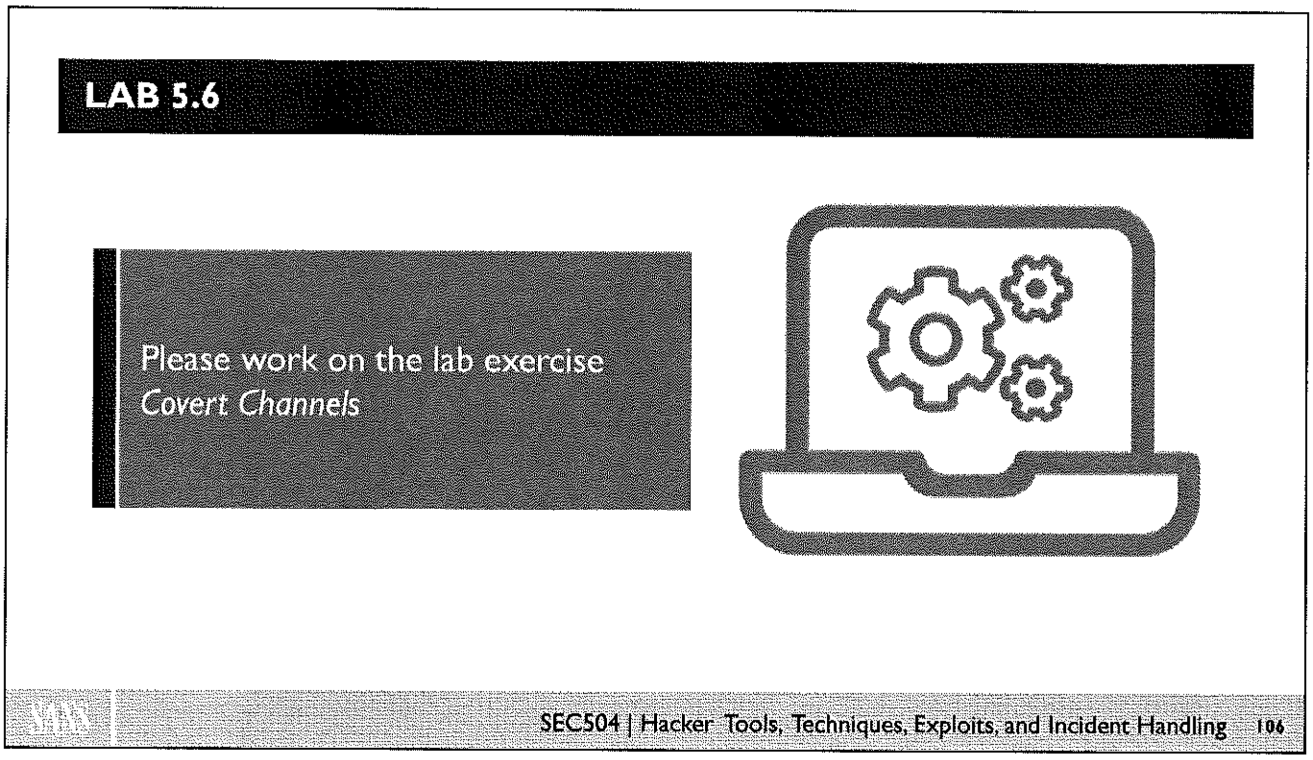
Covering Tracks in Windows
- Hiding Files
- Lab 5.4: Alternate Data Streams
- Log Editing
- Lab 5.5: Windows Log Editing

Covering Tracks on the Network
- Reverse HTTP Shells
- ICMP Tunnels
- Covert_TCP
- **Lab 5.6: Covert Channels**

Steganography

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 105

Next we will work on a lab exercise that employs covert channels.

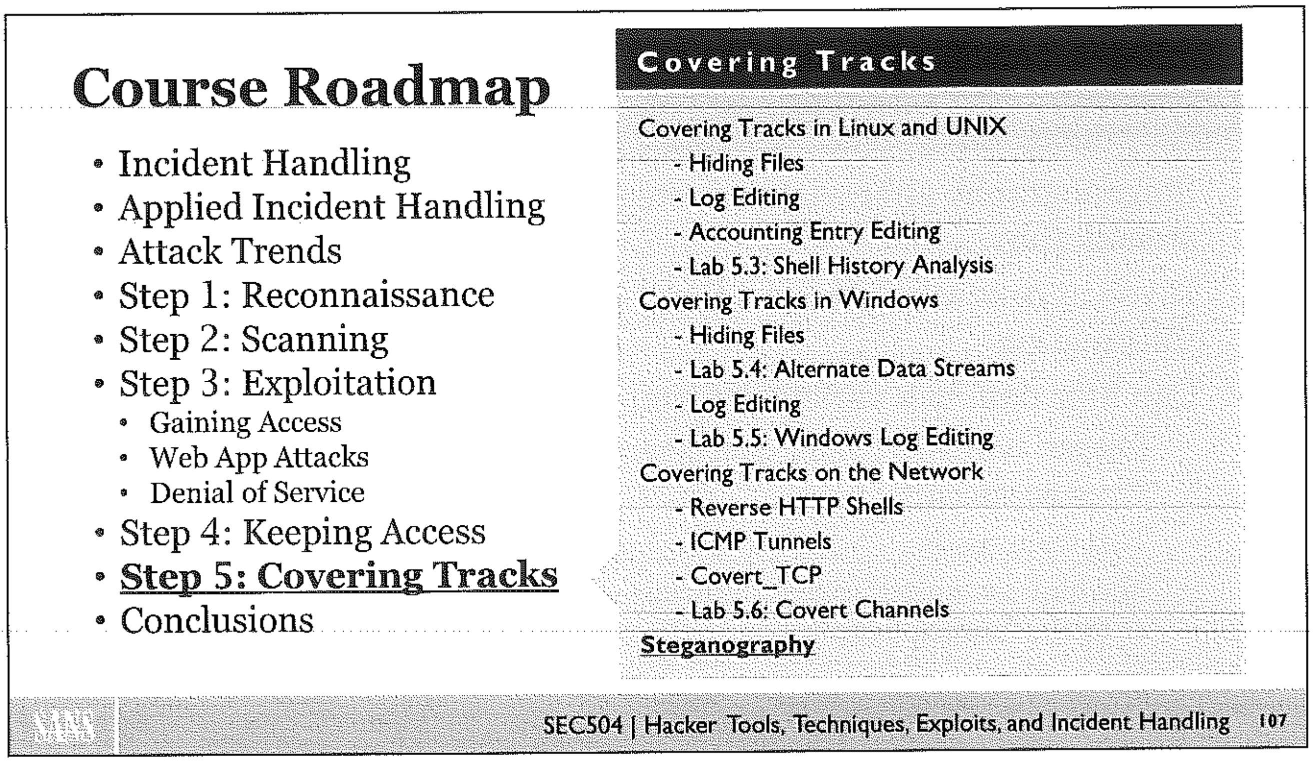


LAB 5.6

Please work on the lab exercise
Covert Channels

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 106

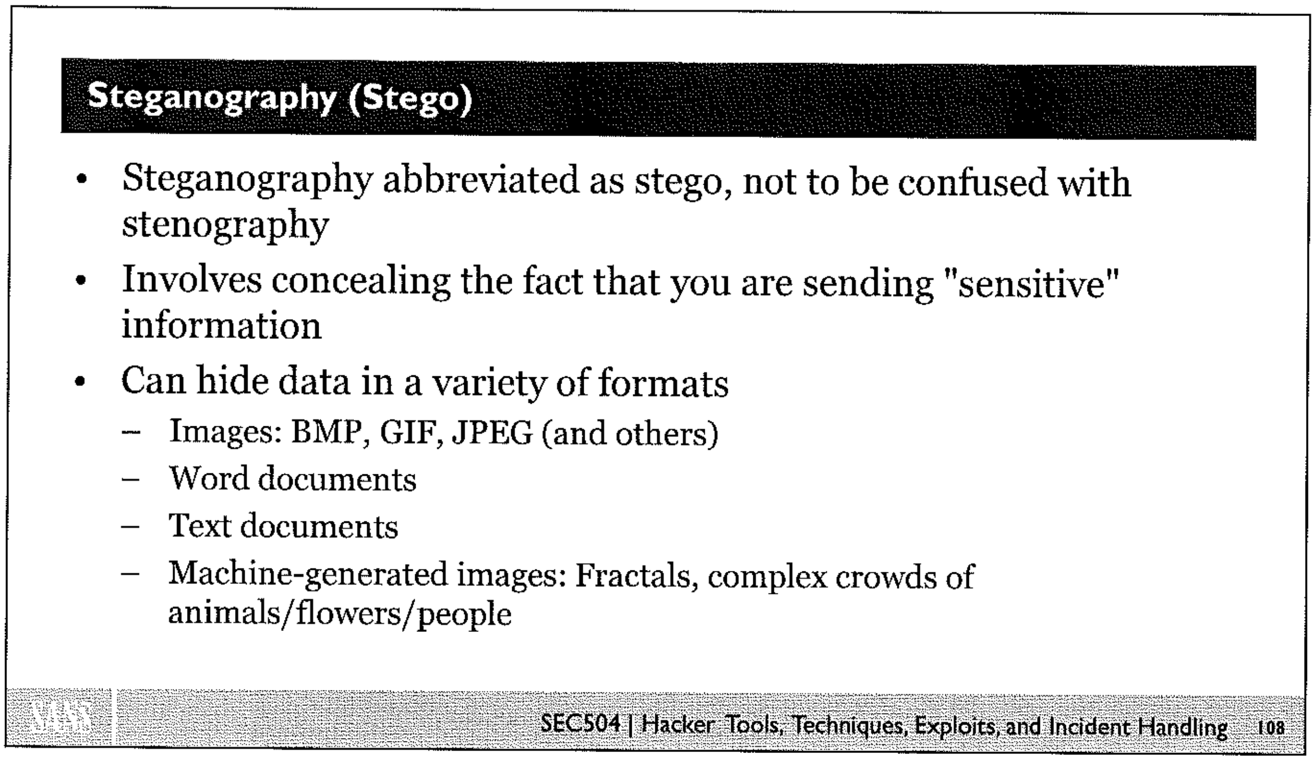
This page intentionally left blank.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- **Step 5: Covering Tracks**
- Conclusions

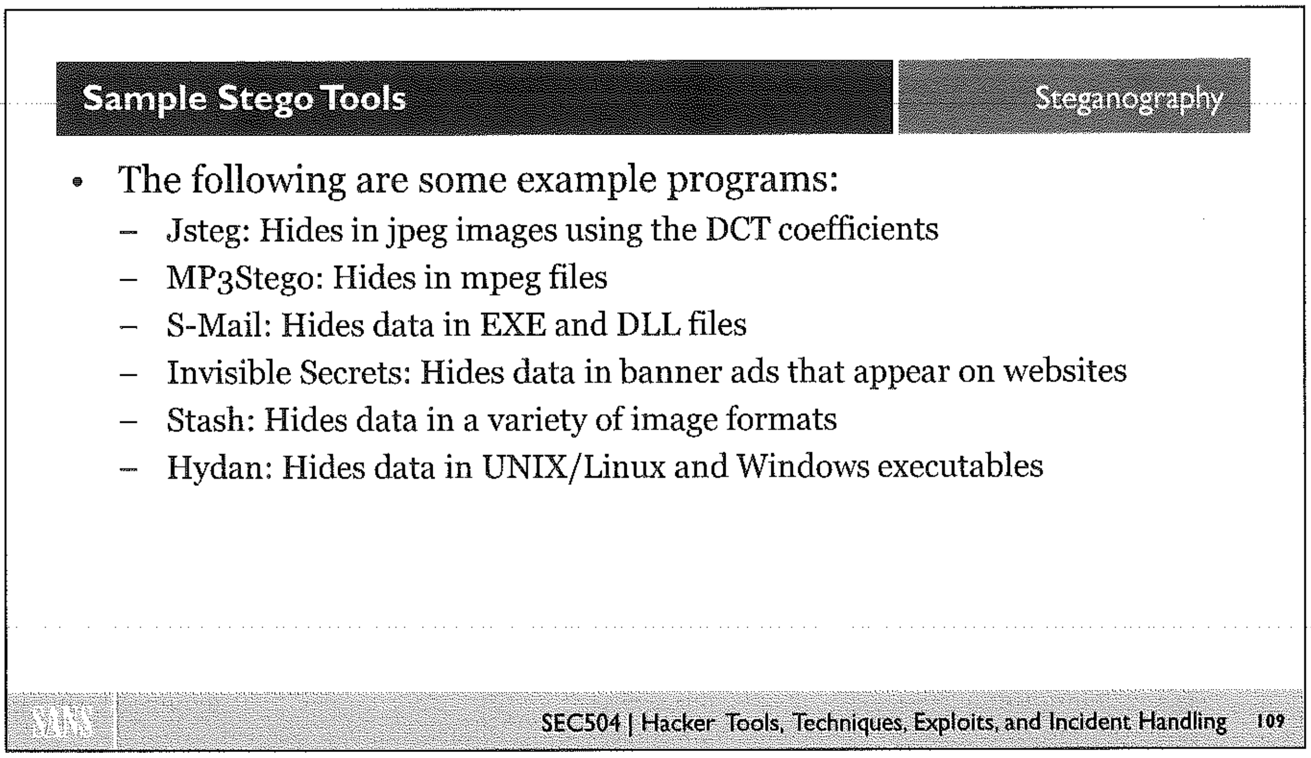
Now that we have analyzed how data can be hidden as it moves across the network, let's look at how data can be hidden in a static file on the filesystem.


Steganography (Stego)

- Steganography abbreviated as stego, not to be confused with stenography
- Involves concealing the fact that you are sending "sensitive" information
- Can hide data in a variety of formats
  - Images: BMP, GIF, JPEG (and others)
  - Word documents
  - Text documents
  - Machine-generated images: Fractals, complex crowds of animals/flowers/people

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 108

Steganography involves hiding data within a file, such as an image or sound file, so that the meaning of the message and the fact that a message is being sent is concealed. There are numerous methods allowing data to be embedded in a wide range of file types. Data can be hidden in images, such as BMP, GIF, or JPEG files; in Microsoft Word documents; or even in computer-generated pictures, such as fractals or complex crowds of animals/flowers/people.

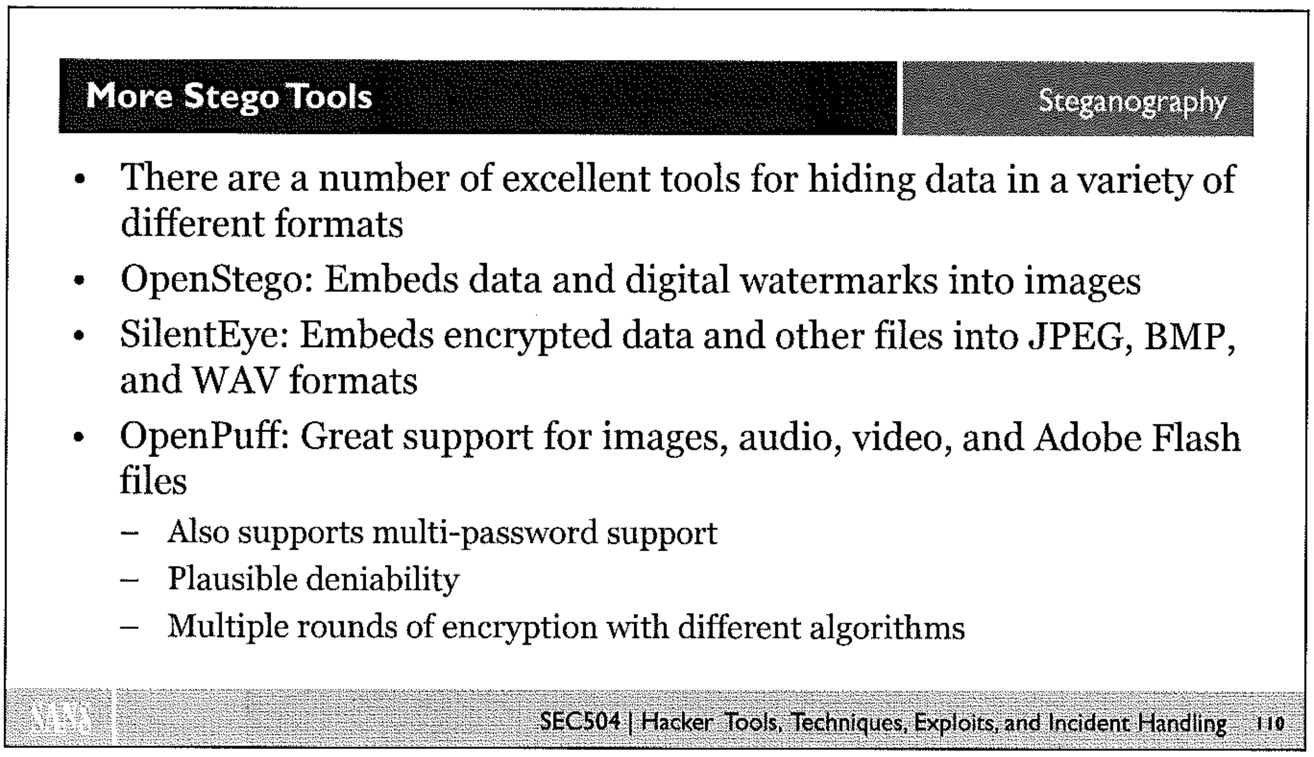

Sample Stego Tools

- The following are some example programs:
  - Jsteg: Hides in jpeg images using the DCT coefficients
  - MP3Stego: Hides in mpeg files
  - S-Mail: Hides data in EXE and DLL files
  - Invisible Secrets: Hides data in banner ads that appear on websites
  - Stash: Hides data in a variety of image formats
  - Hydan: Hides data in UNIX/Linux and Windows executables

To get an appreciation of the large number of stego tools available, let's look at a couple of good examples.

- Jsteg hides data in jpeg images by manipulating the Discrete Cosine Transform coefficients in the jpeg algorithm.
- MP3Stego hides data in mpeg audio files.
- S-Mail hides data in binary executable files, such as EXEs and DLLs.
- Invisible Secrets hides data in banner ads that appear on websites.
- Stash hides data in a variety of image formats.
- Hydan hides data in UNIX/Linux and Windows executables.

These are only a few examples of the many stego tools available.



More Stego Tools

Steganography

- There are a number of excellent tools for hiding data in a variety of different formats
- OpenStego: Embeds data and digital watermarks into images
- SilentEye: Embeds encrypted data and other files into JPEG, BMP, and WAV formats
- OpenPuff: Great support for images, audio, video, and Adobe Flash files
  - Also supports multi-password support
  - Plausible deniability
  - Multiple rounds of encryption with different algorithms

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 110

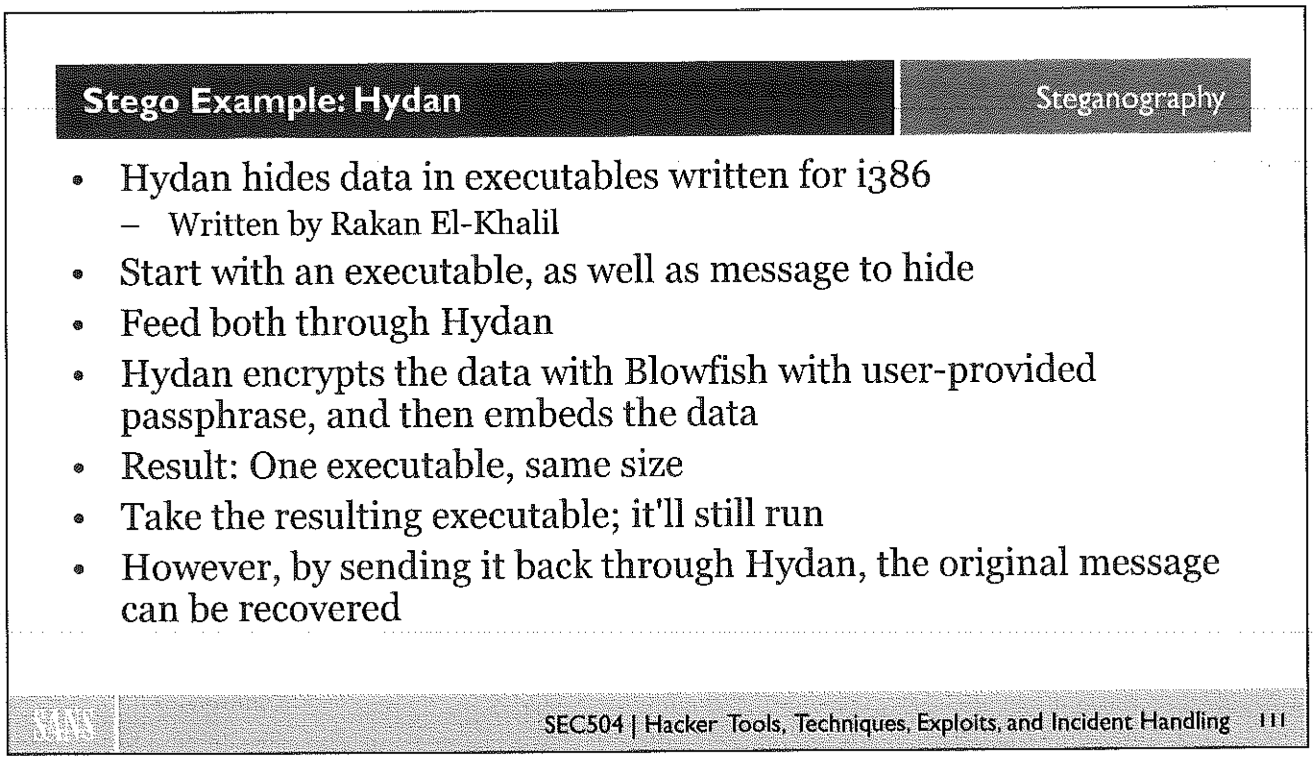
There is a wide variety of different stego tools available today. The following are some of our favorites.

OpenStego is a tool dedicated for embedding data and other files into images. It also has a very cool feature where you can use it to embed digital watermarking in images as well. It can be found at http://www.openstego.com/.

SilentEye is another tool that supports popular formats like JPEG, BMP, and WAVE. It also supports a wide variety of platforms such as Windows, Linux, and OS X.

Finally, we have OpenPuff. This outstanding tool supports far too many formats to be listed here. It also has other unique features such as multi-password support, plausible deniability, and multiple rounds of encryption with different algorithms. It can be found at https://embeddedsw.net/OpenPuff_Steganography_Home.html.

You can also find additional tools at http://stegano.net/tools.



Stego Example: Hydan

- Hydan hides data in executables written for i386
  - Written by Rakan El-Khalil
- Start with an executable, as well as message to hide
- Feed both through Hydan
- Hydan encrypts the data with Blowfish with user-provided passphrase, and then embeds the data
- Result: One executable, same size
- Take the resulting executable; it'll still run
- However, by sending it back through Hydan, the original message can be recovered

In early 2003, Rakan El-Khalil released the "Hydan" tool, available at http://www.crazyboy.com/hydan/. It hides data in executable programs written in the x86 instruction set.

To use the tool, start with an executable program, such as Microsoft Word. You also need a message to hide, such as a secret message, a picture, some other code, or anything.

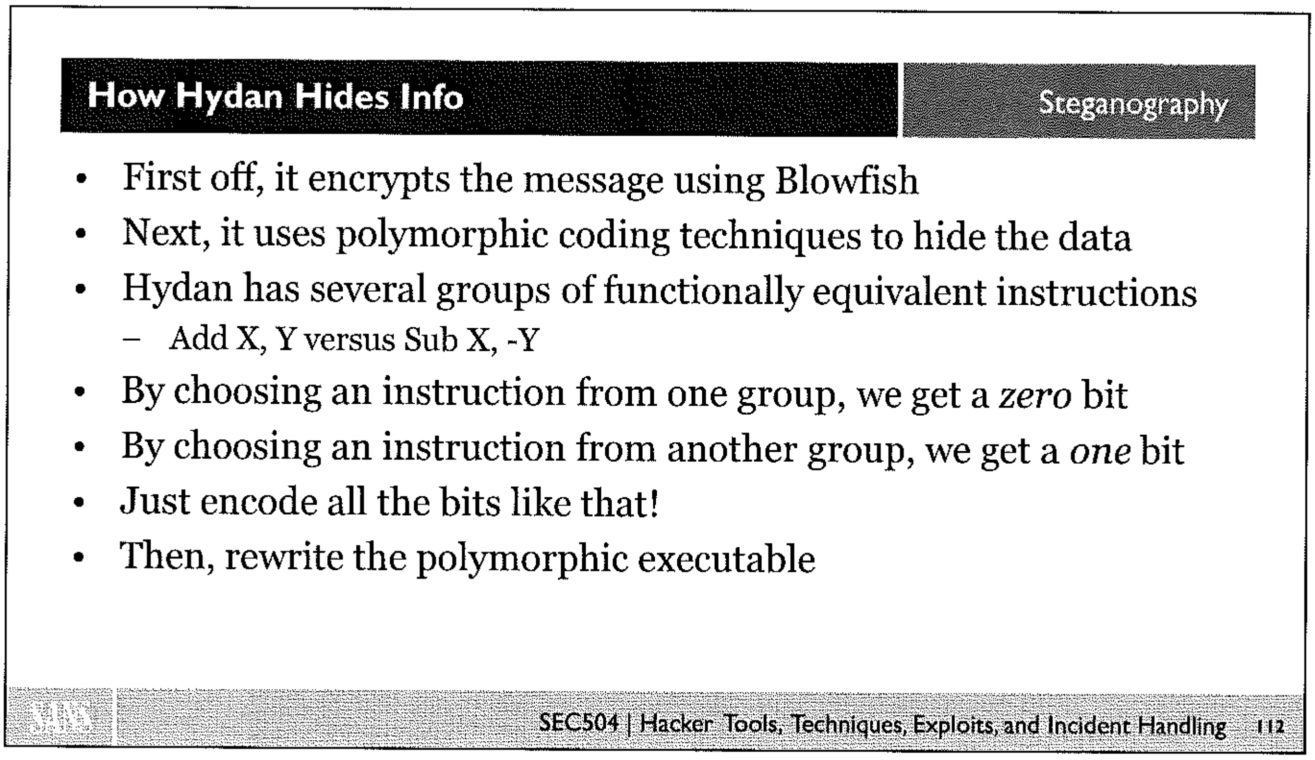
You feed the executable and message into Hydan. You also tell it the passphrase you want to use.

The tool first encrypts the message with the Blowfish encryption algorithm using your passphrase as a key.

It then embeds the encrypted message inside the executable program.

The result is a single executable that includes the hidden encrypted message. This executable is the same size as the original executable, and has the same functionality.

Most importantly, by using Hydan again, the original message can be retrieved from the resulting executable.



How Hydan Hides Info

Steganography

- First off, it encrypts the message using Blowfish
- Next, it uses polymorphic coding techniques to hide the data
- Hydan has several groups of functionally equivalent instructions
  - Add X, Y versus Sub X, -Y
- By choosing an instruction from one group, we get a *zero* bit
- By choosing an instruction from another group, we get a *one* bit
- Just encode all the bits like that!
- Then, rewrite the polymorphic executable

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 112

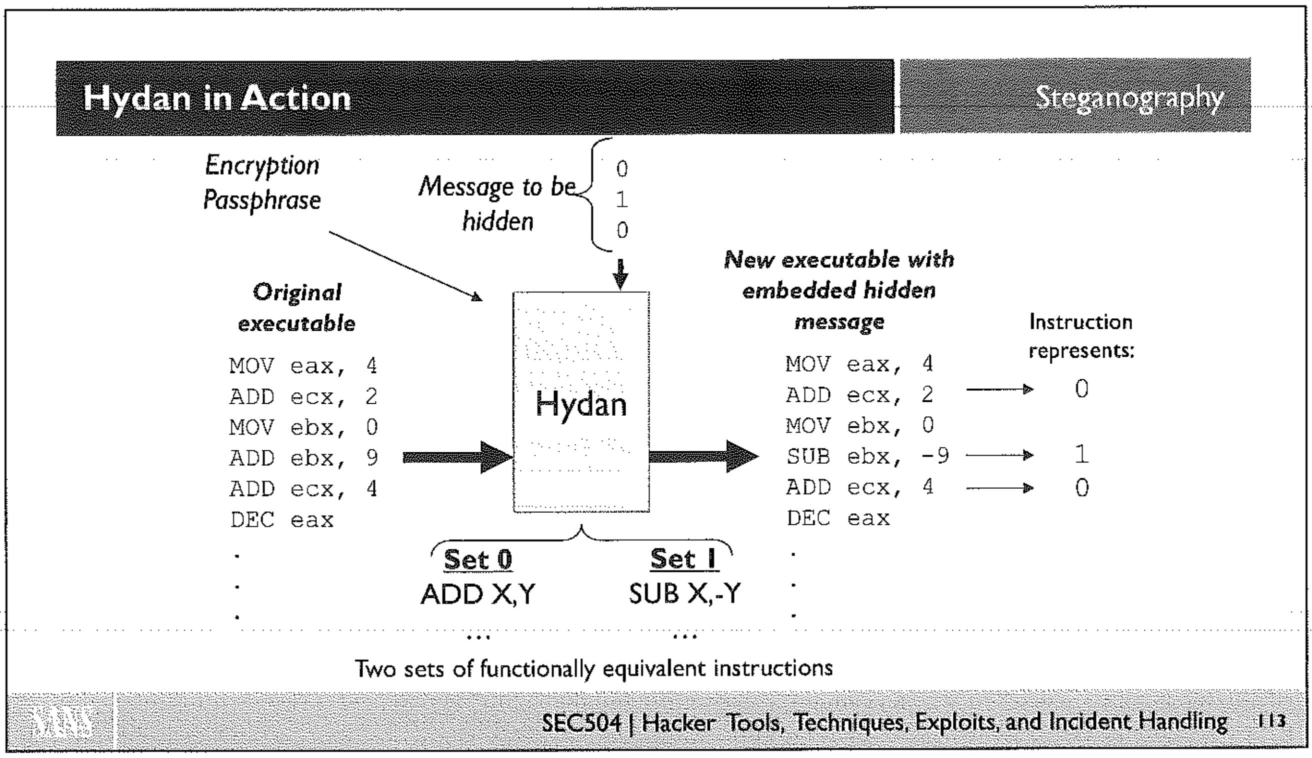
So, how does Hydan accomplish this little feat?

First off, it just encrypts the message to be hidden.

Then, it uses polymorphic coding techniques. Hydan defines different sets of CPU instructions that have the exact same function. For example, when you add two numbers, you can use the "add" or "subtract" instructions. You can add X and Y, or you can subtract negative Y from X. These have the same result.

Hydan takes the original executable and rebuilds it by choosing instructions from one group or the other group of functionally equivalent instructions. If a given bit to be hidden is a zero, we will choose from the first group of instructions. If the bit is a one, we will choose a functionally equivalent instruction from the other group of instructions.

Then, after the entire code is rebuilt with these instructions, the executable is rewritten dynamically to the hard drive. The resulting executable has the same size and the same function!

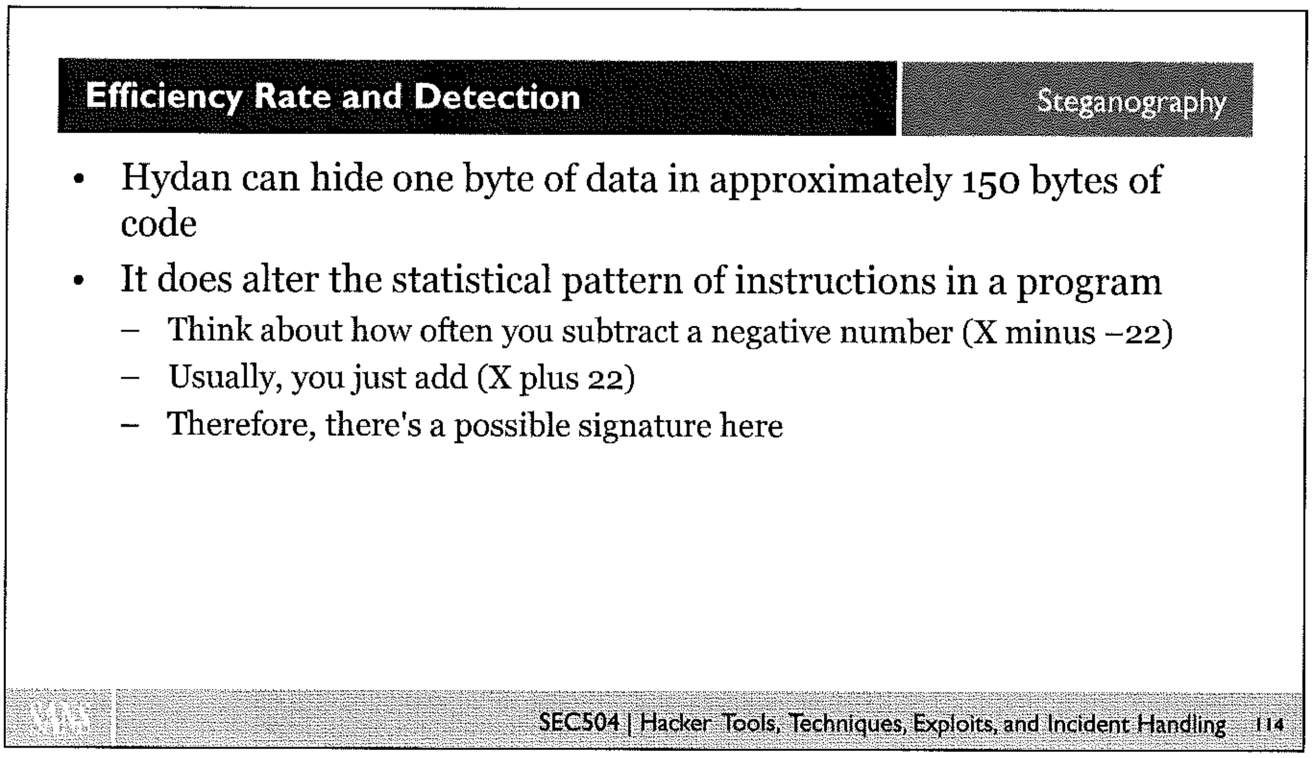


Hydan in Action

```
MOV eax, 4
ADD ecx, 2
MOV ebx, 0
ADD ebx, 9
ADD ecx, 4
DEC eax
.
.
.
```

```
MOV eax, 4
ADD ecx, 2
MOV ebx, 0
SUB ebx, -9
ADD ecx, 4
DEC eax
.
.
.
```

Hydan dynamically rebuilds the executable from the ground up, making substitutions of adds and subtracts to hide the necessary bits. The resulting executable's size is the same because ADD and SUB are the same size.



Efficiency Rate and Detection

Steganography

- Hydan can hide one byte of data in approximately 150 bytes of code
- It does alter the statistical pattern of instructions in a program
  - Think about how often you subtract a negative number (X minus −22)
  - Usually, you just add (X plus 22)
  - Therefore, there's a possible signature here

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 114

Hydan lets its user hide one byte of data in about 150 bytes of code. That's not as efficient as hiding data inside of pictures (which often get up to one byte hidden in 20 bytes of image). Still, it's not bad!

It's also important to note that Hydan does alter the statistical distribution of instructions used in the resulting executable. By creating a histogram showing how frequently various instructions are used, an investigator could determine that the program "just doesn't look right." Think of a histogram of English text: There are lots of "e"s and "t"s, but not many "q"s and "z"s. You could do a similar analysis with x86 instructions. If you see too many subtractions and not enough adds, things might be askew.

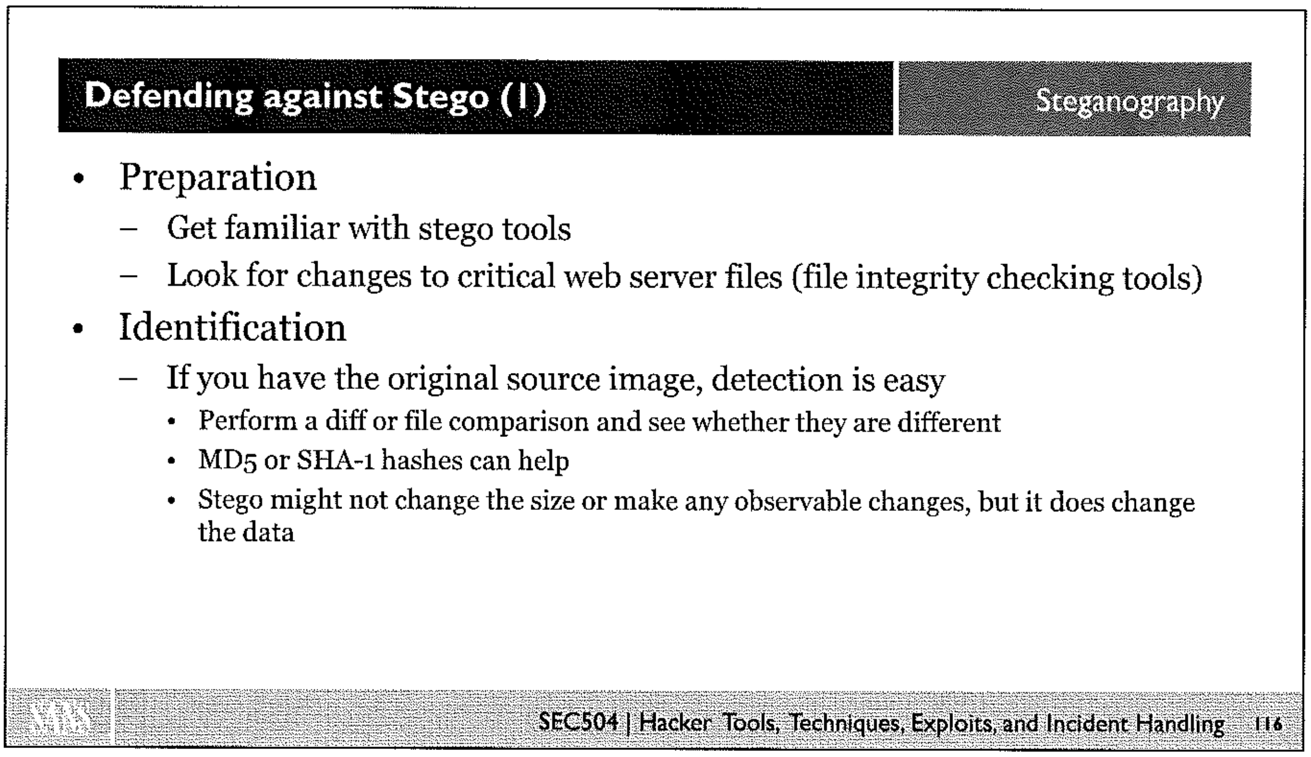
Detecting Stego

- StegExpose: Java utility to detect stego in lossless images where least significant bit (LSB) techniques are used
  - This stego is where the least significant bits, which determine color, are modified
  - This leads to a very slight (think imperceptible) change of color made to the original image
- Supports a number of different *detectors* or mathematical analysis techniques to detect stego
- For quick analysis, it can also use *cheap* or quick analysis methods to detect the presence of stego
- Has the ability to run on a large number of files very quickly

StegExpose is an outstanding tool that can detect stego in which least significant bit (LSB) techniques are used. Because the LSBs will have a minimal impact, we will change those bits for each pixel in the host image. Regardless of what the last 2 LSBs are, the human eye cannot tell the difference. If we take 10001100 and change it to 10001111 or 10001110 to hide two bits of data, it will appear to a human observer like the same color. In this way, the host image can be used to store hidden data, two bits at a time.

StegExpose supports a large number of detection techniques focused on detecting LSB modification. It can either run all of these techniques or just the "cheap" or fast techniques. One of the key features of this tool is the ability to pass the tool a directory, and it will perform its analysis on all the files and save the output to an .xls file.

It can be found at https://github.com/b3dk7/StegExpose.

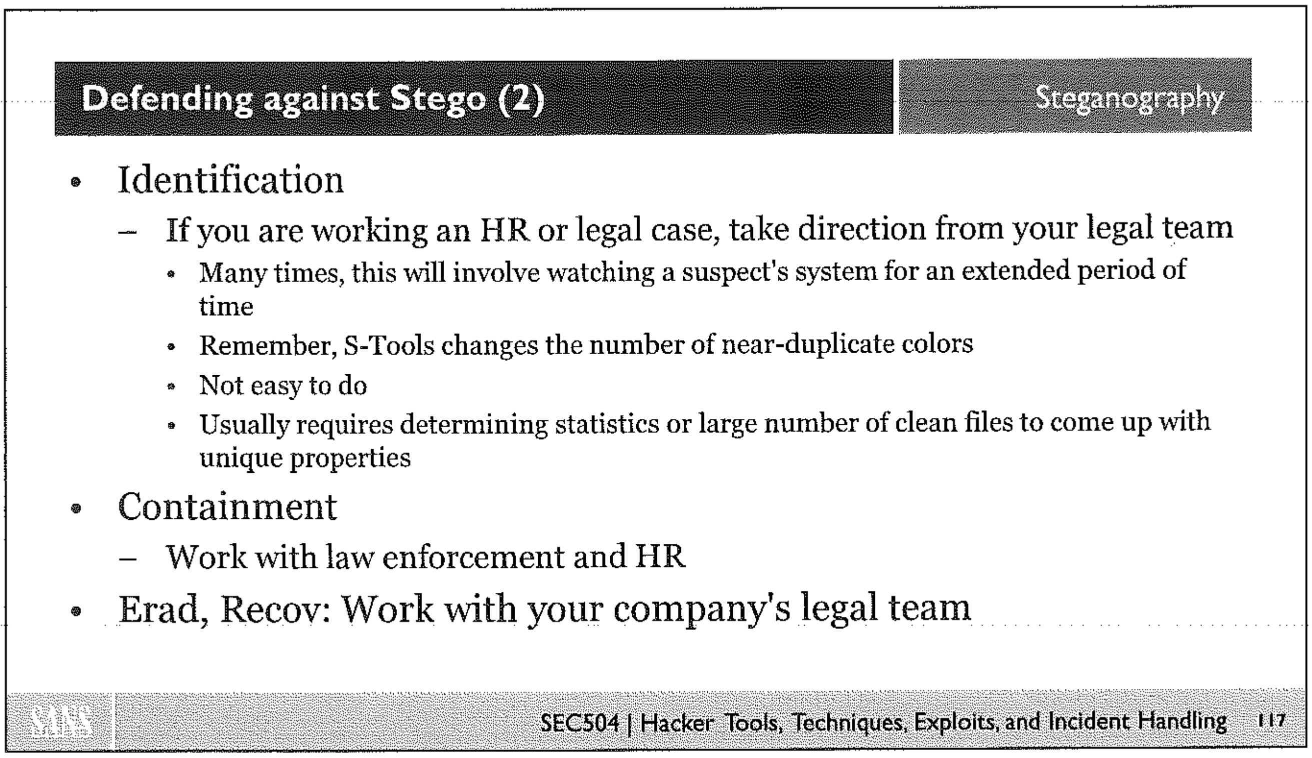

Defending against Stego (1)

Steganography

- Preparation
  - Get familiar with stego tools
  - Look for changes to critical web server files (file integrity checking tools)
- Identification
  - If you have the original source image, detection is easy
    - Perform a diff or file comparison and see whether they are different
    - MD5 or SHA-1 hashes can help
    - Stego might not change the size or make any observable changes, but it does change the data

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 116

Detecting the presence of hidden data is quite trivial if you have the original source image. You can simply compare the two files and see whether they are different using a simple diff program. In most cases, however, you will not have the source image and will not be able to do this.



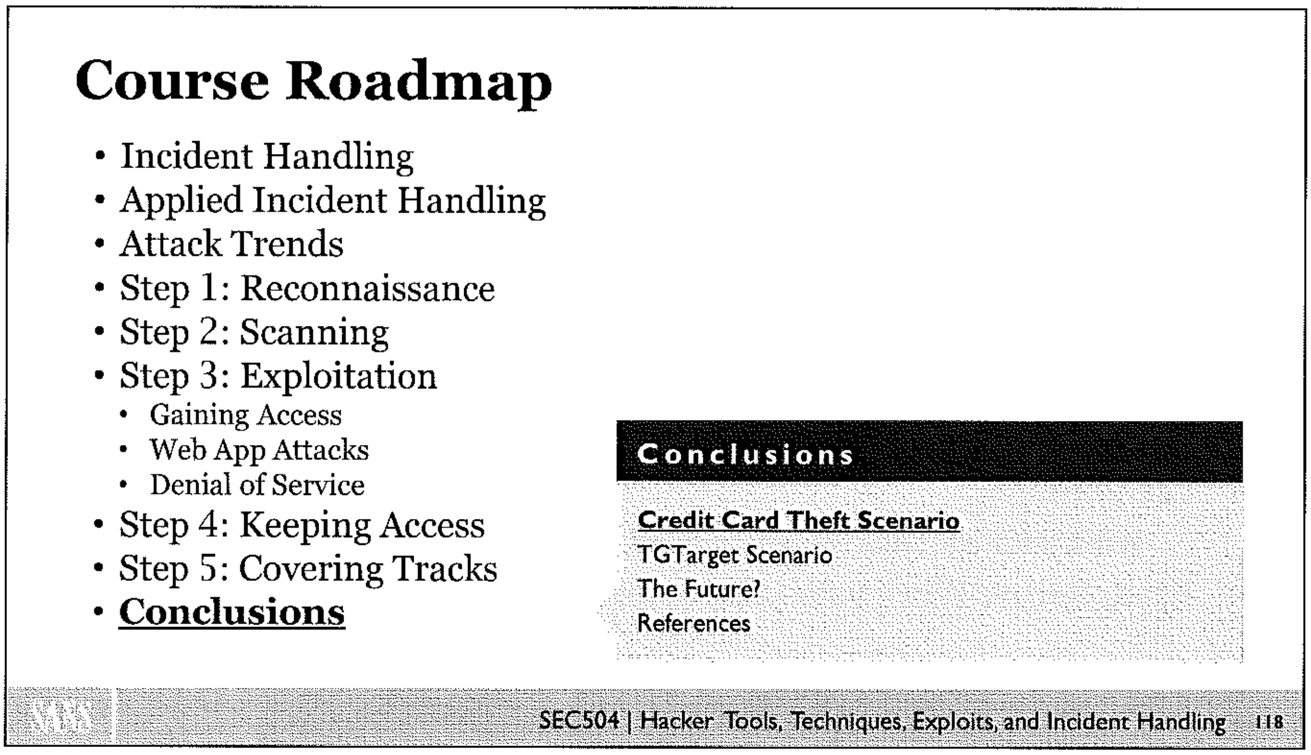
Defending against Stego (2)

Steganography

- Identification
  - If you are working an HR or legal case, take direction from your legal team
    - Many times, this will involve watching a suspect's system for an extended period of time
    - Remember, S-Tools changes the number of near-duplicate colors
    - Not easy to do
    - Usually requires determining statistics or large number of clean files to come up with unique properties
- Containment
  - Work with law enforcement and HR
- Erad, Recov: Work with your company's legal team

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 117

The biggest issue with working with cases where stego is in play is how many different parts of your organization will be involved. HR, legal, and/or possibly law enforcement will be giving direction and advice. This is because most stego cases will involve some sort of espionage or even possibly illegal images. Because of the possibility of having so many different goals and groups attempting to set direction, it is imperative that all requests pass through your organization's HR/legal department. This is because many stego cases will require monitoring the suspect system for an extended period of time.


Course Roadmap

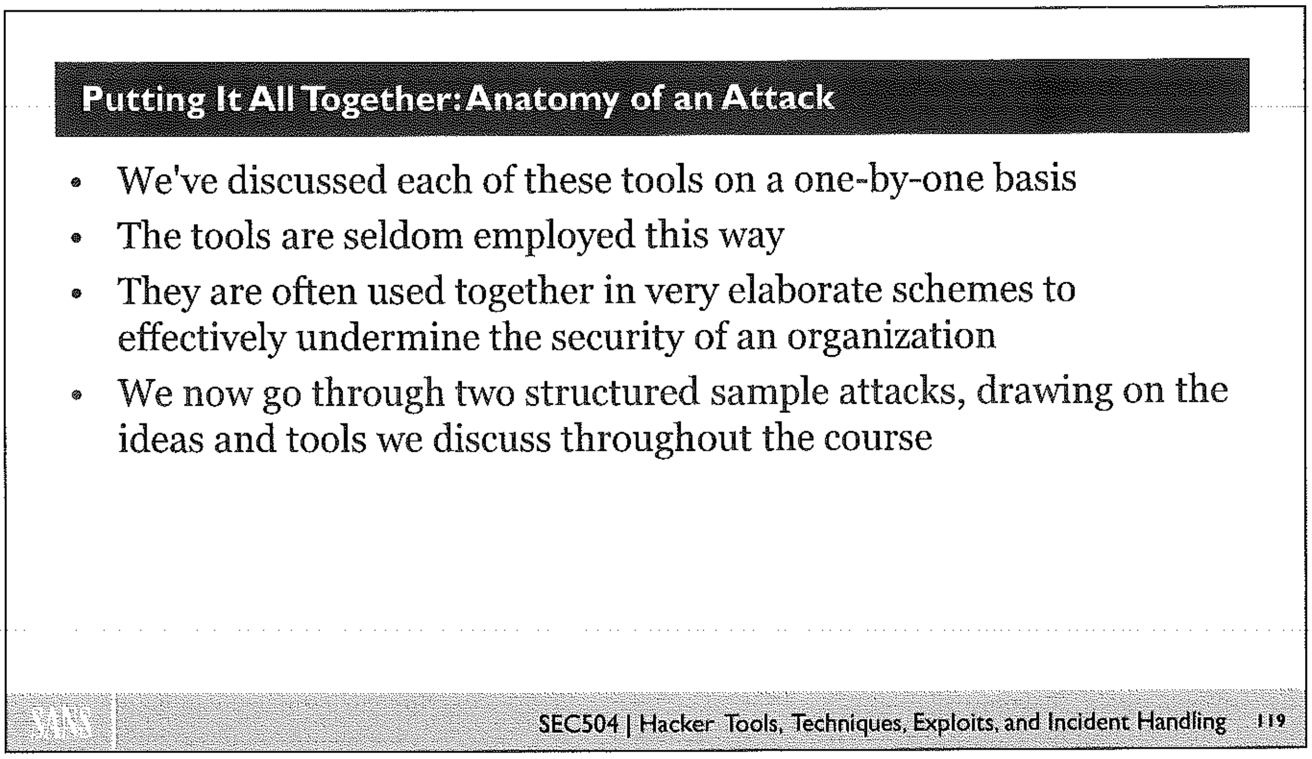
- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- Step 5: Covering Tracks
- **Conclusions**

Conclusions

Credit Card Theft Scenario
TGTarget Scenario
The Future?
References

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 118

Now that we have discussed the step-by-step approach to attacking, let's analyze how an attacker can put all the different techniques we've discussed together in some sample scenarios. In this first scenario, we'll examine a case where attackers used their access for credit card theft.

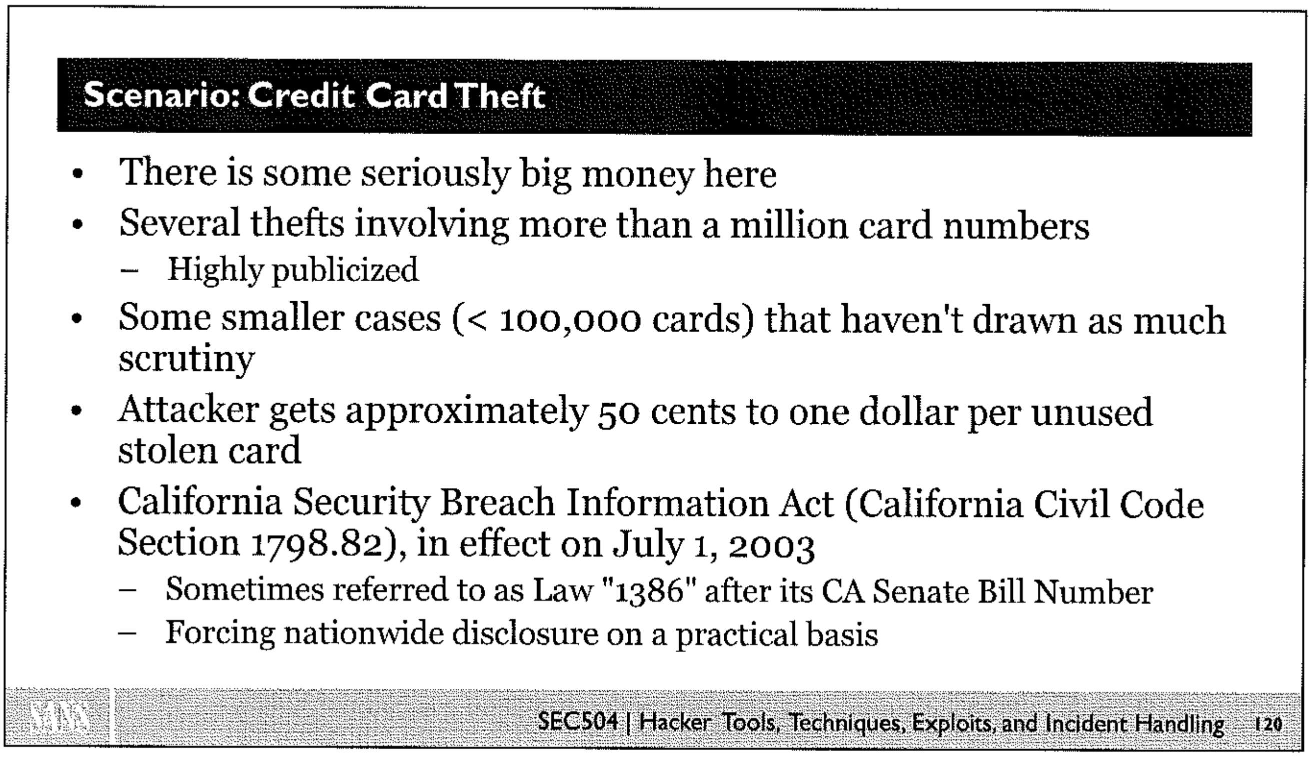


Putting It All Together: Anatomy of an Attack

- We've discussed each of these tools on a one-by-one basis
- The tools are seldom employed this way
- They are often used together in very elaborate schemes to effectively undermine the security of an organization
- We now go through two structured sample attacks, drawing on the ideas and tools we discuss throughout the course

Throughout this course, we discuss attack tools and how they work on a one-by-one basis. In the wild, however, these tools are almost always used in combination with each other. A clever attacker will assemble elaborate attacks using a variety of the tools and techniques we discuss together. You can think of the individual tools like Lego blocks. An attacker will build an entire structure in a unique way to create an elaborate scheme using the individual Lego blocks.

Note that we will cover two all-encompassing scenarios. An infinite number of other scenarios can be constructed. The following scenarios are quite common, however, and illustrate concepts that we discuss throughout the course.


Scenario: Credit Card Theft

- There is some seriously big money here
- Several thefts involving more than a million card numbers
  - Highly publicized
- Some smaller cases (< 100,000 cards) that haven't drawn as much scrutiny
- Attacker gets approximately 50 cents to one dollar per unused stolen card
- California Security Breach Information Act (California Civil Code Section 1798.82), in effect on July 1, 2003
  - Sometimes referred to as Law "1386" after its CA Senate Bill Number
  - Forcing nationwide disclosure on a practical basis

Credit card theft is on the rise, big time. We've seen a number of high-profile cases involving the theft of a million or more credit cards. There have been some smaller cases, with less public scrutiny, as well.

On the black market, an unused stolen credit card number typically sells for 50 cents to a dollar. Thus, with a heist of a million cards, the attacker could get upward of a million dollars.

When personally identifiable information for citizens of the state of California is exposed, those citizens must be informed as a matter of California law. In these credit card thefts, this law is sometimes used to force disclosure, but not just in the state of California. When a theft occurs, companies often begin to notify their California customers. Then, if it's a highly public exposure, various privacy advocates intervene to force the company to treat all of its customers in the same way, whether inside California or not. Thus, folks in other states are sometimes informed of the theft of their personal information based on the presence of the California law and some intense pressure.

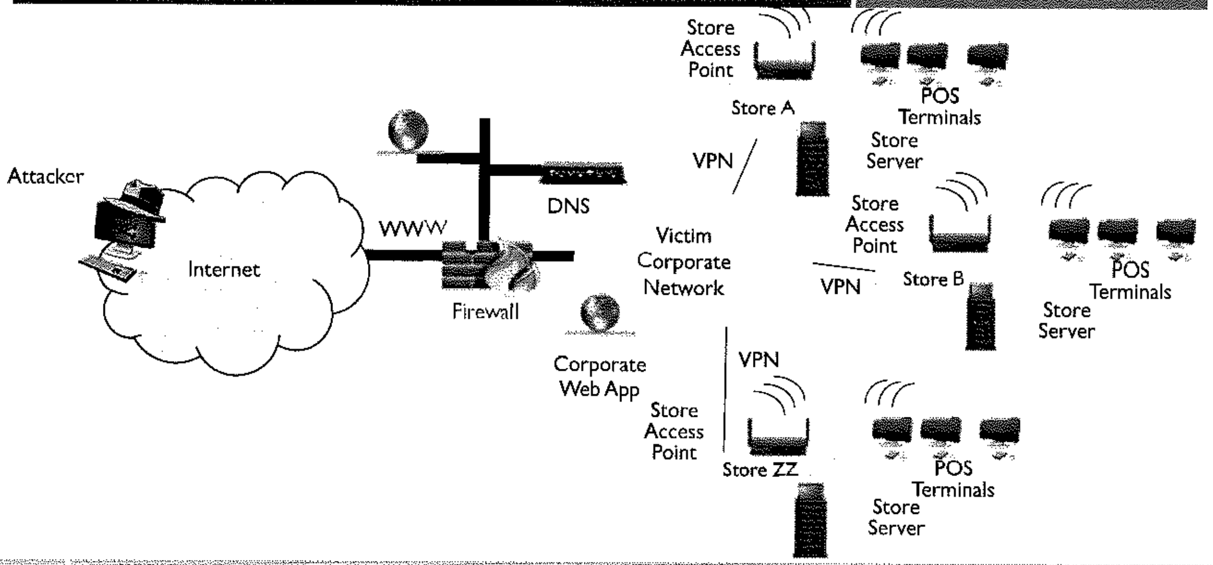
Scenario Details

Credit Card Theft

- The following scenario is based on a synthesis of several real-world cases of credit card theft
- Each technique described actually occurred
- We have blended the cases together and changed the names to protect the innocent

To see how credit cards sometimes get stolen en masse, let's look at an example scenario. This scenario is based on various real-world cases involving the theft of large numbers of credit cards. Each of the techniques we discuss has been used in real-world thefts, but we combine several such cases together into this single narrative to give you a feel for how this happens. We synthesized them and changed the names so you cannot tie it back to any one company.

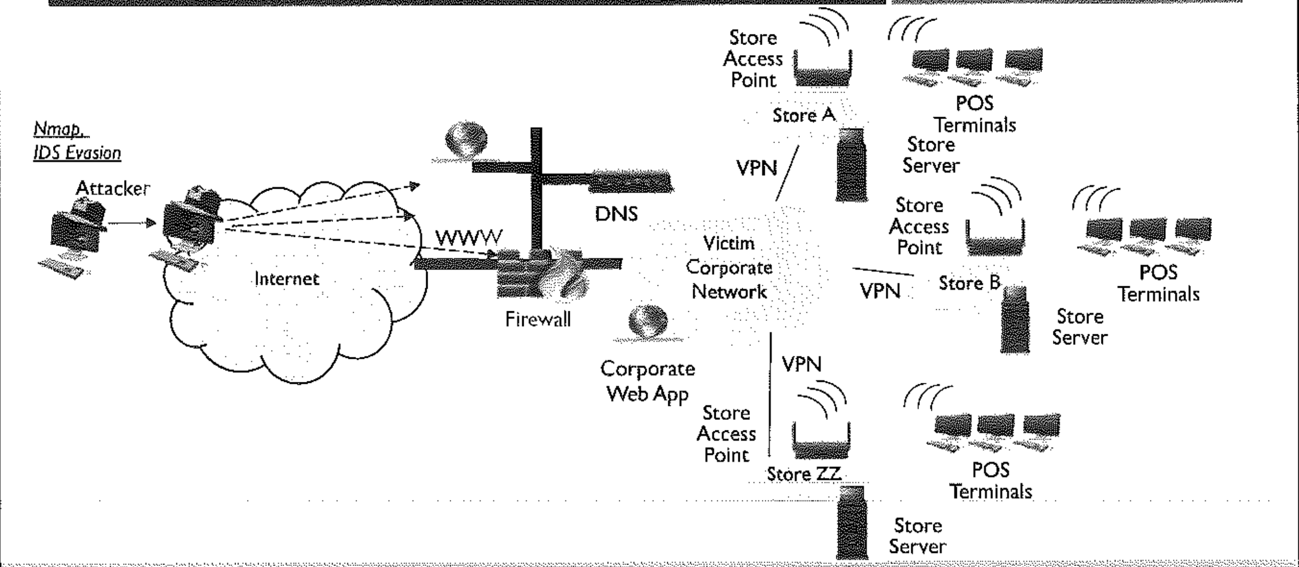
Credit Card Theft: Part 1



A credit card thief, known as "CCT," was in the business of stealing large numbers of credit cards and selling them on the black market, making about a dollar per illicit card. His customers would surely use them to defraud consumers and their banks for many thousands of dollars on each card, but CCT was very happy with his one dollar, knowing that law enforcement often focuses on the big-time fraudsters, not people like CCT. But, at a dollar per stolen card number, it was volume CCT was after. A heist of one hundred thousand credit cards would make his monthly goal, and a million would set him up for almost a year.

One of CCT's biggest hauls yet involved a victim retailer, which operated over 200 stores distributed in cities around the country. Stores were located in shopping centers, strip malls, and standalone buildings. Each outlet communicated with the victim's central corporate network using a VPN. This VPN link was configured to stay up all the time so that credit card transactions and inventory information could be seamlessly moved from individual stores back to the victim headquarters. Each store had several Point of Sale (POS) terminals, a fancy name for its computerized cash registers. To lower costs of deployment and increase flexibility in store layout, the POS terminals accessed the local store network using wireless access points. The victim company's technical and management personnel thought it had improved security by configuring these access points with MAC address filtering, allowing only the hardware addresses of each POS terminal for a given store in through that store's wireless access point. Each outlet also included a store server, which helped in processing credit card transactions. The store server would forward these transactions to a centralized server on the corporate network, which included a web application used by the victim company's management to analyze and manage business operations.

Credit Card Theft: Part 2

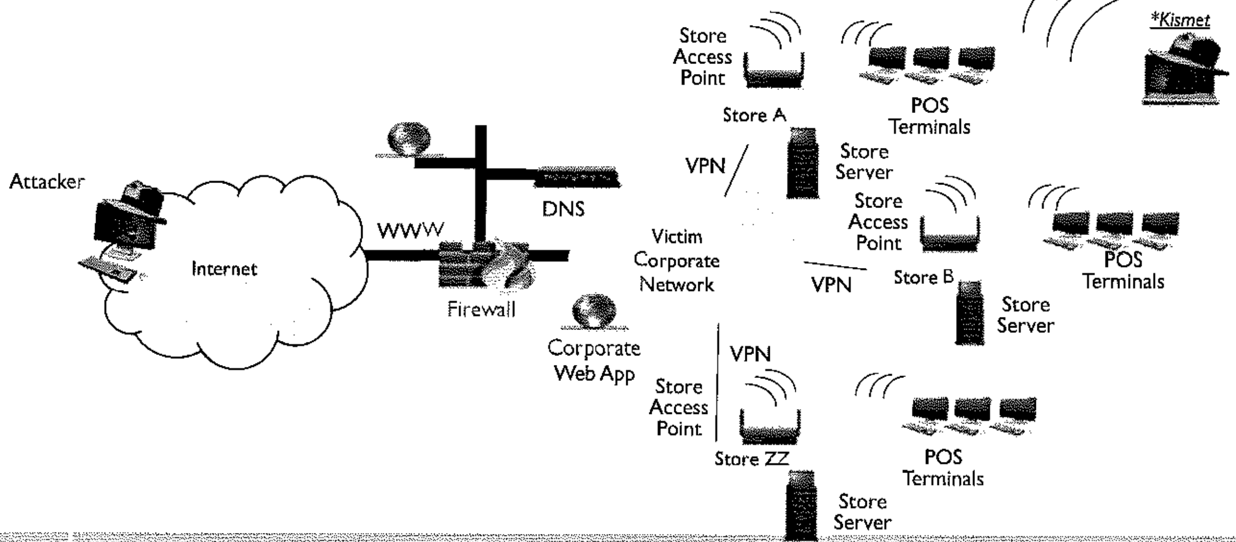


CCT decided to go after this particular victim because of an article he had read about the company's rapid expansion, a possible sign of security weaknesses. Perhaps the victim's quick growth meant that it wasn't as careful with its security as it should be. CCT began his adventure against the victim with some reconnaissance. He had to know some more information about his victim before starting to knock on its (virtual) doors. CCT searched InterNIC to look up information on the victim. The results of his InterNIC search proved quite fruitful, showing a target IP address space of a.b.c.0-255.

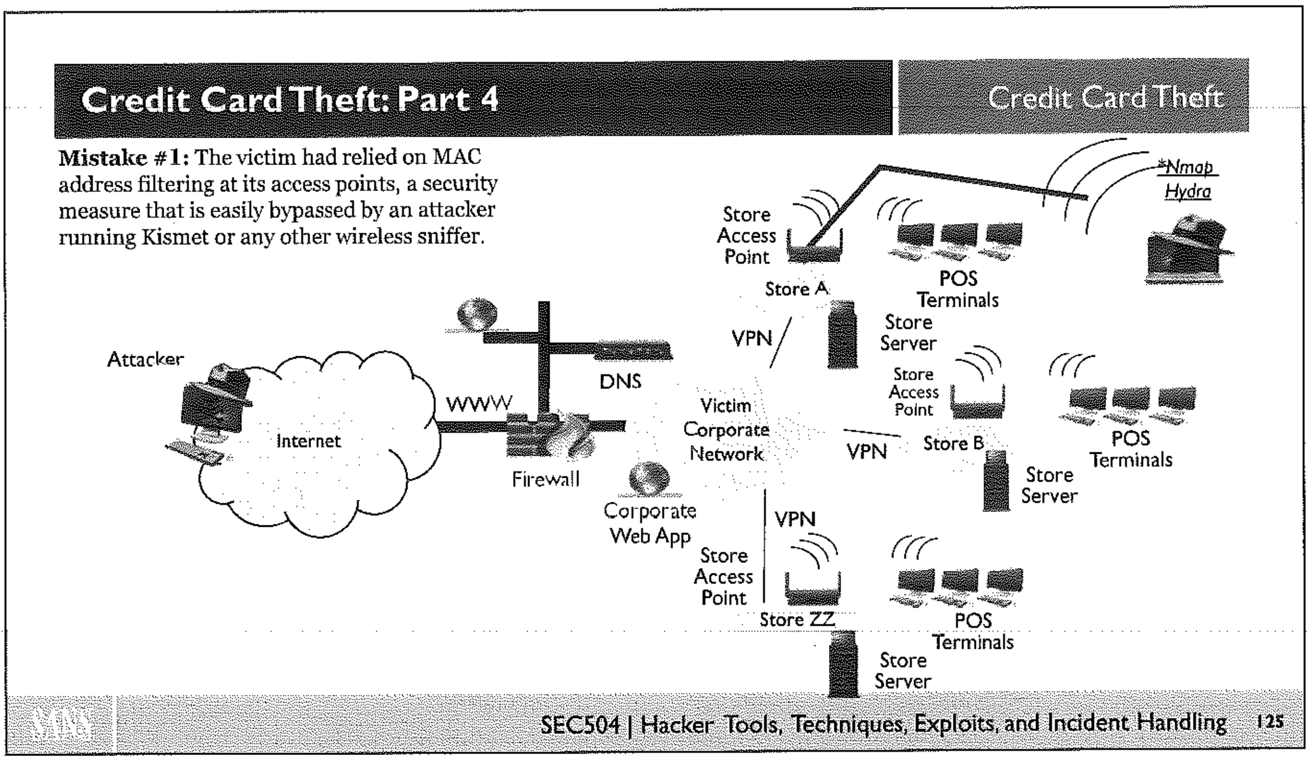
CCT used this information to begin scanning. He routed all scanning traffic through a packet fragmentation tool in an effort to avoid detection. He started scanning the victim's internet gateway using a network mapping tool to discover which systems were alive on the target network, resulting in the discovery of three internet-accessible systems. One of the three systems was in front of the other two. A quick port scan revealed TCP port 80 open on one of the systems, clearly a web server. The other system displayed no open TCP ports, but the UDP port scanner showed port 53 open. CCT had found a DNS server. The other system had no ports open, but a firewall assessment tool showed that it was indeed a packet filter firewall with rules allowing TCP port 80 and UDP port 53 to the DMZ machines. At this point, CCT discerned the general architecture of the victim's internet DMZ and firewall. He scribbled down all of this information, creating a basic sketch of the target. CCT also ran a vulnerability scan, just to see whether the victim made any simple mistakes, such as leaving vulnerable or unpatched services accessible on the internet. Unfortunately for CCT, the vulnerability scan came up dry. No known vulnerabilities were present on the DMZ.

Credit Card Theft: Part 3

Credit Card Theft

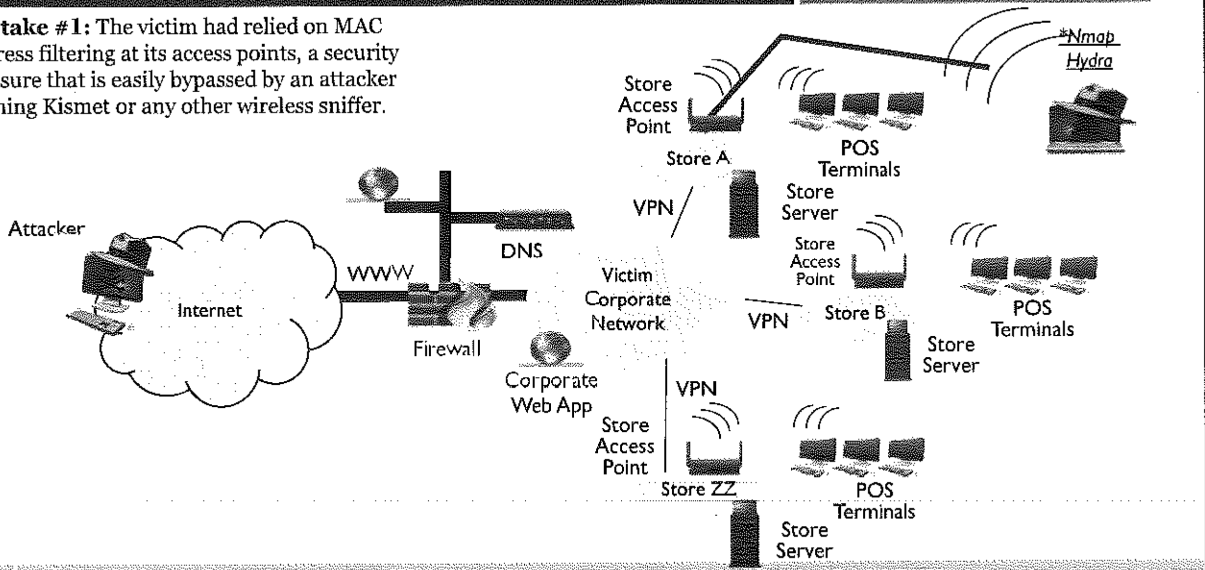


With the internet attack vector lacking promise, CCT surfed to the victim's website and stumbled upon a webpage that listed each of its outlets. A victim outlet store was located in a shopping mall just across town from CCT's home. Hopping in his car, CCT drove to the mall, which featured over a hundred stores and a food court. In this food court, CCT sat down and began looking for available wireless access points using a wireless LAN assessment tool. Using a passive tool, CCT was even able to see access points that were configured not to include their SSIDs in their beacon packets, as well as those set up not to respond to probe packets. By just gathering legitimate traffic, he noticed several access points nearby, but one had a particularly interesting SSID: victim041, a likely sign of his intended target.



Credit Card Theft: Part 4

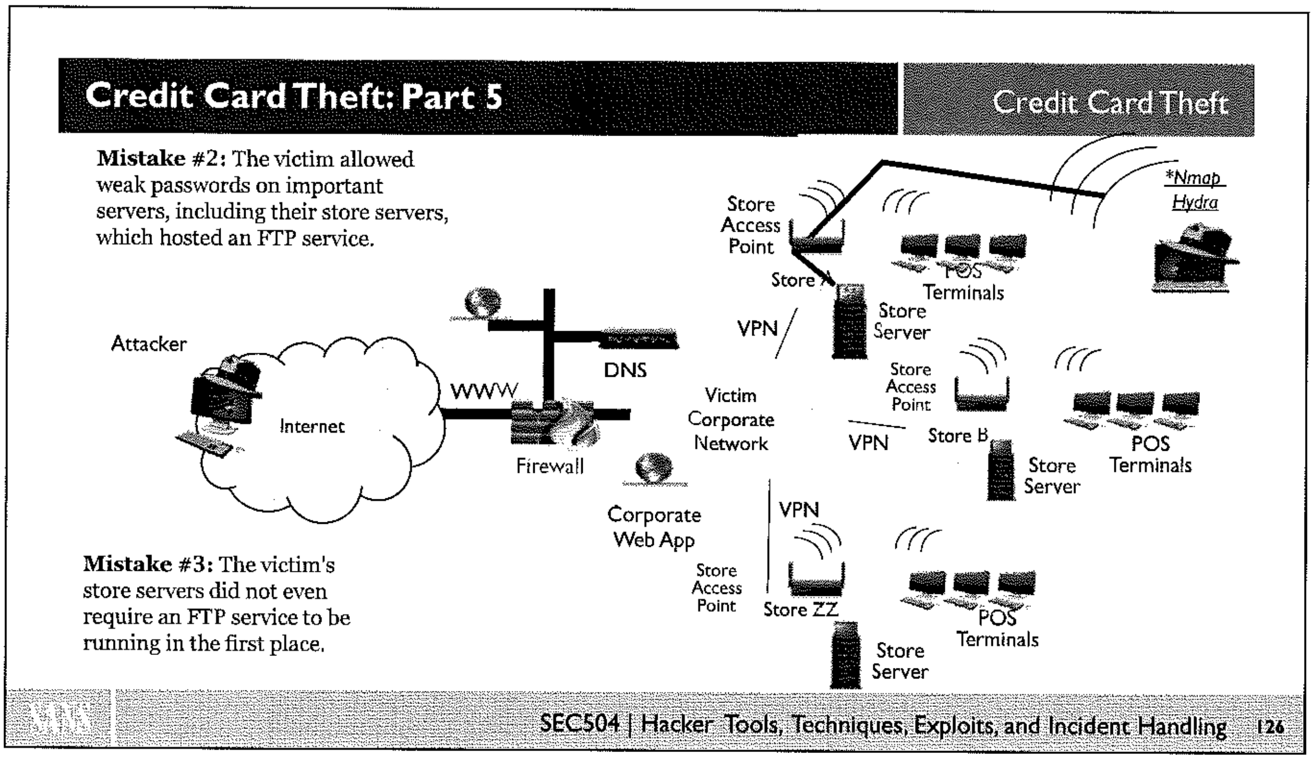
Mistake #1: The victim had relied on MAC address filtering at its access points, a security measure that is easily bypassed by an attacker running Kismet or any other wireless sniffer.



After configuring his wireless client with the victim041 SSID, CCT ran into a snag—he could not access the network for some reason. The access point appeared to be dropping all of his packets, as though it had a filter. Turning back to his wireless assessment tool's display, CCT looked through it carefully. He noticed the MAC address of one of the other devices using the victim041 SSID and quickly hypothesized that the victim was allowing only certain MAC addresses into its network. He configured his Linux laptop with the MAC address grabbed by the wireless LAN assessment tool by simply using the ifconfig command.

Mistake #1: The victim had relied on MAC address filtering at its access points, a security measure that is easily bypassed by an attacker running Wellenreiter or any other wireless sniffer. Likewise, configuring access points to remove SSIDs from their beacons and disabling responses to probe requests with "any" SSIDs are only marginal increases in security, easily bypassed using these same tools. The victim should have relied on cryptographic authentication for access to its store networks, using protocols such as 802.11i.

The spoofed MAC address worked well, allowing CCT through the access point with an IP address that he hardcoded that was on the same subnet as the other systems detected by Wellenreiter. With access to this network, CCT now turned his attention to determining the lay of the land. He launched a ping sweep of the target network, discovering the other POS devices, as well as the store server. With a reverse DNS lookup of the store server's IP address using the dig command in Linux, CCT saw what he wanted: The store server was named "store041.internal_victim.com."



Credit Card Theft: Part 5

Credit Card Theft

Mistake #2: The victim allowed weak passwords on important servers, including their store servers, which hosted an FTP service.

Mistake #3: The victim's store servers did not even require an FTP service to be running in the first place.

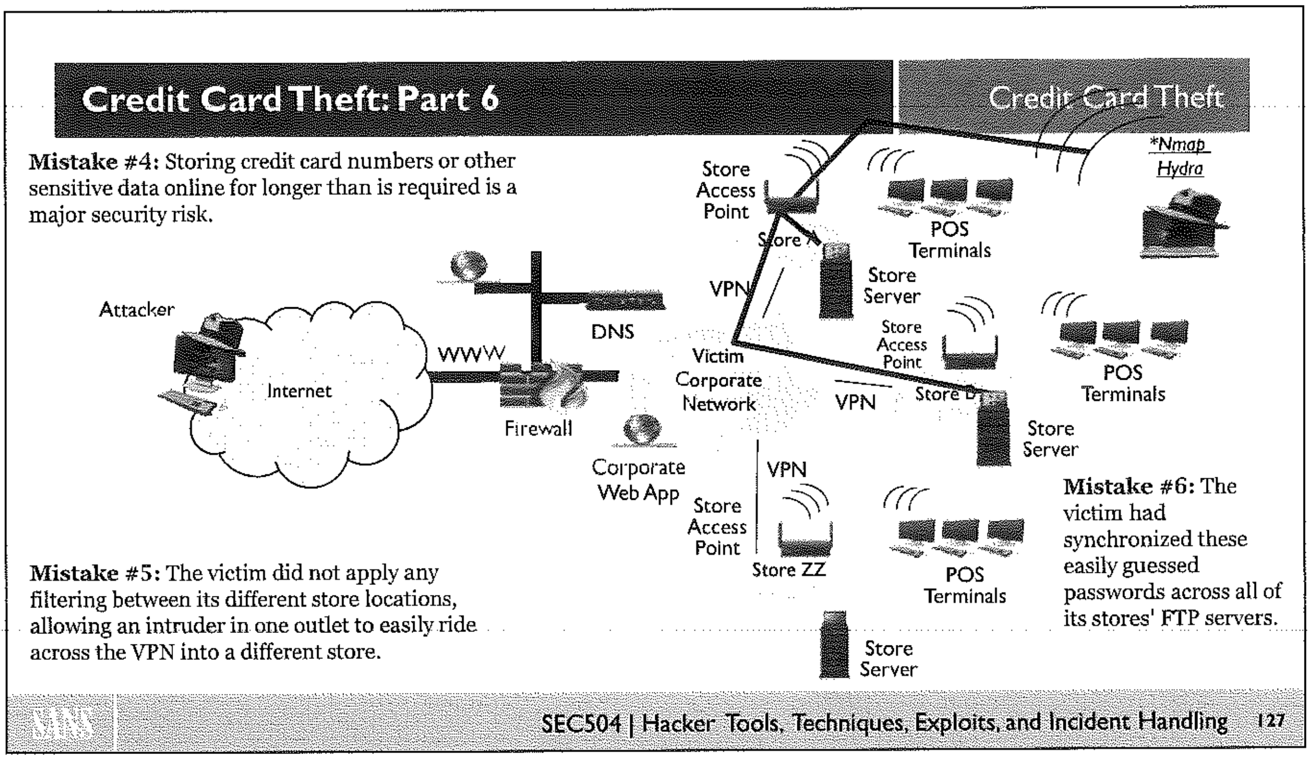
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 126

He then conducted a port scan of the store server. On this machine, CCT found TCP port 21 open, a likely sign of an FTP server. Based on this result, CCT ran an automated password-guessing tool trying to log in to the store server's FTP service. This tool guessed password after password for a variety of standard user IDs, including "root," "admin," and "operator." After five minutes of guessing, CCT discovered a user ID and password of "operator" and "rotarepo123," which is merely the word operator spelled backward, followed by 123.

Mistake #2: The victim allowed weak passwords on important servers, including their store servers, which hosted an FTP service. The victim should have configured its systems with tools that force password complexity so that users and administrators cannot choose trivial-to-guess passwords.

Mistake #3: The victim's store servers did not even require an FTP service to be running in the first place. However, because it was "inside" its network, the company did not bother shutting off this non-vital service. All services without a defined business need should be disabled, lest they offer an avenue for an attacker to gain access.

With his FTP access of the store server, CCT rifled through its files, searching for interesting information. After half an hour of pulling back files, he found his desired target. In an obscurely named directory, CCT located a file containing transaction history placed on the store server by the POS terminals. This history file included all credit card information, including account number, name, and expiration date, for all transactions at the store since the store server was initially deployed, over 100 days ago. All told, this single system provided over one hundred thousand credit card numbers for CCT. With a fine day's work completed, CCT went home to sell his newly found treasure.



Credit Card Theft: Part 6

Mistake #4: Storing credit card numbers or other sensitive data online for longer than is required is a major security risk.

Mistake #5: The victim did not apply any filtering between its different store locations, allowing an intruder in one outlet to easily ride across the VPN into a different store.

Mistake #6: The victim had synchronized these easily guessed passwords across all of its stores' FTP servers.

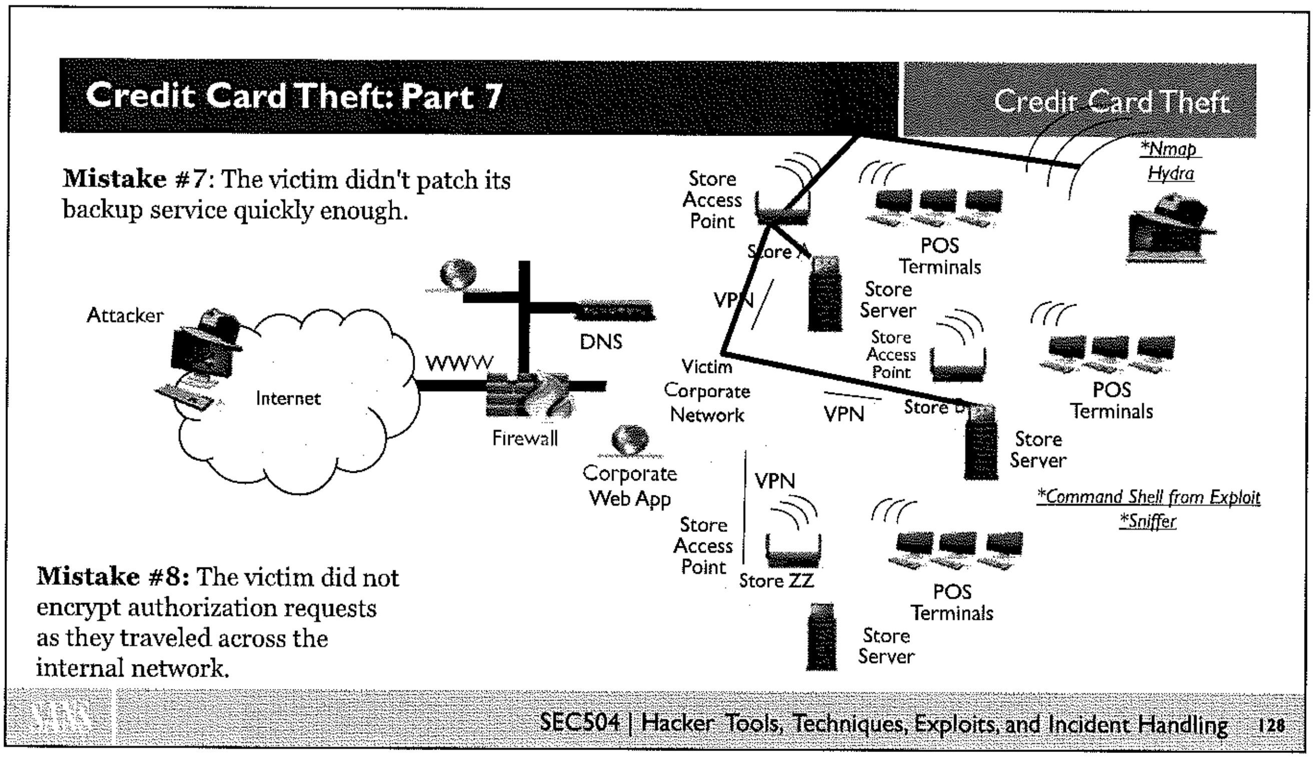
Mistake #4: Storing credit card numbers or other sensitive data online for longer than is required is a major security risk. Most organizations don't need to retain credit card numbers at all, or, if they do, they require only the data for a maximum of several days to support returns and refunds. The victim should have purged all transaction information quickly, rather than allowing it to linger.

While at home, CCT thought through the events of the day. He had gotten into a single store's server and grabbed numerous credit cards. He thought about looking up other victim outlet store locations on the website, but physically moving from store to store might be a lot of work and could get him caught. Instead, he thought hard about what he had found during the day: An FTP server at the store. Surely, other stores had similar configurations. With its rapid growth, the victim likely had each store built in as cookie-cutter a fashion as possible. That insight would dictate CCT's next move.

The next afternoon, CCT drove back to the mall and its food court. He configured his laptop to go through the access point he used yesterday. Next, instead of attacking the local store FTP server, he changed the IP address of the target, simply altering its third octet. Instead of trying to FTP to w.x.y.z, CCT tried to connect to w.x.y+1.z. He was very happy to see his FTP client connect to a different FTP server, this time at a different outlet of the victim corporation. He tried the same password from the first store and got right into this newly discovered FTP server.

Mistake #5: The victim did not apply any filtering between its different store locations, allowing an intruder in one outlet to easily ride across the VPN into a different store. The attacker didn't even realize a VPN was used to interconnect the stores because all store-to-store access was transparent. Organizations should apply filters at the routers or firewalls between their branches, outlets, or business units, allowing only those services required by the business to go through.

Mistake #6: Not only were one store's FTP passwords guessable, but the victim had synchronized these easily guessed passwords across all of its stores' FTP servers. This weakness made the attacker's job easier. The organization should have used difficult-to-guess passwords that were different for the servers in each store.



Credit Card Theft: Part 7

Credit Card Theft

Mistake #7: The victim didn't patch its backup service quickly enough.

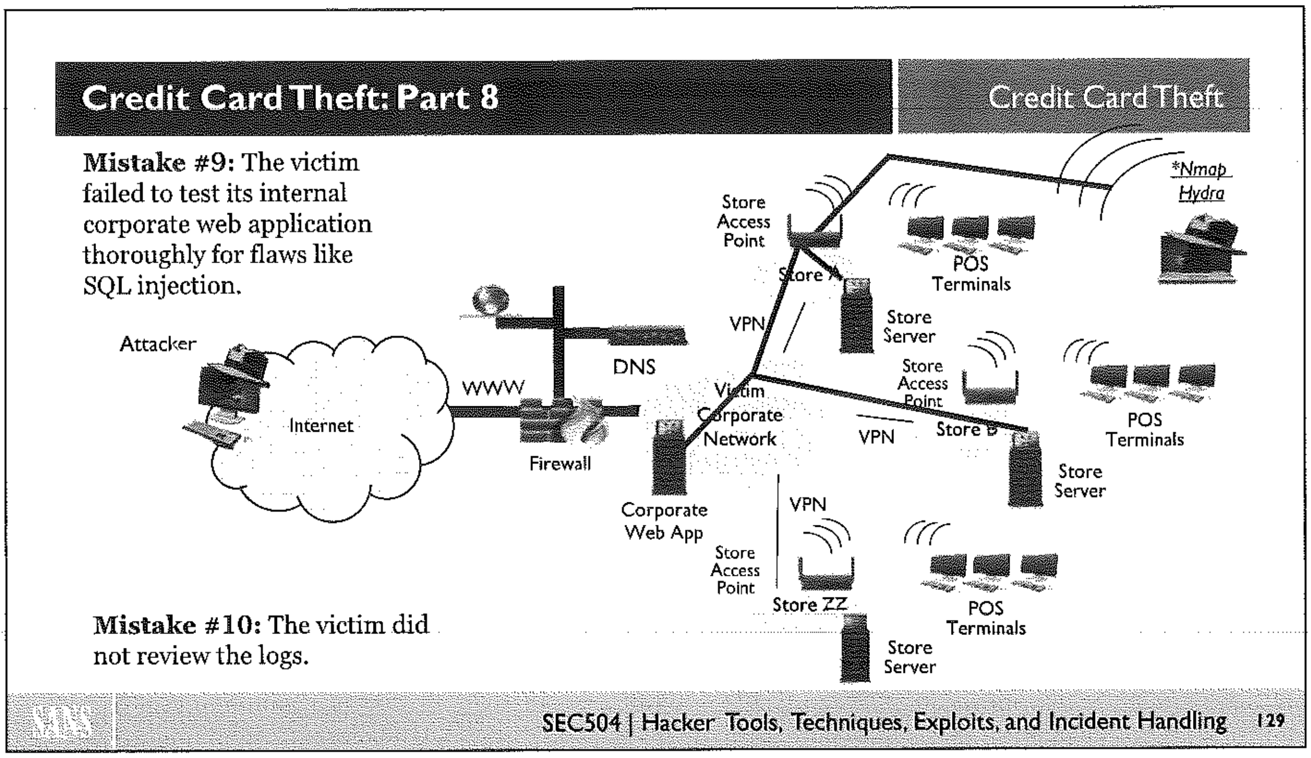
Mistake #8: The victim did not encrypt authorization requests as they traveled across the internal network.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 128

After grabbing another hoard of credit card numbers from Store B, CCT reflected on the attack briefly. Stepping back, he looked at the results of the port scan from the server in Store A. Sure enough, it was running a popular backup program widely known to have a buffer overflow vulnerability. CCT launched a system exploitation tool from his own Linux box against the backup service running on the Store B server, assuming it was running there just as it was running in Store A, taking advantage of that cookie-cutter architecture. Now, with full command shell access on the Store B system provided by the exploitation tool, he installed a sniffer to gather information passing across the store LAN. The sniffer grabbed transaction information as it passed from the POS terminals to the store server, a point that didn't interest CCT that much because he already had such information from the FTP service in the store. But the sniffer turned up a more subtle and important point. The store server itself was sending transaction requests to another server on a different network. These transactions were sent in cleartext, letting CCT rapidly discern that he was looking at credit card authorization requests flowing across the internal network.

Mistake #7: The victim didn't patch the backup service quickly enough. Numerous buffer overflows and other vulnerabilities are discovered on a regular basis, so organizations need to diligently and thoroughly patch their systems, including not only the underlying operating system but also all of the applications they've installed.

Mistake #8: The victim did not encrypt authorization requests as they traveled across the internal network. Such information is quite sensitive and should be carefully encrypted throughout its journey across even an internal network.



Credit Card Theft: Part 8

Credit Card Theft

Mistake #9: The victim failed to test its internal corporate web application thoroughly for flaws like SQL injection.

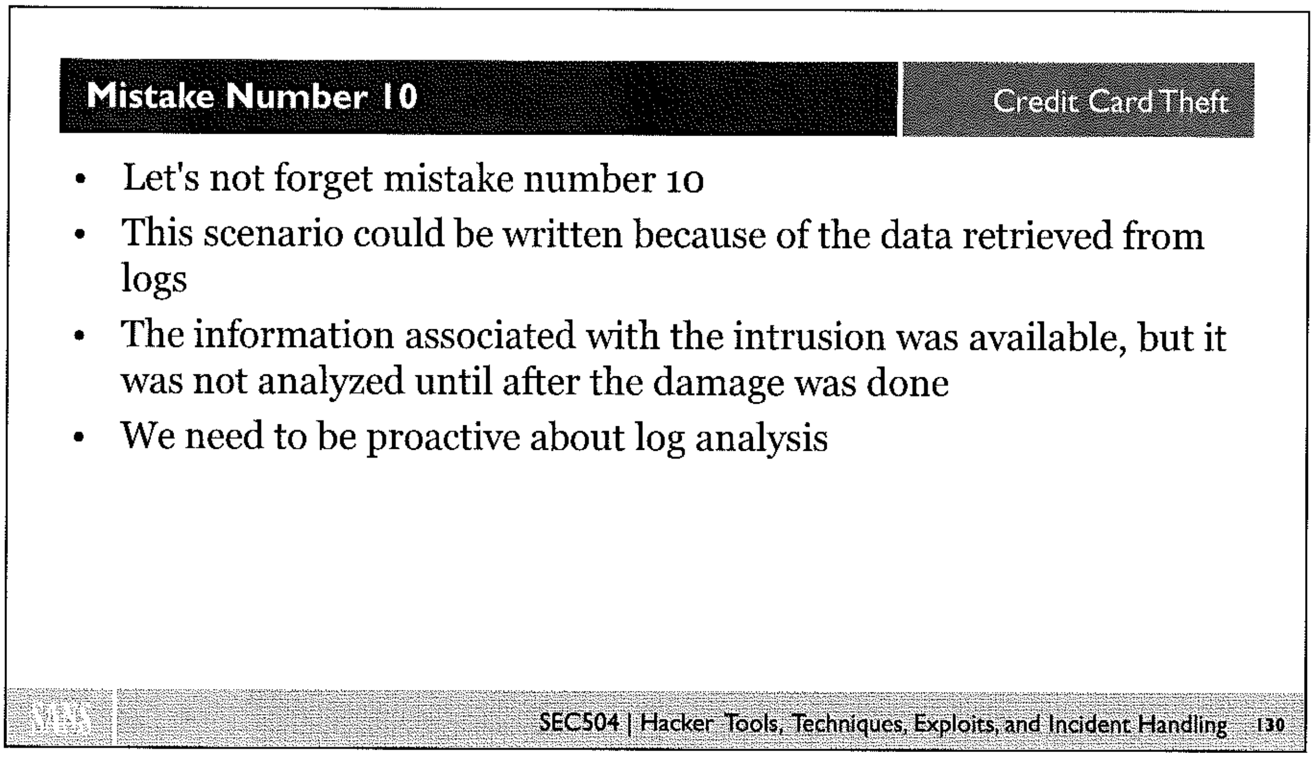
Mistake #10: The victim did not review the logs.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 129

Using the destination address information gathered from the sniffer, CCT ran the port scanner again. This time, he was scanning a server on the victim's corporate network, that crucial system it used for processing all credit card transactions as well as managing its business. The port scanner rapidly identified TCP port 443, a sure sign of HTTPS access to the box. On his own Linux machine, CCT launched a browser to surf to the given website on the victim's corporate server, only to be presented with a webpage describing the internal management application. This particular webpage did not have any sensitive information on it but instead allowed internal users to log in to a web application that provided detailed business information. Without a user ID and password, though, the attacker was stuck. He tried the user ID and password that got him into the FTP servers but to no avail. Likewise, his password-guessing tool turned up nothing either.

Next, CCT fired up a web application manipulation proxy tool. Using its automated web application scanning capabilities, CCT searched the target for Cross-Site Scripting and SQL injection flaws. After about five minutes of intense scanning, the web application manipulation proxy returned with some good news for CCT: A SQL injection flaw in a web cookie associated with the user ID component of the target application. By setting up the proxy to manipulate this cookie manually, CCT tweaked the SQL injection syntax to explore the database underlying the corporate web application. He discovered a table in this database that held a set of customer records from across all 200 victim outlet stores, including over one million credit card numbers.

Mistake #9: The victim failed to test its internal corporate web application thoroughly for flaws like SQL injection. This corporation believed that such internal applications were safer, given their location on a trusted internal network. "What's more, we trust our people," its management frequently stated. However, systems storing sensitive data, even on internal networks, should be carefully scrutinized for vulnerabilities.



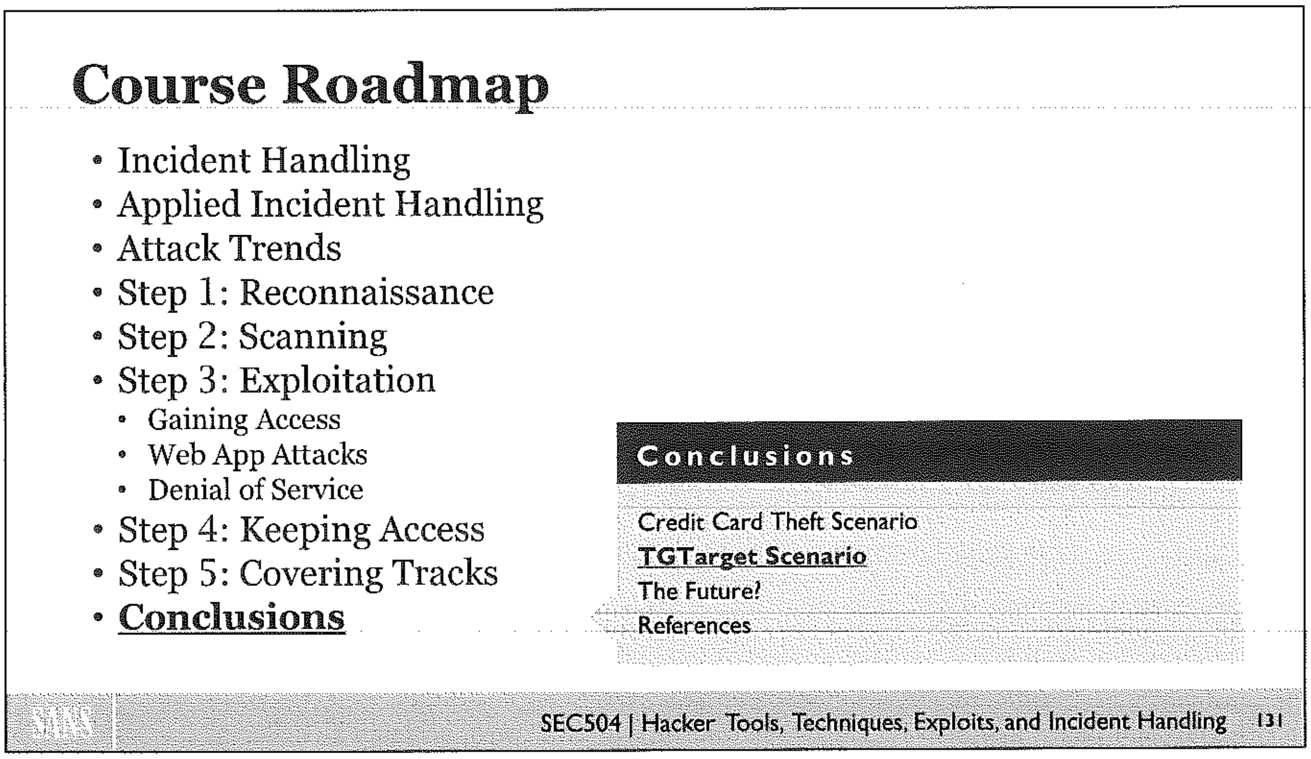
Mistake Number 10

Credit Card Theft

- Let's not forget mistake number 10
- This scenario could be written because of the data retrieved from logs
- The information associated with the intrusion was available, but it was not analyzed until after the damage was done
- We need to be proactive about log analysis

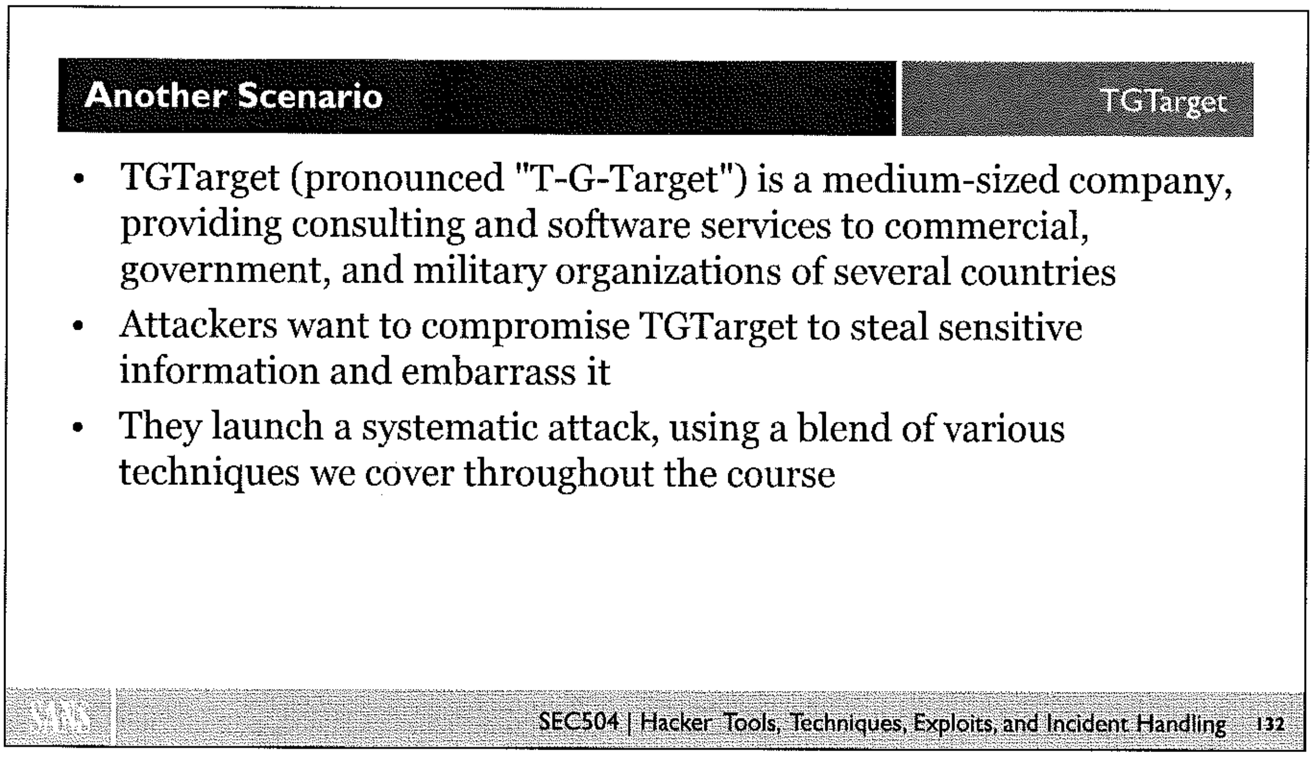
Mistake #10: The victim did not review the logs from its store access points (which might have identified CCT's unusual access), store FTP servers (which would likely have identified the password-guessing attack and stolen files), and the corporate web application (which would almost surely have shown the password-guessing attacks and SQL injection attempts). Although diligent log review might not have stopped the attack entirely, it would have allowed the victim to discover the attacker early in the process, minimizing the damage to the victim's reputation and finances.

Now, with his million credit card numbers, CCT left the mall. He rapidly sold the account information to his underworld contacts and then destroyed all aspects of the data he had stolen.



Course Roadmap

In this next scenario, we'll examine a case where attackers exploit a fictitious company we call *TGTarget*.



Another Scenario

TGTarget

- TGTarget (pronounced "T-G-Target") is a medium-sized company, providing consulting and software services to commercial, government, and military organizations of several countries
- Attackers want to compromise TGTarget to steal sensitive information and embarrass it
- They launch a systematic attack, using a blend of various techniques we cover throughout the course

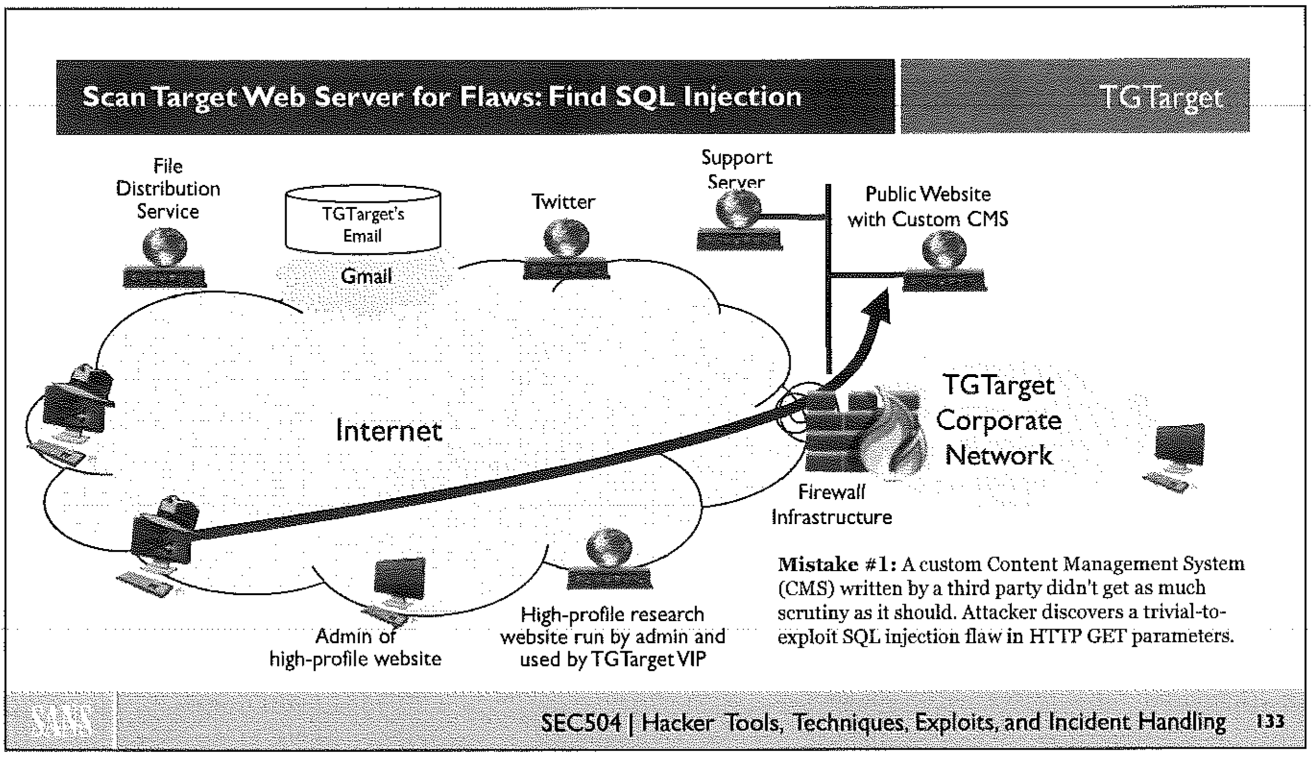
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 132

Next, let's consider a scenario that illustrates an attack using the various techniques we discuss throughout the course. In this scenario, the target organization is known as TGTarget, pronounced "T-G-Target." It is a mid-sized company providing professional services and software to customers that include commercial companies, several government agencies in countries around the world, and military organizations of various countries.

Some computer attackers target this organization with the goals of stealing its sensitive information and embarrassing the company.

To achieve these goals, the attackers launch an attack that blends together a variety of disparate techniques, including SQL injection, password cracking, social engineering, and more.

As we walk through this scenario, we will highlight the mistakes made by TGTarget personnel that allowed the attackers to compromise their systems thoroughly.



Scan Target Web Server for Flaws: Find SQL Injection

TGTarget

File Distribution Service
TGTarget's Email
Gmail
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #1: A custom Content Management System (CMS) written by a third party didn't get as much scrutiny as it should. Attacker discovers a trivial-to-exploit SQL injection flaw in HTTP GET parameters.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 133

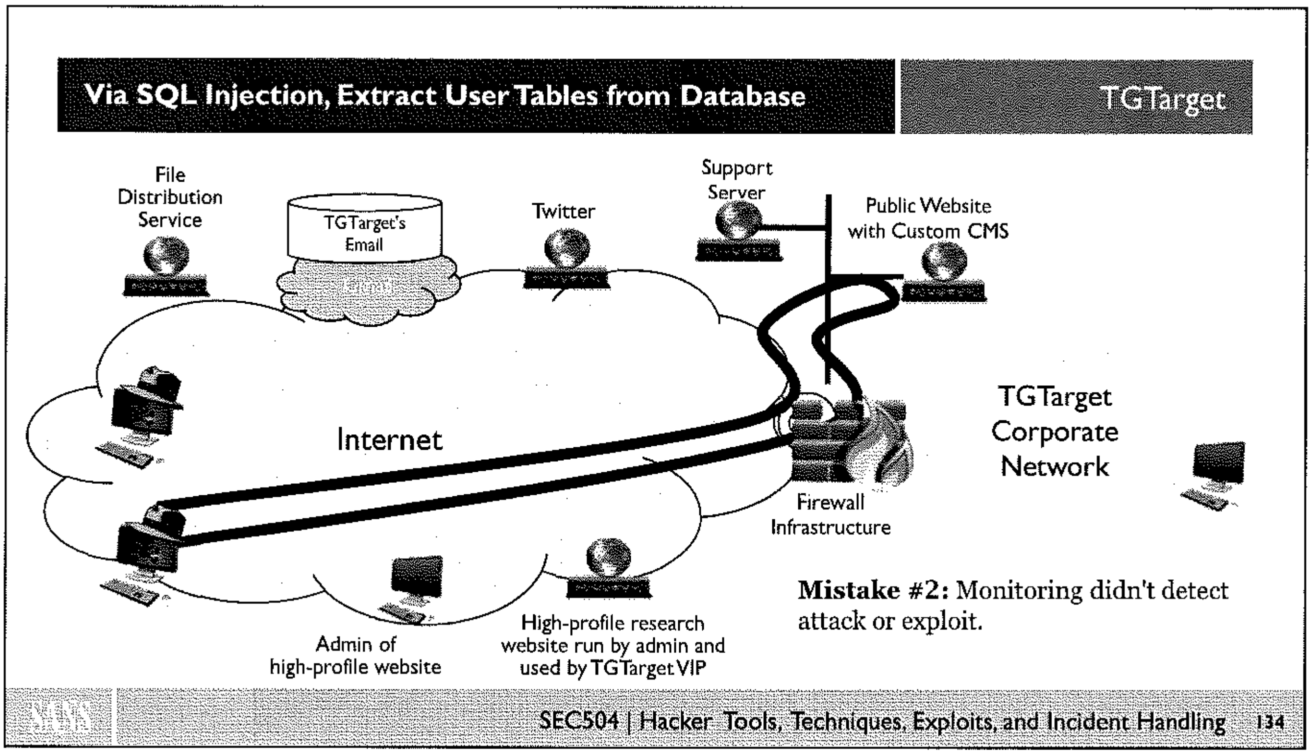
Here, we see an illustration of TGTarget's infrastructure. It has a corporate network on the right that includes a firewall infrastructure. Hanging off this infrastructure is a DMZ that includes a public website, which is managed using a custom Content Management System (CMS) that TGTarget procured from a software development company. This CMS was used to post and update content on the website. We also see a support server on this DMZ, used to help various TGTarget clients.

We also see Twitter, a service used by various VIPs of TGTarget to make public announcements about its business. We also see the Gmail service, which TGTarget relied on for all of its corporate email. In addition, we see a file distribution service, such as BitTorrent or one of the myriads of peer-to-peer (P2P) file-sharing services.

One of the corporate officers of TGTarget (which we refer to as a "TGTarget VIP") was associated with a high-profile public website that wasn't directly related to TGTarget's business. Instead, this website was used for general purpose research about topics of interest to this TGTarget VIP. This website was administered by another person, who didn't work for TGTarget, but who was an associate of the TGTarget VIP.

Finally, on the left side of the figure in the slide, we see the attackers.

The attack begins with a scan of the TGTarget DMZ, specifically focused on the web server. The attackers use a variety of vulnerability-scanning tools and discover a trivial-to-exploit SQL injection flaw in the website's CMS. By simply adding some SQL statements to the variables passed on a URL via HTTP GET messages, the attacker could access the database that housed content for the website. TGTarget's first mistake was using a custom CMS that did not receive detailed security scrutiny through its own vulnerability assessment or penetration tests, allowing a major vulnerability to go undiscovered until the attackers found it.



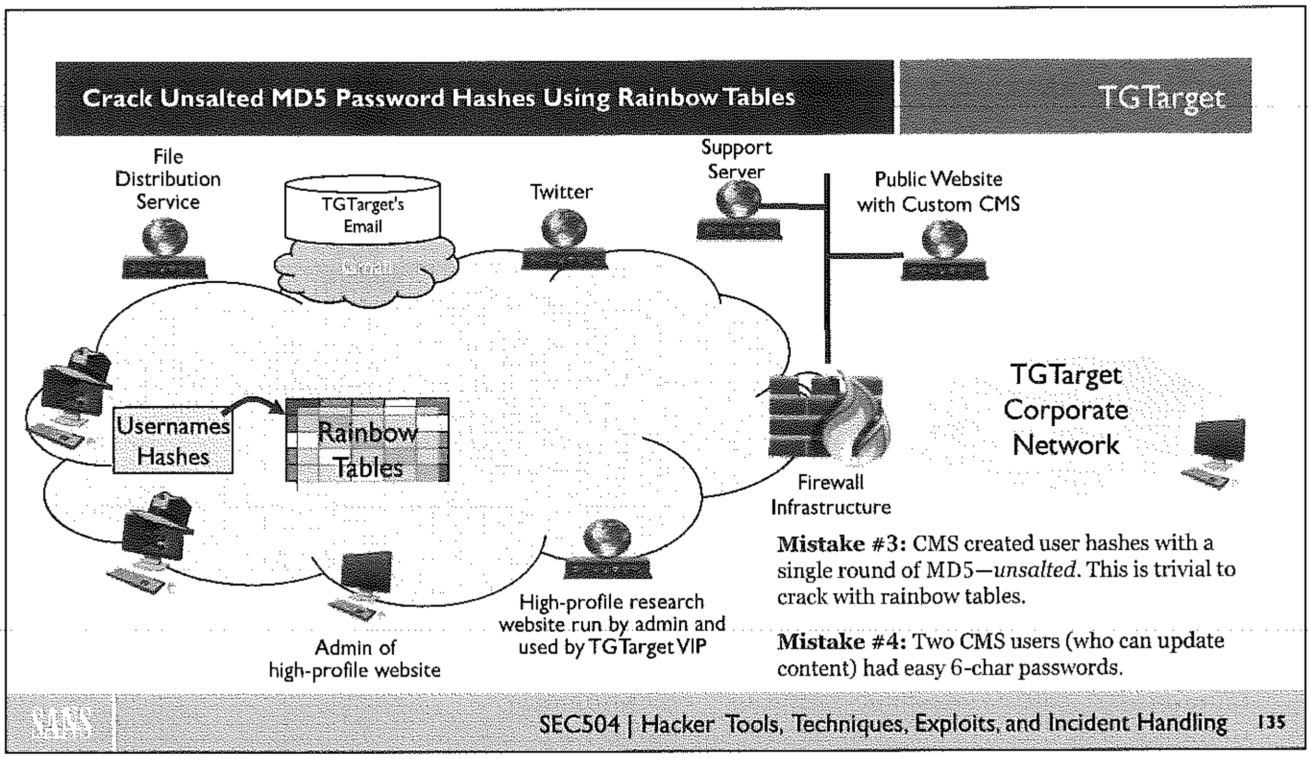
Via SQL Injection, Extract User Tables from Database

TGTarget
File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 134

Mistake #2: Monitoring didn't detect attack or exploit.

The CMS database not only housed the content of the website but also held a table with the usernames and password hashes for all users of the CMS that could update the website. Instead of merely defacing the website, the attackers used SQL injection to extract this table, giving them a list of CMS administrator usernames and password hashes.

Mistake #2 involved TGTarget's lack of monitoring for attacks. Its assailants were able to scan for, find, and exploit the SQL injection attack without being discovered. Furthermore, they were able to extract CMS usernames and password hashes without anyone at TGTarget noticing. A robust monitoring program might have detected the attack and thwarted the follow-on activity before significant damage occurred.



Crack Unsalted MD5 Password Hashes Using Rainbow Tables

TGTarget
File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Usernames Hashes
Rainbow Tables
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #3: CMS created user hashes with a single round of MD5—*unsalted*. This is trivial to crack with rainbow tables.

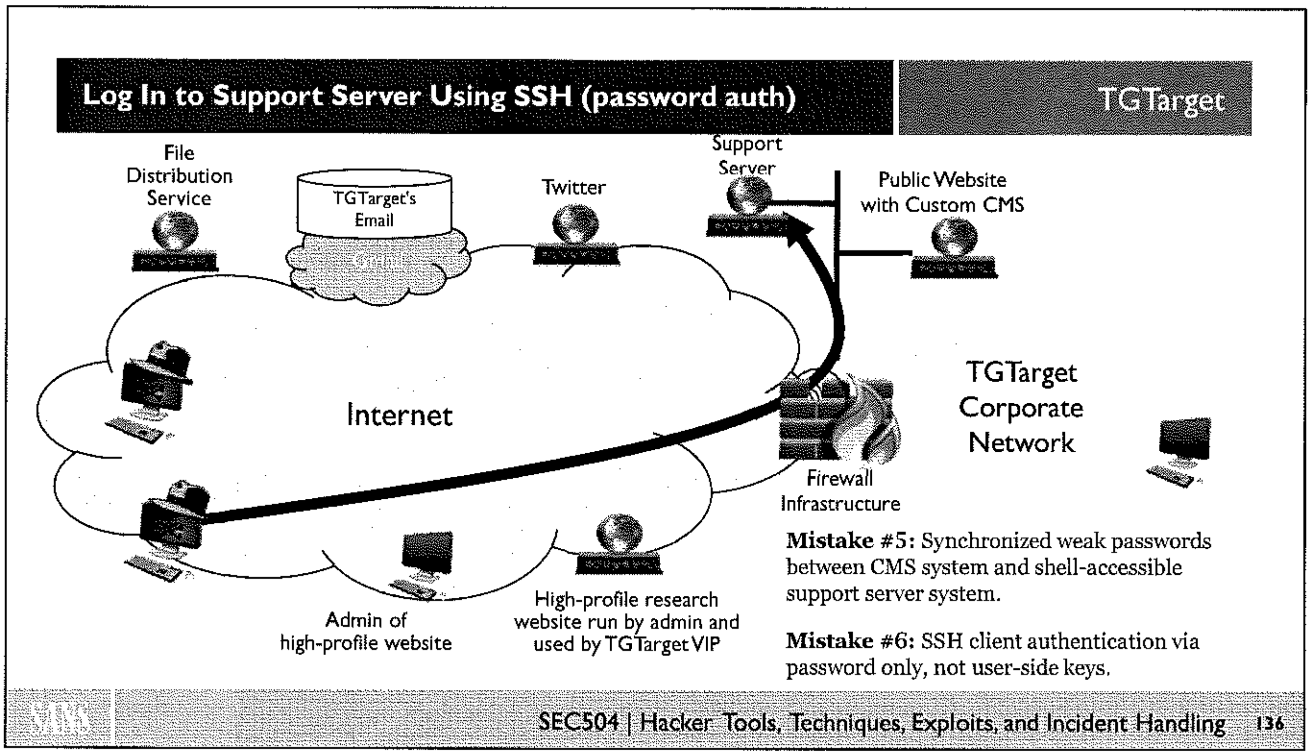
Mistake #4: Two CMS users (who can update content) had easy 6-char passwords.

The attackers now analyzed the structure of the password hashes they had extracted. It turns out that these hashes were simply the MD5 hash of the original password, with only a single round of MD5 hashing with no salt. Although modern Linux systems use a salt and apply 1,000 or more rounds of hashing, this custom CMS had its own application-level passwords that were based on a single round of MD5. That is **Mistake #3**: TGTarget used a CMS with a trivial password hashing algorithm that didn't include salts. It should have used a far stronger password mechanism, such as salted MD5 with a thousand or more rounds.

Without salts, the attackers were able to conduct a rainbow table attack against the password hashes from the CMS. Numerous unsalted MD5 rainbow tables are available on the internet, which the attackers used to crack at least two passwords from the CMS. If the password algorithm used by the CMS had involved salts, the attackers could have still cracked them, but it would have taken more time and effort.

The two passwords cracked by the attackers were actually quite weak: Each was only 6 characters long. That is **Mistake #4**. Those passwords should have been longer and more complex. If they had been 15–20 characters in length, it is far less likely that the attackers would be able to get rainbow tables that represent hashes for such long passwords.

These two CMS users had update rights to the website, so the attackers could have altered content or planted a backdoor on the CMS, but they instead focused elsewhere.



Log In to Support Server Using SSH (password auth)

TGTarget

File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #5: Synchronized weak passwords between CMS system and shell-accessible support server system.

Mistake #6: SSH client authentication via password only, not user-side keys.

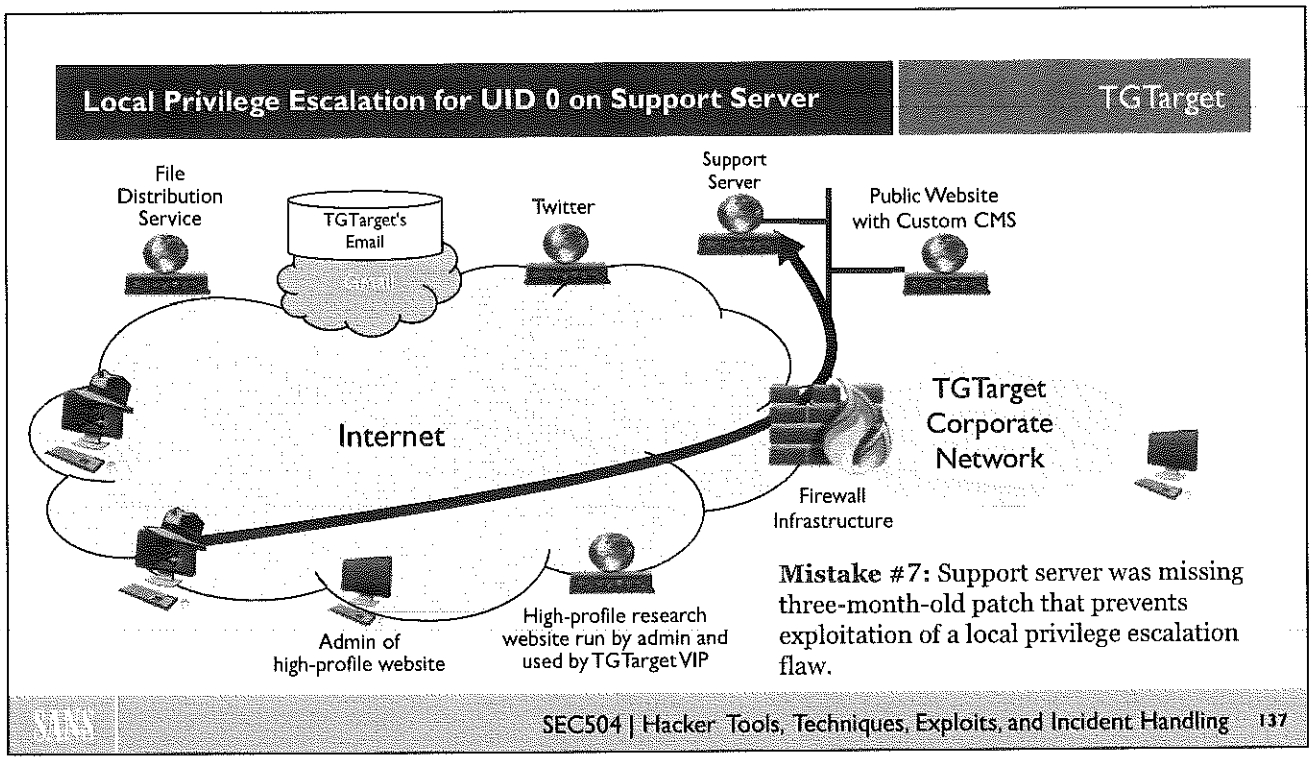
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 136

The attackers began scanning for other systems on the TGTarget DMZ and noticed that Secure Shell was available on another box: The support server. They tried to log in to this server with a username and password that they got from the CMS. With their login successful, they now had non-root access to the customer support server.

The attackers were able to get such access based on two additional mistakes made by TGTarget personnel. In **Mistake #5**, the CMS user had relied on the same username and weak password on both the CMS and the support server. For very different job functions, different passwords should have been used. Thus, the attackers were able to leverage account information and password hashes stolen via SQL injection to get shell access on the TGTarget DMZ.

Furthermore, the ssh daemon on the support server was configured to allow user authentication via passwords. In **Mistake #6**, the ssh service did not require client-side ssh keys to authenticate. Therefore, the attackers never needed to steal the public key of the user. Using client-side public keys (in addition to server-side keys), an attacker has to jump through additional hoops to log in to the server (namely, stealing the client ssh key). Instead, with just a username and easily cracked password, the attackers got shell access on the DMZ.

However, this shell access had limited privileges. So far, the attackers could access only files owned by their compromised account, as well as world readable information on this system. They needed deeper access to the machine to be able to grab files associated with other users on the box.



Local Privilege Escalation for UID 0 on Support Server

TGTarget

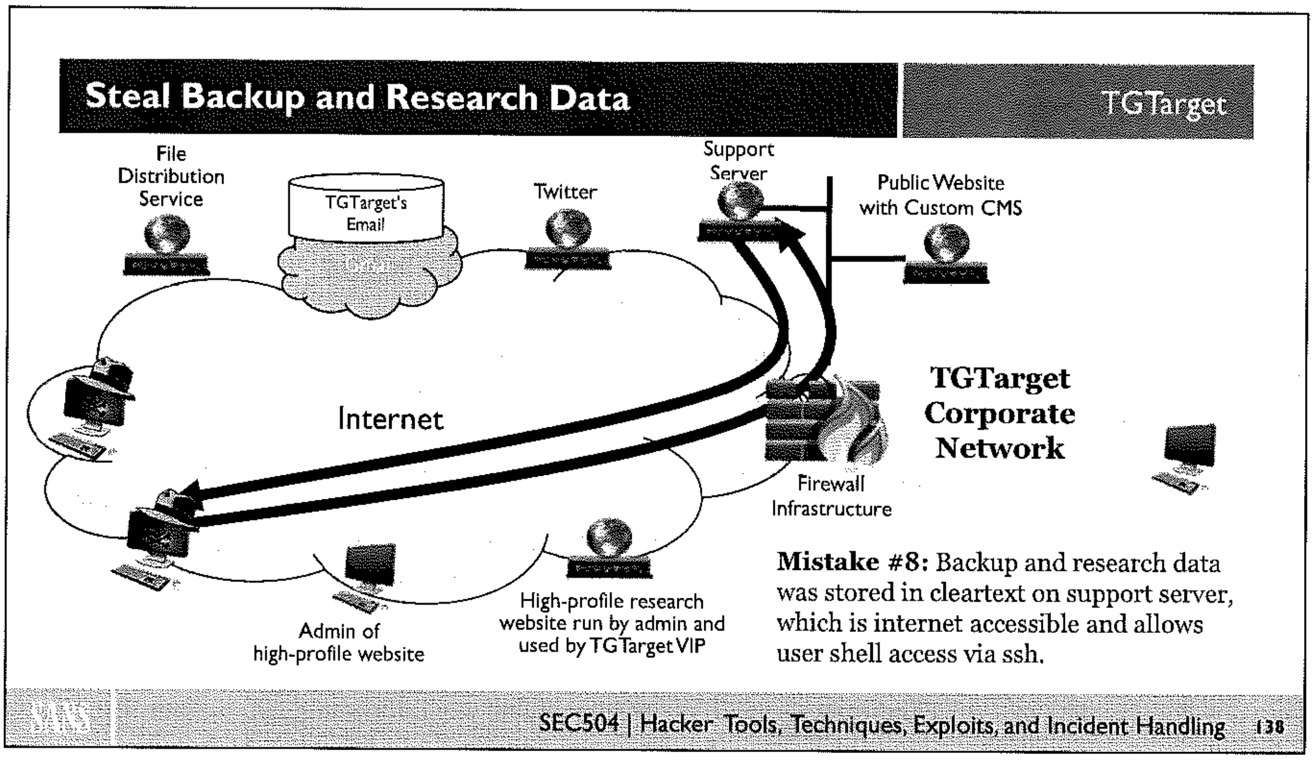
File Distribution Service
TGTarget's Email
Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #7: Support server was missing three-month-old patch that prevents exploitation of a local privilege escalation flaw.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 137

Unfortunately, although TGTarget personnel worked to keep their systems patched for remotely exploitable vulnerabilities, they did not vigorously apply patches that protect against local privilege escalation. In fact, the support server was missing a three-month-old patch that fixed a Linux kernel flaw that leads to local privilege escalation. **Mistake** #7 was failing to apply these important patches for local privilege escalation vulnerabilities over at least a three-month time span.

By uploading and running a free and widely available exploit for the Linux kernel, the attackers were able to gain UID 0 access (full root privileges) on the support server box.



Steal Backup and Research Data

TGTarget

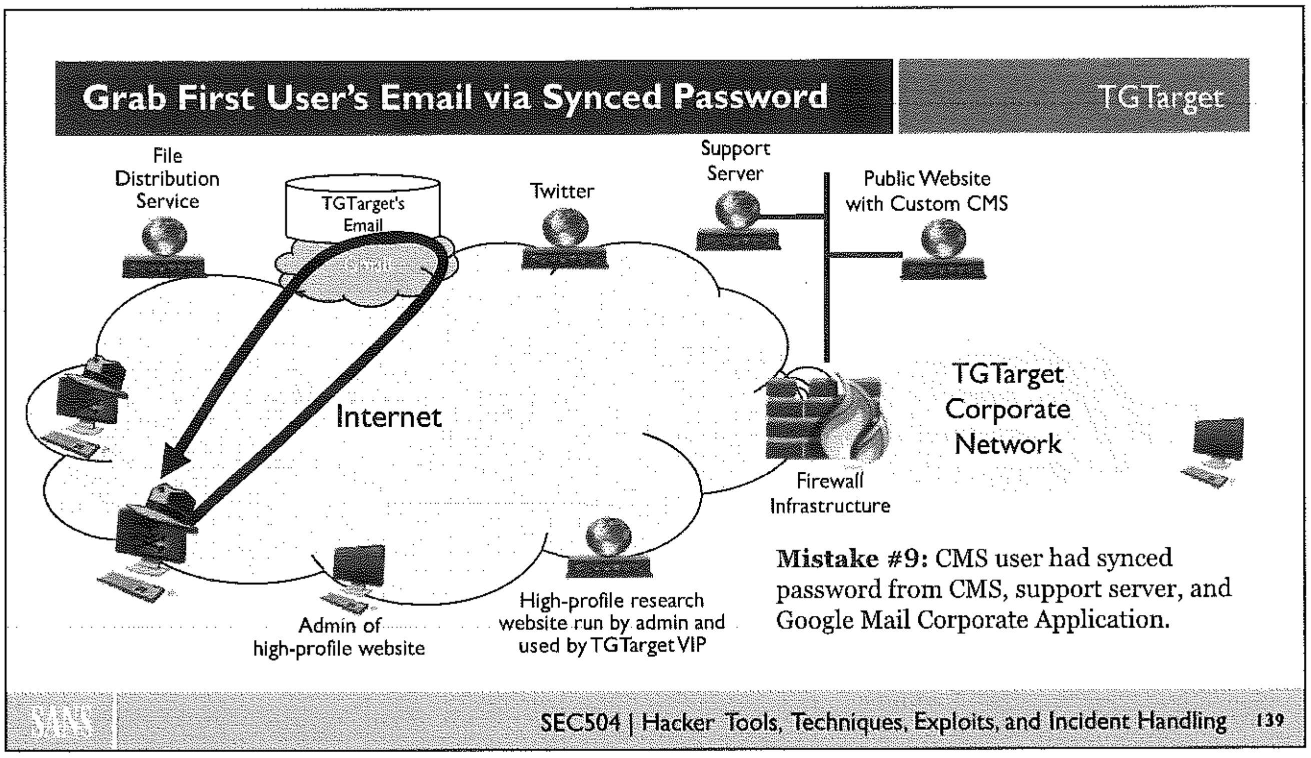
File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #8: Backup and research data was stored in cleartext on support server, which is internet accessible and allows user shell access via ssh.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 138

With their high privileges on the support server machine, the attackers scoured through the filesystem looking for interesting data. They managed to discover backup and research data on this machine, associated with various TGTarget personnel. The attackers grabbed these files for later analysis.

In **Mistake #8**, TGTarget stored backup and research data on an internet-accessible system associated with customer support. If there was no business need for having this data on this machine, it should have been removed. Alternatively, the research data should have been moved to a separate machine that was not associated with customer support.



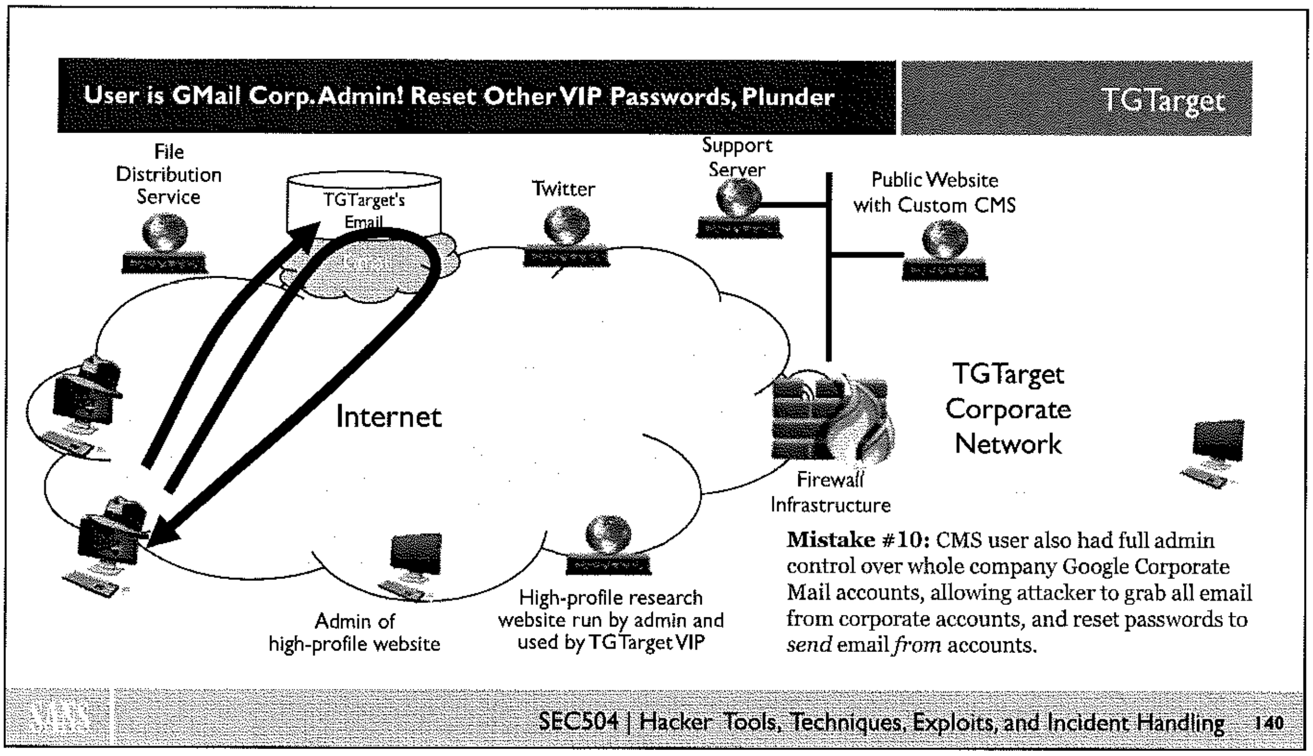
Grab First User's Email via Synced Password

TGTarget
File Distribution Service
TGTarget's Email
Gmail
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP
SANS
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling
139

Mistake #9: CMS user had synced password from CMS, support server, and Google Mail Corporate Application.

The attackers now looked at the MX record in DNS for TGTarget, observing that all of its email is actually sent through Google's cloud service for email, a corporate-branded version of Gmail. On a whim, the attackers surf to the Gmail page for the company and attempt to log in using the original two usernames and passwords they had retrieved from the CMS. One of them worked! The attackers now had access to this user's email. They grab all of the email archived for this user in the Gmail account.

Mistake #9 was yet another problem with manually synchronized user passwords. The weak password used on the CMS (and also used on the support server) was likewise used for this email account in Gmail. For different levels of access (such as managing a CMS versus accessing Google mail), different passwords should be used, especially for a high-profile user in the organization such as the CMS administrator.



User is GMail Corp. Admin! Reset Other VIP Passwords, Plunder

TGTarget

File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #10: CMS user also had full admin control over whole company Google Corporate Mail accounts, allowing attacker to grab all email from corporate accounts, and reset passwords to *send* email *from* accounts.

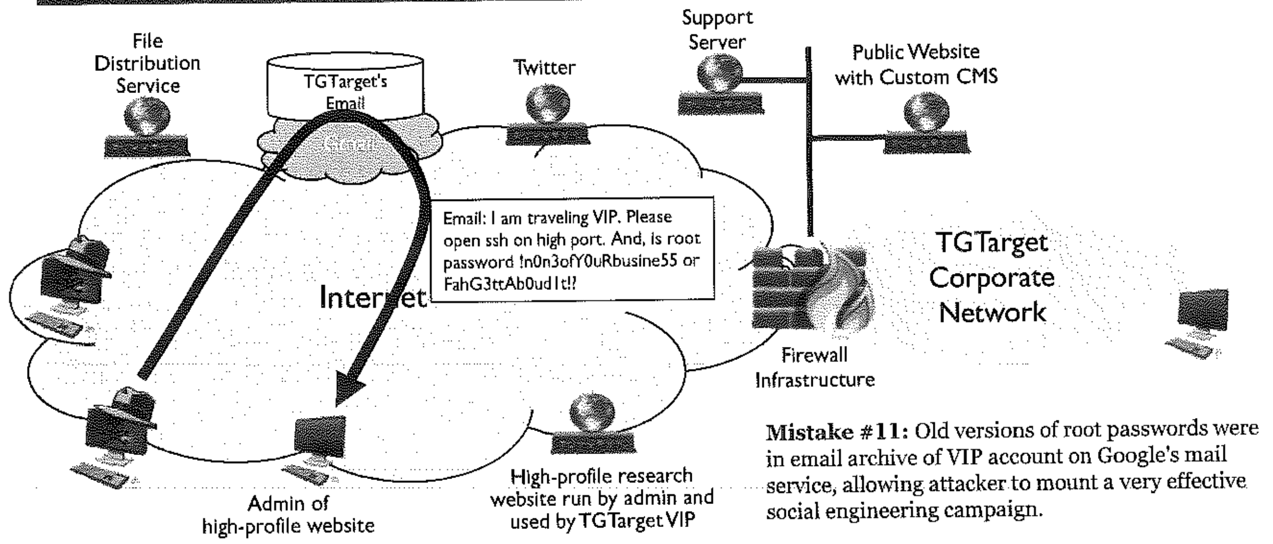
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 140

Now we see a problem that allowed the attackers to radically undermine TGTarget's security. The username and password for the CMS user, which the attackers had used to log in to the Gmail account, had complete administrative rights over TGTarget's entire corporate Google Mail system. Therefore, the attackers could now access any of the Google mail accounts for all TGTarget personnel. The attackers used this access to download a vast archive of sensitive emails from user accounts.

Still, this administrative access offered far more possibility than merely plundering the email history. The attackers could also use this access to change the passwords for these accounts. They could then log in to these Google App Mail accounts for TGTarget personnel and *send* email *from* any of them.

Mistake #10 was giving this account (with the weak 6-character password synchronized with the CMS and support server systems) full administrative privileges over the Google Mail infrastructure. Separation of duties should be applied, with different passwords for these radically different components of the TGTarget infrastructure.

Analyze Email: Learn Admin of High-Profile Site, Root Passwords


File Distribution Service
TGTarget's Email
Gmail
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Email: I am traveling VIP. Please open ssh on high port. And, is root password !n0n3ofY0uRbusine55 or FahG3ttAb0ud1t!?

Mistake #11: Old versions of root passwords were in email archive of VIP account on Google's mail service, allowing attacker to mount a very effective social engineering campaign.

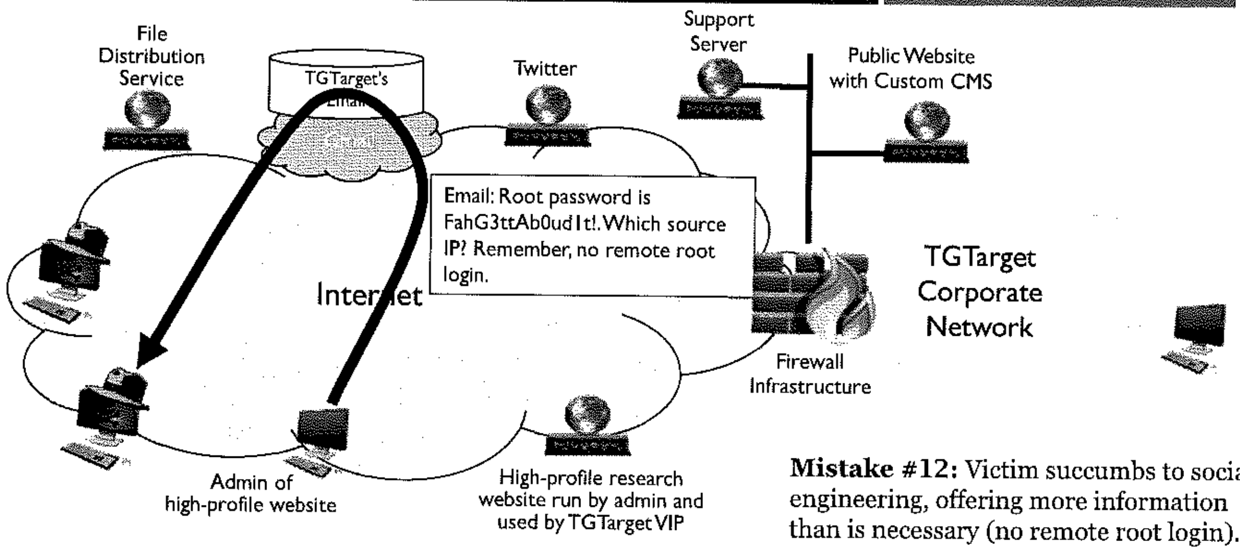
As the attackers browsed through the email history, they focused on the email of the TGTarget VIP. In his mail history, they discovered old emails that contained previous root passwords for the high-profile research website this VIP was affiliated with. They also discovered the email address of the administrator of the high-profile website, someone whom the VIP interacted with on a regular basis. The attackers tried to log in using ssh with these passwords to the high-profile website, but with no success. Either the ssh daemon was filtering their access, these old passwords were no good, or the website didn't allow ssh access for root-level accounts.

Still, leaving old root-level passwords in email history is dangerous and represents **Mistake #11**. With these old versions of passwords available to the attacker, a very effective social engineering campaign can be mounted. Password information should not be exchanged via email, and, if it is, these emails should be immediately destroyed.

Social engineering via email was the attackers' next tactic. Using their access to the VIP's email account, they sent email from this VIP's account to the admin of the high-profile website. In the email, the attackers claimed to be the VIP on a business trip to another country. They said they were in a hurry, and needed access to ssh on the website on some unusual high-numbered port. Based on their knowledge of previous root passwords of the website, they asked the administrator which of two old passwords was still in use on the website. Containing these fairly complex passwords that were actual root passwords at some time in the past on this website made the email very convincing.

Email Response

TGTarget

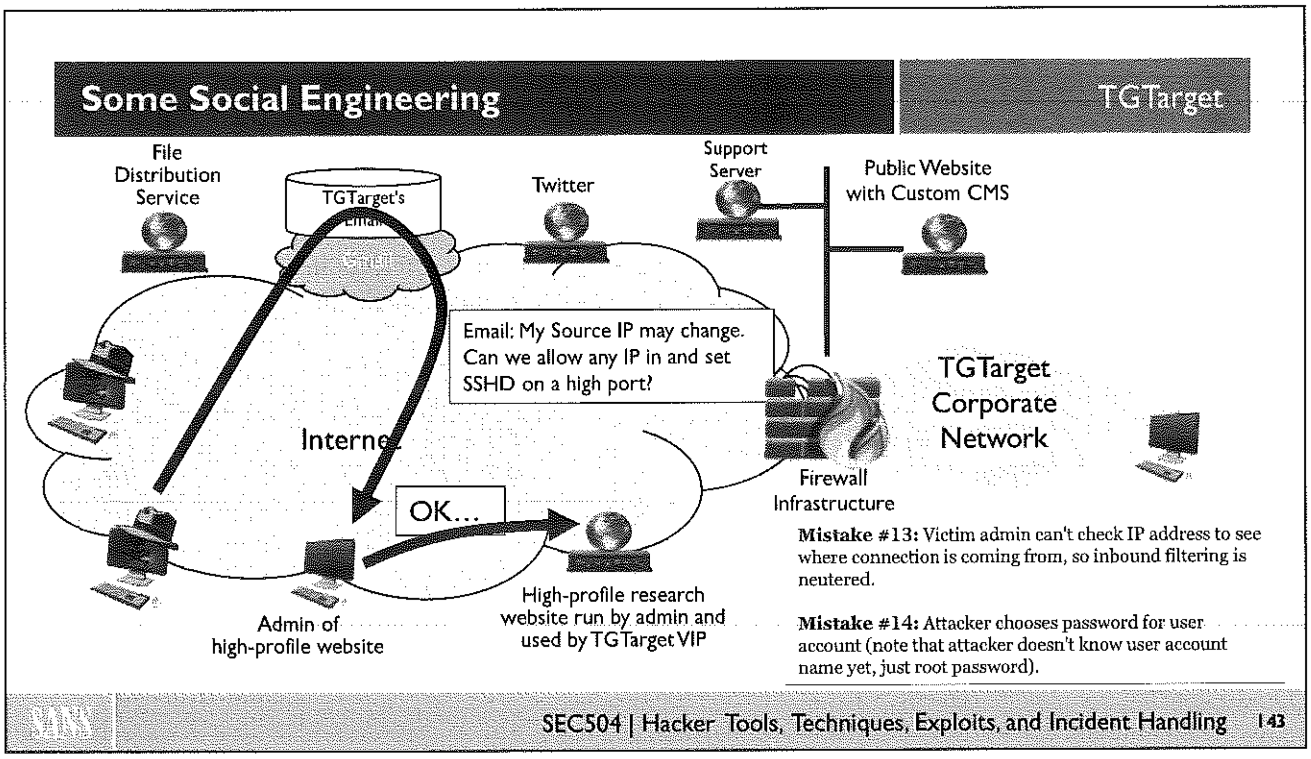

File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Email: Root password is FahG3ttAb0ud1t!. Which source IP? Remember, no remote root login.

Mistake #12: Victim succumbs to social engineering, offering more information than is necessary (no remote root login).

In the next step, the admin of the high-profile site succumbs to the social engineering attempt. He confirms that the root password still had the same value as one of the passwords from the VIP email history, a fairly complex and long password. The admin's response email asks the VIP which source IP address he'll be coming from so that he can limit the inbound ssh access to just that address. What's more, the admin reminds the VIP (actually the attacker) that no remote root login is allowed on the web server.

Mistake #12 was succumbing to this social engineering attack while offering more information back than the attacker actually asked for (the reminder about no remote root login). Given that the attacker possessed the real root password (a fairly complex password), though, made this mistake understandable.



Some Social Engineering

TGTarget

File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Email: My Source IP may change. Can we allow any IP in and set SSHD on a high port?
TGTarget Corporate Network
Internet
OK...
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #13: Victim admin can't check IP address to see where connection is coming from, so inbound filtering is neutered.

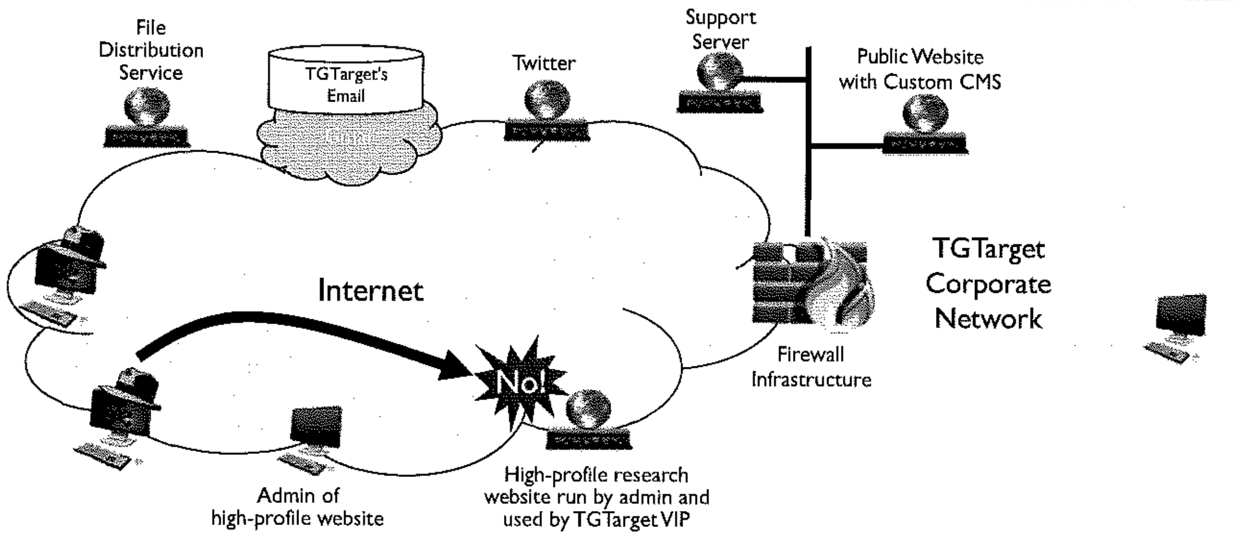
Mistake #14: Attacker chooses password for user account (note that attacker doesn't know user account name yet, just root password).

Next, the attackers further their social engineering attack by responding to the admin's email. Still pretending to be the VIP, they explain that their source IP address might change and that they don't have a fixed address to come from. They request that the admin allow any source IP address in, but to use an obscure destination port for ssh to listen on. In **Mistake #13**, the admin allowed inbound access to the ssh server listening on this port, from any source IP address. Thus, the attacker convinced the victim admin to essentially remove any network filtering from ssh based on IP address. Furthermore, without a source IP address from the attacker, the victim admin couldn't research the geographic location of the IP address to see that it was coming from a completely different part of the world than where the VIP was currently traveling. The admin should have insisted on a source IP address so he could verify that it was a reasonable request (and perhaps a telephone call to confirm the request).

In this email, the attacker went further, telling the admin to change the user account password (not the root password) to "changeme123" so that he could log in and quickly change it. This is **Mistake #14**: The attacker chose the password for an account on the target system, and the admin simply set the password to this value. The admin should have chosen a password, and it should have been stronger than changeme123.

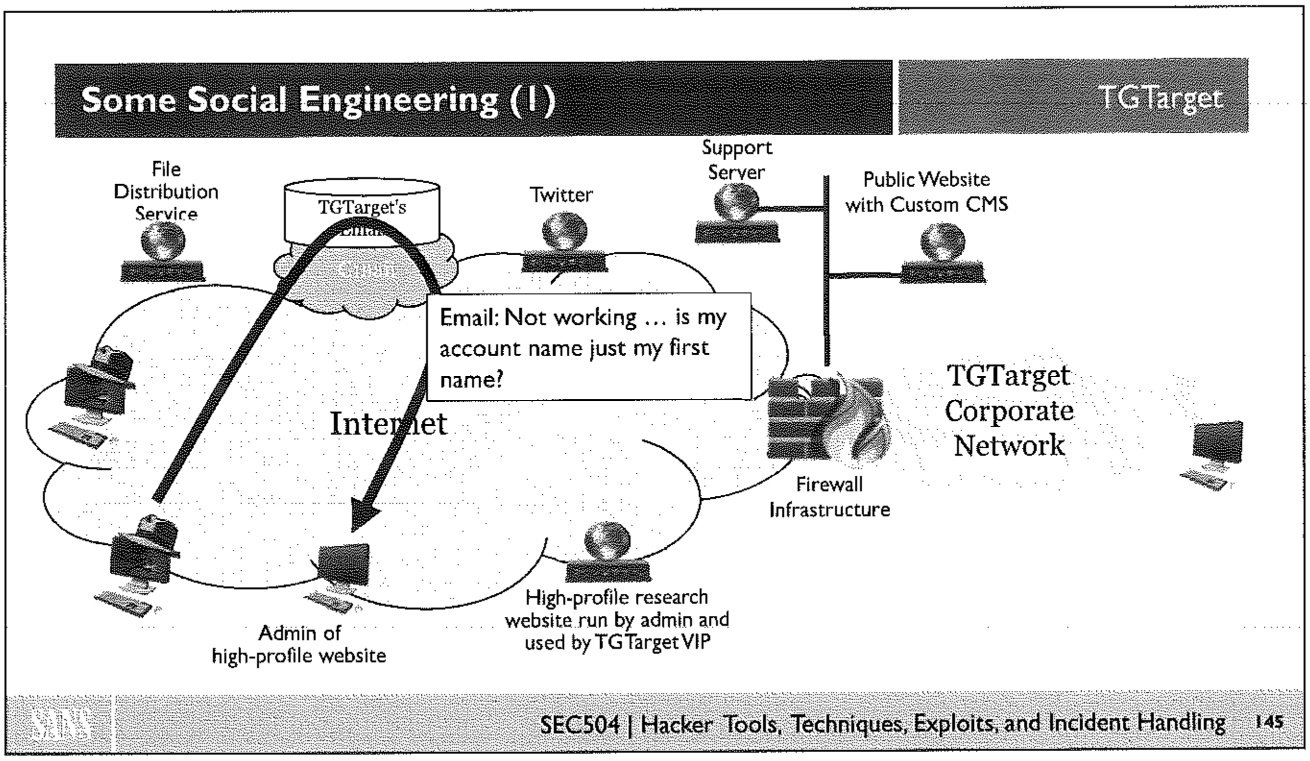
Attacker Tries to Log In but Fails (No ssh in a UID 0)

TGTarget


File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
No!
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 144

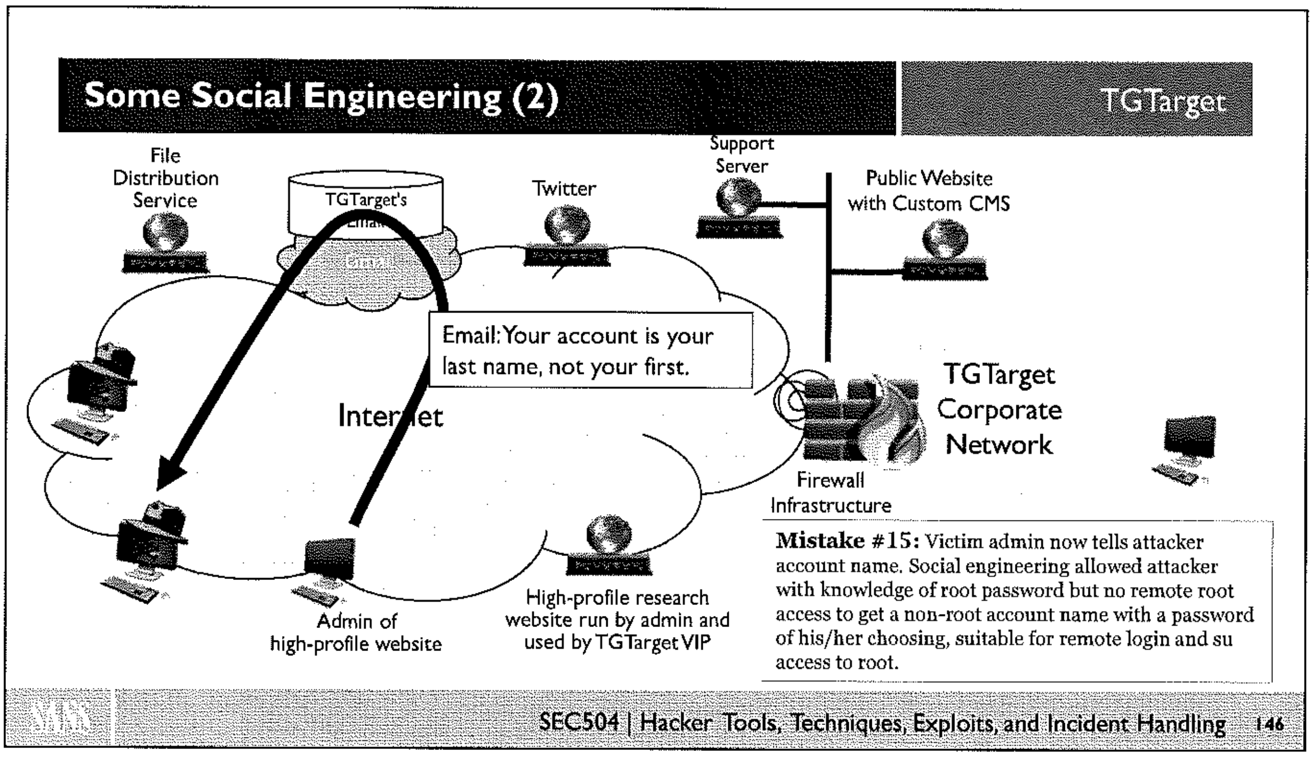
Now the attackers try to log in to the high-profile website, guessing several usernames and trying the password of "changeme123." Yet, for some reason, it just doesn't seem to work for the attackers. Perhaps more social engineering is needed.



Some Social Engineering (1)

TGTarget
File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Email: Not working … is my account name just my first name?
Internet
TGTarget Corporate Network
Firewall Infrastructure
High-profile research website run by admin and used by TGTarget VIP
Admin of high-profile website
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 145

The attackers send yet another email to the admin of the website. This message says that the ssh access is not working and asks whether the account name for the target machine is just the VIP's first name.



Some Social Engineering (2)

TGTarget
File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Email: Your account is your last name, not your first.

Mistake #15: Victim admin now tells attacker account name. Social engineering allowed attacker with knowledge of root password but no remote root access to get a non-root account name with a password of his/her choosing, suitable for remote login and su access to root.

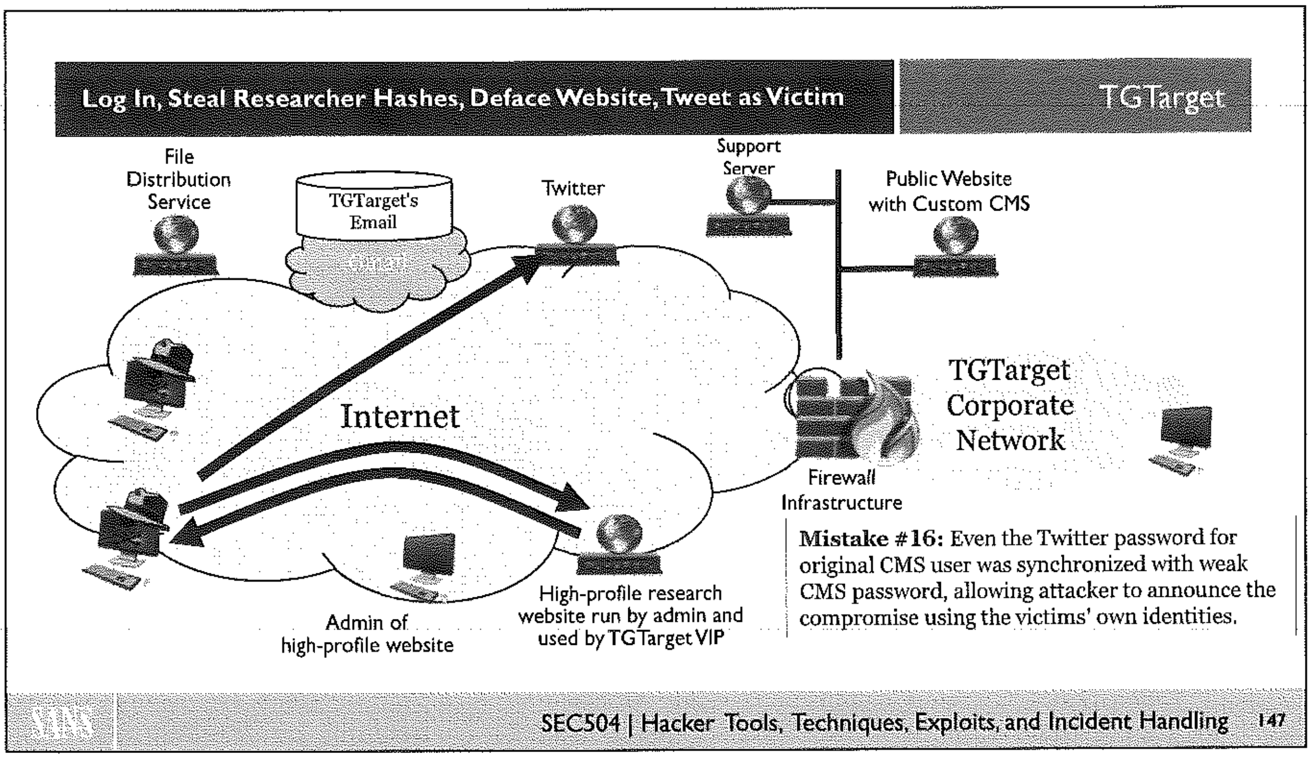
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 146

The admin responds telling the attackers that the account name is the VIP's last name, not the first name. **Mistake #15** was revealing the username for the account, giving the attackers everything they needed to log in to the high-profile website.

It is important to analyze the flow of this social engineering attack. The attackers started knowing only potential root passwords for the high-profile website. Armed with only this knowledge and access to the VIP's email account, they were able to:

1) Verify which of those root passwords was still in use.
2) Learn that remote root login was blocked for ssh.
3) Get inbound ssh access from any source IP address.
4) Get that ssh access on a specific port the admin told them about.
5) Get a password reset for a non-root account to a value of their choosing.
6) Determine the username for the non-root account.

With all of this information and access granted to the attackers, they now can remotely access the website via the non-root account using ssh and then su to get full root-level access to the machine.



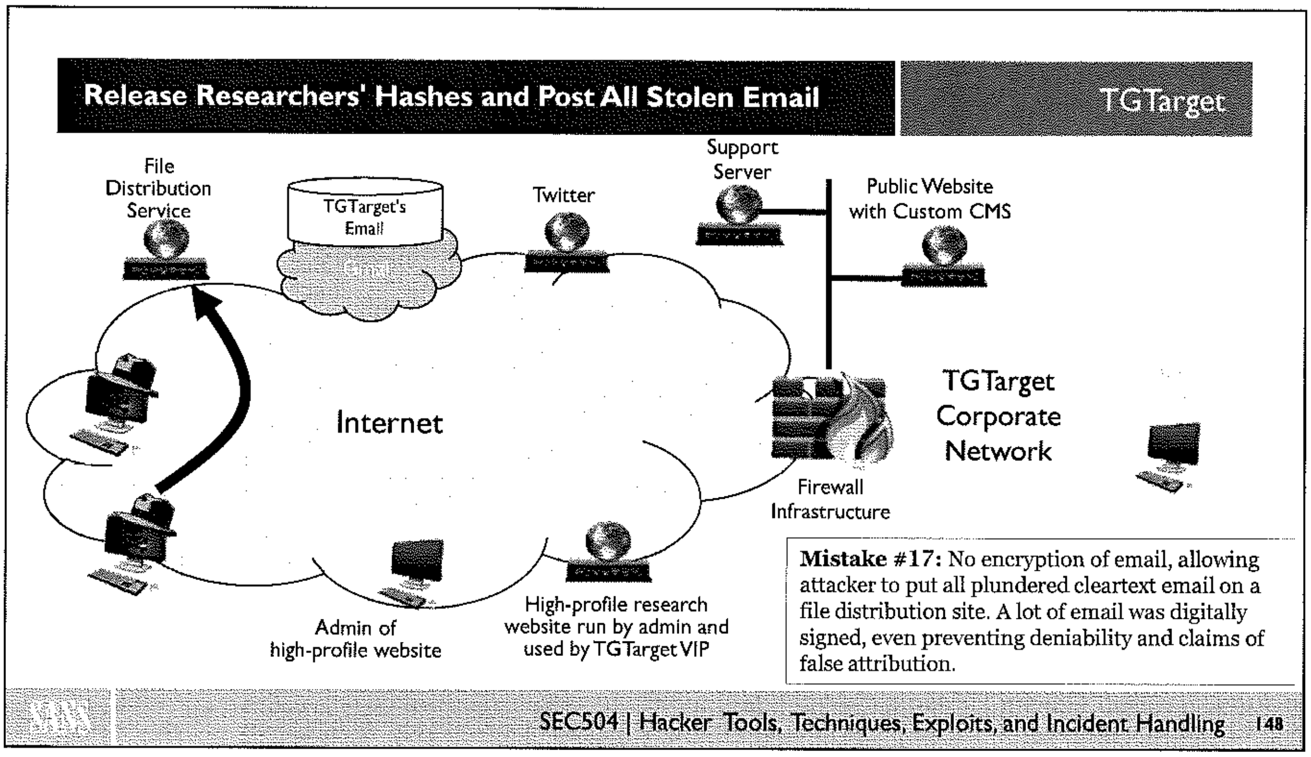
Log In, Steal Researcher Hashes, Deface Website, Tweet as Victim

TGTarget
File Distribution Service
TGTarget's Email
Gmail
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

Mistake #16: Even the Twitter password for original CMS user was synchronized with weak CMS password, allowing attacker to announce the compromise using the victims' own identities.

The attackers now ssh into the high-profile research website. They steal the hashes for accounts on this website that belong to researchers around the world.

With their domination of TGTarget's environment nearly complete, the attackers now move onto the embarrassment phase of their goal. They start by defacing the research website, announcing their attack. They then log in to the Twitter account of the original CMS user. This user had synchronized even his Twitter password to the passwords used elsewhere in the environment, leading to **Mistake #16**. The attackers were, therefore, able to announce their conquest using the organization's own communication mechanisms with the public.



Release Researchers' Hashes and Post All Stolen Email

TGTarget
File Distribution Service
TGTarget's Email
Twitter
Support Server
Public Website with Custom CMS
Internet
TGTarget Corporate Network
Firewall Infrastructure
Admin of high-profile website
High-profile research website run by admin and used by TGTarget VIP

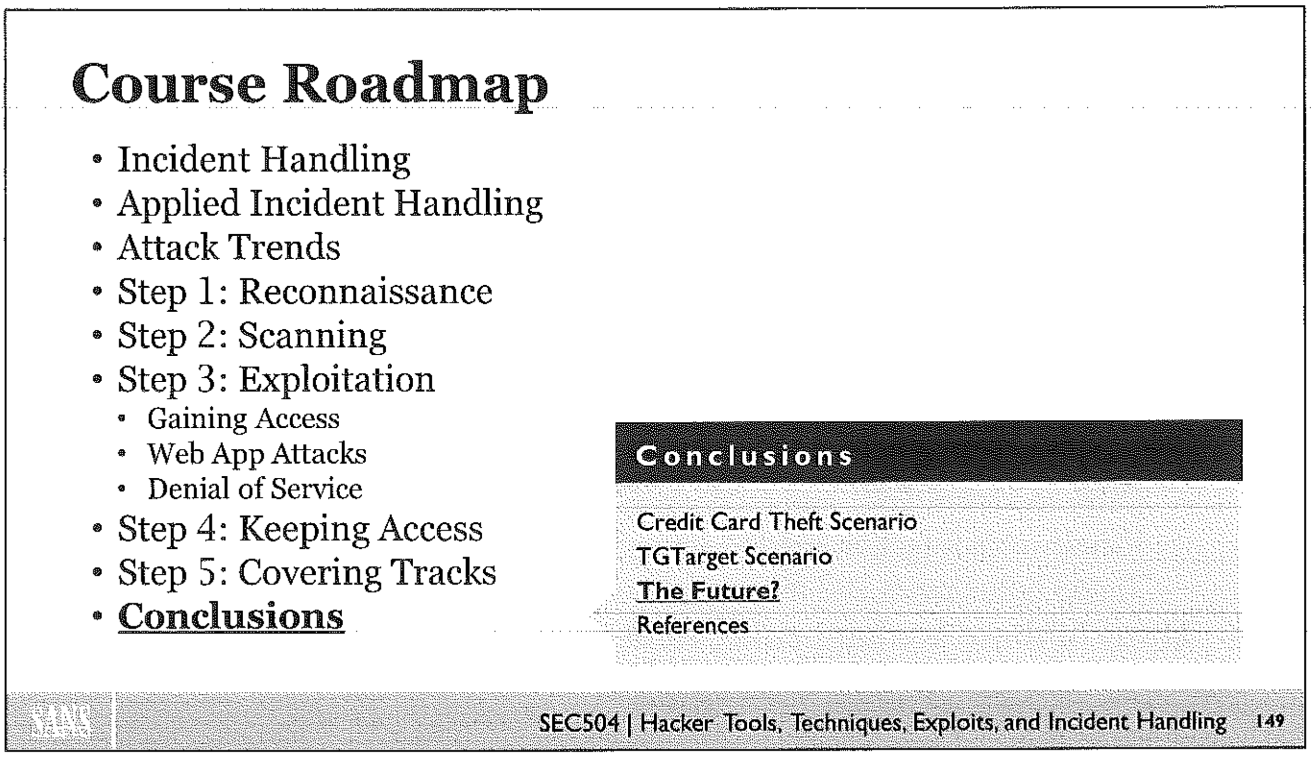
Mistake #17: No encryption of email, allowing attacker to put all plundered cleartext email on a file distribution site. A lot of email was digitally signed, even preventing deniability and claims of false attribution.

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 148

Finally, the attackers package up all of the information retrieved from the target organization, especially the email archive, and post it on a file distribution service, making it available to the world. They also release all of the hashes of researchers retrieved from the high-profile research website, leading to further embarrassment.

Compounding the public release of the email archive was the fact that very little TGTarget email was encrypted, **Mistake #17**. Thus, the press and others were able to read the email and the secrets it contained about TGTarget's business. Furthermore, although the email was generally not encrypted, some of it was digitally signed by TGTarget employees, allowing reporters and others to verify the integrity and authenticity of the leaked emails, making it difficult for TGTarget personnel to deny the veracity of these messages.

In the end, TGTarget made a series of mistakes that led to the compromise of their sensitive information. Your organization might not make some of these mistakes, perhaps leading you to think a situation like this might not occur for your organization. Unfortunately, it is still possible, even if you don't make *all* of these mistakes. Even if you make a small subset of these errors, a determined attacker might still find other problems and gain advantage over your organization. That's why careful detection mechanisms and thorough response strategies have to be designed and implemented in advance.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- Step 5: Covering Tracks
- **Conclusions**

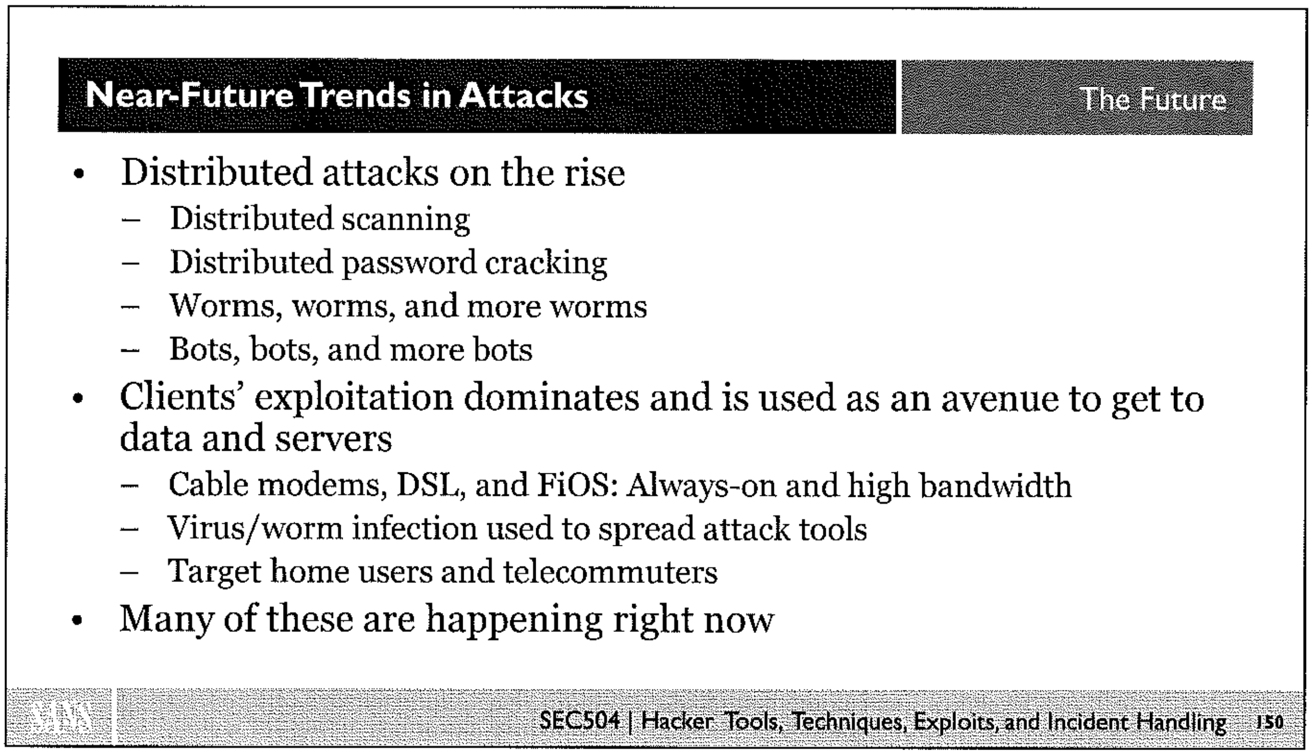
Conclusions

Credit Card Theft Scenario
TGTarget Scenario
The Future?
References

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 149

We are now back to our roadmap and the home stretch for the course. Let's conclude the course by discussing where computer attacks are headed in the future, as well as several useful references.

Where are all these attacks leading us and how will things evolve over the next several years? Let's explore this topic in more detail.



Near-Future Trends in Attacks

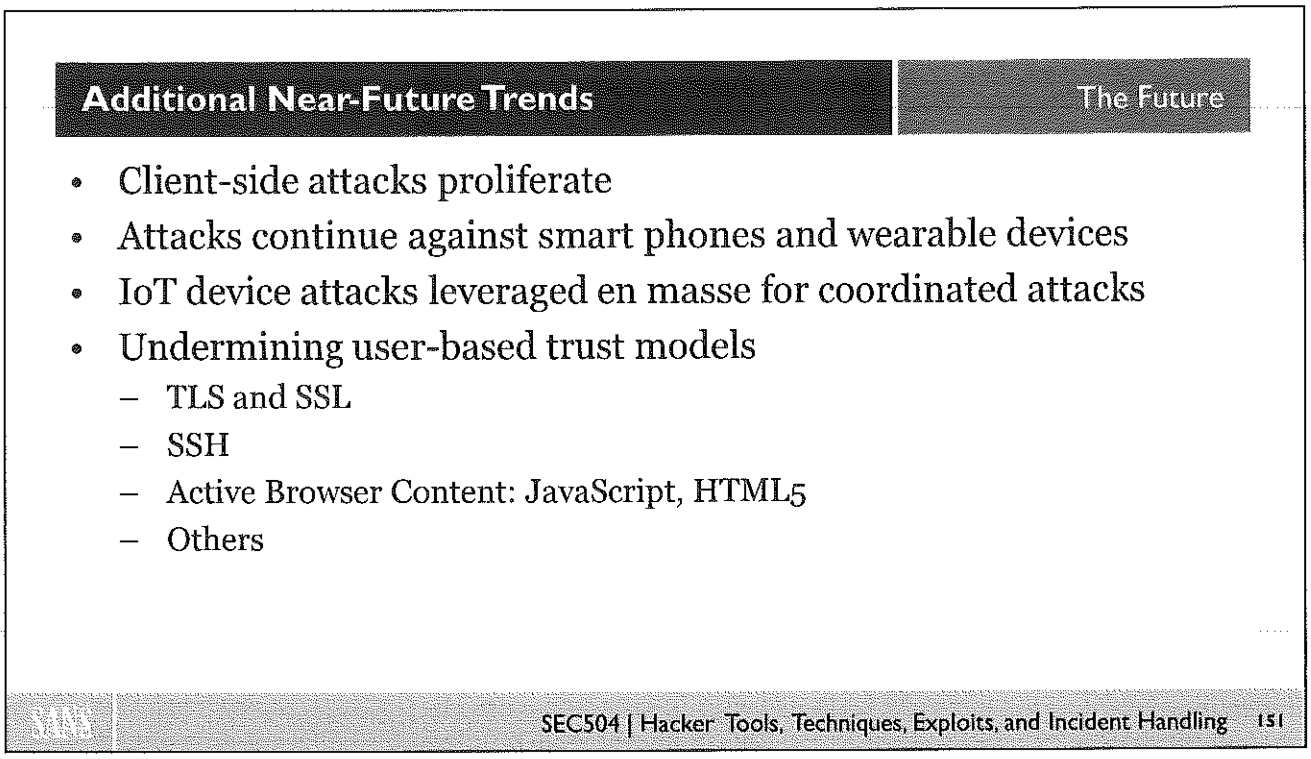
The Future

- Distributed attacks on the rise
  - Distributed scanning
  - Distributed password cracking
  - Worms, worms, and more worms
  - Bots, bots, and more bots
- Clients' exploitation dominates and is used as an avenue to get to data and servers
  - Cable modems, DSL, and FiOS: Always-on and high bandwidth
  - Virus/worm infection used to spread attack tools
  - Target home users and telecommuters
- Many of these are happening right now

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 150

Distributed attacks help attackers by speeding things up, allowing them to consume more resources, and making them harder to detect.

In addition, attacks are focused increasingly on clients rather than servers. Server attacks still occur, but exploitation of clients is extremely popular and growing even more so.



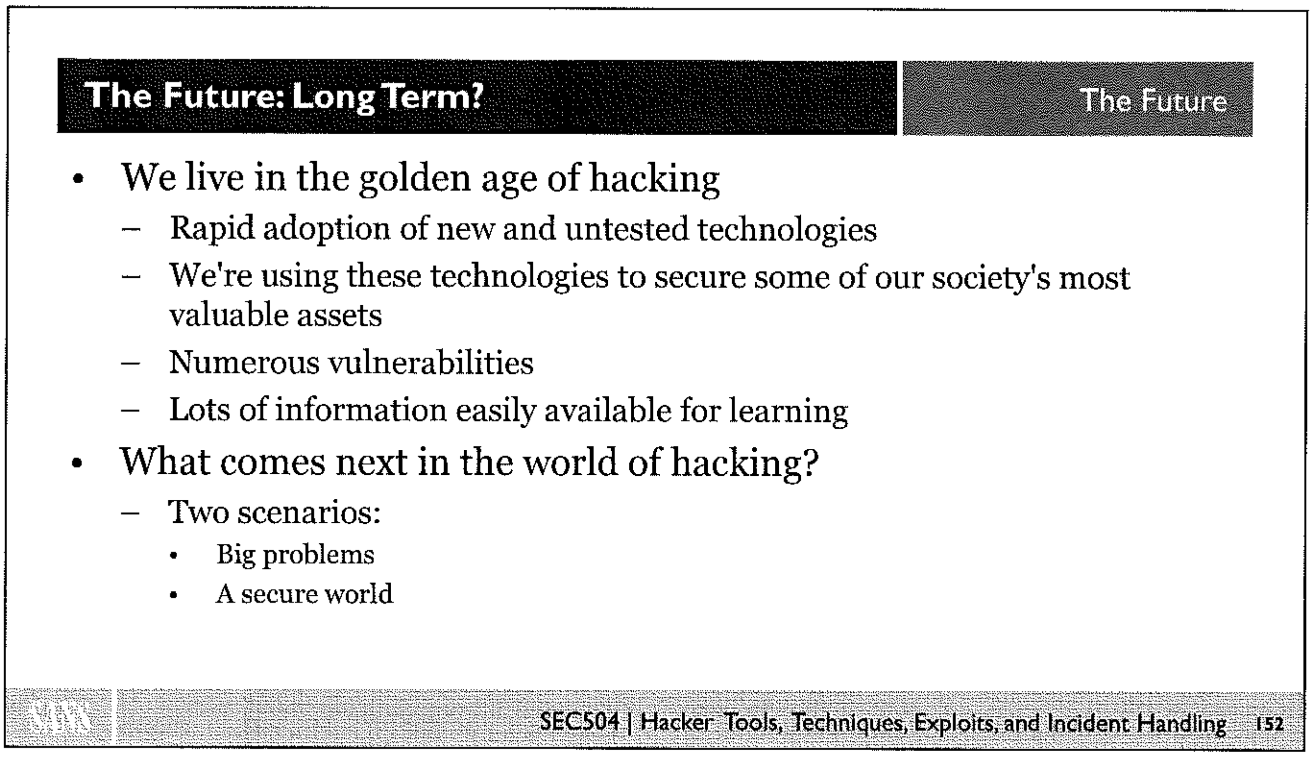
Additional Near-Future Trends

- Client-side attacks proliferate
- Attacks continue against smart phones and wearable devices
- IoT device attacks leveraged en masse for coordinated attacks
- Undermining user-based trust models
  - TLS and SSL
  - SSH
  - Active Browser Content: JavaScript, HTML5
  - Others

We are seeing many attacks against client-side software, such as browsers, music players, and image-viewing tools. This will continue to be a dominant vector for some time.

Also, as more features and power are added to smaller platforms like smart phones and wearable devices, attackers will likely increasingly target these types of tools. With full-fledged operating systems and useful development environments, attackers are starting to create malicious code for cell phones. Watch for more of that in the near future.

User-based trust models are just plain scary because users do not understand what's going on from a technology perspective. Whenever we have a technology that pops up a dialog box asking the user a question (in effect, "Something really dangerous is about to happen; do you want to let it?"), we are going to have trouble. Nevertheless, that is what we have created with SSL, SSH, HTML5, and several other technologies. We either have to move users out of the trust loop (unlikely) or build our tools so that they understand what the implications are.



The Future: Long Term?

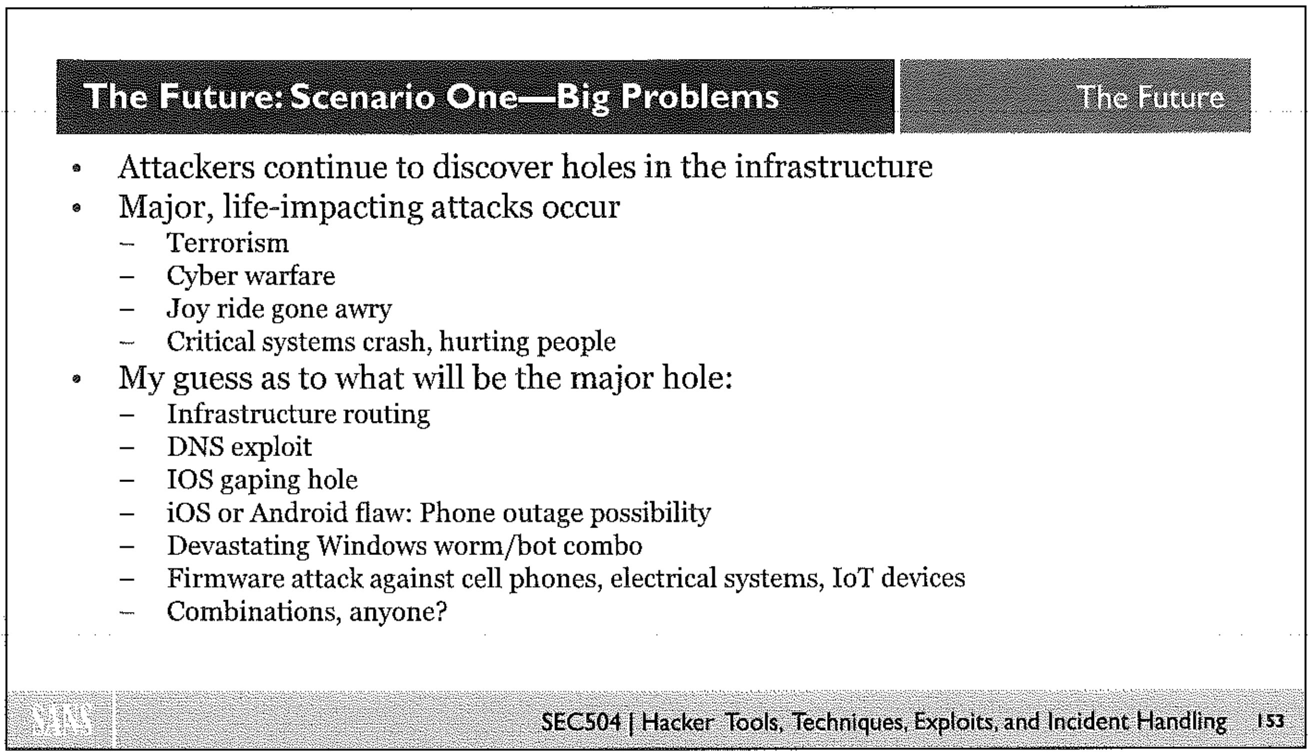
The Future

- We live in the golden age of hacking
  - Rapid adoption of new and untested technologies
  - We're using these technologies to secure some of our society's most valuable assets
  - Numerous vulnerabilities
  - Lots of information easily available for learning
- What comes next in the world of hacking?
  - Two scenarios:
    - Big problems
    - A secure world

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 152

Currently, an enormous number of new vulnerabilities are discovered every week. Software is rolled out into production when it is little better than alpha code. Vulnerabilities are widely publicized, despite the long duration it takes for many software vendors to release fixes. It usually takes even longer for companies to deploy effective countermeasures and patches. In addition, crackers have teamed up around the globe to share information and coordinate attacks. It is the golden age of hacking.

We were once having a deeply philosophical talk with an old-time security guru at Bellcore. We asked him what the future holds. He responded, "There'll either be massive attacks and our jobs will be very much in demand… Or, the vendors will get their act together, and we'll become the electronic equivalent of the night watch person." Next we explore these two scenarios in more detail.



The Future: Scenario One—Big Problems

The Future

- Attackers continue to discover holes in the infrastructure
- Major, life-impacting attacks occur
  - Terrorism
  - Cyber warfare
  - Joy ride gone awry
  - Critical systems crash, hurting people
- My guess as to what will be the major hole:
  - Infrastructure routing
  - DNS exploit
  - IOS gaping hole
  - iOS or Android flaw: Phone outage possibility
  - Devastating Windows worm/bot combo
  - Firmware attack against cell phones, electrical systems, IoT devices
  - Combinations, anyone?

SANS

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 153

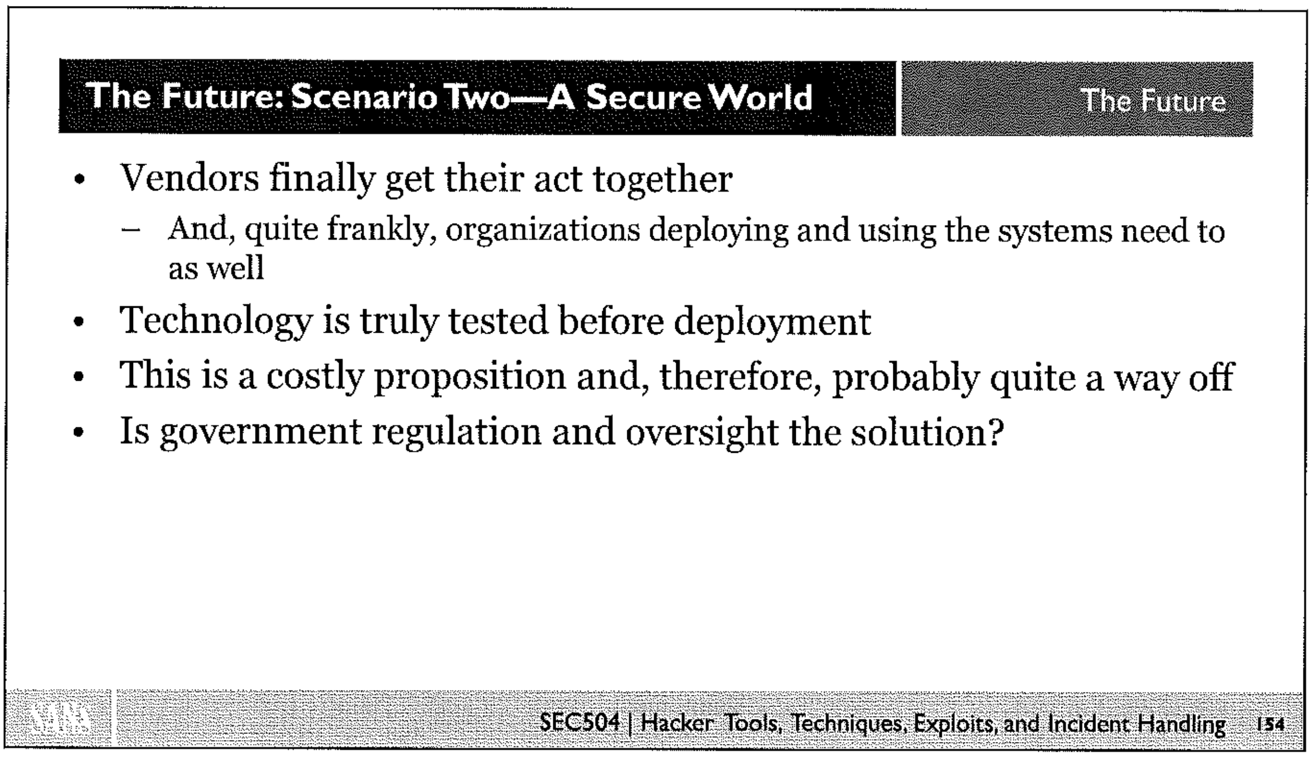
We don't want to be prophets of doom; we don't want to overhype the situation. However, there are some worrisome possible future scenarios. We've already seen precursors to this with the Morris Worm and other significant attacks. As we rely more on the internet and other computer-based technologies, we become much more vulnerable to attacks against our gadgets.

Many countries actively develop cyber warfare capabilities. Likewise, terrorist organizations have taken to using computer technology to carry out their agenda. If your organization is a critical component of the infrastructure of the world or your country, you might be a target.

Also, we've seen some cases involving a youth who plays with a new worm and accidentally releases it into the wild, causing much damage. In the future, we could face a massive incident where a computer joy ride goes wrong, crashing systems.

There are many points of attack against the worldwide computing infrastructure. A major hole in any of the technologies listed in this slide could result in a significant amount of damage. For example, if attackers are able to manipulate infrastructure routing, they can steal massive amounts of data or shut down the internet entirely. If they find a flaw in the Border Gateway Protocol (BGP4), they could cause major disruptions. Similarly, the domain name system underlies so much of internet functionality. If attackers discover a major flaw in DNS systems, they can manipulate DNS to conduct numerous types of attacks. In addition, so many routers on the internet and on enterprise networks use Cisco's Internetwork Operating System (IOS). Or a flaw in hundreds of millions of mobile devices, such as Apple's iOS or Google's Android devices, could cause major phone outages. If a significant flaw is found in IOS, we would have major problems. A worm/bot combo exploiting Windows machines could also cause grief. Or attacks against the firmware of cell phones or electrical systems could cause outages with major implications.

In the Robert Tappan Morris, Jr. worm incident of 1988, we saw an attack that exploited vulnerabilities in a couple of different services, namely Finger and Sendmail. It is possible that attackers could find and exploit vulnerabilities in several of these newer technologies as well, such as a bug in infrastructure routing and a hole in DNS, that could be exploited together with dramatic results. This scenario is certainly not a cheery one.



The Future: Scenario Two—A Secure World

The Future

- Vendors finally get their act together
  - And, quite frankly, organizations deploying and using the systems need to as well
- Technology is truly tested before deployment
- This is a costly proposition and, therefore, probably quite a way off
- Is government regulation and oversight the solution?

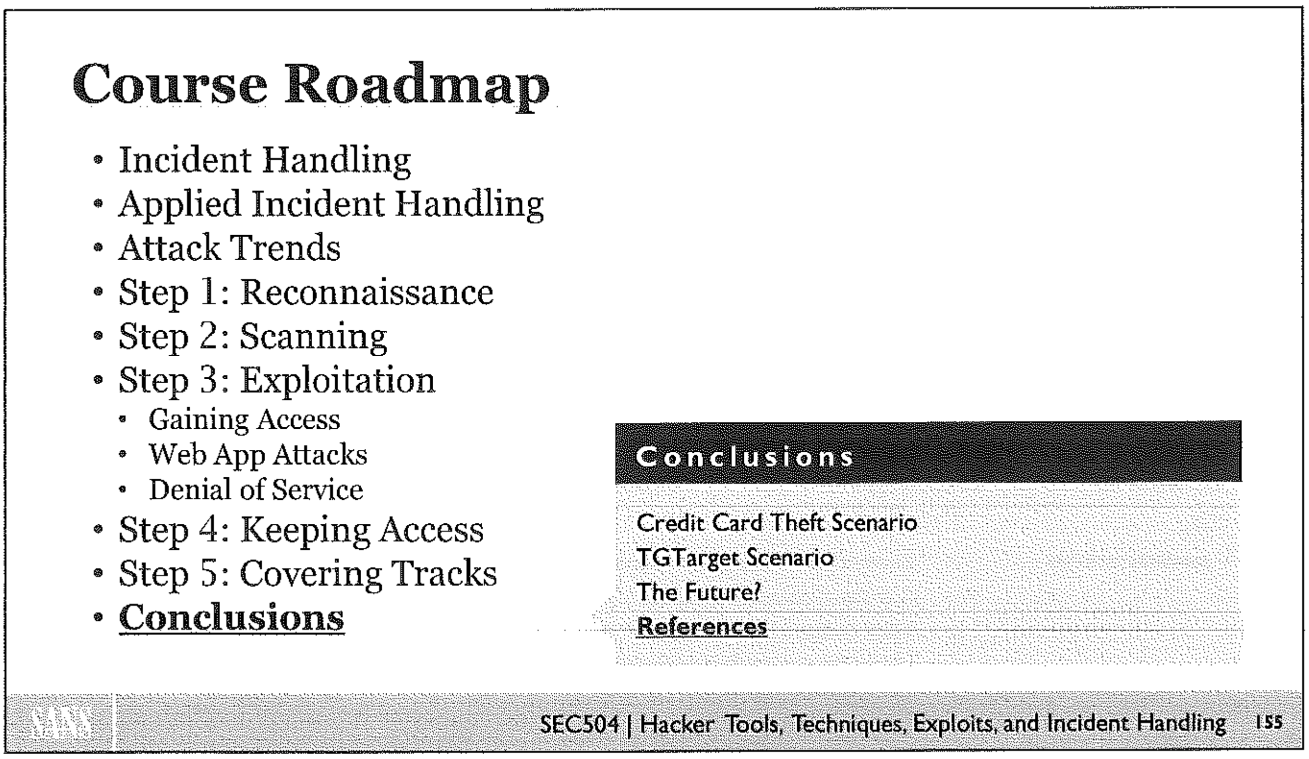
SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 154

At some point in the future, software vendors, governments, companies, and other organizations will devote the resources necessary to be truly secure. Software will be tested before it is put into production. Security will be built into the requirements, design, implementation, and testing of our hardware and software components.

Unfortunately, this is not the trajectory we're on with software release cycles shrinking every day and the rush to be first to market.

Still, in the long term (which, in internet years, might be less than a decade away), we will likely be much more secure. Think about the early automotive revolution, with crank-started engines and dangerous roadways. Our IT infrastructure is the equivalent of the hand-cranked automobile era. In the future, we will have built-in security controls to help manage risks better. Using technology, we will lower the risk to some acceptable value. At that point, we can use insurance to handle the residual risk.

Again, we believe computer security will become much more like what we do with automobiles. We build safe highways, wear our seat belts, and use air bags. But just in case there's an accident, we buy insurance.



Course Roadmap

- Incident Handling
- Applied Incident Handling
- Attack Trends
- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Exploitation
  - Gaining Access
  - Web App Attacks
  - Denial of Service
- Step 4: Keeping Access
- Step 5: Covering Tracks
- **Conclusions**

Conclusions

Credit Card Theft Scenario
TGTarget Scenario
The Future?
References

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 155

Throughout this course, we discussed numerous attack techniques from classics, such as IP address spoofing, to more recent trends, such as kernel manipulation and polymorphic code. It is important for you to keep up to speed with attack techniques and tools because they continue to evolve. The references we cover over the next several slides are helpful in keeping in touch with what is happening with computer attack tools.

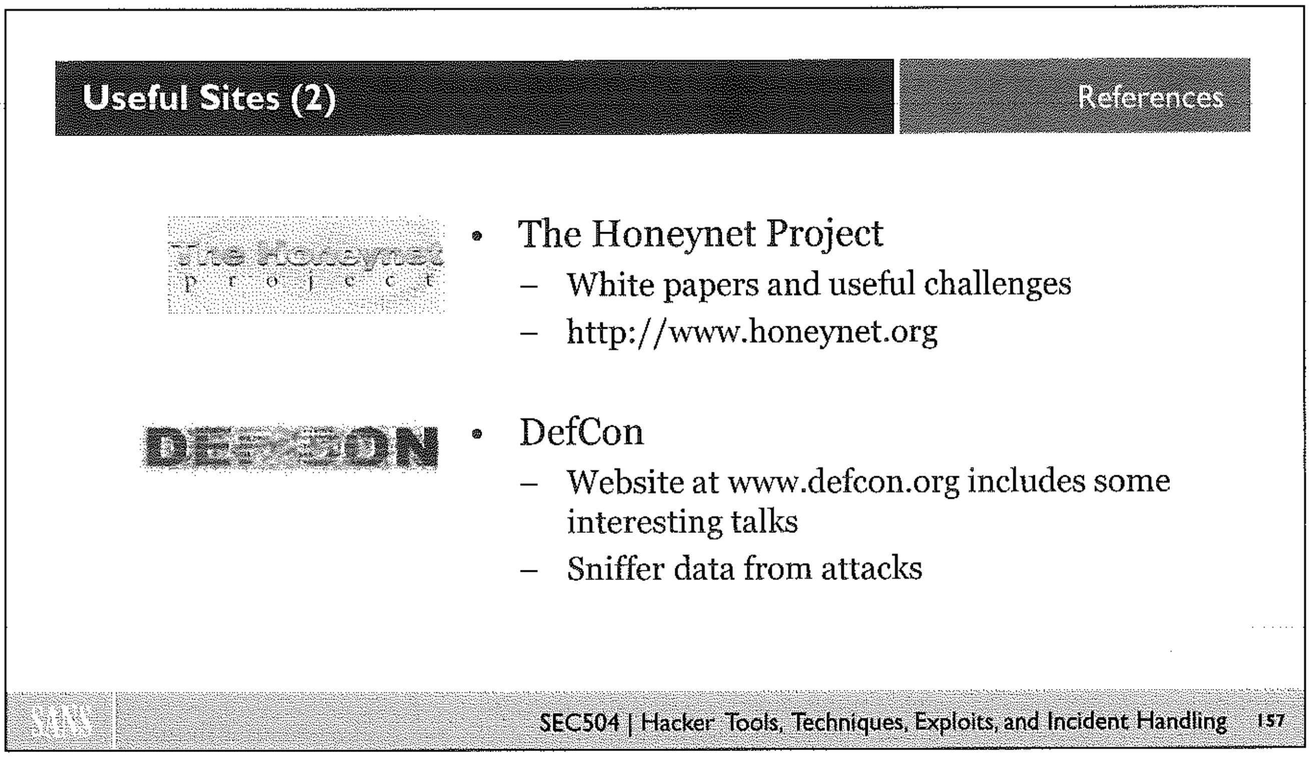
Useful Sites (1)

References

- *Internet Storm Center*
  - Handlers Diary every day
  - Interesting new threats, vulnerabilities, attacks, and defensive tips
  - http://isc.sans.edu
- *Packet Storm*
  - Every day or two, new exploit code and tools are posted
  - A tremendous amount of hacking and security information
  - Very useful; I read it every day
  - http://www.packetstormsecurity.com

The *Internet Storm Center* is a wonderful source of interesting security vulnerability and defense information. I read it every single day, focusing on the handlers diary.

The *Packet Storm* security site is incredibly useful. Whenever a new attack tool is released, its authors usually send a message to *Packet Storm*. *Packet Storm* maintains a huge inventory of attack and defense tools on its website. Every day or two, new exploits and attack tools are listed in its "New Tools" section. The site also includes a useful search engine to find tools and capabilities. *Packet Storm* is now run as a nonprofit organization and is an indispensable resource. Whenever looking for a specific tool, check out *Packet Storm* first.



Useful Sites (2)

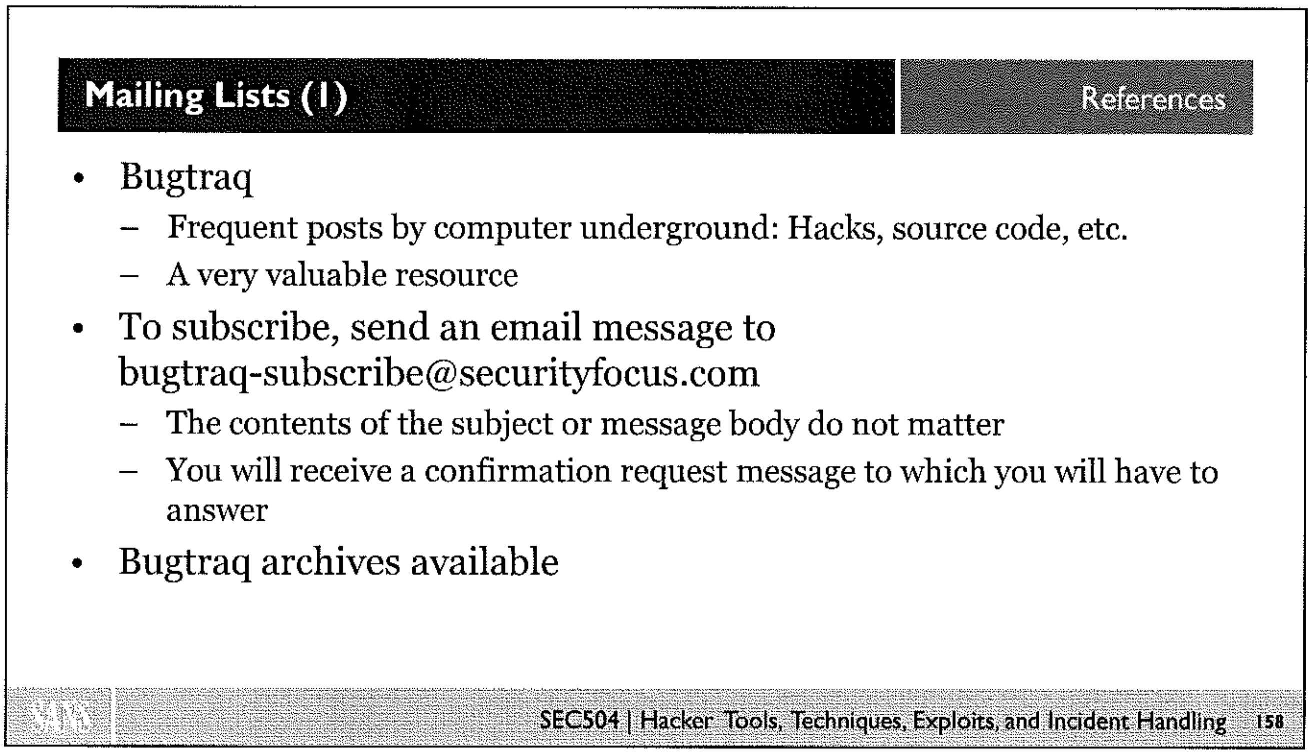
References

- The Honeynet Project
  - White papers and useful challenges
  - http://www.honeynet.org
- DefCon
  - Website at www.defcon.org includes some interesting talks
  - Sniffer data from attacks

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 157

The Honeynet Project website is also useful. You should look for the challenge of the month to get a feel for what the bad guys are up to.

DefCon has the best-attended hacker conference every summer in Las Vegas. Its site has some interesting archives from earlier conferences, including sniffer data from the Capture the Flag contest it holds every year.



Mailing Lists (1)

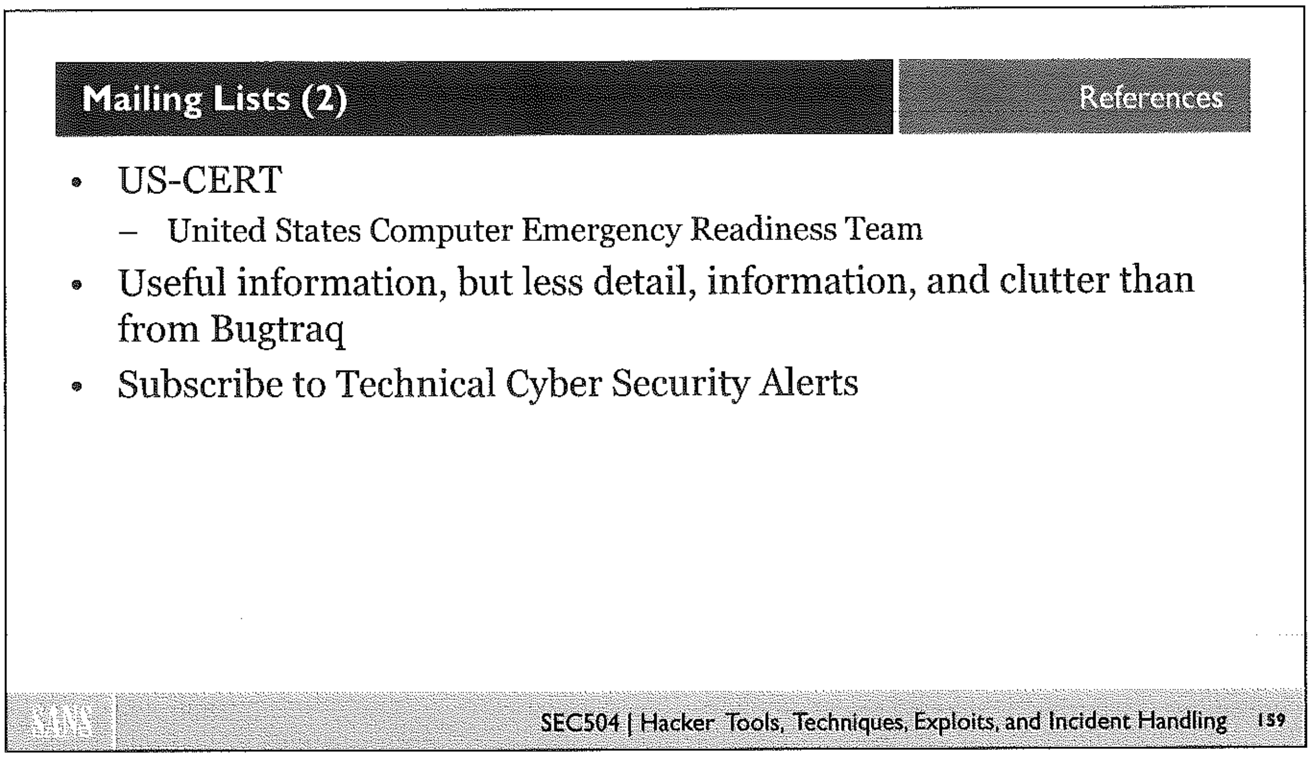
References

- Bugtraq
  - Frequent posts by computer underground: Hacks, source code, etc.
  - A very valuable resource
- To subscribe, send an email message to bugtraq-subscribe@securityfocus.com
  - The contents of the subject or message body do not matter
  - You will receive a confirmation request message to which you will have to answer
- Bugtraq archives available

SANS SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 158

The Bugtraq mailing list is perhaps the single most valuable resource about computer attacks and defenses that is available today. It's definitely worth your time if you need to keep up on the latest and greatest attacks and defenses. It gets a large amount of traffic, ranging from a dozen to upward of 50 messages per day, so it isn't for the faint of heart. However, it is well-moderated and most of the information is useful. You can subscribe to Bugtraq using information shown on this slide.

The archives for the Bugtraq mailing list are available at http://www.securityfocus.com (in the forums area under Bugtraq).



Mailing Lists (2)

References

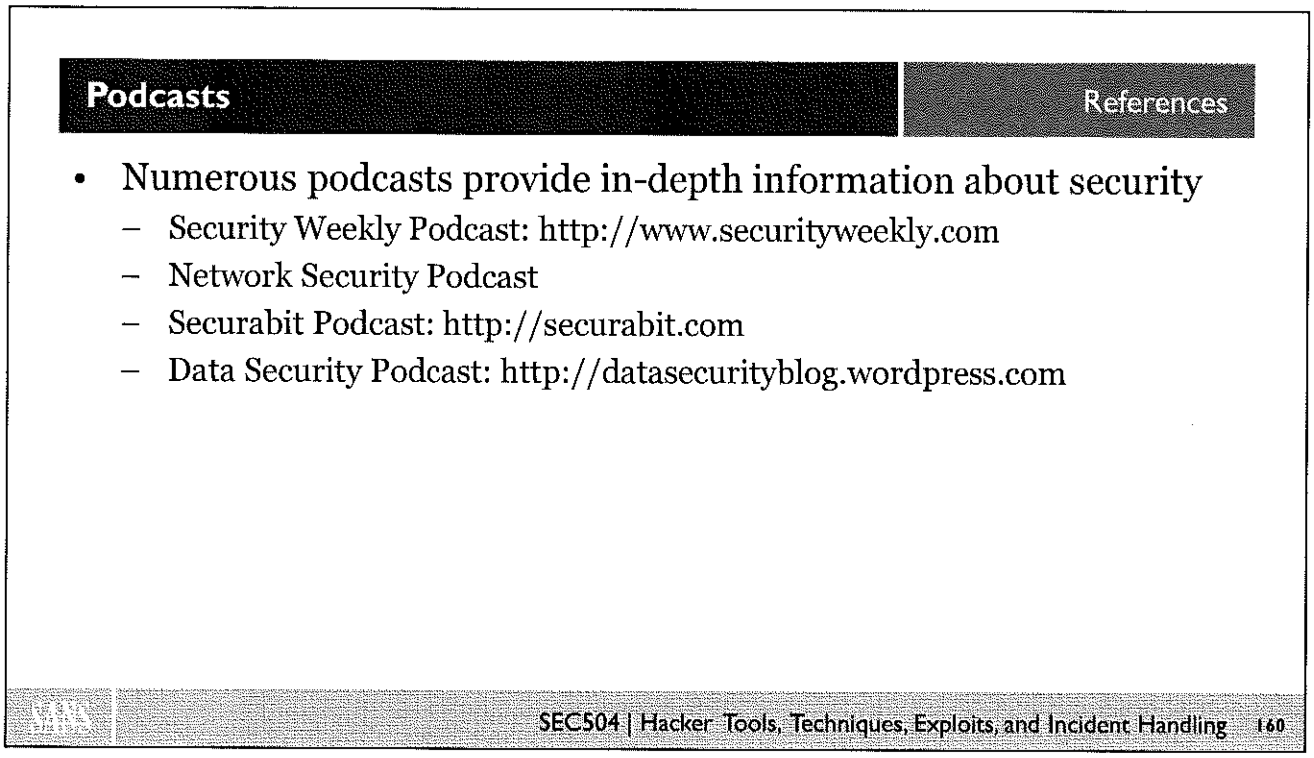
- US-CERT
  - United States Computer Emergency Readiness Team
- Useful information, but less detail, information, and clutter than from Bugtraq
- Subscribe to Technical Cyber Security Alerts

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 159

If you are pressed for time and cannot monitor the other lists, the US-CERT mailing list is sort of a bare minimum of advisory information for system administrators and security professionals.

Reading US-CERT Technical Cyber Security Alerts, you will get all of the major, ultra-important vulnerabilities and attack scenarios in nicely written summary advisories. The advisories will tell you how to defend yourself and what to look out for. However, you will likely get them much later than folks who read Bugtraq, allowing attackers to get a jump on you. On some occasions, however, CERT does release advisories before they appear in other forums.

Subscribe to the Technical Cyber Security Alerts at http://www.us-cert.gov/cas/signup.html.



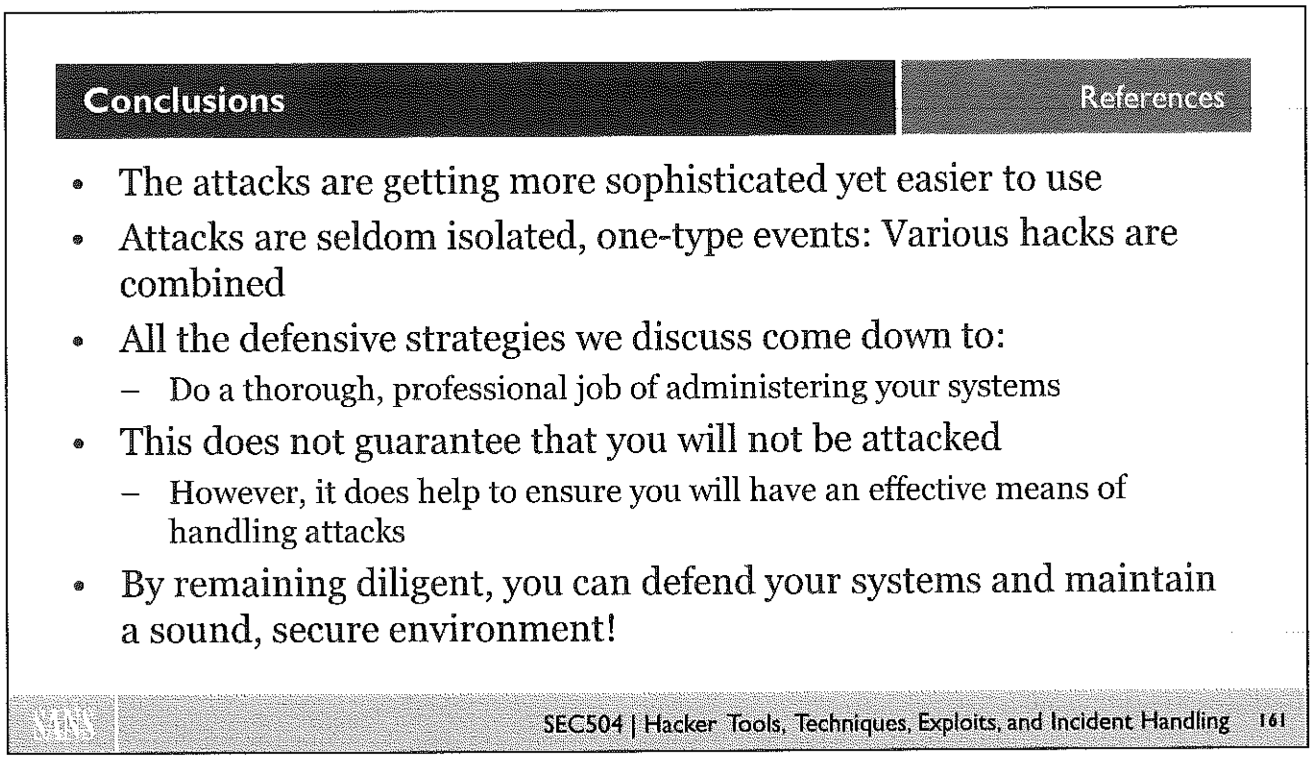
Podcasts

References

- Numerous podcasts provide in-depth information about security
  - Security Weekly Podcast: http://www.securityweekly.com
  - Network Security Podcast
  - Securabit Podcast: http://securabit.com
  - Data Security Podcast: http://datasecurityblog.wordpress.com

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 160

Each of the podcasts on this slide publishes a new episode each week or two, highlighting late-breaking security issues, attacks, and defenses. Listening to them during exercising or morning work commutes can help make otherwise wasted time much more valuable for your information security practices.



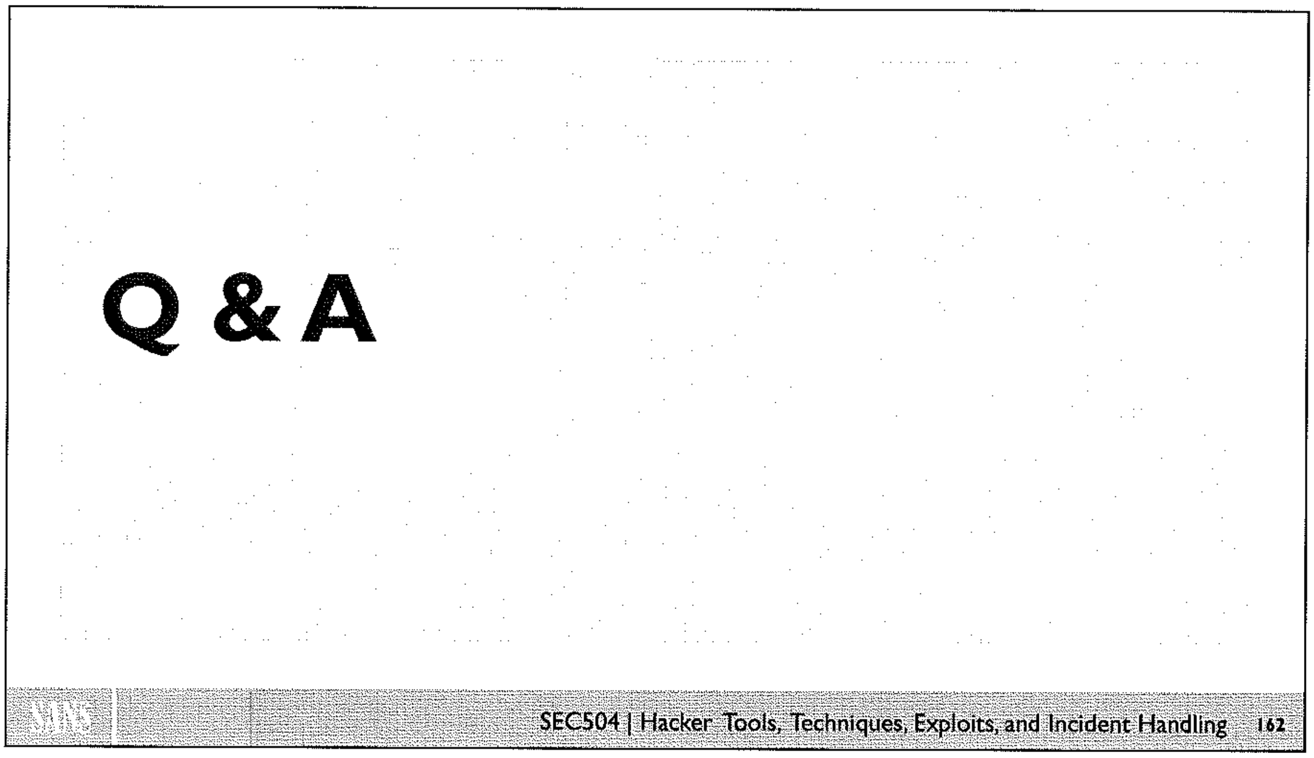
Conclusions

References

- The attacks are getting more sophisticated yet easier to use
- Attacks are seldom isolated, one-type events: Various hacks are combined
- All the defensive strategies we discuss come down to:
  - Do a thorough, professional job of administering your systems
- This does not guarantee that you will not be attacked
  - However, it does help to ensure you will have an effective means of handling attacks
- By remaining diligent, you can defend your systems and maintain a sound, secure environment!

We want you to leave this course with a healthy respect for the capabilities of the attack tools that are available today. We do not want you to leave this course fearful of these tools. That would be counterproductive. Instead, think of the defenses that we discuss. They all come down to doing a good job of administering and securing your systems. Sure, they're not easy and can require a lot of work. However, by remaining diligent, you can defend your environment from attack.

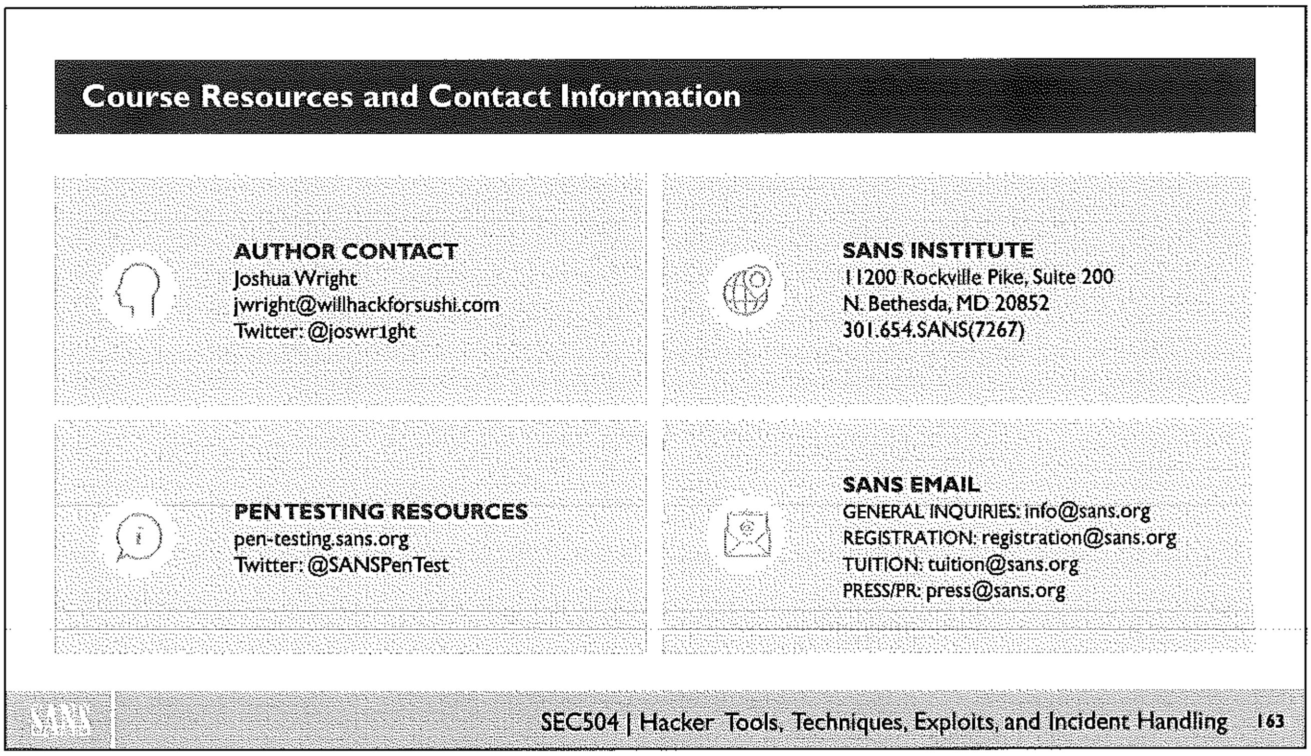
Also, feel free to contact your instructor if you'd like to talk further.



Q & A

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 162

Ask questions. If there's not any course time left, feel free to come up afterward and ask some questions of the instructor.



Course Resources and Contact Information

AUTHOR CONTACT
Joshua Wright
jwright@willhackforsushi.com
Twitter: @joswr1ght

SANS INSTITUTE
11200 Rockville Pike, Suite 200
N. Bethesda, MD 20852
301.654.SANS(7267)

PEN TESTING RESOURCES
pen-testing.sans.org
Twitter: @SANSPenTest

SANS EMAIL
GENERAL INQUIRIES: info@sans.org
REGISTRATION: registration@sans.org
TUITION: tuition@sans.org
PRESS/PR: press@sans.org

SEC504 | Hacker Tools, Techniques, Exploits, and Incident Handling 163

This page intentionally left blank.

"As usual, SANS courses pay for themselves by Day 2. By Day 3, you are itching to get back to the office to use what you've learned."
Ken Evans, Hewlett Packard Enterprise - Digital Investigation Services

SANS Programs
sans.org/programs

GIAC Certifications
Graduate Degree Programs
NetWars & CyberCity Ranges
Cyber Guardian
Security Awareness Training
CyberTalent Management
Group/Enterprise Purchase Arrangements
DoDD 8140
Community of Interest for NetSec
Cybersecurity Innovation Awards

Search SANSInstitute

SANS Free Resources
sans.org/security-resources

- E-Newsletters
  NewsBites: Bi-weekly digest of top news
  OUCH!: Monthly security awareness newsletter
  @RISK: Weekly summary of threats & mitigations
- Internet Storm Center
- CIS Critical Security Controls
- Blogs
- Security Posters
- Webcasts
- InfoSec Reading Room
- Top 25 Software Errors
- Security Policies
- Intrusion Detection FAQ
- Tip of the Day
- 20 Coolest Careers
- Security Glossary

SANS Institute
11200 Rockville Pike | Suite 200
North Bethesda, MD 20852
301.654.SANS(7267)
info@sans.org