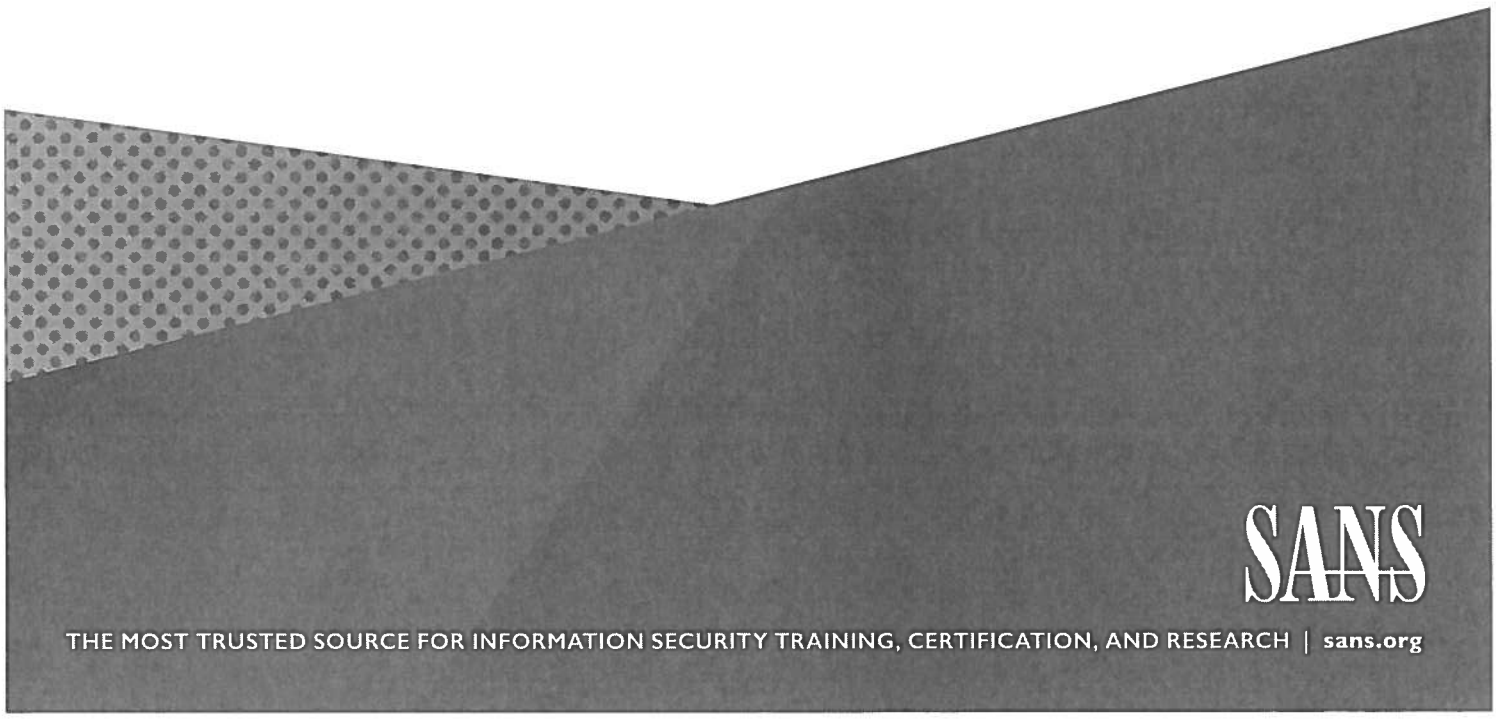
508.1

Advanced Incident Response & Threat Hunting



SANS

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

FOR508.1 | Advanced Digital Forensics, Incident Response, & Threat Hunting

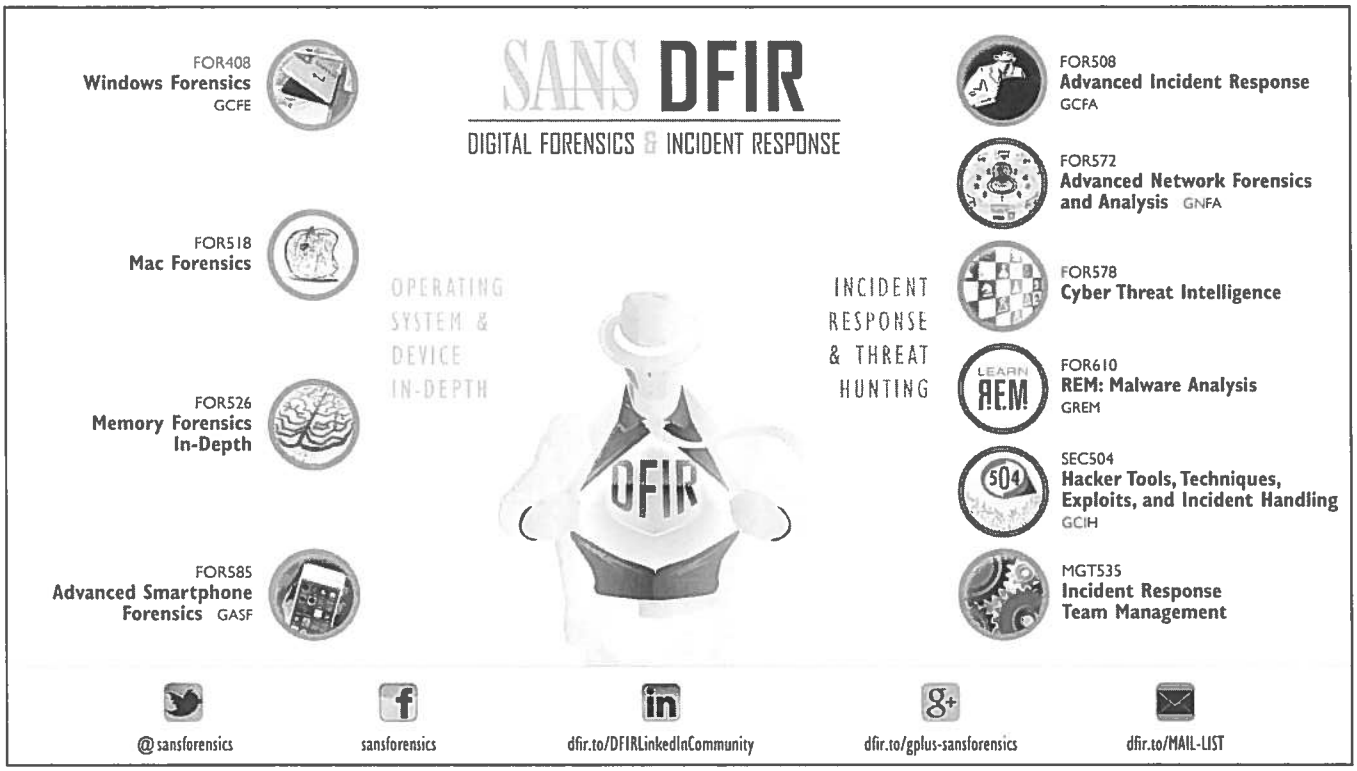
Advanced Digital Forensics, Incident Response, & Threat Hunting

This page intentionally left blank.

Exercise 0

Before Class Begins -- SIFT Lab Installation

This page intentionally left blank.



This page intentionally left blank.

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

OPERATING SYSTEM & DEVICE IN-DEPTH

FOR408
Windows Forensics

FOR518
Mac Forensics

FOR526
Memory Forensics In-Depth

FOR585
Advanced Smartphone Forensics

INCIDENT RESPONSE & ADVERSARY HUNTING

FOR508
Advanced Incident Response

FOR572
Advanced Network Forensics and Analysis

FOR578
Cyber Threat Intelligence

FOR610
REM: Malware Analysis

SEC504
Hacker Tools, Techniques, Exploits, and Incident Handling

MGT535
Incident Response Team Management

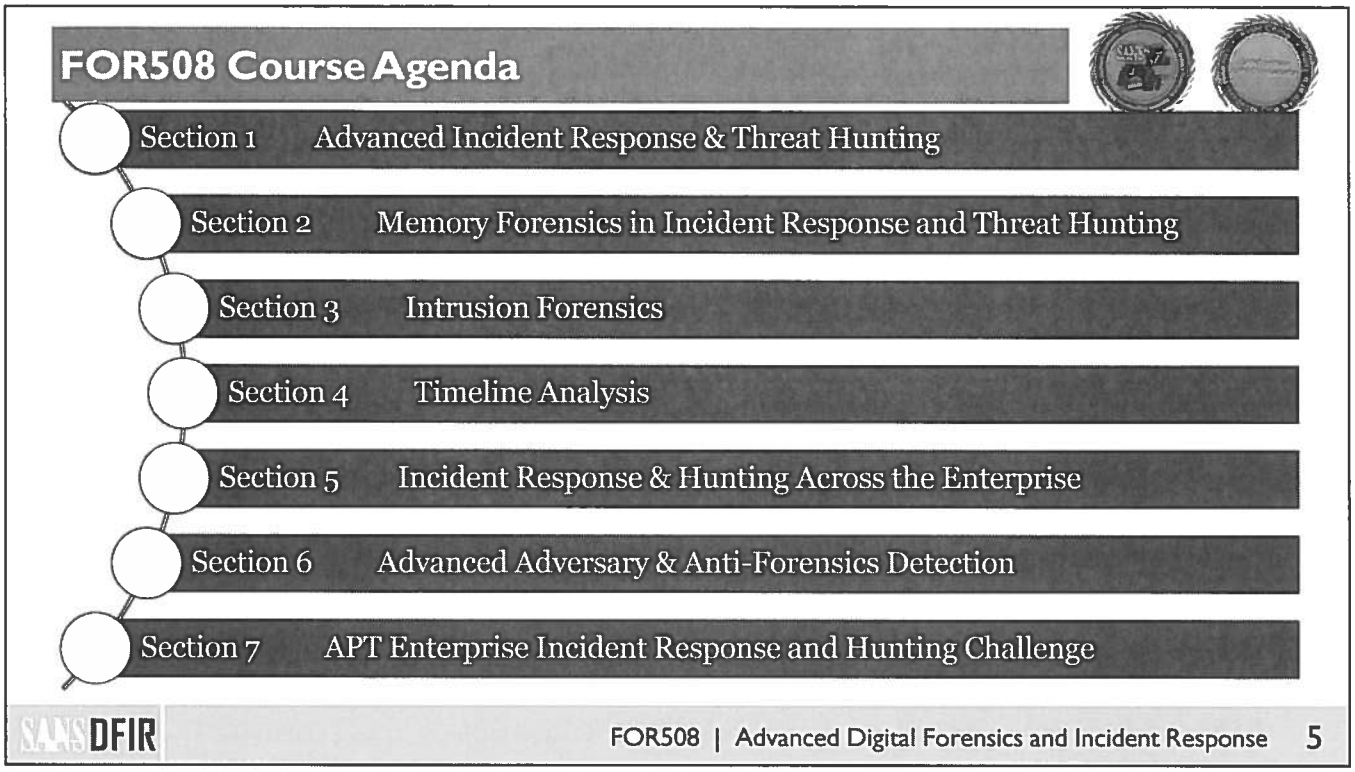
@sansforensics

sansforensics

dfir.to/DFIRLinkedInCommunity

dfir.to/gplus-sansforensics

dfir.to/MAIL-LIST



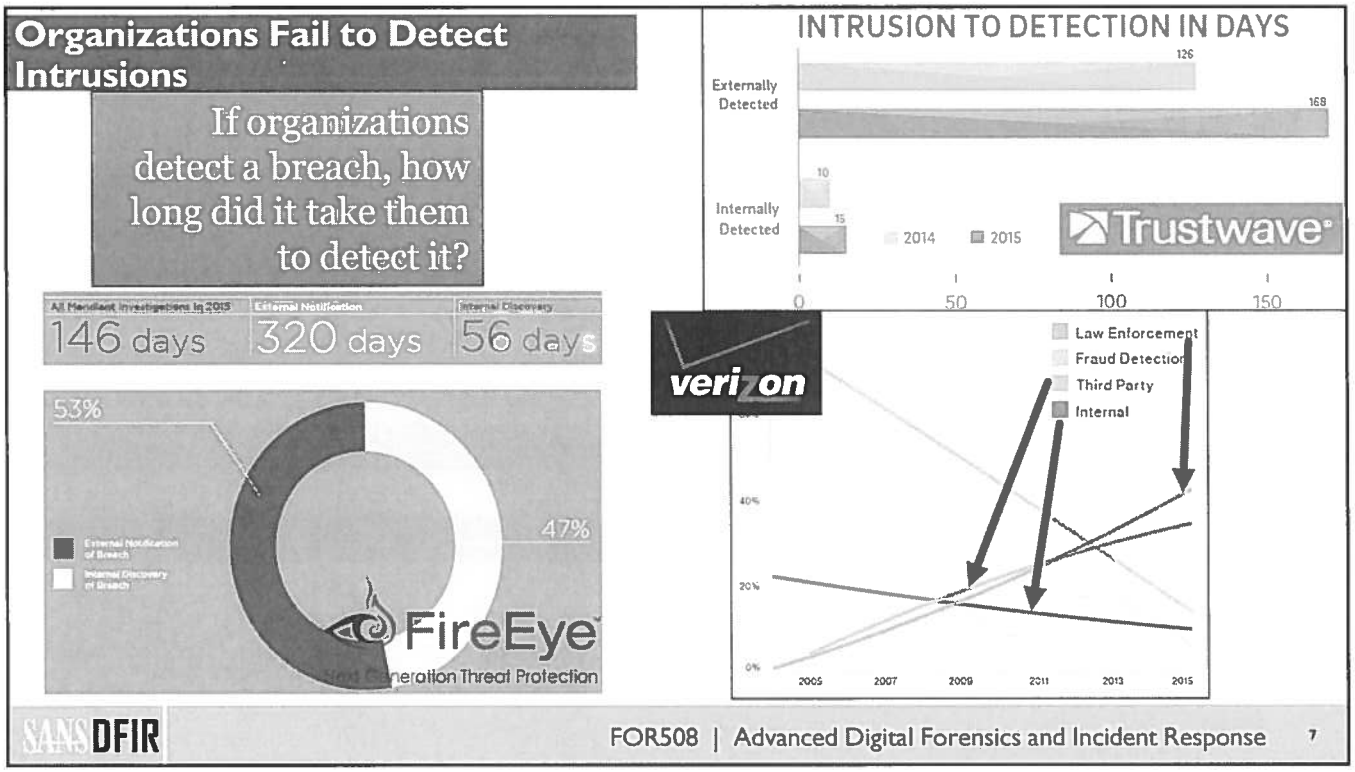
This page intentionally left blank.

Advanced Incident Response & Threat Hunting

Rob Lee – rlee@sans.org
Chad Tilbury – ctilbury@sans.org

Author: Rob Lee
rlee@sans.org
http://twitter.com/robtlee
http://twitter.com/sansforensics

Author: Chad Tilbury
ctilbury@sans.org
http://twitter.com/chadtilbury



Organizations Fail to Detect Intrusions

FACT: Most organizations cannot detect intrusions.

This is really startling considering that our adversaries are increasing their attacks against our systems. In multiple reports, between 2011 and 2016, it has been detailed that organizations cannot detect the intrusions themselves. They find out about the intrusion through third-party notification. In many cases, the attacks are detailed to the victims through law enforcement channels.

- **64%**: Percentage of victim organizations that took more than 90 days to detect the intrusion (*Trustwave Global Security Report*)
- **66%**: Percentage of breaches that remained undiscovered for months or more (*Verizon Data Breach Report*)
- **146 days**: "Median number of days that the attackers were present on a victim network before detection" (*Mandiant M-Trends*); Longest Presence: 2,287 days until detected

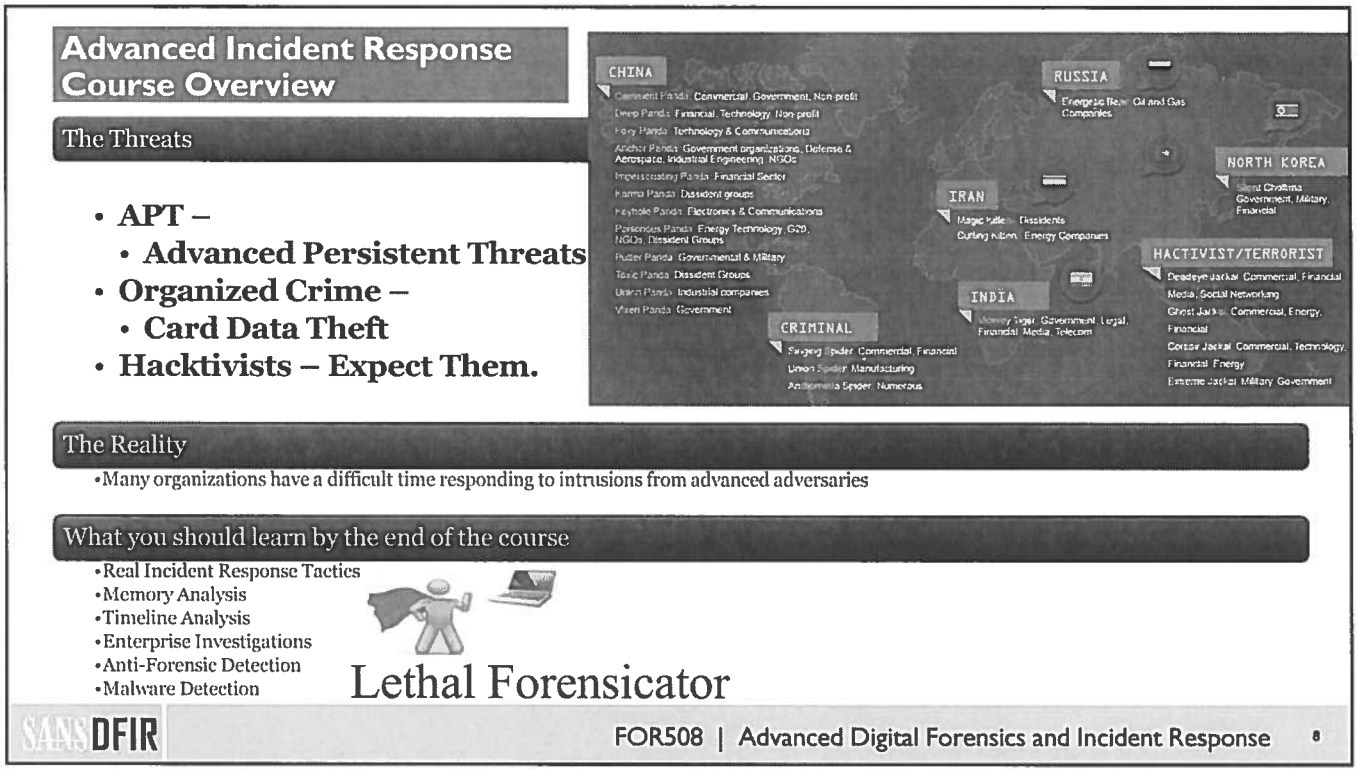
To give you a good perspective on the problem, I highly encourage each attendee to read the following three reports. Each has a slightly different perspective, but you can realize the extent of the challenge.

This course is aimed to help organizations increase their capabilities to detect and respond to intrusions by teaching you the tools and techniques that are critical to overcoming the problem outlined previously. We will not be winning the battle in cyberspace until most organizations (above 50%) can detect their own intrusions. With the millions being poured into cyber detection methods, you would think these numbers would be something from 1999. However, the reality is that our adversaries are good. We aren't. This course is designed to make you and your organization much, much better.

[1] *Trustwave Global Security Report* – https://www.trustwave.com/global-security-report

[2] *Verizon Data Breach Report* – http://www.verizonenterprise.com/DBIR/

[3] *Mandiant M-Trends* – www.fireeye.com – https://www.fireeye.com/current-threats/threat-intelligence-reports.html



Course Overview

Over the past two years, we have seen a dramatic increase in sophisticated attacks against organizations. Cyber-attacks originating from China, named the Advanced Persistent Threat (APT), have proved difficult to suppress. Financial attacks from Eastern Europe and Russia obtain credit card, and financial data resulting in millions of dollars stolen.

Commercial and Federal IT Security are battling multiple intrusions attributed to the Advanced Persistent Threat during the past several years. The adversary is good and getting better. Are we learning how to counter them? Yes we are. Learn how.

Over the past 2 years, we have been updating the forensic and incident response courses at SANS to include the latest tactics at hunting and defeating the APT. The course where I have added most of our efforts to train incident responders to deal with this threat is FOR508: Advanced Computer Forensic Analysis and Incident Response.

Over the past year we have added and updated key sections aimed at directly responding to advanced adversaries that organizations currently face.

Is there malware on this machine? Ever been handed a hard drive and your task is to "Find Evil" but you don't know where to start looking? In FOR508, there is a new section that deals solely with examining compromised systems to look for unknown malware. This process utilizes many of the skills a forensicator must have in order to simply "FIND EVIL" when they do not know where to look.

Timeline Analysis and Super-Timeline Analysis: Critical to any case, the past two years have seen a dramatic increase in the necessity of timeline analysis for incident response and digital forensics. Students

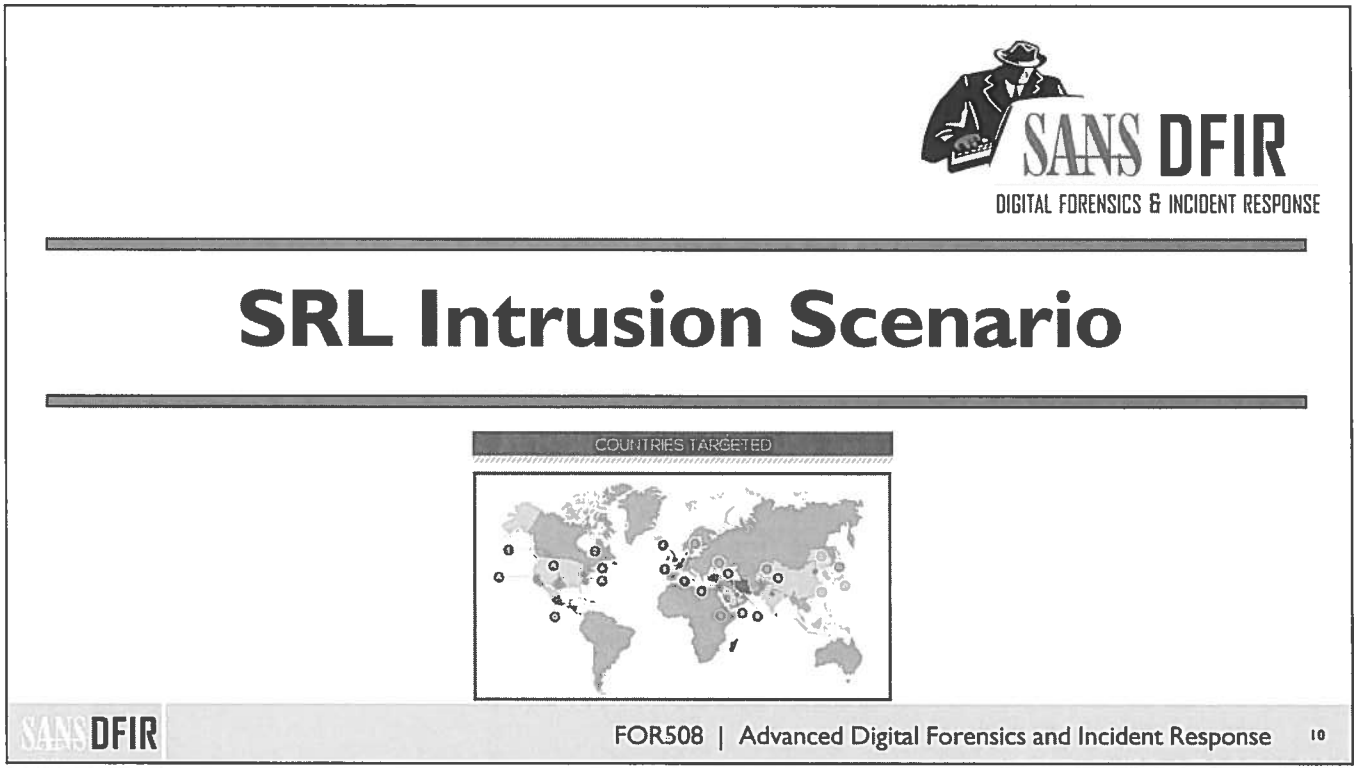
will appreciate being able to automatically track system activity at a glance. Through examining file system, Windows OS artifacts, and registry entries from a single machine, an examiner can determine exactly what happened at any time.

Memory Analysis: Being able to sort through network and active processes from a memory snapshot is a critical skill during an intrusion case to find malware. Moving from malware identification during live response to recovering APT "command and control" channel data, memory analysis is now critical during modern incident response situations.

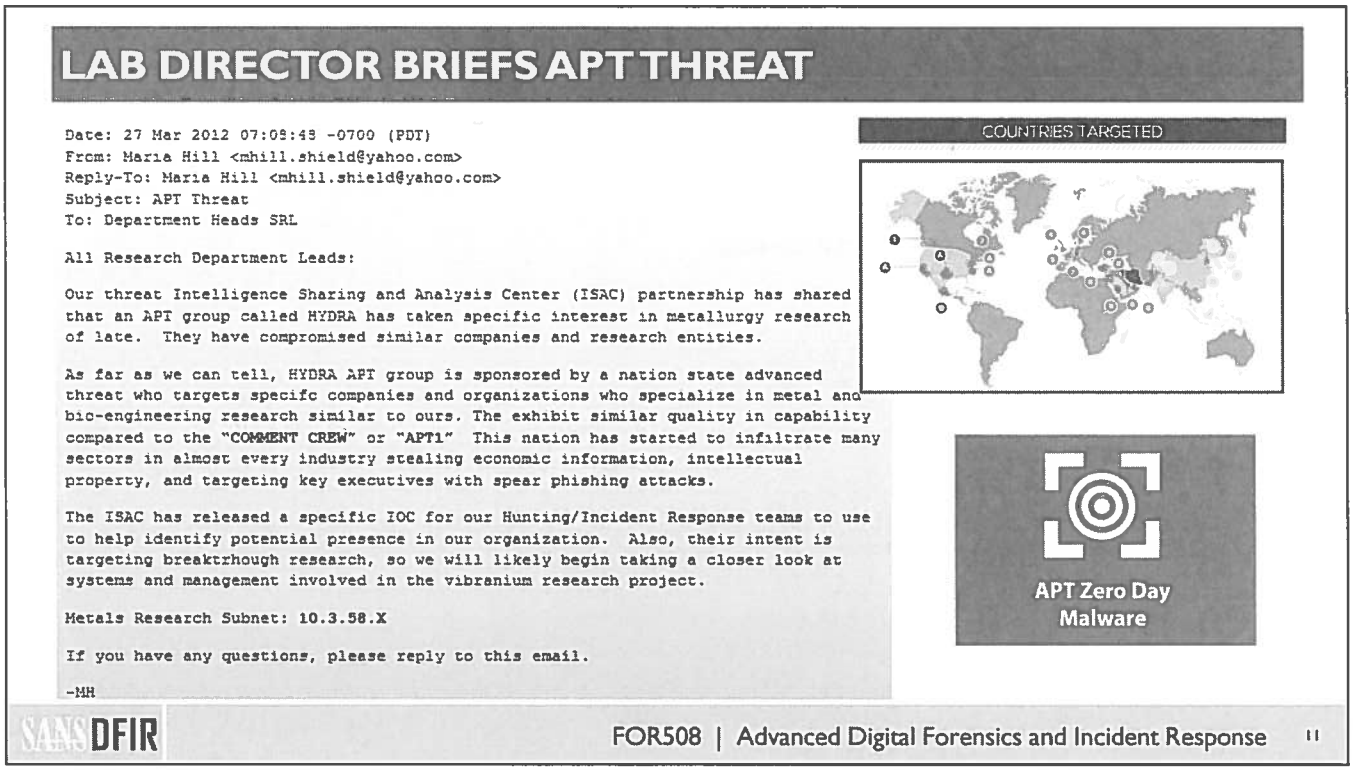
Enterprise Investigations: Investigators must utilize new techniques to not only investigate a single system, but hundreds simultaneously. As a part of this class, we equip each student with F-Response Tactical which allows each student to remotely examine a system without first having to image it. This increase in efficiency is needed in order to quickly scan systems during a large scale breach. Imaging each system to perform forensics is now considered only in rare specific situations. This new addition will change the way you are currently responding to your breaches across your enterprise.

FOR508 has been updated with the latest investigative techniques to help arm you with the correct knowledge to counter advanced adversaries. Our cyber enemies are growing in knowledge and sophistication. FOR508: Advanced Computer Forensic Analysis and Incident Response arms you with the tools and tactics to counter them.

The graphic is from *Crowdsstrike Intelligence Report* and used with permission.



This page intentionally left blank.



LAB DIRECTOR BRIEFS APT THREAT

Stark Research Labs is a government-sponsored laboratory that researches specialized metal alloys and bio engineering capabilities. Lately, SRL has been tasked to find the secret and once lost alloy formula for VIBRANIUM-Alloy. The lead researcher, Timothy Dungan, has been making progress and it looks like with the help of others, he was finally able to replicate the formula. Timothy Dungan has been working tirelessly on trying to find the missing formula for the past two years.

Date: 27 Mar 2012 07:08:48 -0700 (PDT)

From: Maria Hill <mhill.shield@yahoo.com>

Reply-To: Maria Hill <mhill.shield@yahoo.com>

Subject: APT Threat

To: Department Heads SRL

All Research Department Leads:

Our threat Intelligence Sharing and Analysis Center (ISAC) partnership has shared that an APT group called HYDRA has taken specific interest in metallurgy research as of late. They have compromised similar companies and research entities.

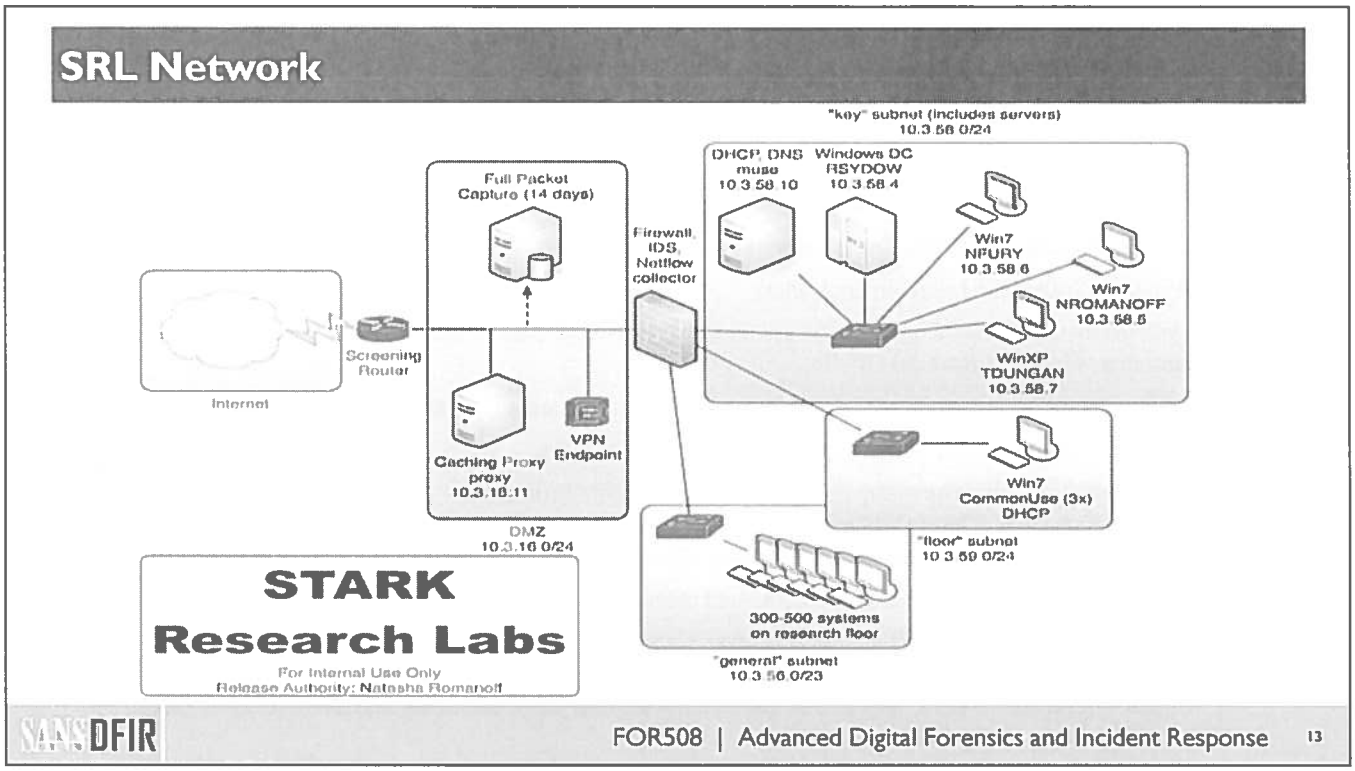
As far as we can tell, HYDRA APT group is sponsored by a nation state advanced threat that targets specific companies and organizations that specialize in metal and bio-engineering research similar to ours. They exhibit a similar quality in capability compared to the "**COMMENT CREW**" or "**APT1**." This nation has started to infiltrate many sectors in almost every industry by stealing economic information and intellectual property and by targeting key executives with spear phishing attacks.

The ISAC has released a specific IOC for our Hunting/Incident Response teams to use to help identify potential presence in our organization. Also, their intent is targeting breakthrough research, so we will likely begin taking a closer look at systems and management involved in the Vibranium research project.

Metals Research Subnet: **10.3.58.X**

If you have any questions, please reply to this e-mail.

-Marie Hill



Host	IP Address	Role
Router	10.3.16.1	Inside interface of Internet screening router
Firewall	10.3.56.1/23 10.3.58.1/24 10.3.59.1/24 10.3.16.99/24	General administration and users Key users & SHIELDBASE servers subnet Production floor systems External (DMZ) interface
VPN	10.3.16.2/24	VPN Concentrator external interface (internal addresses assigned from "general" pool)
Proxy	10.3.16.11/24	Caching proxy for outgoing HTTP
Sniffer	10.3.56.254/24	Tcpdump rotating buffer. Approx 14 days of storage
Webserver	10.3.16.3/24	External presence of SRL
rsydow	10.3.58.4	SHIELDBASE domain controller – Windows 2008r2
tdungan	10.3.58.7	Tim Dungan's R&D Workstation
nromanoff	10.3.58.5	Natasha Romanoff's Workstation
nfury	10.3.58.6	Nick Fury's Workstation

The SRL network is representative of many medium-sized enterprises, with several major internal segments separated by a single firewall, but very little network protection between hosts once on the inside of that perimeter.

SRL-LABS BASE DOMAIN INFORMATION

- Full auditing turned on per recommended guidelines.
- Users are restricted to being users (cannot even install a program if they wanted to).
- Windows 2008R2SP1 domain controller.
- Systems installed and have real software on it that is used (Office, Adobe, Skype, Tweetdeck, E-mail, Dropbox, Firefox, Chrome).
- Fully patched as of 6 April 2012 (today); patches are automatically installed.
- Enterprise Incident Response agents (F-Response Enterprise).
- Enterprise A/V and on-scan capability (McAfee Endpoint Protection – Anti-virus, Anti-spyware, Safe surfing, Anti-spam, Device Control, Onsite Management, Host Intrusion Prevention [HIPS]). Network using HBSS (host-based security system – per DOD recommendations).
- Firewall allowed only inbound 25,80, 443 and outbound 25, 80, 443.
- Users have been "using" this network for over a year prior to the attack. That way, it has the look and feel of something real. These users have setup social media (yes, they are on twitter... you might be friends with them), e-mail, Skype, etc. Each character user has a backstory and a reason to be there working.
- Rsydow is the sole domain administrator.
- The SRL-HELPDESK account is local admin account with shared password.

ADDITIONAL INFORMATION REGARDING SRL DOMAIN

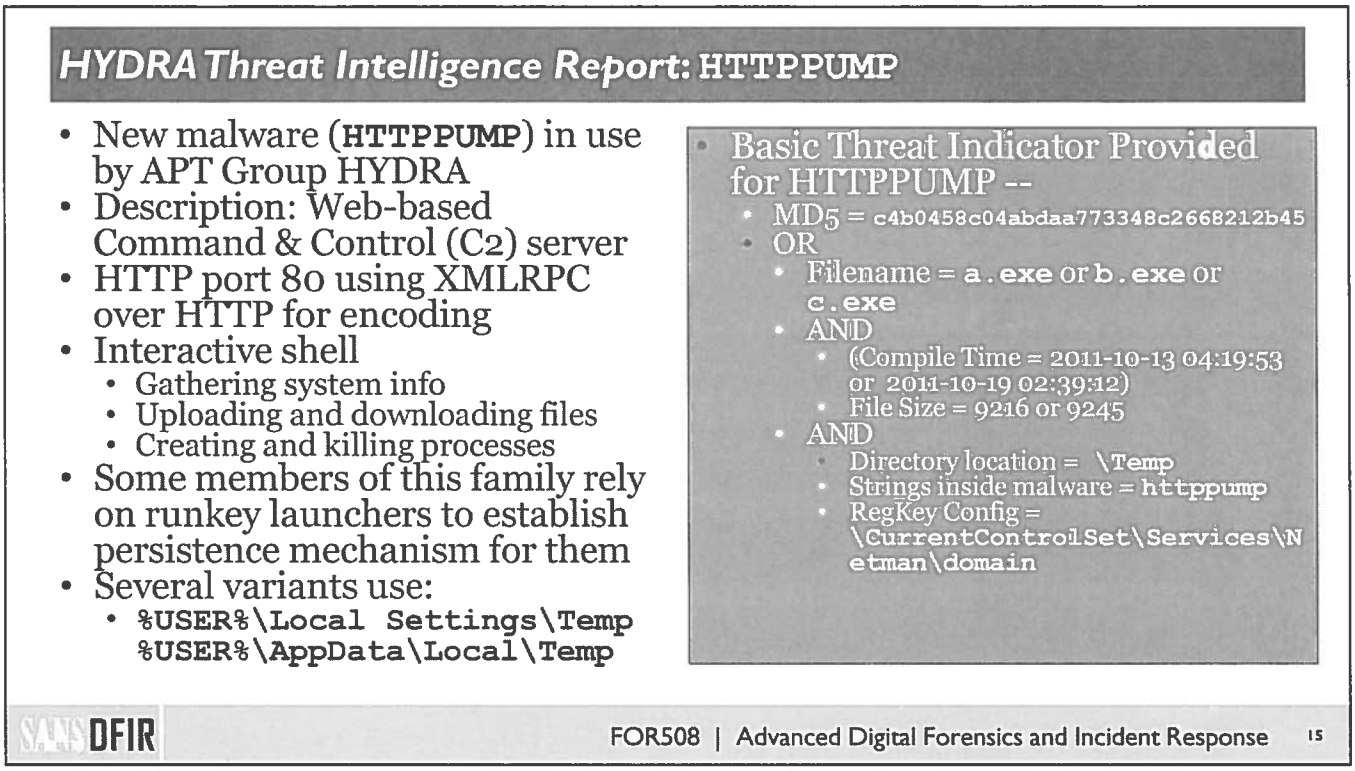
- Local Admin User (SRL-Helpdesk) found on each system with the same password.
- Not every user has migrated to Win7 and Win2008. We do still have some legacy WinXP systems.
- Most of the employees telecommute from home to the lab. The VPN concentrator is located on the \\FALCONIII system. Most users RDP into their systems from the VPN.

Enterprise network with over 1,000 users and systems.

The target of the analysis will begin looking at the subnet involved in the Vibranium research, which also contains a domain controller for that segment of the network.

A few of the key systems on that subnet include:

- 10.3.58.4 Win2008R2 Domain controller
- 10.3.58.5 Win7-SP1-32bit N. Romanoff Workstation: Vibranium Program Manager
- 10.3.58.6 Win7-SP1-64bit N. Fury Workstation: SRL Management
- 10.3.58.7 WinXP-SP3-32bit Timothy Dungan Workstation: Vibranium Lead Researcher



***HYDRA Threat Intelligence Report*: HTTPPUMP**

THREAT INTELLIGENCE REPORT – New Malware (HTTPPUMP) in use by APT Group HYDRA.

Description: Members of this family are full-featured backdoors that communicates with a Web-based Command & Control (C2) server over HTTP port 80 using XMLRPC over HTTP for encoding. Features include interactive shell, gathering system info, uploading and downloading files, and creating and killing processes, Malware in this family usually communicates with a hard-coded domain using XMLRPC over HTTP on port 80. Some members of this family rely on runkey launchers to establish persistence mechanism for them. Several variants use **%USER%\Local Settings\Temp** or **%USER%\AppData\Local\Temp** as working directories, additional malware artifacts might be found there.

Basic Threat Indicator Provided for HTTPPUMP –

MD5 = **c4b0458c04abdaa773348c2668212b45**

OR

Filename = a.exe or b.exe or c.exe

AND

(Compile Time = 2011-10-13 04:19:53 or 2011-10-19 02:39:12)

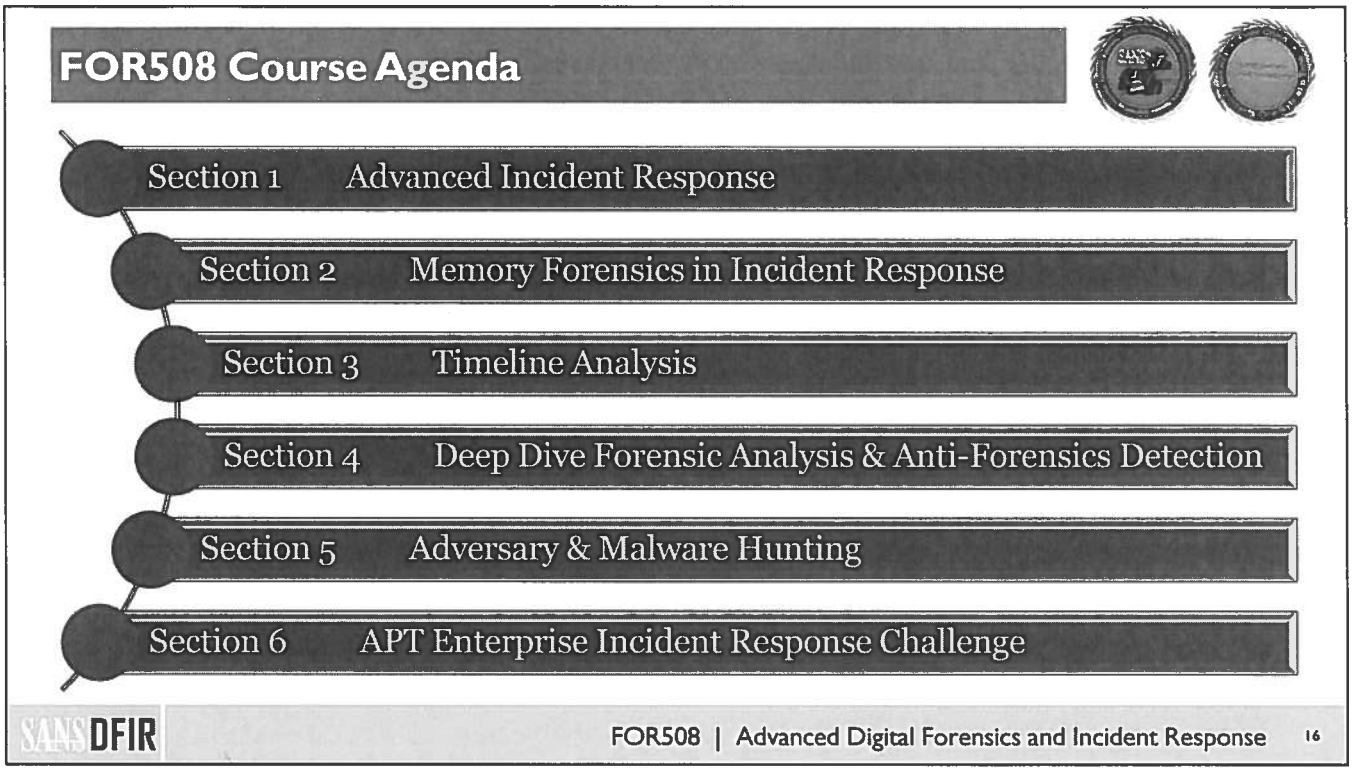
File Size = 9216 or 9245

AND

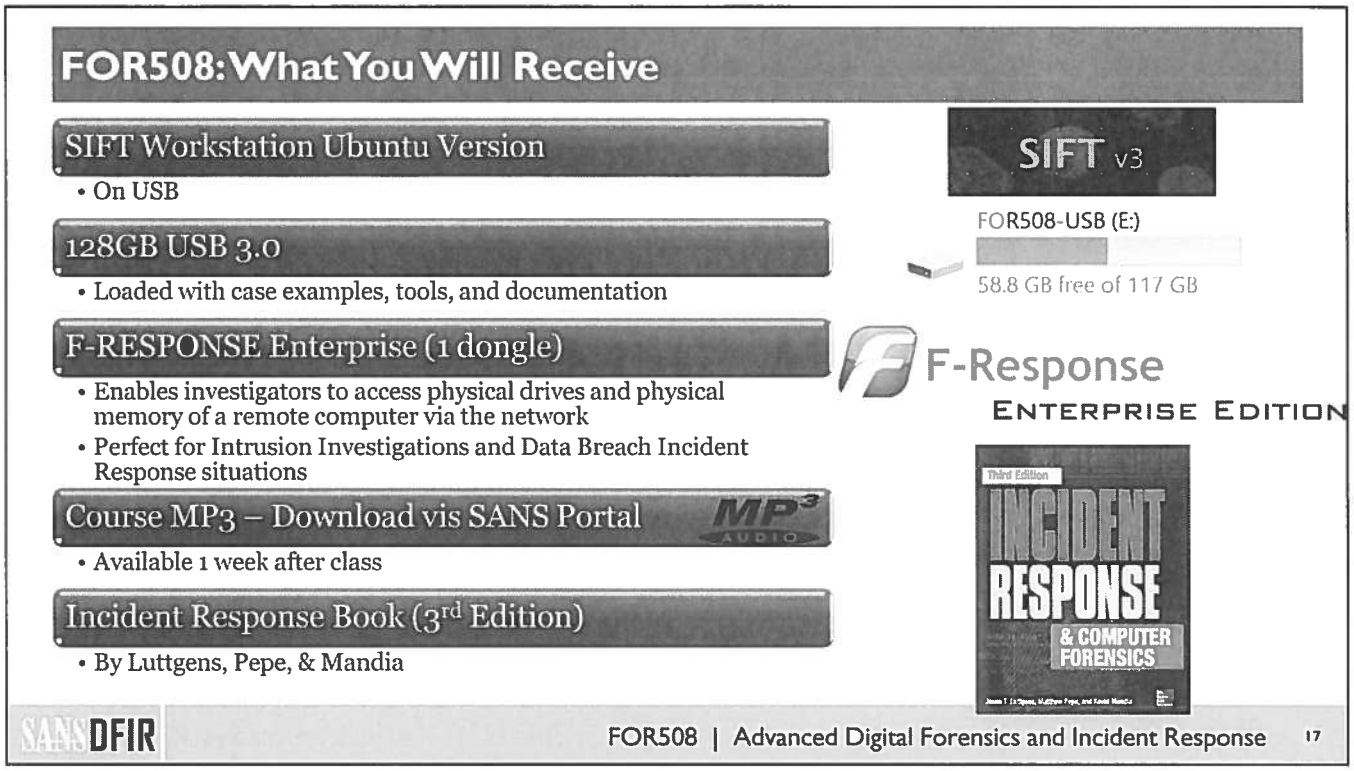
Directory location = **\Temp**

Strings inside malware = **httppump**

\CurrentControlSet\Services\Netman\domain

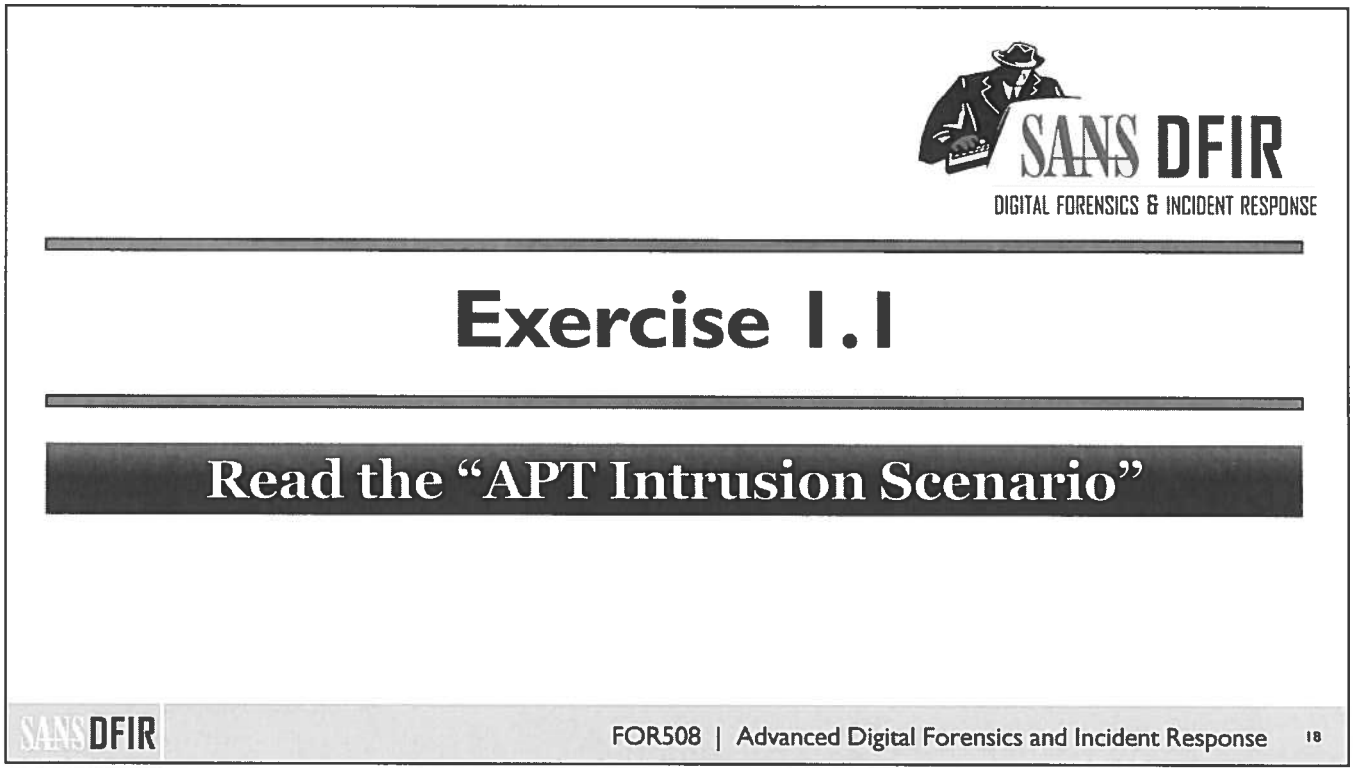


This page intentionally left blank.

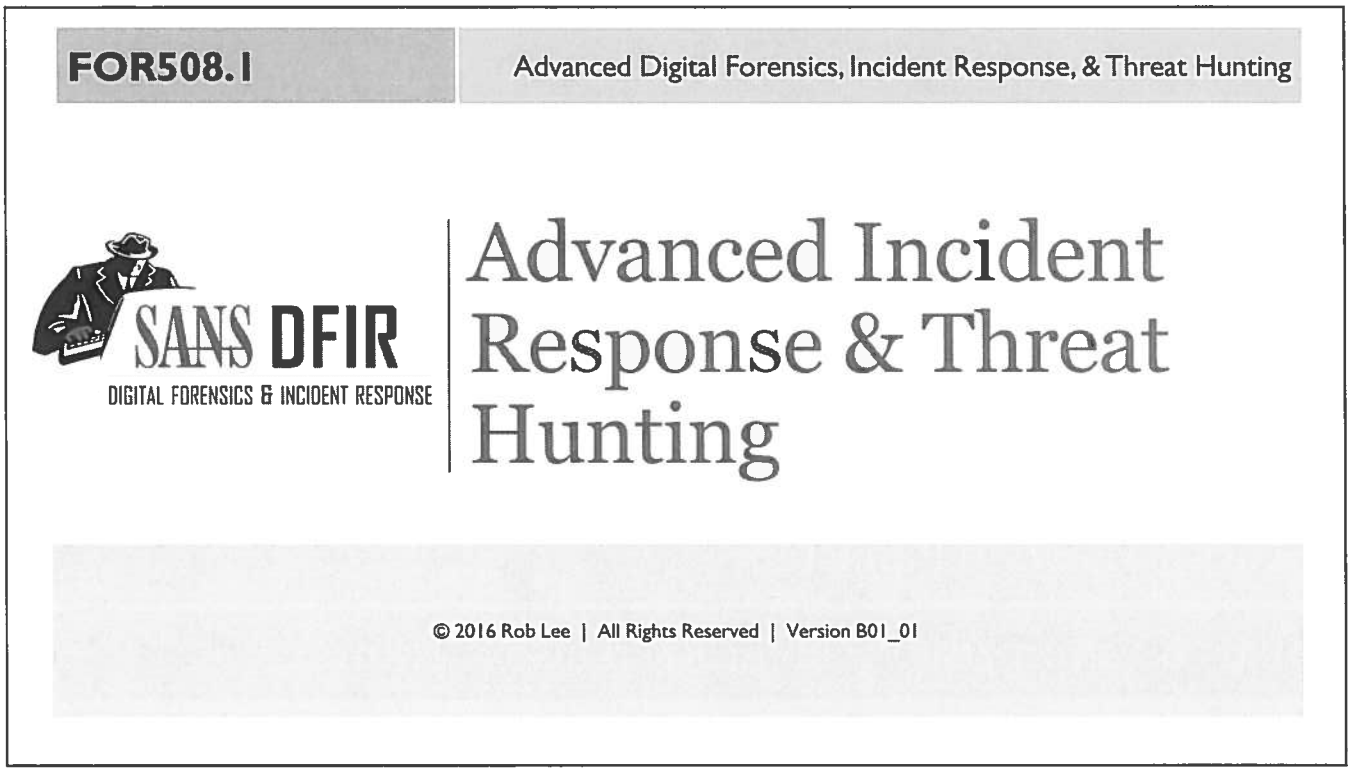


FOR508: What You Will Receive

As part of the course, you will receive the SANS Investigative Forensic Toolkit (SIFT). Using the hardware and software in this toolkit, you will gain first-hand experience in collecting and analyzing evidence recovered from a system under investigation. You will learn best practices on how to investigate and recover deleted data. The course will demonstrate how forensic tools recover evidence so you can articulate how the tool works in-depth. We will examine various investigation methodologies and techniques discovering new places to find evidence and discover the tracks of a motivated suspect who is trying to stay hidden.



This page intentionally left blank.



Welcome to Forensic and Investigative Essentials.

Author: Rob Lee
rlee@sans.org
http://twitter.com/robtlee
http://twitter.com/sansforensics

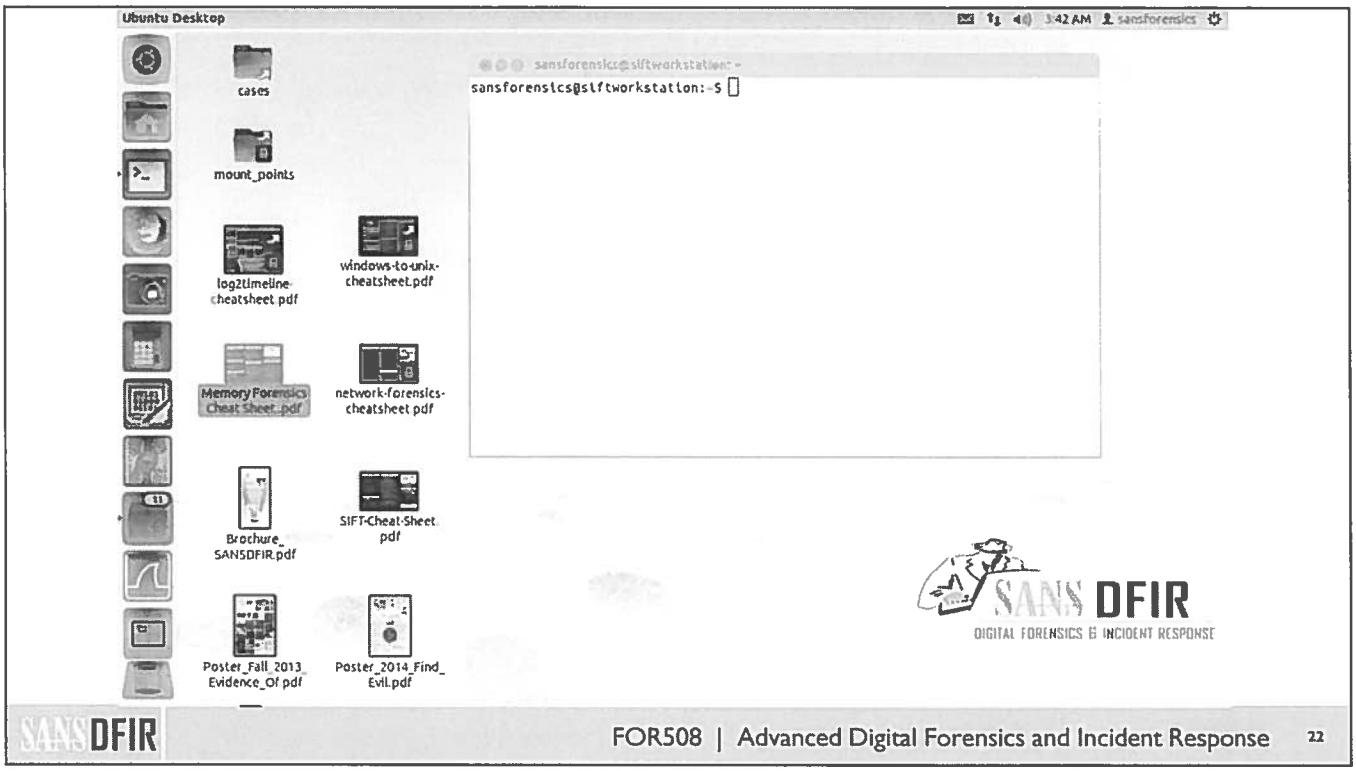
This page intentionally left blank.

The Ubuntu Linux SIFT VMware Forensic Workstation

"If you try and lose then it isn't your fault. But if you don't try and we lose, then it's all your fault."
— Orson Scott Card, *Ender's Game*

SANS DFIR
DIGITAL FORENSICS & INCIDENT RESPONSE

This page intentionally left blank.



Download from http://digital-forensics.sans.org/downloads

Once installed, press **Play** and log in to your new virtual machine:

Login: **`sansforensics`**

Password: **`forensics`**

SANS teaches the latest tools and techniques available. In addition, our aim is also to create a laptop setup that is simple to use to accomplish the daily exercises utilizing forensic tools. To solve both challenges, the SANS Institute has released a cutting-edge course DVD that includes preconfigured **VMware Forensic Workstation** that is ready to tackle forensics right off the DVD. This will allow for you to have either a Windows or a Linux base installation. The only requirement for you to have followed is that VMware is installed and working correctly following the forensic installation guide.

The new VMware image is already pre-configured with all the tools, so you can just concentrate on learning the material and not configuring your machine. You will need to copy over only a gzipped tar archive from your DVD to your VMware directory.

Why a SIFT Forensic Workstation?

Windows forensic tools have a lot of capabilities, but in many cases, you need something with a little more versatility and compatibility. By default, the SIFT Workstation comes pre-installed with many tools that allow you to perform in-depth forensic analysis of multiple operating system types.

You should get out of the class that most of your forensic examination beyond evidence collection can be performed using your SIFT Workstation. If you work mainly in a Windows environment, you should still consider using SIFT to examine your compromised Windows platforms. Most seasoned investigators use both Linux and Windows at the same time to ensure evidence is not missed. You will find your job much easier to perform once you get a feel for the powerful forensic options provided to you by a Linux workstation.

Again, it should be noted, that although you start out in the Windows side collecting your evidence using dd, IRCR, pstools, regedit, etc., you could then examine your collected evidence on your Linux system, or use your Linux system as your anchor to ensure the evidence doesn't change while you examine the evidence.

An international team of forensics experts, led by SANS Faculty Fellow Rob Lee, created the SANS Investigative Forensic Toolkit (SIFT) Workstation and made it available to the whole community as a public service. The free SIFT toolkit, that can match any modern forensic tool suite, is also featured in SANS' Advanced Computer Forensic Analysis and Incident Response course (FOR 508). It demonstrates that advanced investigations and responding to intrusions can be accomplished using cutting-edge open-source tools that are freely available and frequently updated.

The SIFT Workstation is a VMware appliance that is pre-configured with the necessary tools to perform detailed digital forensic examination in a variety of settings. It is compatible with Expert Witness Format (E01), Advanced Forensic Format (AFF), and raw (dd) evidence formats. The brand new version has been completely rebuilt on an Ubuntu base with many new capabilities and tools such as log2timeline that provides a timeline that can be of enormous value to investigators.

Some examples of SIFT TOOLS FROM FOR408 Artifacts that you will likely see in this course include the following:

`# rip.pl -r <HIVEFILE> -f <HIVETYPE>`

[Useful Options]

- **`-r`** Registry hive file to parse <*HIVEFILE*>
- **`-f`** Use <*HIVETYPE*> (**`sam, security, software, system, ntuser`**)
- **`-l`** List all plugins

On your SIFT Workstation

```
# rip.pl -r /mnt/windows_mount/WINDOWS/system32/config/SAM -f sam >
/cases/windowsforensics/SAM.txt
```

RegRipper is an automated HIVE parser that can parse the forensic contents of the SAM, SECURITY, SYSTEM, SOFTWARE, and the NTUSER.DAT HIVES that it is pointed at. You can even use this to forensically mine the contents of restore point registry files. RegRipper, written by Harlan Carvey, can be found at https://regripper.wordpress.com/regripper/ and is based off a plugin system. Each plugin in the plugins directory will parse a separate forensic artifact in a specific registry hive. You can add plugins to the directory as a result, and it will parse the new data the next time the tool runs.

For regripper to run properly, you can run it from your SIFT Workstation or a Windows machine. The regripper files are in /usr/local/src/regripper on your SIFT Workstation.

rip.pl can be invoked by pointing the –r HIVEFILE at the hive you would like to mine forensically. You also need to tell regripper (rip.pl) the type of hive file it is (sam, security, software, system, and ntuser) using the –f option with the HIVETYPE option. Note that the HIVETYPE should be written in lowercase.

LIBPFF: Library and tools to access the Personal Folder File (PFF) and the Offline Folder File (OFF) format. PFF is used in PAB (Personal Address Book), PST (Personal Storage Table) and OST (Offline Storage Table) files.

Usage: pffexport [-c codepage] [-f format] [-l logfile] [-m mode]
[-t target] [-dhqvV] source

source: the source file

-c: codepage of ASCII strings, options: ascii, windows-1250, windows-1251, windows-1252,(default), windows-1253, windows-1254, windows-1255, windows-1256, windows-1257
or windows-1258

-d: dumps the item values in a separate file: ItemValues.txt

-f: preferred output format, options: all, html, rtf, text (default)

-h: shows this help

-l: logs information about the exported items

-m: export mode, option: all, debug, items (default), recovered. 'all' exports the (allocated) items,orphan and recovered items. 'debug' exports all the (allocated) items, also those outside the root folder. 'items' exports the (allocated) items. 'recovered' exports the orphan and recovered items.

-q: quiet shows minimal status information

-t: specify the basename of the target directory to export to (default is the source filename) pffexport will add the following suffixes to the basename: .export, .orphans,.recovered

-v: verbose output to stderr

-V: print version

To update SIFT, type **sudo update-sift**; however, please do not do this during class though.

Forensic Workstation Layout

These are the common directories that a new user of the SIFT should be familiar with while using the workstation. The /usr/local/src/ and /usr/local/bin/ directories are used for source and executable files respectively.

The /cases directory is where your case files will be located during your investigation. Typically, the /cases/YYYYMMDD-casename format is useful to identify data surrounding a specific case. I typically create sub-directories based on the type of data I have collected (for example, Image, Carved Data, and Live Response Output).

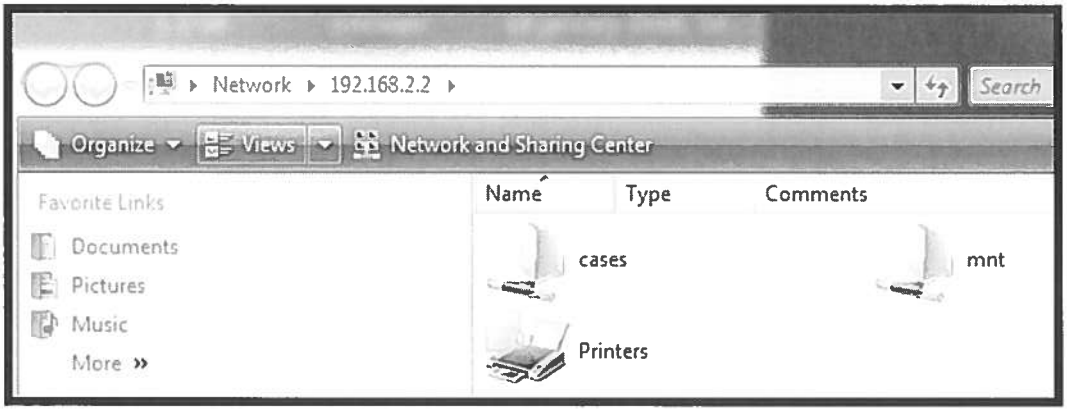
The /mnt directory is the location in the SIFT Workstation where you will mount your image files so you can browse the directory structure of the disk image you have collected of your victim system.

The cases directory is a perfect place to create a case for each of the images for every case you encounter. For this course, we will examine several sets of images. Each of these images has a separate directory created in the /cases directory. For future cases you might encounter, you could use this structure as a way to separate your cases.

Once all images for this course are installed, they will be in the **`/cases/`** directory.

Each subdirectory will be the name of your case (for example, **`windowsforensics`** and **`forensicchallenge`**).

A best practice is to place your host and virtual machine on the "Host-Only" standalone air-gapped network. This ensures only the two machines in question can talk between themselves.



To gain access to your shares from a local Windows machine:

- Start -> RUN or type **Explorer Bar**.
- Type **`\\SIFTWORKSTATION`**.
- You should now see directories **`/mnt`** and **`/cases`**.

You will see the two shares that were created for you (**`cases`** and **`mnt`**). All of the images that we use in class can now be viewed underneath the cases directory. When you mount a specific day's files, you will be able to examine them from either Windows or Linux. Files in the **`mnt`** folder are shared as read only AND mounted as read only. You will not be able to change the files so you can run any variety of scanning tools such as virus scanners against the files without worrying that you would be adjusting the file system.

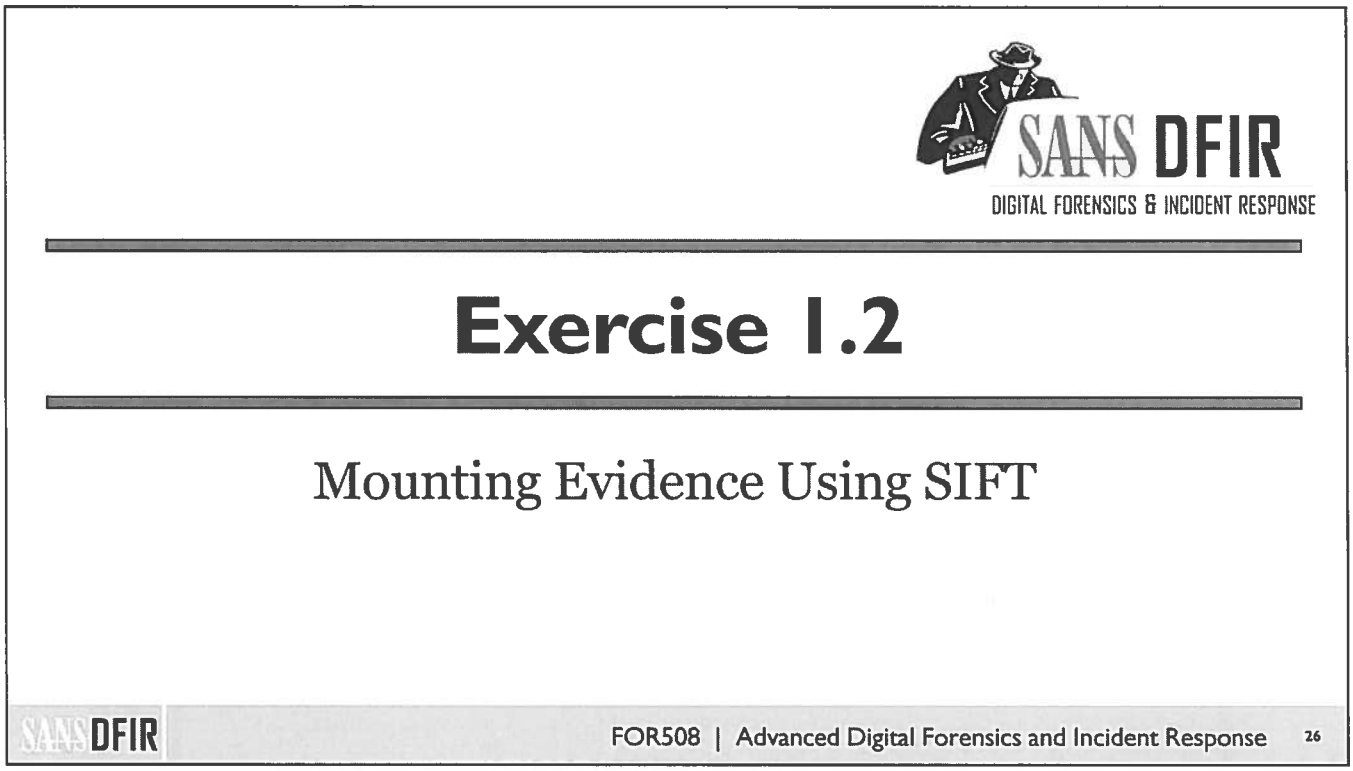
`/mnt` is a READ ONLY share.

`/cases` is a READ/WRITE share.

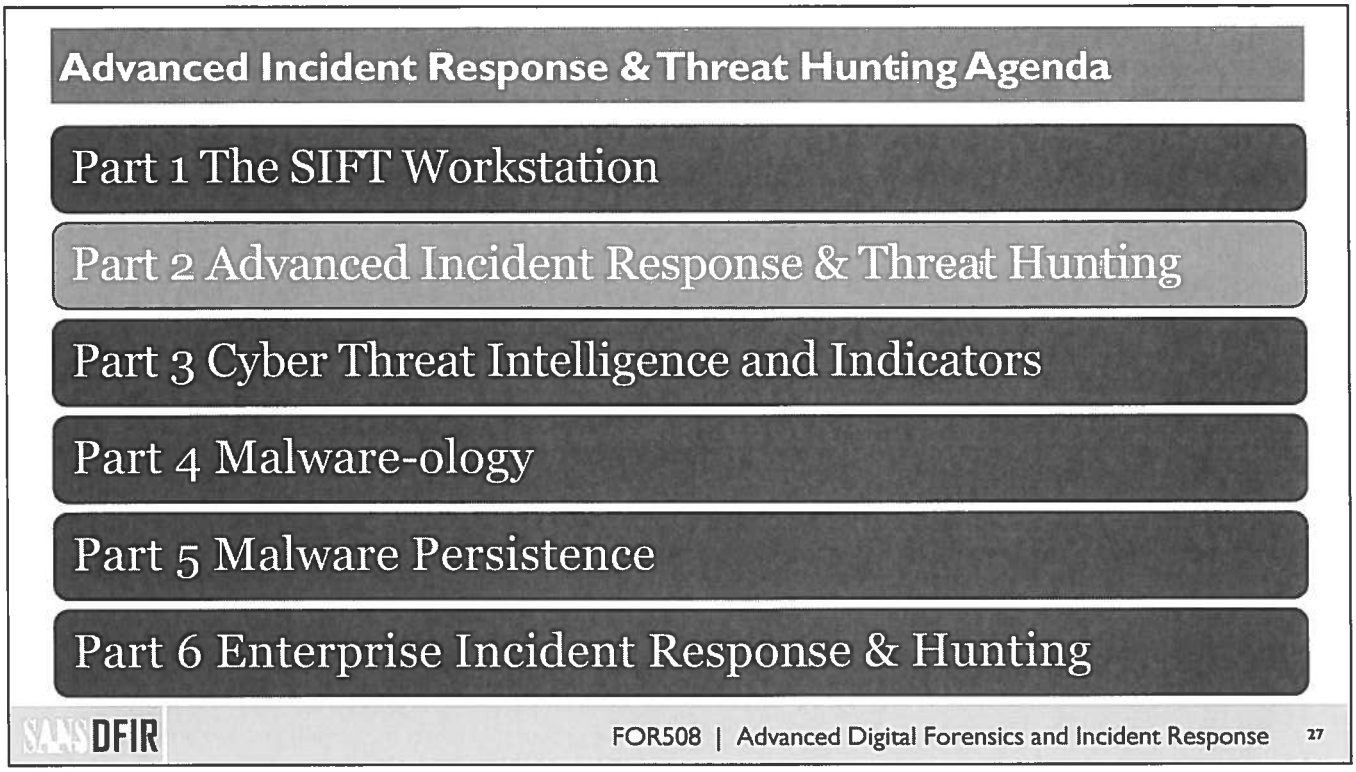
You can use the shares to transfer and examine files from your Windows machine to your Linux machine.

Remember the **`mnt`** directory is where you will mount your images as file systems and be able to browse through the mounted system. Your cases folder will be the location containing your forensic images.

If you are having problems seeing your connectivity, you will not need this until later in the course. Let your instructor know so he or she can help identify the problem.



This page intentionally left blank.



This page intentionally left blank.

Advanced Threat Hunting & Incident Response

"I need you to be clever, Bean. I need you to think of solutions to problems we haven't seen yet. I want you to try things that no one has ever tried because they're absolutely stupid."
— Orson Scott Card, *Ender's Game*

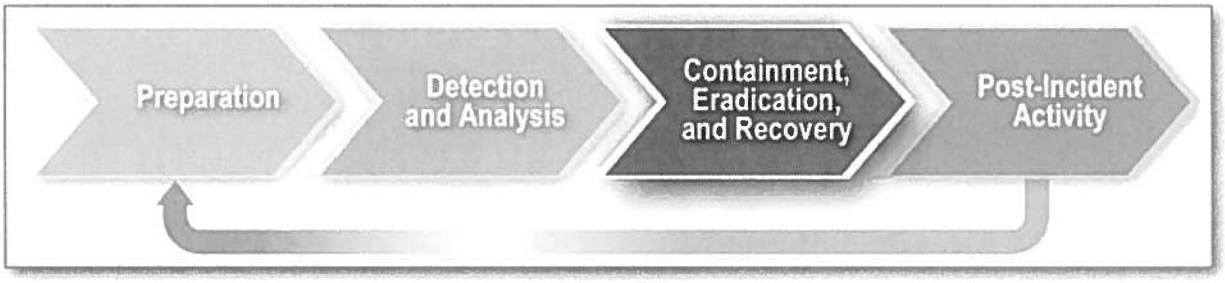
What is cyber threat intelligence?

"I am your enemy, the first one you've ever had who was smarter than you. **There is no teacher but the enemy.** No one but the enemy will tell you what the enemy is going to do. No one but the enemy will ever teach you how to destroy and conquer. Only the enemy shows you where you are weak. Only the enemy tells you where he is strong. And the rules of the game are what you can do to him and what you can stop him from doing to you. I am your enemy from now on. From now on I am your teacher." — Orson Scott Card, *Ender's Game*

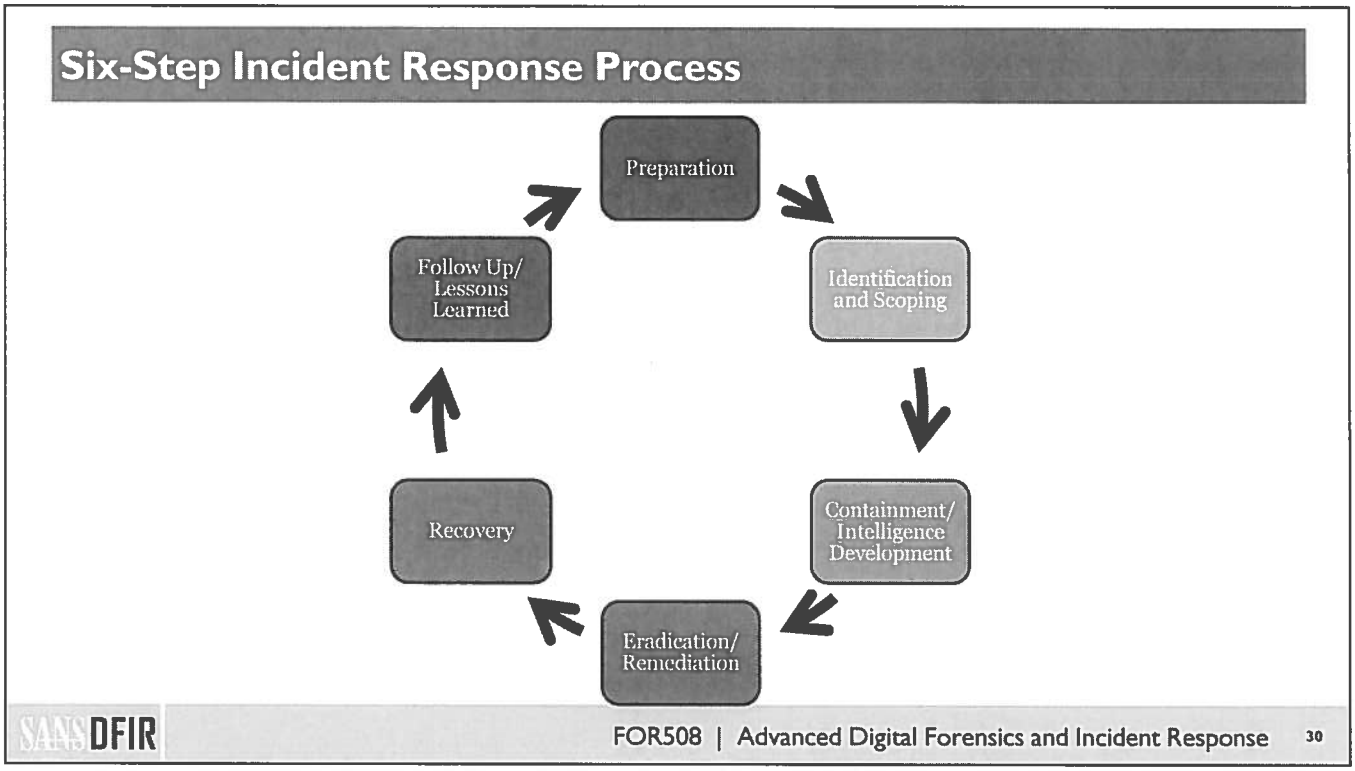
Incident Response Today

Average # of days from compromise to detection: ½ Year

- NIST's *Computer Security Incident Handling Guide*
- Use pre-designed forms, and ask for help
  - http://www.sans.org/score/incidentforms
  - Forms include Incident Contact List, Identification Checklist, Survey, Containment Checklist, Eradication Checklist, and Comm Log



This page intentionally left blank.



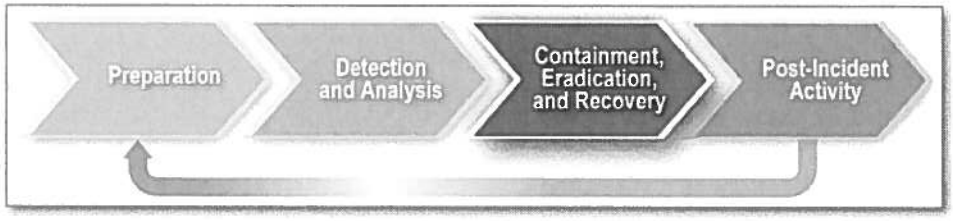
Six-Step Incident Response Process

Incident Response has remained the same for the overall management of the incidents. Typically, we follow the six-step process even with today's adversaries. However, having said that, it should be noted that today's adversaries do require different tactics in order to successfully remediate events today. Remember, your adversaries might have been in your network for over a year. According to Mandiant, the average # of days from compromise to detection is 243.[1]

Most of the detection of incidents should originate from your security team or the security operations. Take for example the concept of an active defense. Organizations should use their entire security architecture and their defenders to identify and scope the problem. This is currently not the case though as detection is likely to originate from a third party. Moving the security maturity of an organization to a place where the majority of detection of incidents is performed in-house is a good goal.

To help you stick to the six-step process, please use the forms at the www.sans.org website. They provide a template for useful information you need to capture during an incident. The free forms at this site include Incident Contact List, Identification Checklist, Survey, Containment Checklist, Eradication Checklist, and a Communications Log.[2]

And, for further materials, NIST has developed a *Computer Security Incident Handling Guide* that covers the same bases we do here. It's a solid read, and goes hand-in-glove with this material as well. The figure below is excerpted from the NIST document.[3]



The six steps in incident handling are preparation, identification, containment, eradication, recovery, and lessons learned. The steps serve the handler as a compass or a roadmap, a way to keep in mind what they are trying to do and the things they need to do next.

Overview of the Six-Step Incident Response Process

Preparation

Incident response methodologies typically emphasize preparation—not only establishing an incident response capability so the organization is ready to respond to incidents, but also preventing incidents by ensuring that systems, networks, and applications are sufficiently secure.

Identification

Out of the six steps, most of the detection of incidents should originate from your security team or the security operations. However, today it is likely that detection of an incident originates from a third party. In many cases, it will be law enforcement.

So you have detected an incident? You spring into action—right? The first step of incident response is critical. It is the proper identification of the systems compromised. This means ALL systems compromised, not just one or two. Advanced intruders install malware on 54% of the systems compromised. This means there are other systems compromised in your enterprise without active malware on them. You need to be able to detect those systems as well.[4]

Unfortunately, most organizations seem to skip directly to containment and eradication in a knee-jerk response without proper scoping. The adversaries tend to react quickly and could deploy countermeasures to ensure continued access or worse, begin to exfiltrate collected data harvested from your enterprise.

As a result, identification is the first step toward proper remediation. The IR team must scope every compromised system. However, in order to do this, you need to gather intelligence.

Containment and Intelligence Development

While analyzing the intrusion, you will learn exactly how the intruders breached the network, how they are laterally moving from system to system, and how they are identifying malware used. All of these traits can be used in order to help identify additional systems compromised and help you engineer countermeasures that can be used during remediation. Threat intelligence is one of the key products of the IR team during an incident.

Remediation

Remediation recommendations are actions that are required to be completed in a very short period of time to mitigate the current incident. Most companies choose to perform these activities over a weekend. These recommendations must be performed in the order listed to ensure a comprehensive remediation.

- Block malicious IP addresses
- Blackhole malicious domain names
- Rebuild compromised systems
- Coordinate with cloud and service providers
- Enterprise password change
- Verify all remediation activities

Recovery

Recovery tends to start to move the enterprise to move back to day-to-day business. However, long-term solutions should start to be implemented. Usually, many recovery items are used to improve the overall security of the network to prevent and also detect another incident that happens in a similar way.

Some options could include:

- Improve Enterprise Authentication Model
- Enhanced Network Visibility
- Establish Comprehensive Patch Management Program
- Enforce Change Management Program
- Centralized Logging (SIM/SIEM)
- Enhance Password Portal
- Establish Security Awareness Training Program
- Network Re-Design

Follow Up

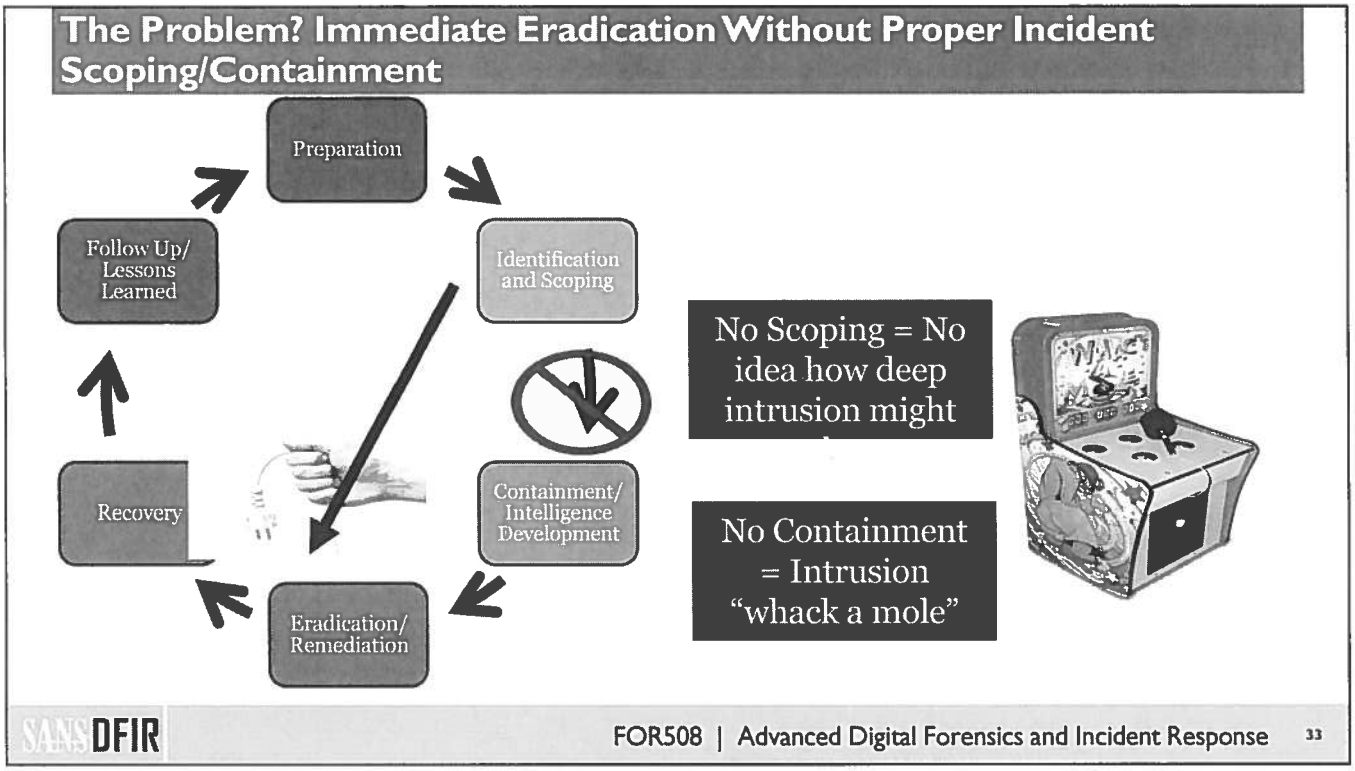
Typically, follow up is needed to verify that the incident has been mitigated, the adversary has been removed, and additional counter measures are being implemented. This is a combination of additional monitoring, network/host sweeps looking for new breaches and beach heads, and finally auditing the network (Pen Test and Compliance) to ensure that the new security mechanisms are in place and functioning normally.

[1] MANDIANT – https://www.mandiant.com/resources/mtrends/

[2] SANS IR Forms – http://www.sans.org/score/incident-forms/

[3] NIST Document – http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf

[4] MANDIANT – http://www.uscc.gov/sites/default/files/3.26.12bejtlich.pdf



The Problem? Immediate Eradication Without Proper Incident Scoping/Containment

The real problem is that not many teams actually follow the six-step incident response process as it is prescribed. They tend to skip immediately to "Eradication/Remediation." It is possible to begin remediation quickly, but not if you are skipping key steps such as containment and the role cyber threat intelligence has in modern intelligence driven incident response doctrine.

A search of the NIST *Computer Security Incident Handling Guide* for the words "power," "plug," "reinstall," or "pulling" resulted in zero hits being found. Many organization's incident response policies trend toward the "immediate eradication" through "pulling the plug/power loss" as their stop gap maneuver to prevent the additional spread of an attacker inside their environment. This traditional incident response doctrine focused on each infected system in isolation. The problem with this lies in the fact that most organizations cannot detect the intrusion early enough to prevent the beachhead system from infecting others. Statistics show that most intrusions have lasted weeks/months and in many cases years prior to detection. Reacting to the intrusion rapidly without properly following the six-step process might lead to a situation many incident responders call "whack a mole" incident response.

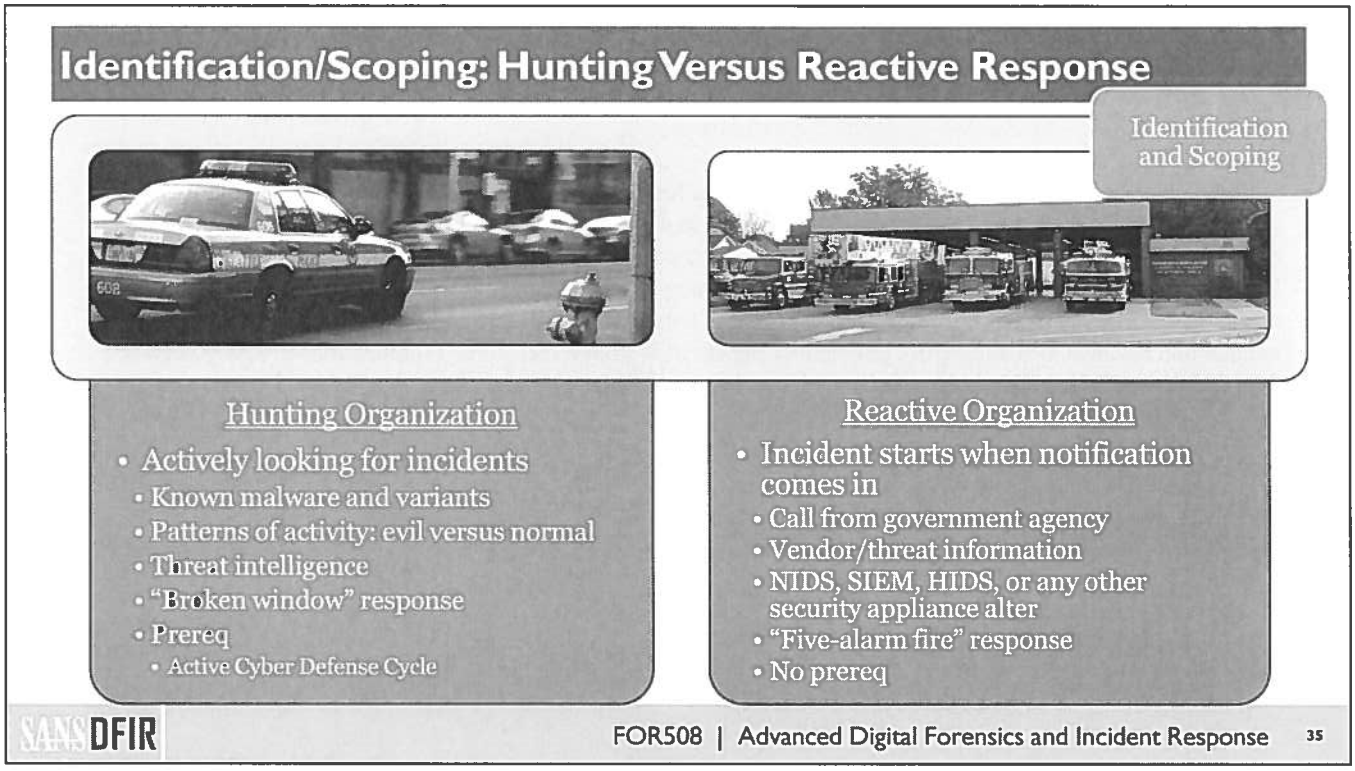
Other methods that are automated similar to "pulling the plug" include blocking IP addresses, rebuilding systems, domains, resetting, and compromised user accounts. These types of activity are automated in many cases and in the end might exasperate the incident by forcing the response team into "whack a mole" incident response.

"Whack a mole" response occurs when you move too fast toward eradication without proper cyber threat intelligence helping direct containment by encircling your adversary so his own gasps for survival will be snuffed out. Without properly containing your adversary, the adversary is free to redeploy his assets around your network for greatest survivability. When you remove only one of the adversary controlled systems, he is usually not bothered. By the time your IR team is done patting itself on a "job well done" and a waiving a "mission accomplished" banner over its head, your adversary is back to compromising additional systems. These systems will inevitably be discovered and the IR team will ask, "Didn't we already remove this hacker before?" This process of eradicating an adversary without

proper scoping of the intrusion or containment will occur many, many times until your IR team begins to realize that what it is doing is having little to no effect on the adversary's control of the systems in your network. In many cases, it took many organizations months to several years to realize this.

What is driving the immediate eradication/remediation call to arms? Many organizations fear losing the data stored on the system to the adversary. It is simply too valuable and the risk is too high. As a result, many IR teams know it is a bad idea to pull the plug but are compelled to do so by management's fear of the horse leaving the barn. But as the analogy goes, closing the barn door after the horses have been let out is useless. Trying to rush to eradicate at this stage leads also leads the APT to possibly react and counteract to your actions of remediation. They might assume this is the beginning of a full-scale remediation and begin a major exfiltration process on other systems they control. This act/react/counteract model is the norm for APT intrusion response. As a result, the longer you can scope your incident and learn where your adversary exists on your network, the more eventual control you will have over them.

Bottom line: Do not react too quickly to an incident by pulling the power. We need to move toward intelligence-driven incident response.



Identification/Scoping: Hunting Versus Reactive Response

Hunting is about taking a proactive versus a reactive approach to identifying incidents.

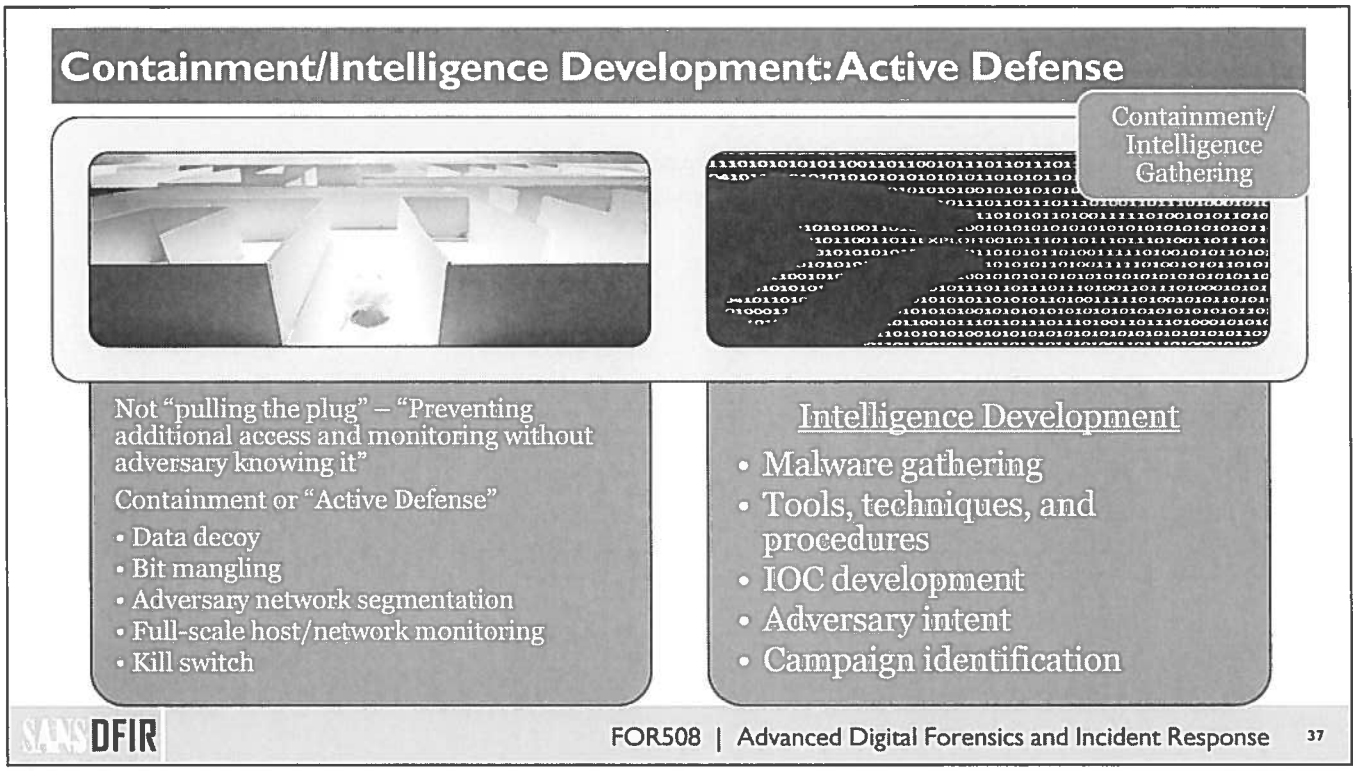
A reactive organization begins incident response when an alert or notification comes in. The alert could come from a third party such as the FBI, or it could come from the organization's own security sensors. The best analogy to a reactive approach is that the IR team is largely waiting to be called into action and relying on accuracy of the notifications it is receiving. Most organizations start building their incident response teams as a reactive organization and there is nothing wrong with that. In many cases, the IR team is largely comprised of augmentation staff that normally fulfill other duties during their regular jobs. As the organization grows larger or if it has an increasing number of incidents, the team is likely to become permanent. The best analogy here is small towns with "volunteer fire departments" vs. "full-time fire fighters." Even larger organizations likely still augment their IR teams with additional personnel internally or might even contract to third-party contractors who provide incident response services.

Organizations move from a reactive organization to a hunting organization when they realize they are not detecting their incidents early enough. The idea of hunting-based response doesn't mean it is an "either or" approach. Most hunting organizations are also reactive organizations, but they begin to task their incident response team to actively engage and hunt for adversaries inside their environment. To accomplish this task, the hunting team typically will be armed with known malware, patterns of activity, or accurate threat group intelligence aiding them in their search.

Organizations who decide to create a hunting organization sometimes fail to see the importance of proper threat intelligence for driving the search in the right areas. Simply tasking a team to "find evil" isn't enough. The team needs to know the difference between normal and abnormal. It needs to know typical hacker tools and techniques. It needs to be skilled in both network and host-based forensics and response to look for the

footprints of these adversaries. Finally, it helps if the organization has invested heavily into a cyber threat intelligence capability that will accurately help guide the team to the right locations on the network to look for specific indicators associated with threat groups interested in that particular data or capability the organization owns.

Without any type of threat intelligence even in its basic form of patterns of typical hacker activity, most hunting groups are simply tasked with looking for "things that look weird" without knowing what weird versus normal even looks like. A trained hunter must know the difference between normal and abnormal as a prerequisite. Even better, if a threat intelligence capability is informing the team, it would look for specific threat groups targeting specific programs using specific techniques. This is an achievable goal. Hunting involves both a manual and automated scanning of systems looking for evil. I have seen some organizations use only network data and do not have proper system/host interrogation capabilities paired with it.



Containment/Intelligence Development: Active Defense

Regardless of how you identify your adversary through hunting or receiving an alert, you will need to contain your adversary. To properly scope an incident and maintain the trust of your organization, you will need to contain your adversary when you find her. Containment essentially degrades the capabilities of your adversary or denies her from achieving her goals. During an intrusion, the more you can learn about your adversary's true intent, the easier it is to achieve true containment. For example, if the adversary simply wants to steal intellectual property and spy on your organization, limiting the adversary from exfiltrating the stolen data would essentially prevent her from achieving operational success. Any possible technique you employ to restrict, limit, and degrade the capabilities of your adversary while she is moving around your network, collecting data, and exfiltrating it would be implemented. This is also called the "cyber kill chain" based response, which we will cover more in-depth later.

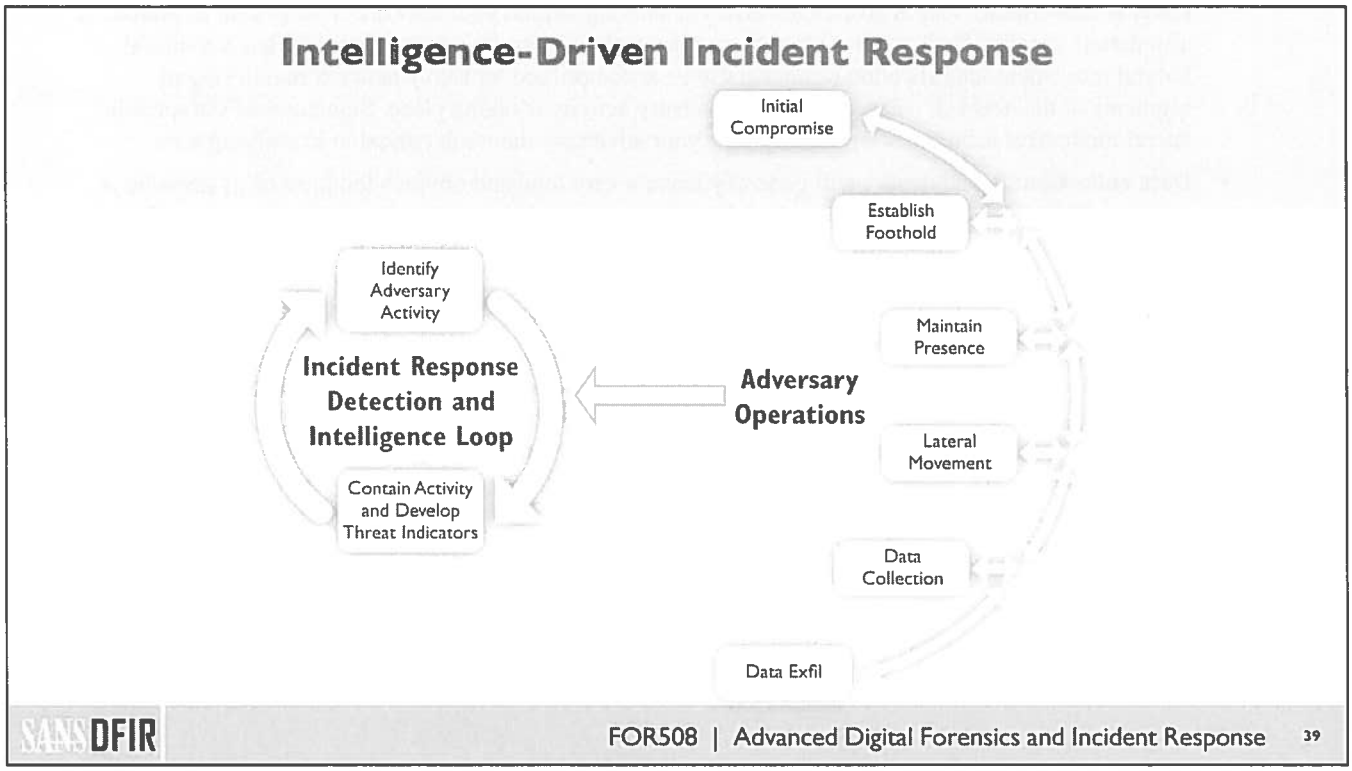
Capabilities employed by responders directly that enable decoy data sets, corrupting data, preventing data exfiltration or lateral movement, or employing kill switches on the network to prevent the adversary from achieving her goals are all considered proper containment options. In most cases, most organizations will utilize full-packet capture monitoring of adversary-controlled systems and network segments. In some cases, organizations have deployed their own host-based monitors and key loggers to specifically spy on the adversary's every move.

After you have successfully limited the capabilities of the adversary, you should also develop initial intelligence about the techniques and capabilities of your adversary. The aim here would help guide your network and host sweeps to help you identify additional compromised hosts in your environment. For example, if you find an instance of evil.exe in a particular directory on a system, then it is likely the adversary also placed a similar file in the same directory on another system. This is called an indicator of compromise or an IOC.

IOC development is extremely important at this phase because it will enable your hunting and IR teams to look for specific evidence that would prove whether a system is truly compromised. An indicator doesn't have to be malware.

A compromised host would be any system that the adversary has examined, utilized, or infected during her campaign on your network. Discovering systems with malware is easier than identifying a system that simply was examined for critical data the adversary might be interested in.

Eventually with enough intelligence data, predicting the adversary's intent is possible. With this information, you are likely to predict the type of data she would be most interested in and deploy additional security capabilities around systems that house that data.



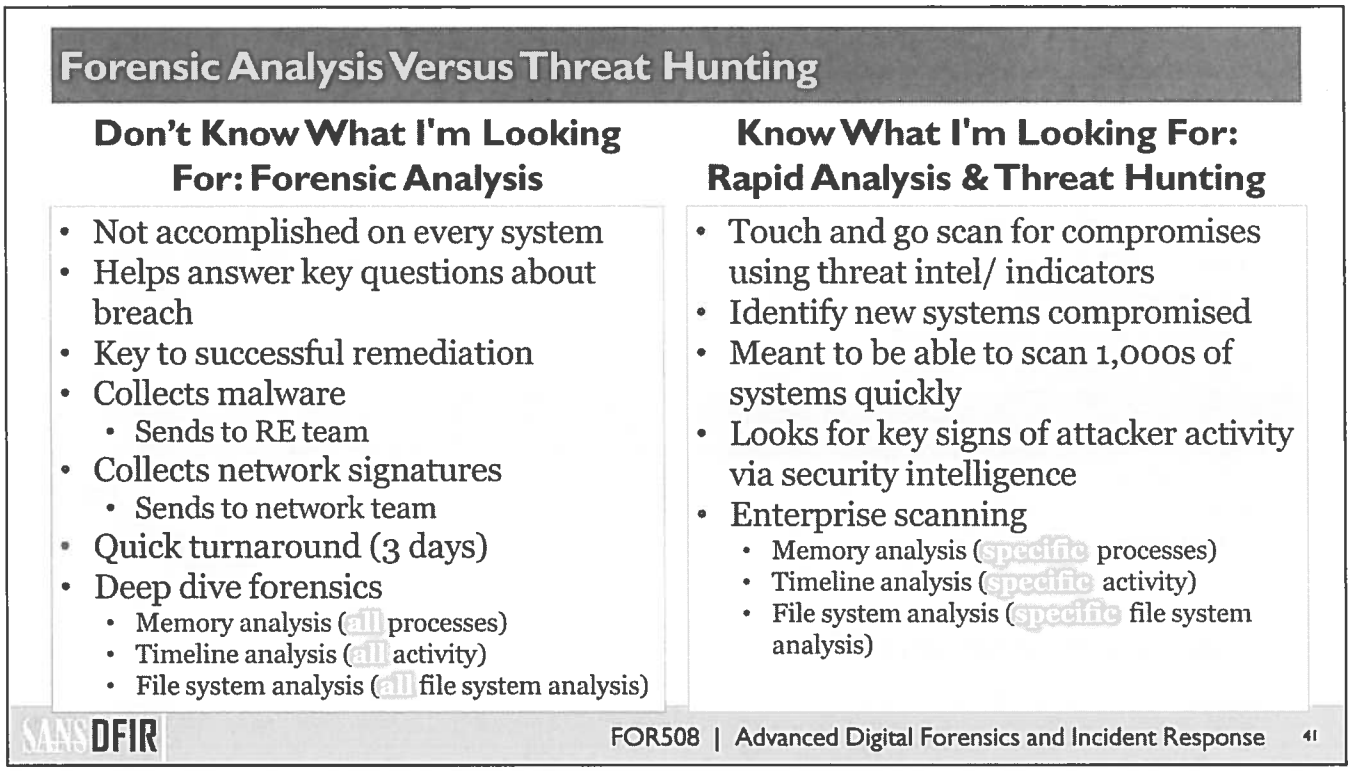
Intelligence-Driven Incident Response

This process of identification/scoping and containment/intelligence development will generally continually loop during and after an incident. In fact, most hunting teams will consistently and continually use this feedback process to help identify actively new systems compromised quicker. It is likely the same adversary will attempt to compromise your network again and an IR hunting team armed with the right knowledge will be in place and actively monitoring the most likely locations that adversary will be found. It will also employ mechanisms, both automated and manual, to identify adversary TTPs that would pop up in the environment.

Specifically, the types of TTPs that are most useful to hunting teams to identify are those associated with specific goals the adversary is trying to achieve. These include:

- **Initial compromise:** Initial attack that gave the intruder threat access to your network. Most initial compromised systems exploits are not persistent and the level of access given to an adversary at this stage is generally very fragile. If a response team can eliminate an adversary before he establishes a beachhead or foothold on your network, then survivability of the adversary drops to nearly zero. This is extremely hard to accomplish because most spear phish attacks are hard to detect and generally the forensic artifacts that drive intelligence-driven incident response by alerting a team to its presence do not yet exist. For example, a registry key setting used for C2 malware. Another example is looking for and collecting data about a specific project in the environment. These types of indicators end up being used quite frequently by incident response teams after the initial compromise transitions to establishing a foothold on the network.
- **Establish foothold/maintain presence:** This is how the adversary has maintained his presence on systems despite rebooting the system. He does not need to recompromise the system again. Any C2 channel that can survive reboot is likely to be a good candidate here. Removing the way an adversary maintains presence on a network will force him to recompromise the system again by establishing a new foothold generally by a new spear phish attack.

- **Lateral movement:** This is how an adversary is moving around your network from system to system. It also details specifically how he compromises new hosts in your environment once he has a foothold. Lateral movement identification can generally be accomplished by heavy network monitoring of segments of the network where suspected adversary activity is taking place. Signatures of the specific lateral movement techniques iareexactly how your adversary moves is critical to identifying him.
- **Data collection:** An adversary will generally leave a very loud and obvious footprint on systems he is trying to identify and collects data and intellectual property of interest on a host. The process of finding specific data on a remote network is challenging. We have a hard time doing it on our own systems, imagine what it would be like to try and find a key file out of millions across hundreds of hosts. Adversaries usually leave many footprints across the network and hosts in an environment as they search for and collect data they are interested in.
- **Data exfiltration:** Once an adversary obtains the data he seeks, he needs to steal it. Stealing that data is easier said than done because moving a large quantity of data outside the network is likely to be caught by a decent SOC analyst or an automated SIEM. In many cases, utilizing a throw-away system is sometimes necessary because it is likely the system will be discovered during the data exfiltration. Many adversaries use this "staging system" in order to accomplish this goal. Data exfiltration is by far the easiest to detect. However, denying the adversary the capability to extract data from the network is a key skill. Bit rotting the exfiltrated data by manipulating outbound packets is a useful capabilities to employ. In addition, setting a network segment "kill switch" that will suddenly terminate all connections from an adversary-controlled subnet that is used for data staging is likely wise.



Forensic Analysis vs. Threat Hunting

One of the keys of this course is to discuss the concept of both using digital forensics to perform deep-dive and enterprise forensics in order to identify new systems compromised and to answer key questions about a data breach. Without proper forensics, your team will be left blind to know what to look for. The other side of the fence includes network-based analysis, which would follow a similar mechanism.

During deep-dive forensics, it should be noted that this is not performed on every system during a data breach. However, it should be used and required on the most likely systems to contain the most intelligence data on them. Sometimes it is a guess as to which system it is, but in some cases it is quite obvious—you might have a system that is simply "louder" than the rest on your network.

Deep-dive analysis usually focuses on being able to quickly analyze a system in less than 3 days that takes a full look at all processes in memory, key activity in the timeline, and analyzing and recovery items deleted and even wiped. The goal of deep-dive analysis is to be able to gather enough intelligence to pass to the RE malware team and to the network team so that proper threat and security intelligence can be learned and distributed to the rest of the team.

Enterprise scanning is meant to cover a lot of ground quickly. However, you need to know what to look for specifically. These are specific scans looking for a specific process, file, registry key, or activity on a system that would indicate that it is also compromised. If your scanning capability is decent and quick enough, you could even use this methodology to find new intrusions and move up the kill chain detection earlier.

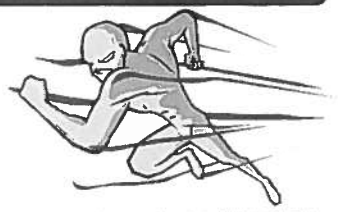
Deep-dive forensics:

Memory analysis (**all** processes)
Timeline analysis (**all** activity)
File system analysis (**all** file system analysis)

Switching from Rapid Analysis to Deep Dive Analysis

Immediate/Quick Response

- Answers in 4 to 6 hours
  - Enterprise Response
  - Memory Forensics
  - Timeline Analysis
  - Initial Incident Assessment
  - Threat Indicators Developed



Deep Dive Analysis

- Answers in 1 to 2 days
  - Deep Dive Analysis
  - Anti-Forensic Detection
  - Malware and Adversary Identification
  - Threat Capabilities and Purpose
  - Additional Threat Indicators Created

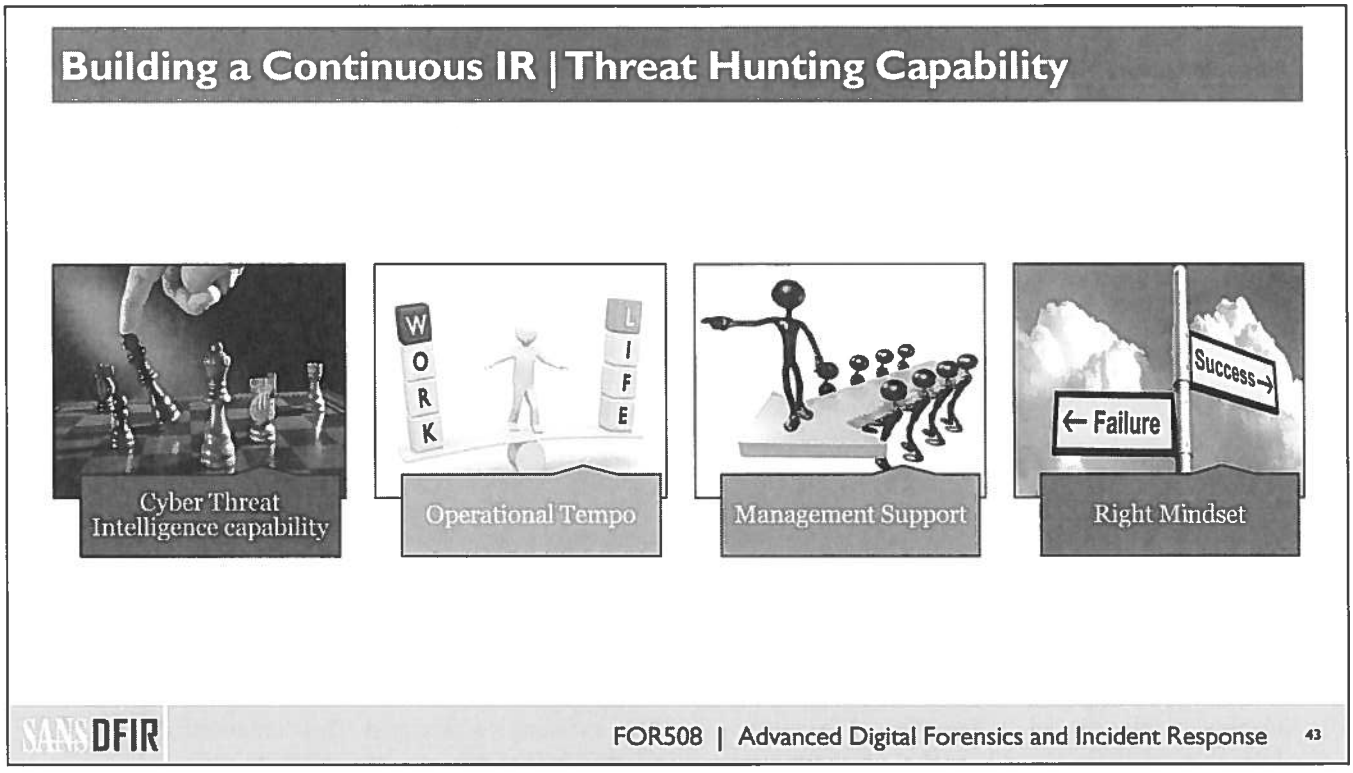
At this point, we will move from our rapid triage and analysis mode designed to produce answers in only a few hours. During the initial phase, we concentrated on **analysis techniques** that were quick and resulted in the ability to provide quick turn-around answers during incident response. During the first 24 hours, the majority of your work will likely be immediate and quick response analysis techniques.

However, not every incident can be solved with our quick analysis techniques. In cases in which adversaries utilize more efficient anti-forensic techniques or are more careful about their movements around your enterprise, to discover their tools, techniques, and procedures to pull out effective cyber threat intelligence triggers, we will need to dive deeper.

From here forward in the class (until the last day) are techniques that we would not recommend you accomplish on **every** system, but systems that you feel might need a deeper look. It is hard to tell exactly which systems might need this additional scrutiny. As a result, the initial assessment might help you triage and identify systems that might yield the most data for a deep dive analysis.

Deep dive analysis will usually also give us a better look into why the threat has decided to target your organization and attempt to answer questions to their intent. "What are they after?" "How skilled is this threat group?" "Do they use anti-forensic techniques to hide?" "Are they good at hiding their malware?"

In the end, deep dive analysis techniques cannot be applied to every system; it is too time-consuming and slow to achieve. But doing it on specific systems would be a wise decision.



Building an IR Hunting Capability

Before you consider how cool it sounds to have a proactive hunt team in your organization, there are a myriad of things to consider. First of all, in addition to a robust containment capability, most organizations need mature capability to deal with remediation and recovery phases of incident response.

Cyber Threat Intelligence Role in Hunt Teaming

Another key component to building a hunt team is a cyber threat intelligence capability that resides inside your security team and feeds directly into your hunt team. It is always surprising to hear organizations that have committed to staffing a hunting team and when asked what the team is looking for on the network, the response is "We don't know." This is tragic because the teams should be able to differentiate between normal vs. abnormal on a system. They could employ mechanisms using "frequency of least occurrence" to identify anomalies. However, if they are tasked with doing this across the entire enterprise, it is especially overwhelming.

A proper cyber threat intelligence capability will arm the hunting team with:

- **Where to look**: What types of data might the various known APT groups be particularly interested in?
- **What to look for**: IP addresses, malware footprints, registry keys, utility tools such as PSEXEC, etc.
- **Likelihood of attack**: Which threat group is most likely to attempt to compromise our network?

Many hunt teams will tell me that they have a registry key that is connected with a known APT group, but have no capability across their enterprise to look for it. Generally, it is suggested to use AD Group Policy or PowerShell commands to sweep an enterprise segment looking on each host for the specific registry key. The hunt team informs us that this is impossible. They do not have administrator rights on the network. My head

usually impacts my keyboard when I hear this because your hunt team should be one of the most trusted capabilities in your IT organization. If there are trust issues, then we generally recommend that the hunt team consist of at least one dedicated system administrator with domain administrator credentials just so the team can operate with agility. Active defense requires agility.

In a nutshell, building a hunt team requires the ability to access a wide variety of data sources and capabilities in the enterprise. The more options a team has for gathering and examining data, the hunt team has a much higher success ratio.

Right Mindset

Hunt teams also require both manual and automated methods of collecting and searching data across an enterprise network. A single hunt team member should be able to scale up to searching thousands of hosts for a single forensic artifact. Alternatively, a member might need to dive deeply into a single system trying to uncover unknown malware she feels exists. The challenge though, like with all forensics, is to know when you are truly done and the hunter should move on. The analysts always will feel like he missed something, didn't have the right skills to find it, or the adversary is simply better than him. No amount of searching will help remove the doubt that comes with hunting and not finding anything of substance. A good hunt team manager will constantly need to nudge analysts on to the next artifact to look for.

Hunt Team Operational Tempo

The other challenge of hunt teams is operational tempo. Usually incident response initiated by reactive response teams is usually a sprint, 24-hour days, 7 days a week, until the incident is remediated. This tempo ends up leaving many husbands and wives at home for weeks on end until their spouse returns from their incident response mini-deployment. This scenario is typical for reactive response teams but will be the death of hunt teams.

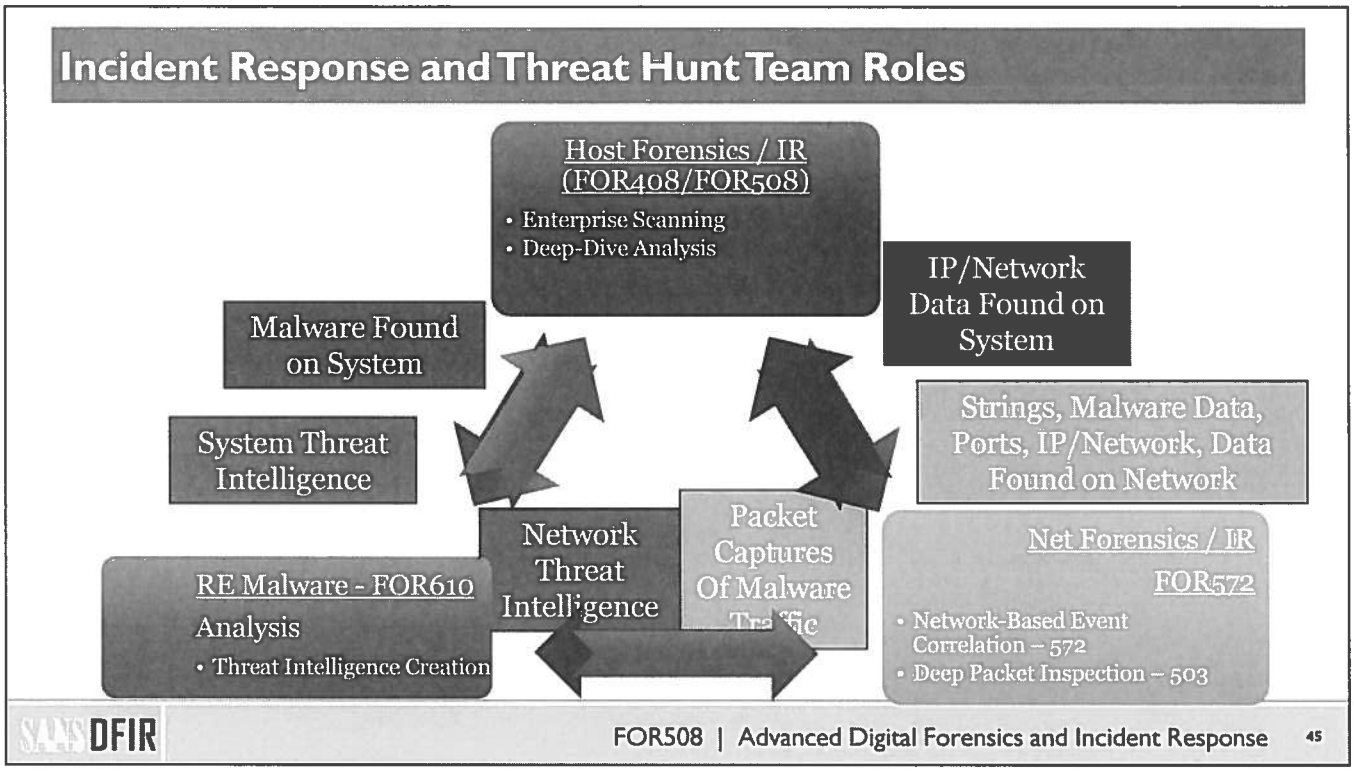
Hunt teams will consistently find new breaches if they are good. If every breach detected is treated like a 24/7 fire drill, then the team will likely always be exhausted. Even fire fighters fighting blazes for week after week in a forest have to take days off to recuperate. In fact, the operational tempo is slower to ensure no one is working a fire line that is exhausted. Although incident response in the cyber world has not yet been attributable directly to a loss of life, exhausting your incident response hunt team isn't a good strategy.

Forcing your incident response hunt teams to take days off, weekends off, vacations, and spending time with family is a must. They should come in at 8 AM and work till 5 PM like everyone else. The only difference is that a hunt team works its 9 hours a day finding evil and responding to it consistently. Good hunt teams will always be in the middle of an engagement fighting an adversary—fighting the adversary is simply what their job is. It is not a fire drill.

Management Support of the Hunt Team

Management buy-in is also a must. In many cases, management thinks it wants to know about breaches, but we have found over the years many organizations are worried about what happens when breaches are found rather than the fact they exist in the first place. This is due to regulatory or fines that the organization could be hit with if a breach is discovered. You might get management buy-in on paper, but in reality management views IR hunt teaming similar to going to the doctor to see whether it contracted a superbug disease or cancer. No one really wants to know even whether it is good for him in the long run. If he knows about it, then he has to do something about it.

Not all management ignores the usefulness of a hunt team. Generally, organizations that are hit enough by adversaries have begun to form a thickness and desensitization to the news of a breach. Management will simply want to know how effective the team is and how many adversaries it is tracking currently. Management generally warms up to the idea of hunt teams eventually, but don't be surprised if it isn't very excited initially when a hunt team ends up doing its job exceptionally well.



IR and Hunt Team Roles

To use security intelligence effectively during the six-step IR process. You need to collect your data to be processed to look for atomic, computed, and behavioral indicators. To accomplish this, set up your team correctly so that your IR team is focused on intelligence creation as much as possible during the identification and containment phase of IR.

Digital forensics and IR is the process to analyze systems (host and network data) in order to properly identify all systems compromised and the ability to learn exactly what remediation steps will need to be taken in order to stop the incident. As a part of forensics, being able to properly collect intelligence of the adversary will depend greatly on your team's ability to analyze the remnants of the activities of the intruder and then use that data as a way to "catch" him on other systems.

As a result, your team is likely to be made up of host, network, and reverse engineers that will be working side by side to help identify new systems compromised, create new threat intelligence data, and finally use that data to identify new systems and engineer additional defenses that can be used to help stabilize the current incident.

The challenge today is that our adversaries might not simply "go away." It is likely they will come back to your network again and deploy more advanced techniques in order to maintain persistence.

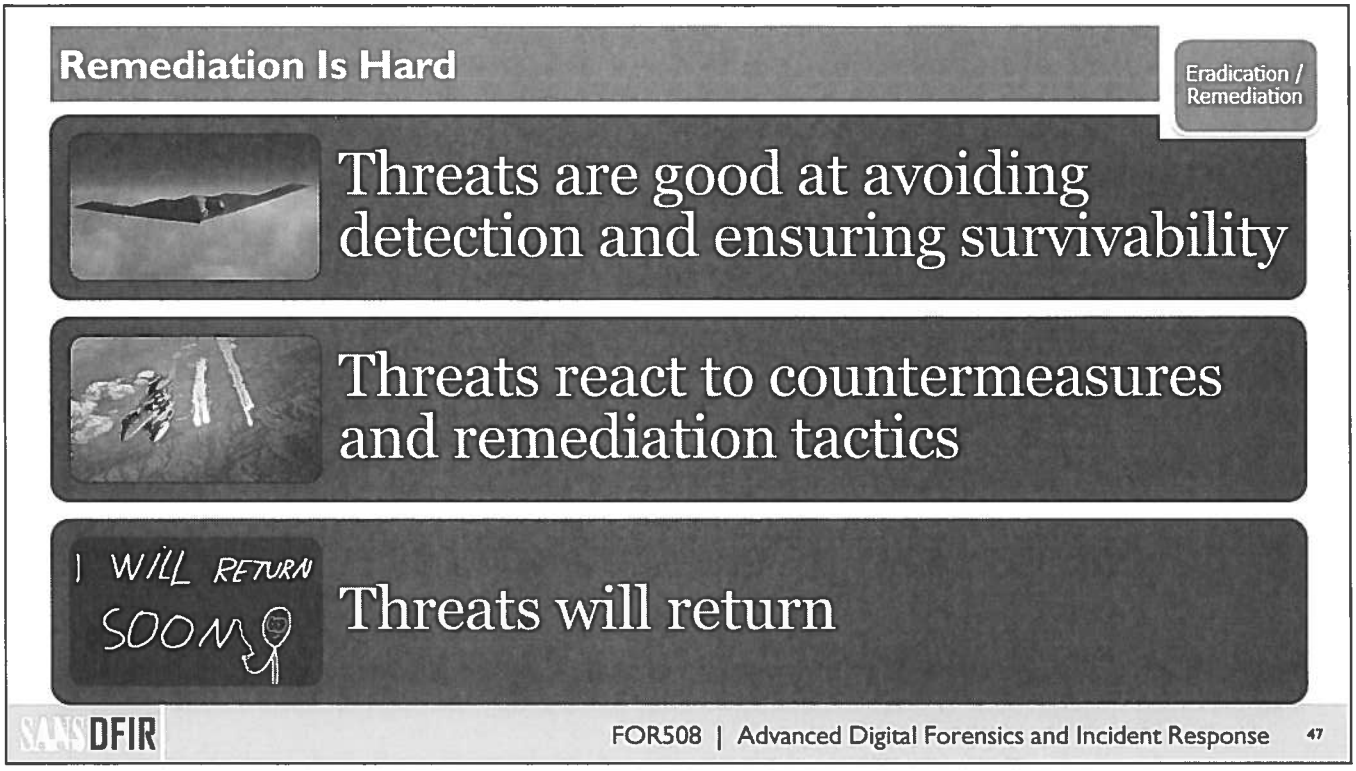
To enact the kill chain successfully, each IR team should have a team with a variety of skills. Each group helps support the other groups. Here is a suggestion for the composition of a team:

- 1 Team lead
- 1-2 System and host forensicators
- 1-2 Network forensicators
- 1 RE malware specialists

Finally, the team should have an enterprise scanning capability looking for both host- and network-based signatures discovered during the intrusion. This will help enable the ability to stop beach heads before they are able to laterally move or exfiltrate your data from the network.

Enterprise scanning

- Memory analysis (**specific** processes)
- Timeline analysis (**specific** activity)
- File system analysis (**specific** file system analysis)



Remediation Is Hard

Nothing is more important to your organization than finally removing the adversary threat from your network. This is much easier said than done and most organizations almost always skip to this step immediately after incident detection. Without a proper scoping and containment, remediation is not possible. Typically, you end up only annoying the adversary and they end up returning very soon. In the end, remediation is a part of an ongoing incident response cycle. The adversaries will always attempt to return and they likely will improve their own TTPs when they do.

What generally happens is a similar life cycle to the following events:

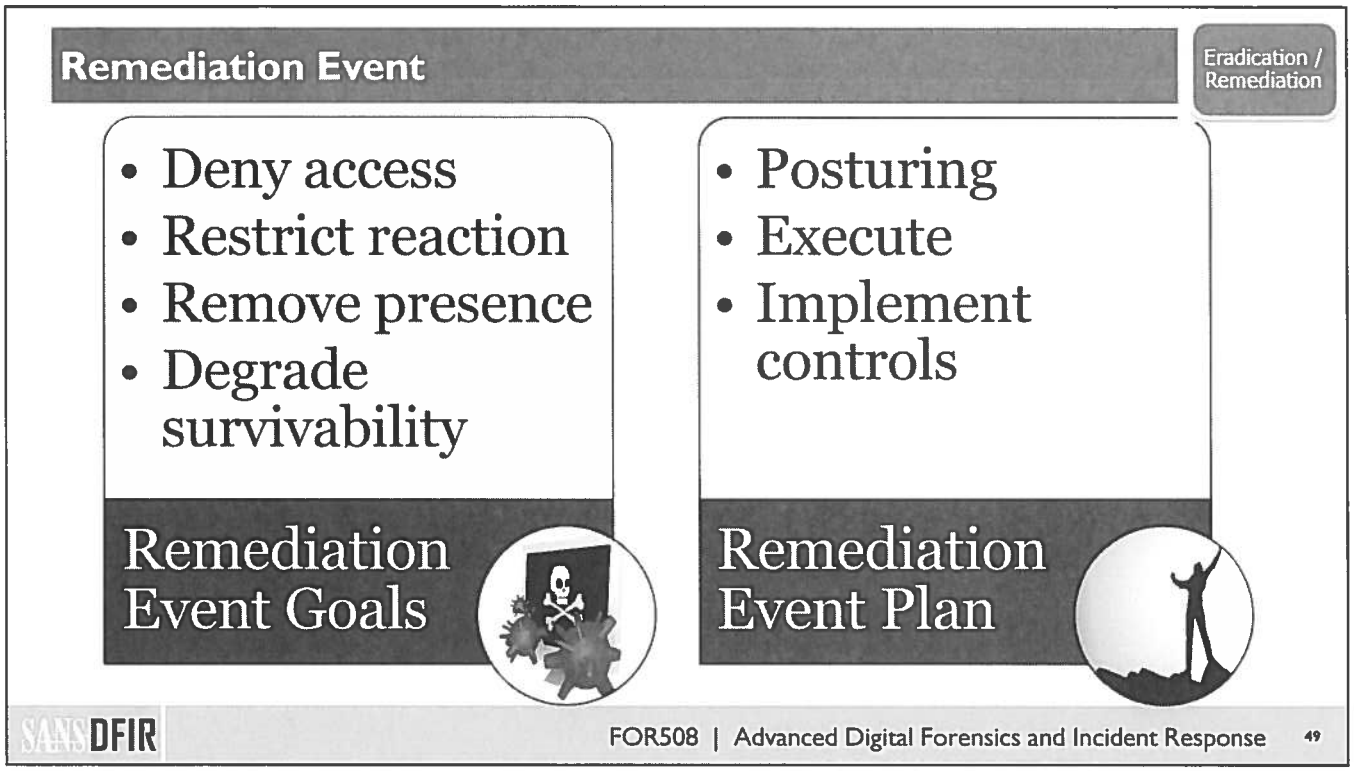
1. Response team removes/rebuilds all known compromised hosts and blocks IP addresses, domains, and resets possible compromised accounts.
2. Response team, not having scoped out the full intrusion before tipping its hand, ends up showing the adversary how it found her by removing specific systems.
3. Adversary, not knowing if the response team is actually any good, deploys new malware that she hasn't used yet in an effort to ensure long-term access to the network.
4. Response team and management feel a sense of satisfaction as they "removed the threat" from their environment.
5. Attacker maintains access but is using new capabilities not seen before by the response team.
6. This continues until the attacker makes a mistake or an external organization notifies the organization of the intrusion and the cycle generally repeats itself.
7. Go to step 1; full eradication did not take place. The adversary has survived.

Why is successful remediation difficult to accomplish successfully?

1. Threat adversaries are good at avoiding detection and plan at being detected at some point. As a result, they will go to great lengths at ensuring survivability beyond remediation.

2. Threat adversaries are likely to react to any countermeasure or remediation tactic employed by the incident response team.
3. Threat adversaries won't simply go away. Continually target organizations regardless of countermeasures used. They will try and come back. In some cases, the adversary begins a new wave of attacks the day following remediation.

A decent white paper on proper remediation steps and posturing has been written by Mandiant and is located on your USB in the documents folder called "Remediating-Intrusions.pdf."



Remediation Event

Remediation takes a bit of time to plan. It usually involves more groups outside of the incident response team to coordinate a series of tightly burst and controlled activities over a short period of time called a "Remediation Event."

Remediation events have generally occurred over a weekend where an organization can commit to purging an adversary from its network without impacting business operations greatly.

A remediation event should:

1. Deny access to the environment to the adversary.
2. Eliminate the ability for the adversary to react to the remediation.
3. Remove the presence of the adversary from the environment.
4. Degrade the ability of the adversary to return.

A remediation event generally consists of activities that would deny access to the environment to the adversary, eliminating the ability for the adversary to react to the remediation, remove the presence of the adversary from the environment, and finally prevent additional compromises by applying additional security controls.

Remediation consists of three steps:

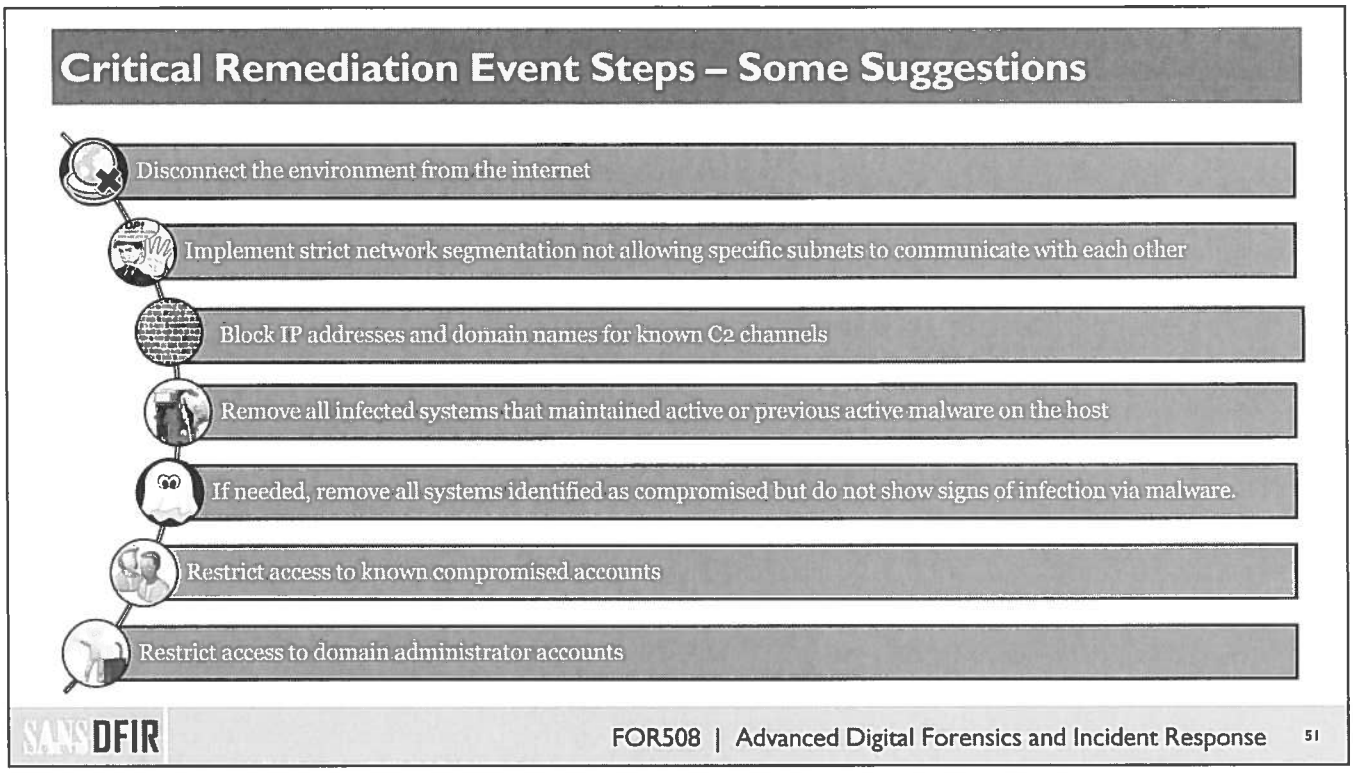
1. Posture for remediation.
2. Execute remediation.
3. Implement and apply additional security controls.

During the planning of the remediation, organizations should begin to tune their existing sensors and add additional capabilities to allow for more monitoring of the network before, during, and post-remediation.

Proper posturing actions should increase monitoring while continually identifying additional systems compromised.

Actions to increase monitoring include but are not limited to:

1. Full content packet captures in compromised segments of the network
2. Full Netflow data originating from all egress points on the network
3. Retain and maintain all DHCP, VPN, firewall, and Web logs



Critical Remediation Event Steps

During the remediation, there is no one right solution to apply. This is one of the reasons that remediation planning takes some time to complete. Understanding and knowing your adversary through intelligence-driven incident response is key. You need to know every host and system compromised by your attackers. You should detail security controls that would degrade and deny the ability of the adversary to function properly. Regardless of the specific tactical options you might consider planning during remediation, here are a few critical recommendations we recommend you consider when you plan for your own remediation.

Critical remediation controls include but are not limited to:

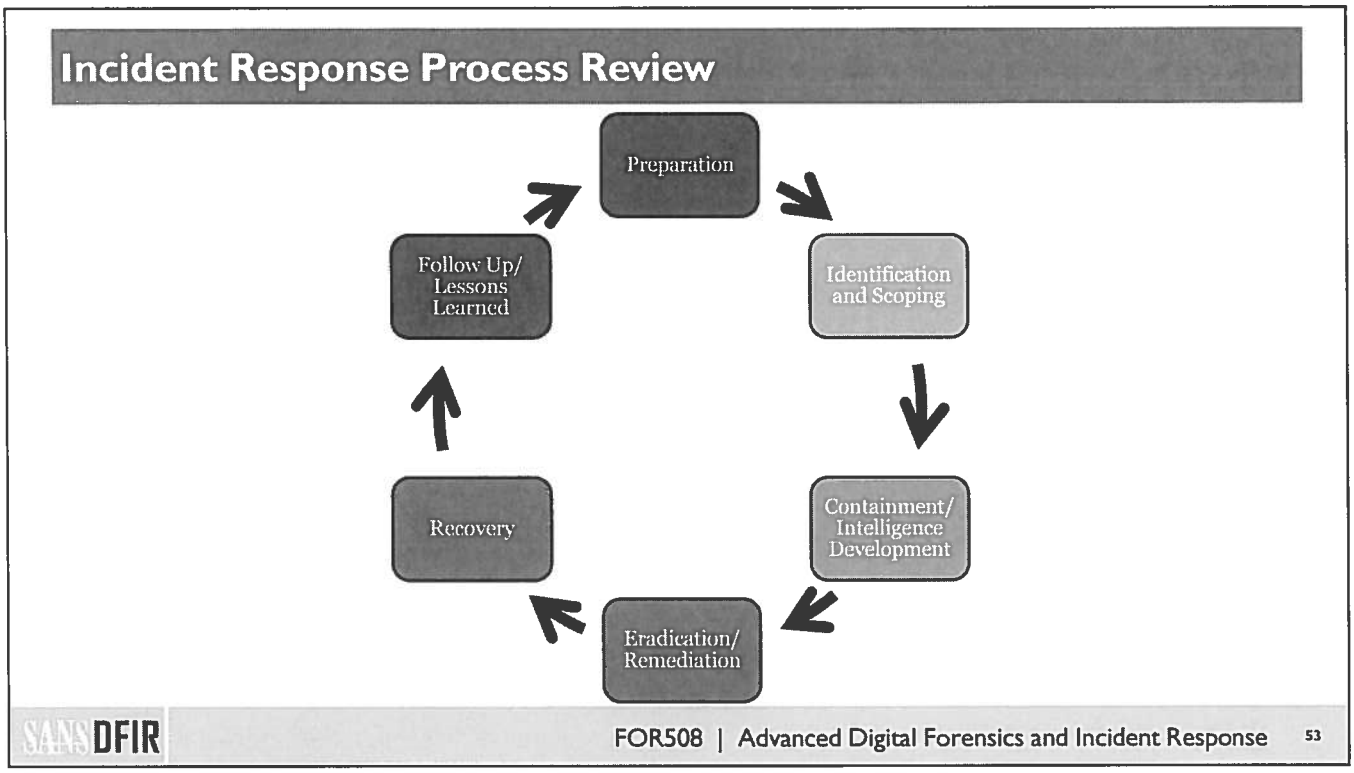
1. Disconnect the environment from the Internet.
2. Implement strict network segmentation not allowing specific subnets to communicate with each other.
3. Block IP addresses and domain names for known C2 channels.
4. Remove all infected systems that maintained active or previous active malware on the host.
5. If needed, remove all systems identified as compromised but do not show signs of infection via malware.
6. Restrict access to known compromised accounts.
7. Restrict access to domain administrator accounts.
8. Validate everything above is done properly.

The last step is incredibly important because people do make mistakes. Leaving one compromised host online after a remediation event likely means you might need to start from scratch in the end. Everyone wants this to succeed, so it is imperative that there are two sets of eyes to verify that each task as it is done properly. A

mistake during this phase is costly because it not only allows continued access for the adversary, but it also tips your hand of what you know about him. The adversary is likely to immediately scramble to deploy new malware, maintain a presence by infecting additional hosts that might have already been cleaned up, and change his tactics so he might become more invisible to the incident response team.

Finally, once the remediation is successful, additional security controls should begin being deployed in the environment. The idea here is that you want to implement additional measures that would increase the chance of detection of the adversary while degrading the ease of maneuverability to the threat. There are many new solutions that can be implemented, but even following the SANS Critical Controls is a good first step.[1] Doing the basics such as following the critical controls or network hygiene actions makes IR, active defense, etc. much more doable and much less costly - it eliminates such significant noise that makes finding the adversaries much easier.

[1] https://www.sans.org/critical-security-controls/



This page intentionally left blank.

This page intentionally left blank.

Cyber Threat Intelligence and Indicators

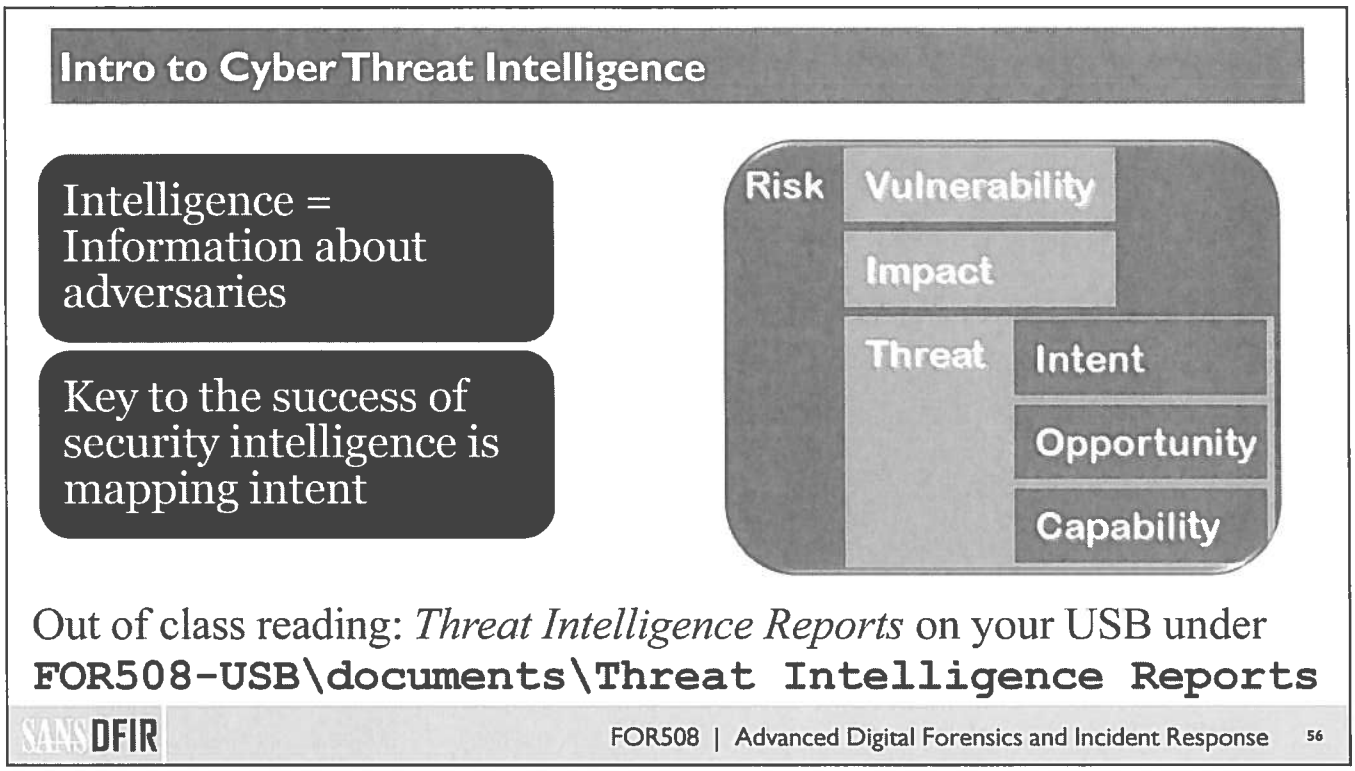
What Is Cyber Threat Intelligence?

"I am your enemy, the first one you've ever had who was smarter than you. **There is no teacher but the enemy.** No one but the enemy will tell you what the enemy is going to do. No one but the enemy will ever teach you how to destroy and conquer. Only the enemy shows you where you are weak. Only the enemy tells you where he is strong. And the rules of the game are what you can do to him and what you can stop him from doing t you. I am your enemy from now on. From now on I am your teacher." —Orson Scott Card, *Ender's Game*

SANS DFIR
DIGITAL FORENSICS & INCIDENT RESPONSE

What Is Cyber Threat Intelligence?

"I am your enemy, the first one you've ever had who was smarter than you. **There is no teacher but the enemy.** No one but the enemy will tell you what the enemy is going to do. No one but the enemy will ever teach you how to destroy and conquer. Only the enemy shows you where you are weak. Only the enemy tells you where he is strong. And the rules of the game are what you can do to him and what you can stop him from doing to you. I am your enemy from now on. From now on I am your teacher." —Orson Scott Card, *Ender's Game*



Intro to Cyber Threat Intelligence

Source with permission from Mike Cloppert originally published on the SANS Computer Forensics Blog: http://digital-forensics.sans.org/blog/2009/07/23/security-intelligence-introduction-pt-2/

Understanding Risk

As I like to say, we are in the business of risk management. To understand security intelligence, it is imperative that we properly scope and carefully define this concept. Different fields define risk in different terms, but in security, risk is the product of three primary components: vulnerability, impact, and threat:

- **Vulnerability**: Vulnerability is sometimes replaced with "exposure." I would argue that they are represented together as one component. Vulnerability is both mutable and ephemeral. This is good, because it means this component of risk can be affected by individuals and organizations. Applying the principle of least privilege, network segmentation, robust system management, and adherence to software development and life-cycle best practices are but a few high-level examples of how vulnerability (or exposure) can be reduced, with a proportional reduction of risk. The operative word here is "reduced," not "eliminated." Again, vulnerability reduction, as you will see, is necessary but insufficient.
- **Impact**: Impact is immutable and changes are either slow or non-existent. This is what happens when security systems fail and the confidentiality, integrity, or availability (but mostly the first two) of data or systems are compromised. This is largely a property of your organization and its operational context—physical, industrial, and what have you. There is typically not much you can do to influence impact.
- **Threat**: Threat is the most important risk component in intelligence-driven response. In fact, one could say that security intelligence is threat-driven security. To understand, differentiate, and properly respond to threats, it is helpful to divide this concept into a further three components: intent, opportunity, and capability

(IOC). These terms are the MMO (means, motive, opportunity) of security intelligence—in fact, they map nicely to one another, but I feel IOC encourages more clarity of thought on threat.

Threats can be examined using the following descriptions:

- **Intent**: Intent stems in a way from impact. It is immutable, and driven by the industry you are in just as impact is. Typically, at a high level, the intent of adversaries to whom security intelligence techniques are applied is data theft—CNE (Computer Network Exploitation), if you will. Of course, for each intrusion, each compromise, or each actor, the intent will most likely be slightly different. Is the goal of the adversary to compromise operational details of a campaign, or technical details of a widget? There is nothing that can be done to influence intent.

- **Opportunity**: Opportunity is about timing and knowledge of the target space. In some cases, it pairs with vulnerability, but not always. It is one thing to use a product with a 0-day vulnerability in it, but quite another when your adversary knows this. In other respects, however, opportunity is less related. For instance, wouldn't a company's benefits open enrollment period be a great time for a targeted attack on users using socially engineered, topically relevant e-mail as a delivery vector?

- **Capability**: Put simply, capability is the ability of adversaries to successfully achieve their intended goal and leverage opportunity. It is influenced by things such as the skills of the adversaries and the resources (financial, human, and technical) available to them. To extend the 0-day example, a target might be vulnerable, the adversary might intend to steal data by exploiting this 0-day, but if he or she cannot write or obtain the exploit, then the risk is lower.

The Threat Environment

The "intelligence" in intelligence-driven response is the information acquired about one's adversaries, or collectively the threat landscape. Each industry has a different threat landscape, and each organization in each industry has a different risk profile, even to the same adversary. Understanding one's threat environment is collecting actionable information on known threat actors for Computer Network Defense (CND), whether that action is purely detection or detection with prevention. Now is the time to mention that there is no such thing as protection without detection, or protection without reaction, in this environment. This will be discussed in more detail in Part 3.

By combining information on a threat with observations of activity, one can more effectively and in some cases heuristically defend one's data and systems. Perhaps a heuristic or anomalous event indicative of malicious activity occurs too frequently across your enterprise to respond to it every time it happens. To borrow some parlance, these are also referred to as Tactics, Techniques, and Procedures (TTPs). If this maps directly to the TTP of a particular adversary, and you know this adversary's intent is to acquire data, which is concentrated in a particular portion of your network, you can investigate the heuristic with this scoping that would otherwise be unreasonable to leverage.

More discretely, discovering the infrastructure, tools, and preferred techniques of each particular adversary, and having processes in place to leverage the data, allow you to detect hostile activity even if all but one minor aspect of an adversary's attempt to break in has changed. Let's take an easy example. If an adversary uses an IP address in an attack, you don't just want to block it at your firewall. You want to detect when it is used in the future, and also not reveal to the adversary that you discovered the attack; otherwise, he'll just switch IPs. You want to let him think subsequent attacks were successful, and then research these attacks for "new" (or "different") techniques, which can then in turn be pivoted on for further defense in case the adversary does ever switch to a new IP.

In this threat environment, you cannot rely on traditional tools like firewalls, IDS, and (especially) anti-virus. These tools can sometimes be leveraged to achieve detection or protection goals, but it will be you that is defining those conditions, based on your security intelligence—not your vendor. These vendors have by and large failed to adapt to targeted attacks, and most are interested only in protecting against the broader, easier problems. This isn't easy, folks, but trust me when I say it's pretty effective.

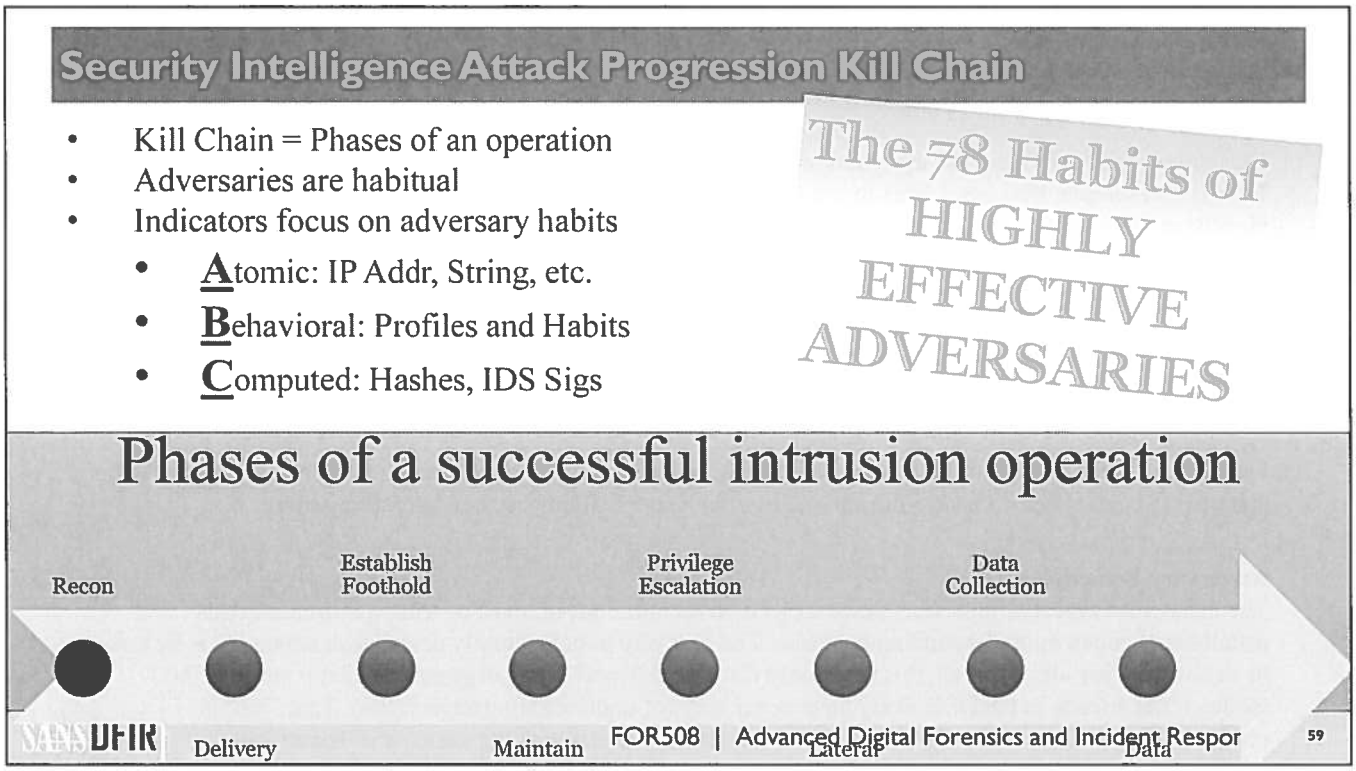
Appropriate Application of Techniques

Key to the success of security intelligence is mapping intent to impact. If your research and compromise response investigations reveal that adversaries are intent on stealing data, then there is little reason to be concerned about denial-of-service attacks from those actors, because the impact of such an activity is completely orthogonal to the goal of a confidentiality breach, and the ancillary goal that is often paired with it, invisibility.

It is also important to understand the threat, which is likely behind certain hostile activities. These techniques are not wisely applied to commodity viruses or massive worms; such rigor provides little ROI from an analytical perspective and tends to waste resources on a problem, which can be adequately addressed with existing security tools and infrastructure. Only APT actors should be subject to such scrutiny. Naturally, this creates a derivative challenge: Not only must you now identify hostile versus benign activity, but further which of that hostile activity corresponds to APT actors! This needle-in-a-needlestack challenge is, at times, very difficult, but as you wrap your head around these techniques it becomes easier in some cases. Unfortunately, our adversaries know all too well that they can hide in the cruft, and can (and do) exploit this.

One way to think about this is by answering the question of whether an attack or intrusion is one of opportunity, or intent. Opportunistic intrusions are generally a problem solved by existing best practices (architecture, AV, patching, classic IR model, etc.), rather than this analytical offshoot we're calling SI. As that last sentence suggests, it is not the end-all, be-all to CND, but rather one component of a large and complicated affair in information security.

Out of class reading: *Threat Intelligence Reports* on your USB under **`FOR508-USB\documents\Threat Intelligence Reports.`**



Security Intelligence Attack Progression Kill Chain
Source with permission from Mike Cloppert originally published on the SANS Computer Forensics Blog at http://digital-forensics.sans.org/blog/2009/10/14/security-intelligence-attacking-the-kill-chain/.

Now, we will introduce the attack progression (known as "kill chain") and briefly describe its intersection with indicators. The next segment will go into more detail about how to use the attack progression model for more effective analysis and defense, including a few contrived examples based on real attacks.

On Indicators

Just like everyone, adversaries have various computer resources at their disposal. They have favorite computers, applications, techniques, websites, etc. It is these fundamentally human tendencies and technical limitations that we exploit by collecting information on our adversaries. No person acts truly random, and no person has truly infinite resources at their disposal. Thus, it behooves us in CND to record, track, and group information on our sophisticated adversaries to develop profiles. With these profiles, we can draw inferences, and with those inferences, we can be more adaptive and effectively defend our data. After all, that's what intelligence-driven response is all about: defending data that sophisticated adversaries want. It's not about the computers. It's not about the networks. It's about the data. We have it, and they want it.

Indicators can be classified a number of ways. Over the years, my colleagues and I have wrestled with the most effective way to break them down. Currently, I am of the mind that indicators fall into one of three types: atomic, computed, and behavioral (or TTPs).

Atomic indicators are pieces of data that are indicators of adversary activity on their own. Examples include IP addresses, e-mail addresses, a static string in a Covert Command-and-control (C2) channel, or fully qualified domain names (FQDNs). Atomic indicators can be problematic, because they might or might not exclusively represent activity by an adversary. For instance, an IP address from where an attack is launched could very likely be an otherwise-legitimate site. Atomic indicators often need vetting through analysis of available historical data to determine whether they exclusively represent hostile intent.

Computed indicators are those which are, well, computed. The most common among these indicators are hashes of malicious files, but can also include specific data in decoded custom C2 protocols, etc. Your more complicated IDS signatures might fall into this category.

Behavioral indicators are those that combine other indicators (including other behaviors) to form a profile. Here is an example: Bad guy 1 likes to use IP addresses in West Hackistan to relay e-mail through East Hackistan and target our sales folks with trojaned MS Word documents that discuss our upcoming benefits enrollment, which drops backdoors that communicate to A.B.C.D. Here, we see a combination of computed indicators (Geolocation of IP addresses, MS Word attachments determined by magic number, and base64 encoded in e-mail attachments), behaviors (targets sales force), and atomic indicators (A.B.C.D C2). Already, you can probably see where we're going with intelligence-driven response. What if we can detect, or at least investigate, behavior that matches that which I describe previously?

One likes to think of indicators as conceptually straightforward, but the truth is that proper classification and storage has been elusive. I'll save the intricacies of indicator difficulties for a later discussion.

Adversary Behavior

The behavioral aspect of indicators deserves its own section. Indeed, most of what we discuss in this installment centers on understanding *behavior*. The best way to behaviorally describe an adversary is by how he or she does his job. After all, this is the only discoverable part for an organization that is strictly CND (some of our friends in the USG likely have better ways of understanding adversaries). That "job" is compromising data, and therefore we describe our attacker in terms of the anatomy of her attacks.

Ideally, if we could attach a human being to each and every observed activity on our network and hosts, we could easily identify our attackers, and respond appropriately every time. At this point in history, that sort of capability passes beyond pipe dream into ludicrous. However mad this goal is, it provides a target for our analysis: We need to push our detection "closer" to the adversary. If all we know is the forged e-mail address an adversary tends to use in delivering hostile e-mail, assuming this is uniquely linked to malicious behavior, we have a mutable and temporal indicator upon which to detect. Sure, we can easily discover when it's used in the future, and we are obliged to do so as part of our due diligence. The problem is this can be changed at any time, on a whim. If, however, the adversary has found an open mail relay that no one else uses, then we have found an indicator "closer" to the adversary. It's much more difficult (though, in the scheme of things, still somewhat easy) to find a new open mail relay to use than it is to change the forged sending address. Thus, we have pushed our detection "closer" to the adversary. Atomic, computed, and behavioral indicators can describe more or less mutable/temporal indicators in a hierarchy. We as analysts seek the most static of all indicators, at the top of this list, but often must settle for indicators further from the adversary until those key elements reveal themselves.

That this analysis begins with the adversary and then dovetails into defense makes it very much a security intelligence technique as we've defined the term. Following a sophisticated actor over time is analogous to watching someone's shadow. Many factors influence what you see, such as the time of day, angle of sun, etc. After you account for these variables, you begin to notice nuances in how the person moves, observations that make the shadow distinct from others. Eventually, you know so much about how the person moves that you can pick him out of a crowd of shadows. However, you never know for sure if you're looking at the same person. At that point, for our purposes, it doesn't matter. If it looks like a duck, and sounds like a duck... it hacks like a duck. Whether the same person (or even group) is truly at the other end of behavior every time is immaterial if the profile you build facilitates predicting future activity and detecting it.

Attack Progression, or the Kill Chain

We have found that the phases of an attack can be described by six sequential stages. Once again, loosely borrowing vernacular, the phases of an operation can be described as a "kill chain." The importance here is not

that this is a linear flow—some phases might occur in parallel, and the order of earlier phases can be interchanged—but rather how far along an adversary has progressed in his or her attack, the corresponding damage, and investigation that must be performed.

Recon

The reconnaissance phase is straightforward. However, in security intelligence, oftentimes this is manifested not in portscans, system enumeration, or the like. It is the data equivalent: browsing websites, pulling down PDFs, learning the internal structure of the target organization. A few years ago, I never would've believed that people went to this level of effort to target an organization, but after witnessing it happen, I can say with confidence that it does. The problem with activity in this phase is that it is often indistinguishable from normal activity. There are precious few cases where one can collect information here and find associated behavior in the delivery phase matching an adversary's behavioral profile with high confidence and a low false positive rate. These cases are truly gems; when they can be identified, they link what are often two normal-looking events in a way that greatly enhances detection. The weaponization phase might or might not happen after reconnaissance; it is placed here merely for convenience. This is the one phase that the victim doesn't see happen, but can very much detect. Weaponization is the act of placing malicious payload into a delivery vehicle. It's the difference in how a Soviet warhead is wired to the detonator versus how a U.S. warhead is wired in. For us, it is the technique used to obfuscate shellcode, the way an executable is packed into a trojaned document, etc. Detection of this is not always possible, nor is it always predictable, but when it can be done, it is a highly effective technique. Only by reverse engineering of delivered payloads is an understanding of an adversary's weaponization achieved. This is distinctly separate and often persistent across the subsequent stages.

Delivery

Delivery is rather straightforward. Whether it is an HTTP request containing SQL injection code or an e-mail with a hyperlink to a compromised website, this is the critical phase where the payload is delivered to its target. I heard a term just the other day that I really like: "warheads on foreheads" (courtesy of the U.S. Army).

Exploitation

The compromise phase will possibly have elements of a software vulnerability, a human vulnerability known as "social engineering," or a hardware vulnerability. Although the latter are quite rare by comparison, I include hardware vulnerabilities for the sake of completeness.

Security Intelligence Using the Kill Chain Successfully

The compromise of the target might itself be multi-phase, or more straightforward. As a result, we sometimes have the tendency to pull apart this phase into separate sub-phases, or peel out "Compromise" and "Exploit" as wholly separate. For simplicity's sake, we'll keep this as a single phase. A single-phase exploit results in the compromised host behaving according to the attacker's wishes directly as a result of the successful execution of the delivered payload. For example, if an attacker coaxes a user into running an EXE attachment to an e-mail, which contained the desired backdoor code. A multi-phase exploit typically will involve delivery of shellcode whose sole function is to pull down and execute more capable code upon execution. Shellcode often needs to be portable for a variety of reasons, necessitating such an approach. We have seen other cases where, possibly through sheer laziness, adversaries end up delivering exploits whose downloaders download other downloaders before finally installing the desired code. As you can imagine, the more phases involved, the lower an adversary's probability for success.

This is the pivotal phase of the attack. If this phase completes successfully, what we as security analysts have classically called "incident response" is initiated: Code is present on a machine that should not be there. However, as will be discussed later, the notion of "incident response" is so different in intelligence-driven

response (and the classic model so inapplicable) that we have started to move away from using the term altogether. The better term for security intelligence is "compromise response," because it removes ambiguity from the term "incident."

C2 – Maintain Presence

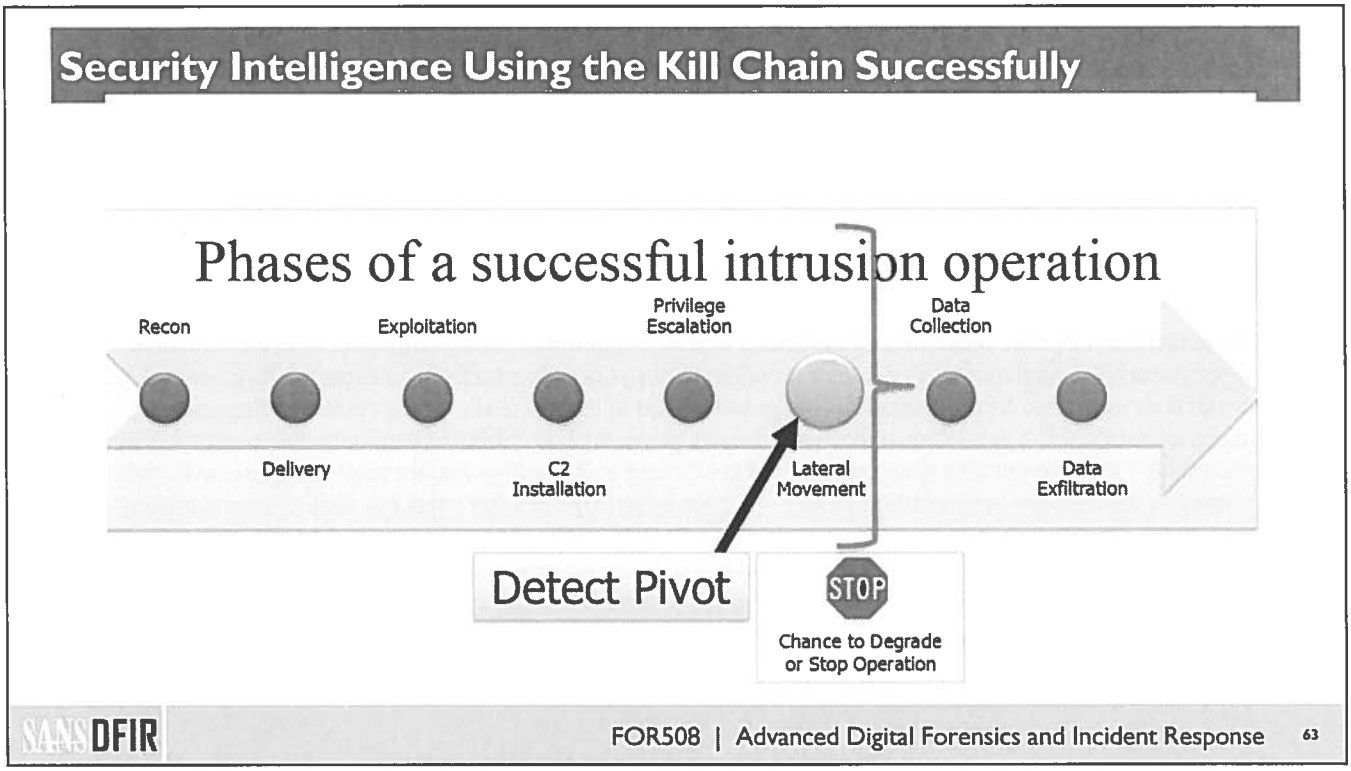
The command-and-control phase of the attack represents the period after which adversaries leverage the exploit of a system. A compromise does not necessarily mean C2, just as C2 doesn't necessarily mean exfiltration. In fact, we will discuss how this can be exploited in CND, but recognize that successful communications back to the adversary *often* must be made before any potential for impact to data can be realized. This can be prevented intentionally by identifying C2 in unsuccessful past attacks by the same adversary resulting in network mitigations, or fortuitously when adversaries drop malware that is somehow incompatible with your network infrastructure, to give but two examples.

In addition to the phone call going through, someone has to be present at the other end to receive it. Your adversaries take time off, too... but not all of them. In fact, a few groups have been observed to be so responsive that it suggests a mature organization with shifts and procedures behind the attack more refined than that of many incident response organizations.

We will also lump lateral movement with compromised credentials, file system enumeration, and additional tool dropping by adversaries broadly into this phase of the attack. Although an argument can be made that situational awareness of the compromised environment is technically "exfiltration," the intention of the next phase is somewhat different.

Exfiltration

The exfiltration phase is conceptually very simple: This is when the data, which has been the ultimate target all along, is taken. Previously, I mentioned that gathering information about the environment of the compromised machine doesn't fall into the exfiltration phase. The reason for this is that such data is being gathered to serve but one purpose, either immediately or longer term to facilitate collection and theft of the target information: the source code for the new O/S, the new widget that cost billions to develop, and access to the credit cards or PII.



Security Intelligence Using the Kill Chain Successfully

Source with permission from Mike Cloppert originally published on the SANS Computer Forensics Blog: http://digital-forensics.sans.org/blog/2010/06/21/security-intelligence-knowing-enemy/

The "persistence" in APT intrusions is manifested in two ways: maintaining a presence on your network, as well as repeatedly attempting to gain entry to areas where presence is not established. The repeatability of these activities inevitably involves attributes that are consistent, because resource constraints typically prevent adversaries from acting differently every time they set foot in your environment. With a way to model intrusions and align these common attributes, network defenders can take advantage of persistence to profile their adversaries, informing strategic response, analysis efforts, and resource investment.

A single intrusion, as we have already discussed, can be modeled as seven phases. Within each of these phases of an intrusion is a highly dimensional set of indicators—computer scientists would call them "attributes"—that together uniquely define that intrusion. For example, a C2 callback domain is an indicator attribute; talktome.bad.com is the corresponding value of the indicator. The targeting used (reconnaissance), the way in which the malicious payload is obscured (weaponization), the path the payload takes (delivery), the way the payload is invoked (exploit), where the backdoor is hidden on the system (installation), the protocol used to call back to the adversary (C2), and habits of the adversary once control is established (actions on intent) are all categorical examples of these indicators. It is up to the analyst to discover the significant or uniquely identifying indicators in an intrusion. In some cases, there are common indicators—for example, the last-hop e-mail relay used to deliver a message will be significant in most like intrusions, excluding webmail. In others, the attributes can be unique and surprising—a piece of metadata, a string in the binary of a backdoor, and a predictably malformed HTTP request to check for connectivity.

Be aware that adversaries shift tactics over time. A campaign is not static, nor are the key indicators or their corresponding values. We've seen adversaries use the same delivery and C2 infrastructure for years, whereas others will shift from consistent infrastructure in the Delivery and C2 phases, to highly variable infrastructure

in the delivery phase but consistent targeting and weaponization techniques. Some adversaries will have consistent key indicators, such as tool artifacts in the Delivery and Weaponization phases, but the specific indicator values might change over time. Without constant and complete analysis of sophisticated intrusions, knowledge of campaigns becomes stale and ineffective at predicting future intrusions.

Gathering Intel through Kill Chain Completion

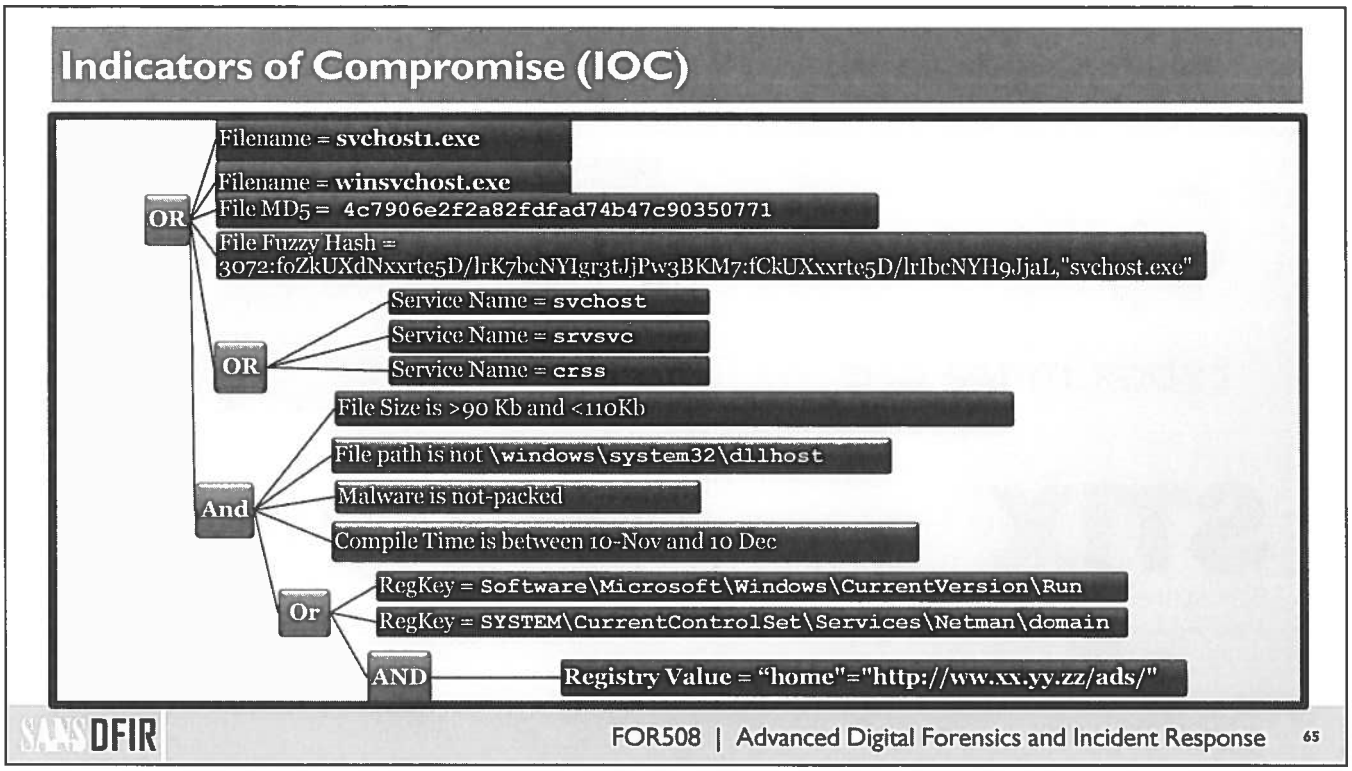
To have the data set necessary to link intrusions and identify key indicators, analysts must understand all phases of every sophisticated intrusion. Initial detection of an intrusion might occur at any point across the kill chain. Even if the attack is unsuccessful, detection is just the first step.

Classic incident response methodology assumes a system compromise. In this situation, where a detection happens after the installation and/or execution of malicious code, adversaries have successfully executed many steps in their intrusion. As the intrusion progresses forward in the kill chain, so the corresponding analysis progresses backward. Analysts must reconstruct every prior stage, necessitating not only the proper tools and infrastructure to do so but also deep network and host forensic skills. Less mature response teams will often get stuck in the delivery to installation phases. Without knowledge of what happened earlier in an intrusion, network defenders will be unable to define campaigns at these earlier phases, and response to intrusions will continue to happen post-compromise because this is where the detections and mitigations are. When walls are hit in analysis that prevents reconstruction of the entire chain, these barriers represent areas for improvement in instrumentation or analytical techniques. Where tools do not already exist for accurate and timely reconstruction, development opportunities exist. Here is but one area where having developers on staff to support incident responders is critical to the success of the organization.

As response organizations mature and are able to more fully build profiles of intrusion campaigns against them, they become more successful at detection prior to compromise. However, just as a post-compromise response involves a significant amount of analysis, the unsuccessful intrusion attempts matching APT campaign characteristics also require investigation. The phases executed successfully by the adversary must still be reconstructed, and the phases that were not must be synthesized to the best ability of the responders. This aspect is critical to identifying any TTP change that may have resulted from a successful compromise. Perhaps the most attention-grabbing example is identification of 0-day exploits used by an APT actor at the Delivery phase, before the exploit is invoked.

Synthesis clearly demonstrates the criticality of malware reverse engineering skills. It is likely that the backdoor that would have been dropped, even if it is of a known family, using a known C2 protocol, also contains new indicators further defining the infrastructure at the disposal of adversaries. Examples include indicators such as C2 callback IP addresses and fully qualified domain names. Perhaps minor changes in the malicious code would produce new unique hashes, or a minor version difference results in a slightly different installation filename that could be unique. Although anti-virus is typically a bad example of detection in the context of APT intrusions, there are times when it can be of value for older variants of code. For instance, how many reading this analyze e-mails that are detected by their perimeter anti-virus system? If the detection is for a particular backdoor uniquely linked to an APT campaign, the e-mail could contain valuable indicators about the adversary's delivery or C2 infrastructure that might be re-used later in an intrusion that your anti-virus system does not detect.

Detecting campaigns enables resilient detection and prevention mechanisms across an intrusion, and engages CND responders earlier in the kill chain, reducing the number of successful intrusions. It should be obvious, but bears repeating that a lack of specific indicators from a single intrusion prevents identification of key indicators from sequential intrusions. A lack of key indicators results in an inability to define adversaries, and an inability to define adversaries leaves network defenders responding post-compromise to every intrusion. In short, inability to reconstruct intrusions should be considered an organizational failure of CND, and intelligence-based detections prior to system compromise a success. Defining campaigns, as demonstrated here, is one effective way to facilitate success.



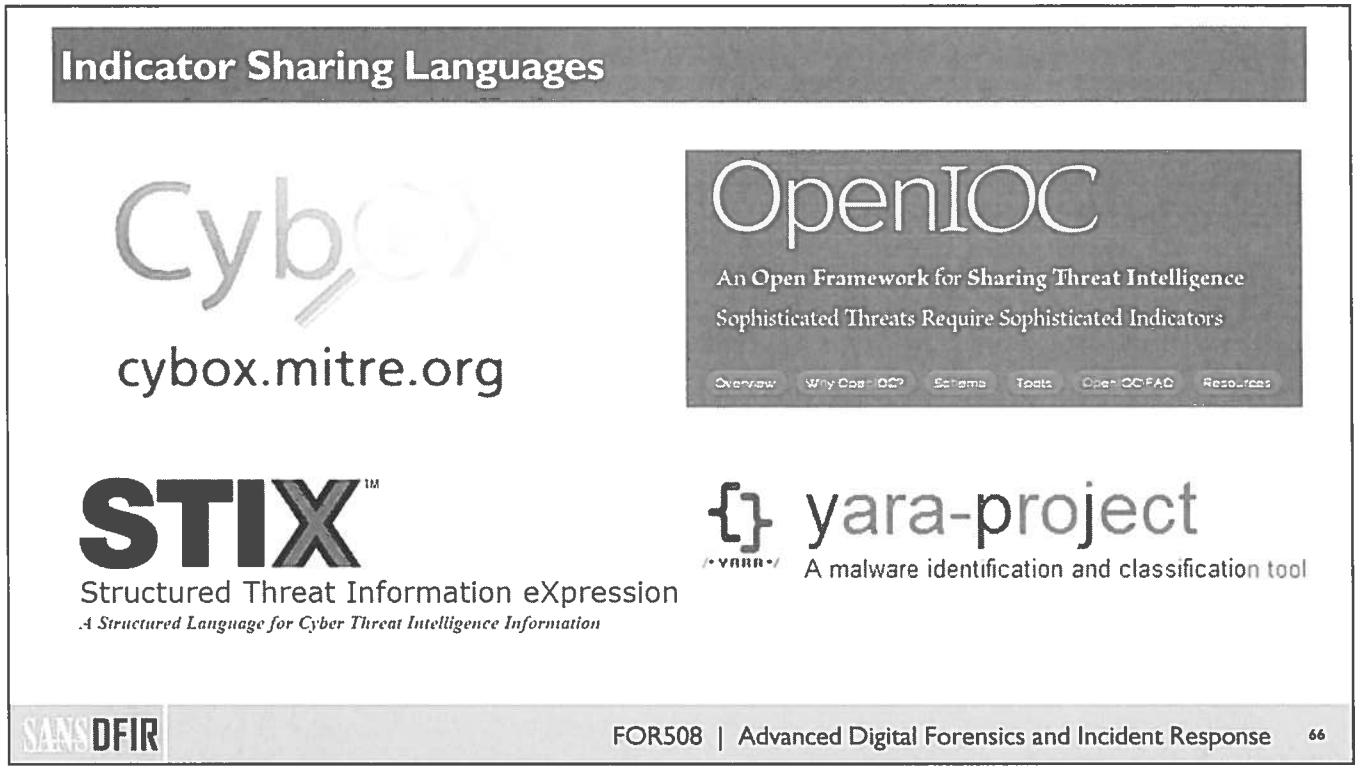
Indicators of Compromise (IOC)

What is an indicator of compromise? It is a very powerful technique to identify malware components on a compromised host. Generally, it is a combination of boolean expressions that can be used to identify general characteristics of malware. If these characteristics are found, then you have a hit.

There are two types of indicators: host-based (as in the slide) and network-based (similar to snort signatures plus additional data).

It is equivalent to narrowing down a suspect through identifying specifics about the suspect: male, 6 ft. 2 in. ~230 lbs., shaved head, blue eyes, and driving a red or orange Nissan Xterra.

Your indicators of compromise usually are created by reversing malware and through application footprinting. Some professional groups that are responding to incidents have massive IOC lists that range in the thousands of indicators collected from previous intrusions that they have collected. Malware exhibits many of these signatures and it is up to your team to use these, once detailed, to identify which additional system that might have also been compromised. IOCs are the difference between having to analyze each system in-depth and by analyzing a few in-depth and using that data to identify similar machines on your network that have the same characteristics.



Indicator Sharing Languages
There are many projects that have started to come about as a result of the idea of threat intelligence becoming more common. In the slide is a listing of some of the more popular projects that exist today.

Cybox = Cyber Observable eXpression: http://cybox.mitre.org

CybOX is a standardized schema for the specification, capture, characterization, and communication of events or stateful properties that are observable in the operational domain. A wide variety of high-level cyber security use cases rely on such information including event management/logging, malware characterization, intrusion detection, incident response/management, attack pattern characterization, etc. CybOX provides a common mechanism (structure and content) for addressing cyber observables across and among this full range of use cases improving consistency, efficiency, interoperability, and overall situational awareness.

STIX = Structured Threat Information eXpression: http://stix.mitre.org/

STIX™ is a collaborative community-driven effort to define and develop a standardized language to represent structured cyber threat information. The STIX Language intends to convey the full range of potential cyber threat information and strives to be fully expressive, flexible, extensible, automatable, and as human-readable as possible. All interested parties are welcome to participate in evolving STIX as part of its open, collaborative community.

CRITS information: http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/collaborative-research-into-threats-crits

Yara-Project: http://plusvic.github.io/yara/

YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples. With YARA, you can create descriptions of malware families (or whatever you want to describe) basedon textual or binary patterns. Each description (known as rule) consists of a set of strings and a boolean expression, which determine its logic.

OpenIOC -> http://openioc.org

OpenIOC was originally designed to enable MANDIANT's products to codify intelligence in order to rapidly search for potential security breaches. Now, in response to requests from across the user community, MANDIANT has standardized and open sourced the OpenIOC schema and is releasing tools and utilities to allow communication of threat information at machine speed.

Conversion of IOC to STIX can import YARA signatures: https://github.com/STIXProject/openioc-to-stix.

OpenIOC

- http://openioc.org/
- **IOC Editor**
  - Allows users to create, edit, and compare Indicators of Compromise in XML format
- **Redline**
  - Allows users to search for Indicators of Compromise on a single host, allowing for everything from testing new IOCs to finding evil on hosts during the course of an investigation

OpenIOC

IOC Editor is a free editor for Indicators of Compromise (IOCs). IOCs are XML documents that help incident responders capture diverse information about threats including attributes of malicious files, characteristics of registry changes, artifacts in memory, and so on. [1] IOCe provides an interface into managing data within these IOCs including:

- Manipulating the logical structures that define the IOC
- Applying meta-information to IOCs including detailed descriptions or arbitrary labels
- Converting IOCs into XPath filters
- Managing lists of "Terms" that are used within IOCs

Redline is a free tool for collecting host system data and reporting the presence of IOCs. IOCs are open-standard XML documents that help incident responders capture diverse information about threats. [2] Redline supports:

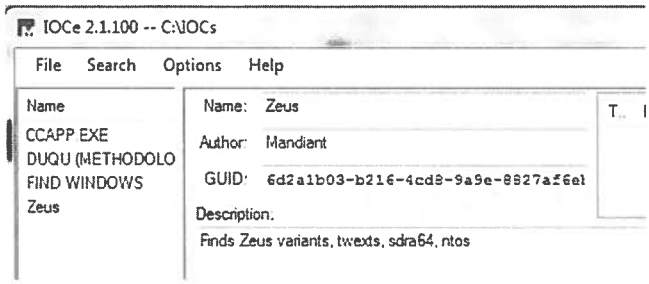
- Collection of full data, sufficient for general IOC matching requirements
- Using a portable storage device allowing the collection from multiple hosts
- IOC hit reporting in simple text, full HTML, and full MS Word XML formats
- Reports generated for specific hosts or all hosts

[1] http://www.mandiant.com/resources/download/ioc-editor/

[2] http://www.mandiant.com/resources/download/ioc-finder/

Building IOCs: IOC Editor

- **IOC Editor**
  - Allows users to create, edit, and compare Indicators of Compromise in XML format



- **IOC Bucket**
  - Virus Total Stub Generator: iocbucket.com/vtstubgenerator

Building IOCs: IOC Editor

IOC Editor is a free GUI editor for IOC. By taking the output from the malware/registry analysis tools, it is possible to use IOCe to build a "signature" that can be used to find other compromised systems.

Alternatively, you can use IOC Bucket to take an uploaded sample that exists in Virus Total to create an IOC Stub for the malware from the details that virus total has on it. To use it, simply drop in an address for a file on Virus Total and select **Generate**. This is a stub of an IOC intended to be used as a base to make a more robust IOC. The IOC stub is generated from data provided by Virus Total. Not all files have the same data available. [1]

[1] https://www.iocbucket.com/vtstubgenerator

Exercise 1.3

Cyber Threat Intelligence – Indicator Creation and Examination

This page intentionally left blank.

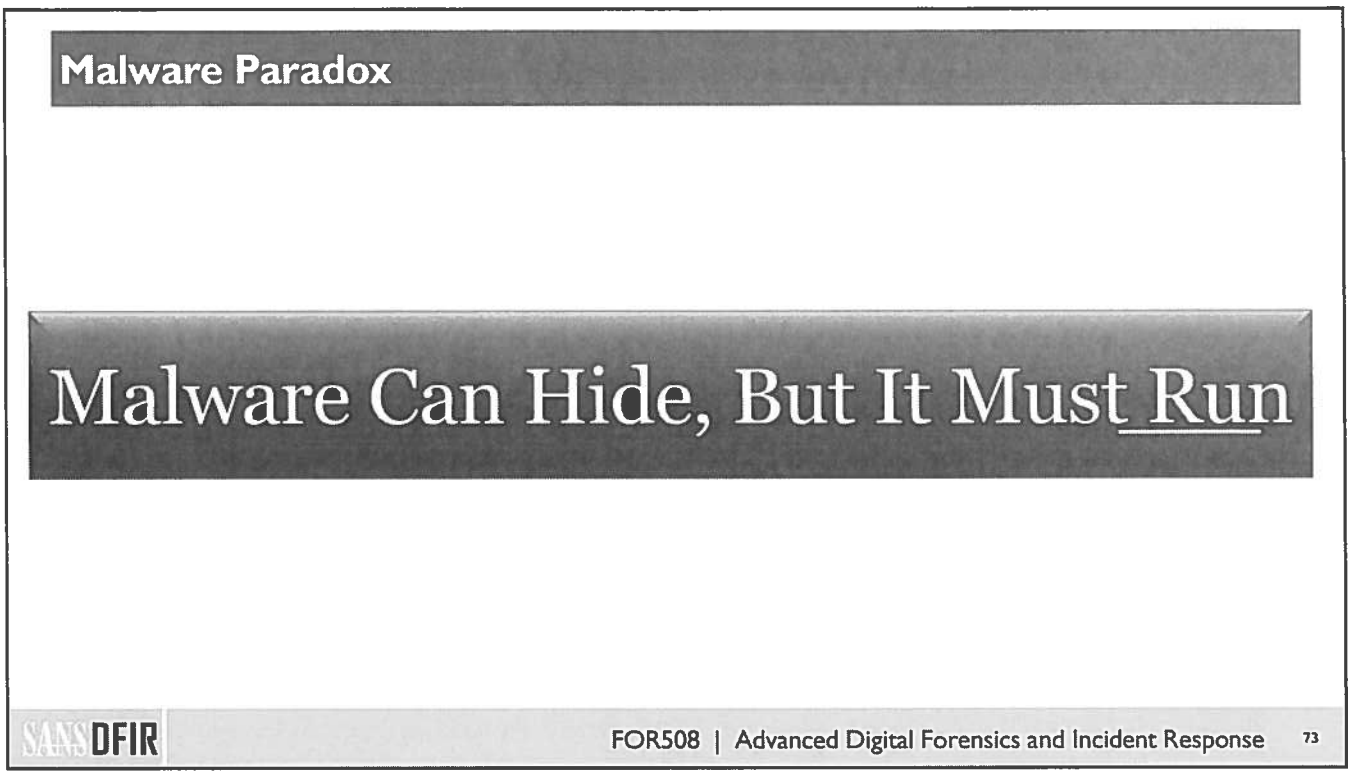
This page intentionally left blank.

What are we looking for? Malware-ology

"We don't use the word 'intelligence' with software. We regard that as a naive idea. We say that it's 'complex.' Which means that we don't always understand what it's doing."
— Orson Scott Card, *Ender's Shadow*

SANS DFIR
DIGITAL FORENSICS & INCIDENT RESPONSE

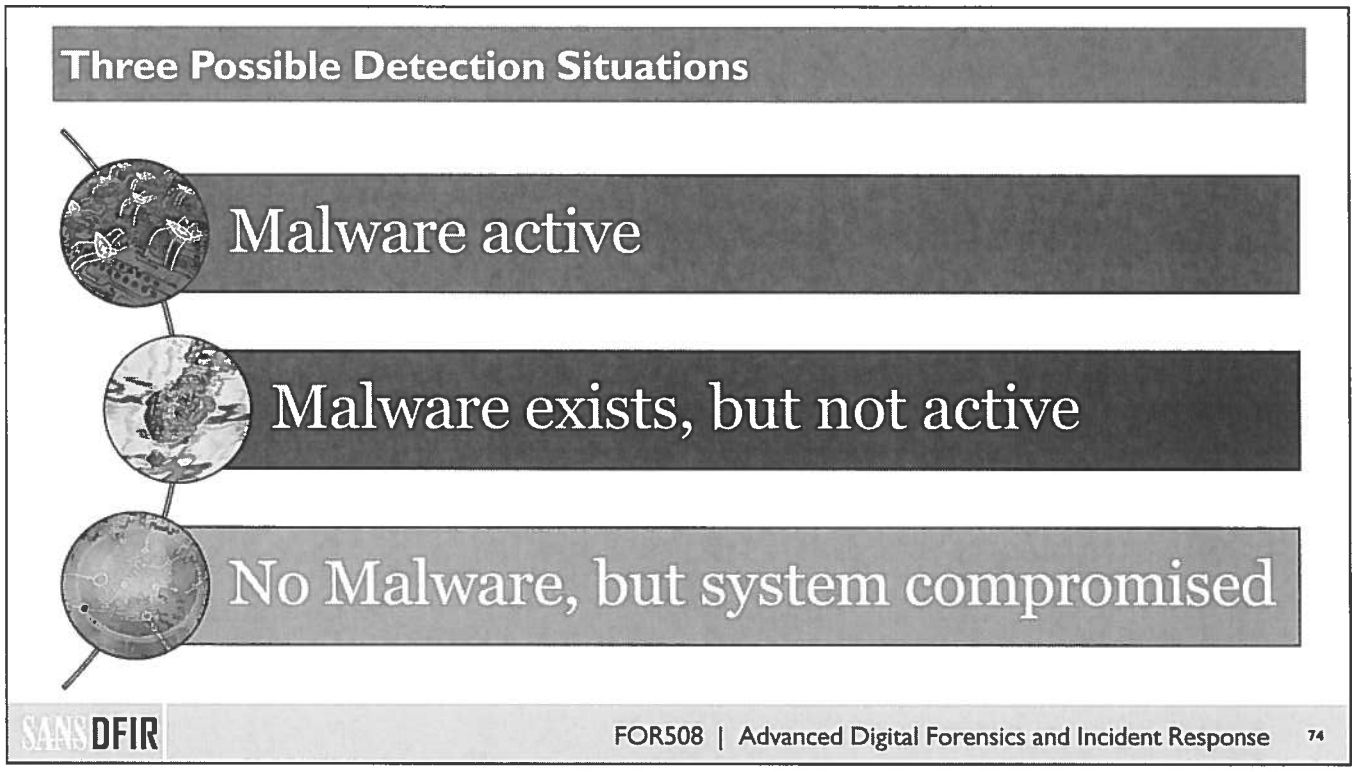
This page intentionally left blank.



Malware Paradox

Several years ago, Jesse Kornblum stated, "Malware Can Hide, But It Must Run," and this became known as the Malware Paradox.[1] The paradox means that malware can exist but sooner or later something must activate it to run. Typically, a method to keep malware "persistent" across multiple reboots on a system is called a "persistence mechanism." It is a key piece of evidence to look for and could possibly help us point, in reverse, back to the malware—more on that shortly.

[1] Exploiting the Rootkit Paradox with Windows Memory Analysis
http://jessekornblum.com/publications/ijde06.html



Three Possible Detection Situations

When hunting for a compromise, it is actually easiest if there is active malware on the system. In most cases, it gives you more places to look. Malware that is not active, but is dormant is harder to detect as we lose the ability to detect the malware in memory. Malware that is dormant could launch through a specific persistence mechanism like a Word add-on or via a scheduled task instead of at boot. Finally, the last situation to consider is the system is compromised and malware doesn't currently exist. In this last situation, it is actually the hardest of the situations. Many adversaries do not need to leave malware on every system they access while trying to pilfer data from your network. Being able to log on to a system using valid credentials and subsequently searching for key data to exfiltrate leaves a very small and difficult-to-detect footprint.

In other situations, more advanced adversary groups are routinely removing their malware from systems in order to limit the possibility it might be captured and threat intelligence created off it. They will employ anti-forensics measures to remove themselves from the systems and an analyst can only hope that they made a mistake and left behind a registry key, a configuration file, or a file fragment that can prove they had accessed this system. Over time, analysts learn that detecting malware is much easier than detecting a compromised host without malware.

Adversary Hiding in Plain Sight

- Common Malware Names
  - **svchost.exe**
  - **iexplore.exe**
  - **iprinp.dll**
  - **winzf32.dll**
- Persistent
  - Process injection
  - Service persistence
- Service Replacement
  - Wireless Zero Configuration service
  - RIP Listener service
  - Background Intelligent Transfer service

Most Common Malware Locations

- `Windows\System32`
- Temp folders
- Windows
- System volume information
- Recycle Bin
- Program files
- Temporary Internet files

Adversary Hiding in Plain Sight

In the study of malware over the past several years, it should be noted that the most popular malware name used to hide as a new service a system would be svchost.exe. Svchost.exe is a process that is found running at least 5-6 times on every system and frequently is hard to pick out a good svchost from a bad one.[1] The following is a list of some of the most common malware names and services that they try and replace most often.

Common Malware Names
- **svchost.exe**
- **iexplore.exe**
- **iprinp.dll**
- **winzf32.dll**

Avoids Detection
- Process injection
- Service persistence

Service Replacement
- Wireless Zero Configuration service
- RIP Listener Service
- Background Intelligent Transfer service

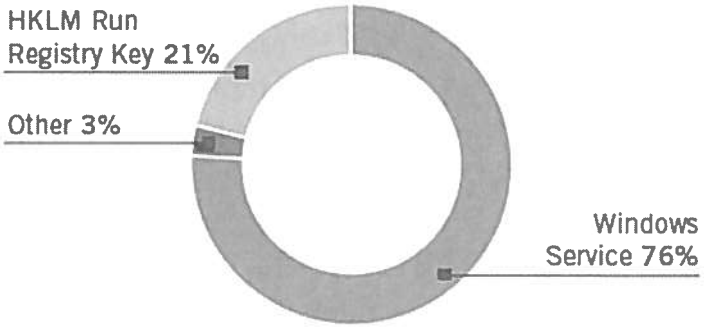
In addition, if you look at the statistics from the types of malware submitted to Virustotal for examination, most of the file types are executable files, libraries, PDFs, ZIPs, and other document types.[2] This is good to note when we first use "sorter" to categorize the data inside of our image. Knowing that our malware is essentially a file type that can be identified using the sorter automated tool will help us during the malware identification process.

[1] "M-Trends" by an author team that includes Rob Lee
[2] https://www.virustotal.com/statistics/
Information on abuse of SVCHOST can be found at: http://www.hexacorn.com/blog/2013/07/04/the-typographical-and-homomorphic-abuse-of-svchost-exe/

Mandiant M-Trends Report

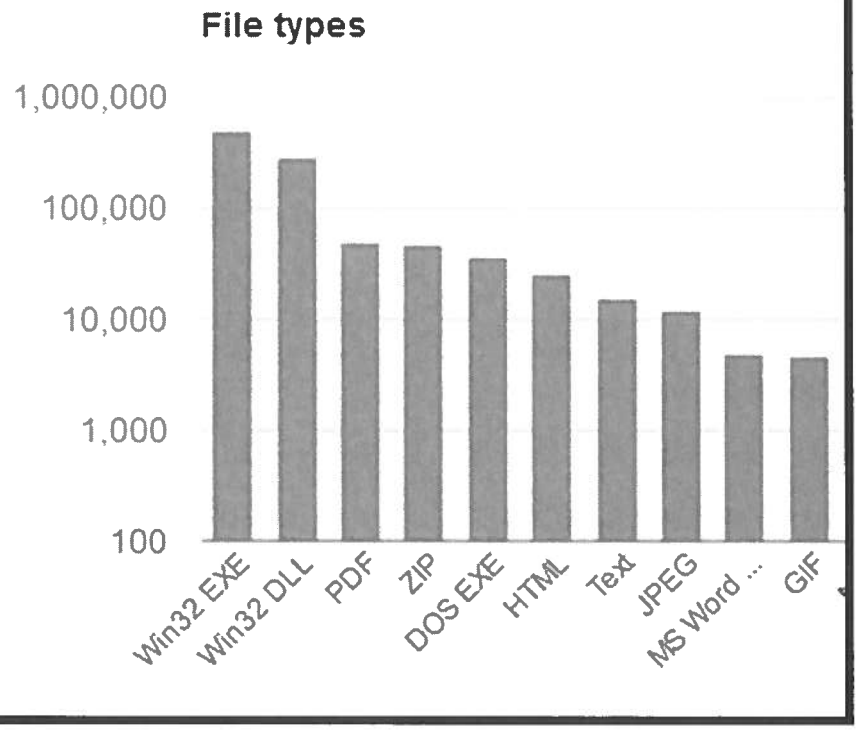
APT MALWARE BACKDOORS

APT: Persistence Backdoors
60% of APT backdoor samples were persistent on the machine



APT: Non-Persistent Backdoors
30% used process injection to avoid detection

VirusTotal Malware Statistics



Malware Evasion Techniques

- Evasion Techniques
  - Windows Services
  - Process Injection
  - File Name/Service Hijacking
  - Alternate Data Streams
  - WebShells/Beacons
  - Frequent Compilation
  - Packing/Armoring
  - Dormant Malware
  - Outbound HTTP beacons
  - Signing Code with Trusted Cert

SANS DFIR FOR508 | Advanced Digital Forensics and Incident Response 77

Malware Evasion Techniques

First, look at how malware is typically found in cases today. Most malware will simply hide in plain sight. No need for fancy rootkits for today's advanced hackers. They know that rootkit technology is often detectable and often creates system instability. As a result, malware will simply try to hide among the thousands of files and directories that exist on a standard windows system.

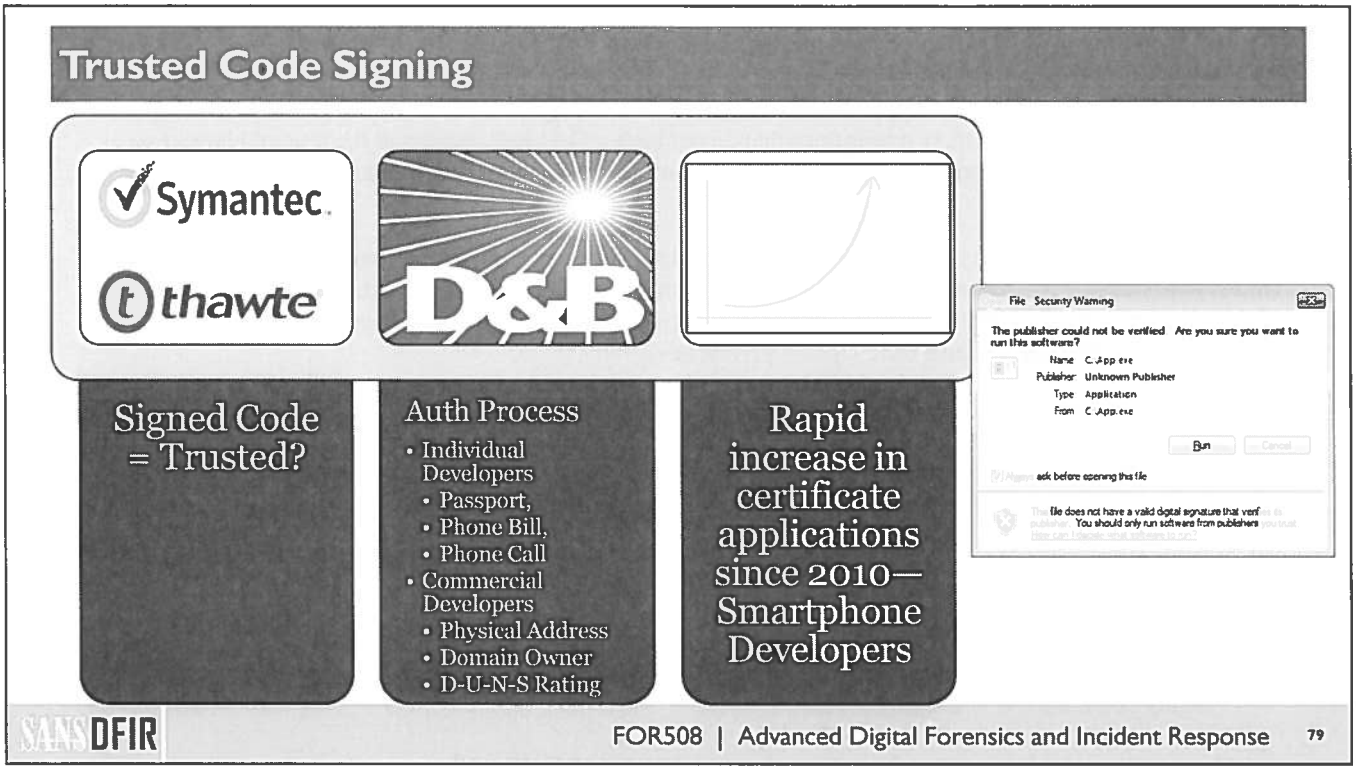
Most analysts who become decent forensicators have no trouble finding where malware might exist after they gain years of experience. It isn't a difficult process to follow what they know, but it is half process and half art. Malware traditionally will probably create a new service, using the scheduler "at" command that makes the malware hide in plain sight among other normal services. However, I doubt the service name will be called "EVIL MALWARE BRU-HAHAHA SERVICE"—more likely it will also look like a normal service. In some cases, but more rare, the malware will replace an existing service such as wireless zero configuration service. The third most popular method that malware will maintain persistence is by registering as an auto-run process in the registry. And finally, process injection is also used, but not as frequently.

Some other traits associated with malware include that malware today is frequently recompiled; use alternate data streams, frequently packed, and armored to prevent A/V detection and anti-reversing. It is also very common to find dormant malware on a machine, either to be activated via a remote scheduler from another machine in your network or that the malware was improperly deleted/wiped when the adversary last used the system.

Let's take a look at how malware is typically found in cases today. Malware often employs tactics that are categorized as "Hiding in Plain Sight." What does this mean? Adversaries know full well that the more hiding techniques employed, the more likely a host-based intrusion detection system or antivirus program will detect them—this includes rootkit techniques.

The more manipulation of the Windows API, the more likely one is to 1.) crash a system and 2.) get detected through protection services. Instead, Hiding in Plain Sight offers a better solution. Many of these techniques involve mimicking known good services, normal Windows processes (or naming their malicious processes, which is something that closely mimics a known good). If it looks like a normal process/service, there is less chance of raising flags or calling attention to itself.

Malware uses tactics such as frequent compilation, packing, and armoring to evade detection via AV or host-based intrusion prevention systems, and might also lie dormant, waiting for a remote signal from a command and control server or another system on the network to activate it.



Trusted Code Signing

Trusted code signing was always intended to increase the security and trustworthiness of programs downloaded from the Internet. In the past, when most programs were bought at PC stores, programs were inherently trusted because they came in original packages that were shrink wrapped. Code signing is intended to produce the same trust in programs purchased and downloaded across the Internet. It is fairly trivial to "self-sign" code. However, trusted code was intended to be a different story completely. With signed and trusted code, one wouldn't have to worry about it being malicious—or at least that was the intent. However, signed malware is a real threat and has been seen multiple times in highly publicized incidents.

How is trusted code signing supposed to work?

A certification authority, such as Verisign or Thawte, would verify and confirm your identity and information and issue a digital certificate to the organization or individual that applied for it. The intent of the certification is to be able to identify the original code author and the issuing authority. Once issued, the developer would use his or her private key to digitally "sign" the code, which is intended to produce attribution.

What is the verification process to obtain a trusted code signing certificate from a Certificate Authority (CA)?

Authentication of organizations and/or individuals applying for a code signing cert varies widely. For some companies, all you would need to provide is a copy of your passport or phone bill and then the company will call the number and make sure you are there.[1] In other instances, they ensure you are not on a restricted organization list from the government, has a business license, exists at physical address, obtained rights to the domain name, and verified by a third-party phone number.

To obtain a commercial software certification, most organizations must also apply for a *Dun and Bradstreet Rating*. According to MSDN, "Applicants must achieve a level of financial standing as indicated by a D-U-N-S number (which indicates a company's financial stability) and any additional information provided by this service. This rating identifies the applicant as a corporation that is still in business. (Other financial rating services are being investigated.) Corporations that do not have a D-U-N-S number at the time of application (usually because of recent incorporation) can apply for one and expect a response in less than two weeks."[2]

The Dun & Bradstreet rating is intended to draw a very hard line to cross between commercial and private (individual) developers. Although not impossible, to create a front company to meet this criteria is length and tedious though not impossible—especially for nation-state adversaries. To this end, many malware developers will opt to steal commercial-level code signing private keys.

The hope is that one would be able to trust that the code has not been tampered with after signing it in addition to aiding in identifying the origination of the code. Hardly foolproof, there have been multiple incidents that have resulted in malware being signed with stolen certificate private keys. Stuxnet is just one example.

A few additional examples include:

- Code-signing certificate private keys stolen from Adobe were used to sign malicious software.[3]
- An attack on the browser company Opera allowed the intruder to access a the private key of a code-signing cert and use it to sign malware.[4]

Major operating systems will require code to be signed by a trusted developer in order to allow them to execute without user interaction. On many servers, code has to be signed in order to run. Malware that is signed has an easier time spreading and hiding on networks, although this might also come with a cost. If the malware is ever discovered, the code-signing certificate could be revoked and added to a Certificate Revocation List (CRL). The CRL list should be checked and any system with that software installed would no longer function. However, although this process has produced faulty results and has resulted in major patches to Microsoft operating systems, to update the CRL list as revoked code will continue to function unless you have updated your system with the latest Microsoft patches.[5] After this update, Microsoft is now able to help flag untrusted software more quickly. In an attempt to have the advantages of signed code without the risk of malware that can be flagged, some adversaries have resorted to more advanced techniques like DLL Side-Loading, which we will cover later in the class.

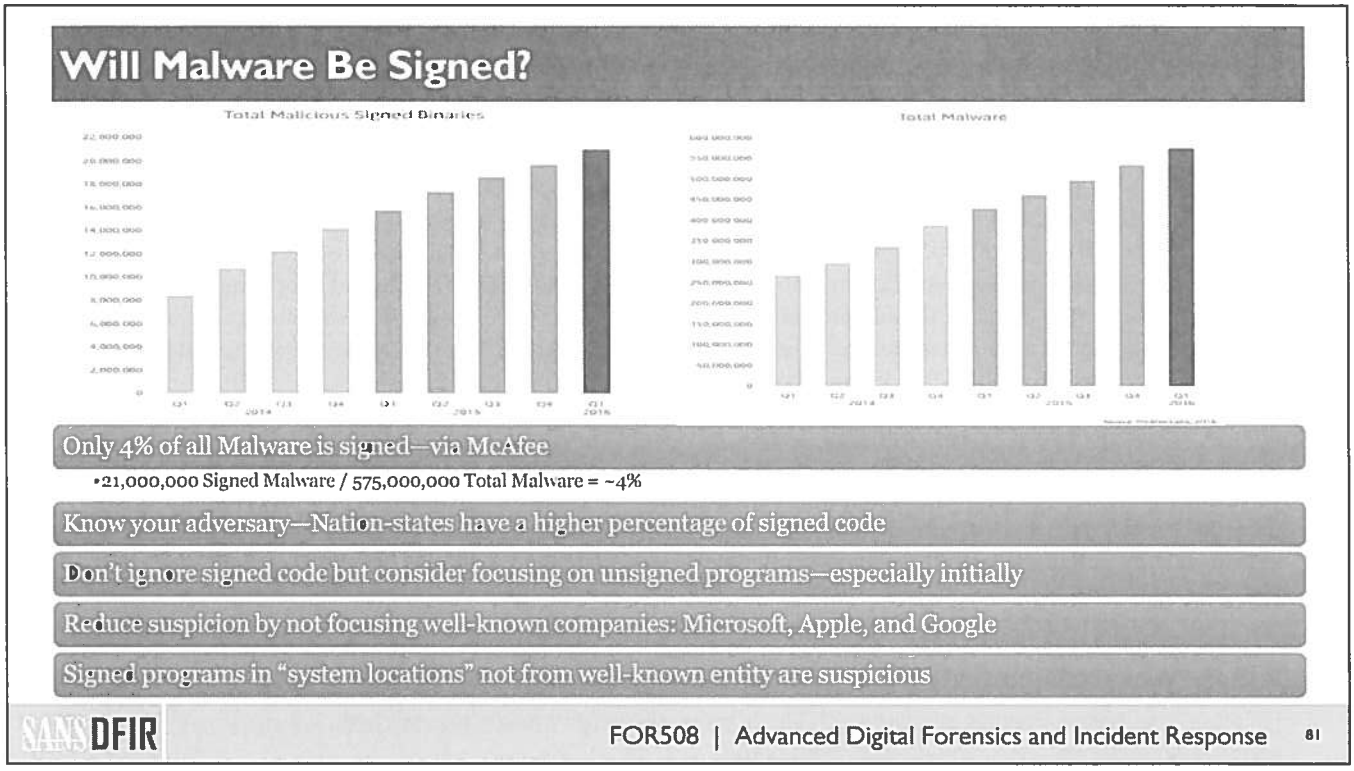
[1] https://www.thawte.com/assets/documents/guides/pdf/enroll_codesign_eng.pdf

[2] https://msdn.microsoft.com/en-us/library/ie/ms537361%28v=vs.85%29.aspx

[3] http://blogs.adobe.com/security/2012/09/inappropriate-use-of-adobe-code-signing-certificate.html

[4] https://threatpost.com/opera-code-signing-certificate-stolen-malware-signed-and-distributed/101107/

[5] http://support.microsoft.com/kb/2677070



Will Malware Be Signed?

It depends. There are some benefits and drawbacks to signing code. According to McAfee, only 4% of all malware is signed. If you look at malware discovered just this past quarter, only 3% of the malware is signed. This is the same percentage of signed malware over total malware that has been seen for the past several years. It is still predicted to increase; however, the fact remains, the total number of signed malware compared to unsigned malware is small. We can use this to our advantage when we are trying to hunt for malware across our enterprise. [1]

If you are like me, you likely would like to know why that is. To be honest, I am surprised that more malware isn't signed. However, some of this could be explained in the fact that there are both benefits and drawbacks to signing code.

Benefits to Signing Malware

Malware that is signed is trusted by the operating system and can stay hidden for a longer period of time without arousing suspicion. Typically, we see espionage malware is signed frequently, such as Flame malware that is intended to stay hidden for as long as possible. It is also advantageous if the developer is not planning on using that malware again and is willing to risk it being revoked if discovered.

Drawbacks to Signing Malware

Rapid development and release of malware will be inhibited. In many cases, malware authors need to rapidly develop alternatives to their code to avoid anti-virus and host-based intrusion detection systems. A malware author would need a plethora code signing certs to avoid burning an entire family of malware active across an enterprise if discovered. If a code signing cert is used and later revoked, all the current locations of the malware will be flagged and responders should be able to easily to locate them. As a result, the code signed malware

would become a liability. For malware that is rapidly released, you might have a limited number of certs available and malware developers might be supporting operating across multiple targets. If just one malware cert is revoked, it would burn all of the malware currently installed across those targets.

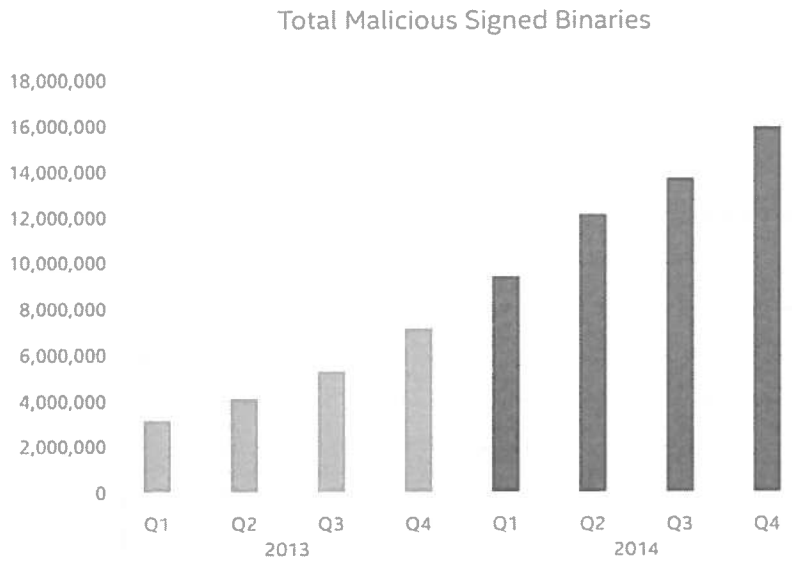
What Is the Likely Percentage of Malware that Is Signed?

Less than 20% of all the malware is signed. According to many sites, most malware is still unsigned but that trend is changing quickly. Due to the drawbacks listed previously, not all malware is intended to be signed. Certificate authorities seem to have relaxed their verification standards as developers are rushing to apply for certificates in order to produce smartphones apps. Specifically, the number of applications for certs have flooded the market since 2010. Signed malware has seen a direct correlation. According to McAfee research, Malware signed with legitimate certificates has soared since 2010 when roughly 1.3% of a sample set was found signed that way. This roughly doubled to 2.9% in 2011, and then rose to 6.6% in 2012. [2]

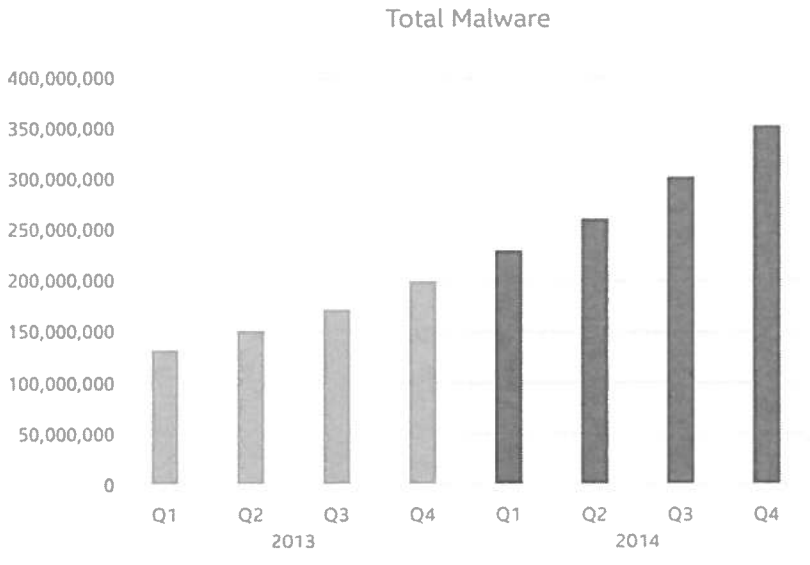
The bottom line: There is still a chance that malware is signed. As a result, both signed malware and unsigned code should still be examined. You might first try to eliminate all unsigned code from your tool output. But you also might include trusted code from unrecognized developers. You can likely eliminate a large portion of signed code by initially ignoring signed code from well-known companies such as Microsoft, Apple, and Google.

[1] http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-may-2016.pdf

[2] http://www.networkworld.com/article/2170526/network-security/mcafee-research-shows-sharp-rise-in-malware-signed-with-legitimate-digital-certific.html



Source: McAfee Labs, 2015.



Source: McAfee Labs, 2015.

Only 4.5% of all Malware is signed - via McAfee

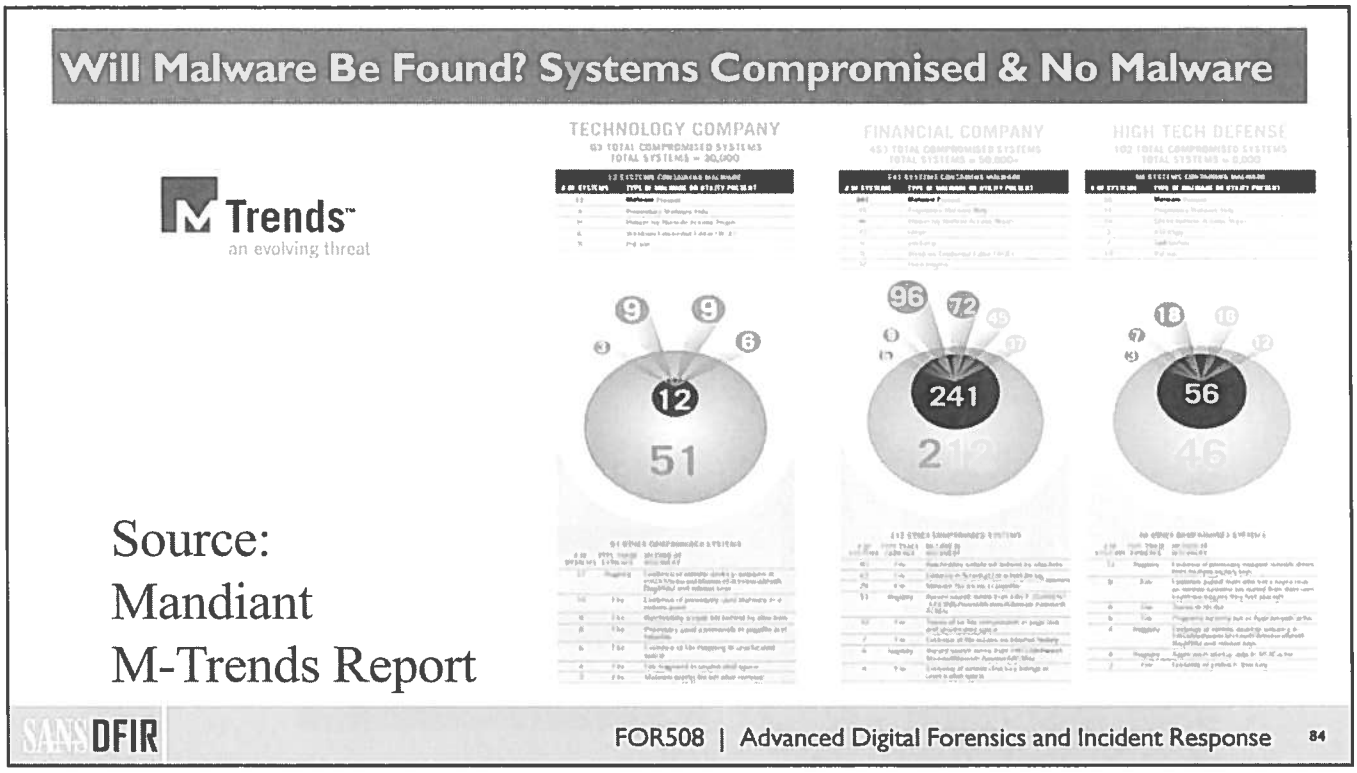
- 16,000,000 Signed Malware / 350,000,000 Total Malware = ~4.5%

Know your adversary – Nations states have a higher percentage of signed code

Don't ignore signed code but consider focusing on unsigned programs – especially initially

Reduce suspicion by not focusing well-known companies: Microsoft, Apple, & Google

Signed Programs in "system locations" not from well-known entity are suspicious



Will Malware Be Found? Systems Compromised & No Malware

According to Mandiant and other sources, malware might NOT be present on every system. The 2012 *M-Trends* report details this fact very well in discussing that malware traces might be left on a machine, but the existence of malware on each system is not guaranteed. Some of the examples that Mandiant uses are listed in the slide. What is useful here is how it determined how the systems were compromised in the end. It had to use traditional forensic techniques to uncover the existence of the compromise.[1]

Some of the methods that are used to uncover compromised machines are found detailed in both FOR408 and FOR508 combined.

So while in this next section, we will detail how to find whether malware still exists on a machine. However, having said that, just because you do not find malware on the system, it does not mean that the system was not compromised.

[1] *Mandiant M-Trends:* www.mandiant.com

TECHNOLOGY COMPANY

63 TOTAL COMPROMISED SYSTEMS
TOTAL SYSTEMS = 30,000

12 SYSTEMS CONTAINING MALWARE	
# OF SYSTEMS	TYPE OF MALWARE OR UTILITY PRESENT
12	Malware Present
3	Proprietary Malware Only
9	Poison Ivy Remote Access Trojan
6	Windows Credential Editor (WCE)
9	PsExec

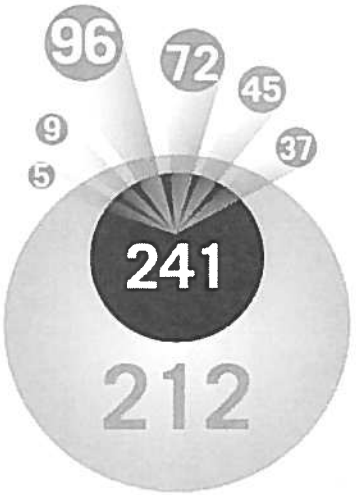
51 OTHER COMPROMISED SYSTEMS

# OF SYSTEMS	TYPE TRACE EVIDENCE	METHOD OF DISCOVERY
12	Registry	Evidence of remote desktop sessions in HKCU\Software\Microsoft\Windows\Shell\BagMRU and related keys
10	File	Evidence of previously-used malware in a restore point
9	File	BatchUtility scripts left behind by attackers
8	File	Previously used commands in pagefile and hiberfile
6	File	Evidence of file mapping in unallocated space
4	File	File fragment in unallocated space
2	File	Malware config file left after removal

FINANCIAL COMPANY

453 TOTAL COMPROMISED SYSTEMS
TOTAL SYSTEMS = 50,000+

241 SYSTEMS CONTAINING MALWARE	
# OF SYSTEMS	TYPE OF MALWARE OR UTILITY PRESENT
241	Malware Present
45	Proprietary Malware Only
96	Poison Ivy Remote Access Trojan
72	Htran
5	pwdump
9	Windows Credential Editor (WCE)
37	Hookmsgina



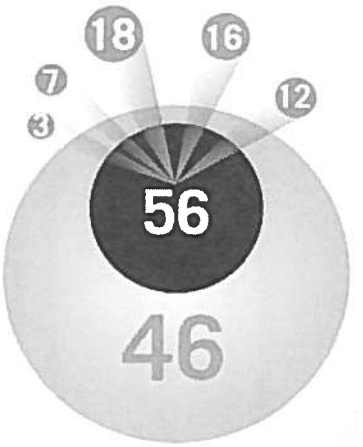
212 OTHER COMPROMISED SYSTEMS

# OF SYSTEMS	TYPE TRACE EVIDENCE	METHOD OF DISCOVERY
80	File	Batch utility scripts left behind by attackers
63	File	Evidence in SchedLgU.txt schedule log
29	File	Malware file traces in pagefile
13	Registry	Recent search terms from HKEY_CURRENT_USER\Software\Microsoft\Search Assistant\ACMru
10	File	Traces of rar file compression in page files and unallocated space
7	File	Evidence of file access via internet history
6	Registry	Recent search terms from HKCU\Software\Microsoft\Search Assistant\ACMru
4	File	Evidence of remote directory listings in unallocated space

HIGH TECH DEFENSE

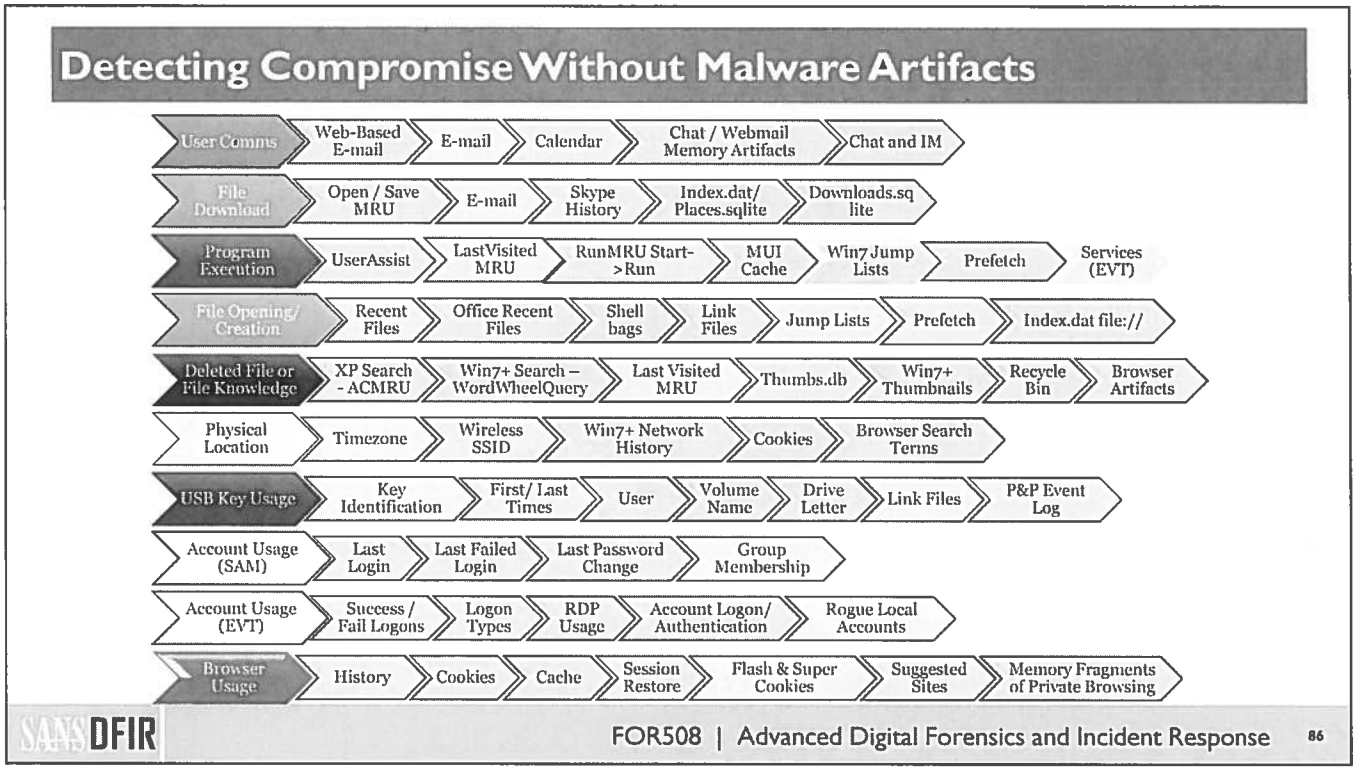
102 TOTAL COMPROMISED SYSTEMS
TOTAL SYSTEMS = 6,000

56 SYSTEMS CONTAINING MALWARE	
# OF SYSTEMS	TYPE OF MALWARE OR UTILITY PRESENT
56	Malware Present
16	Proprietary Malware Only
18	GrOst Remote Access Trojan
3	ASPXSpy
7	GetHashes
12	PsExec



46 OTHER COMPROMISED SYSTEMS

# OF SYSTEMS	TYPE TRACE EVIDENCE	METHOD OF DISCOVERY
13	Registry	Evidence of previously mapped network drives from multiple registry keys
9	File	Evidence pulled from attacker's keystrokes on remote systems (as culled from their own keystroke loggers they had placed)
8	File	Traces in hh.dat
5	File	Programs recently run in AppCompatCache
4	Registry	Evidence of remote desktop sessions in HKCU\Software\Microsoft\Windows\Shell\BagMRU and related keys
4	Registry	Application startup data in MUICache
3	File	Contents of prefetch directory



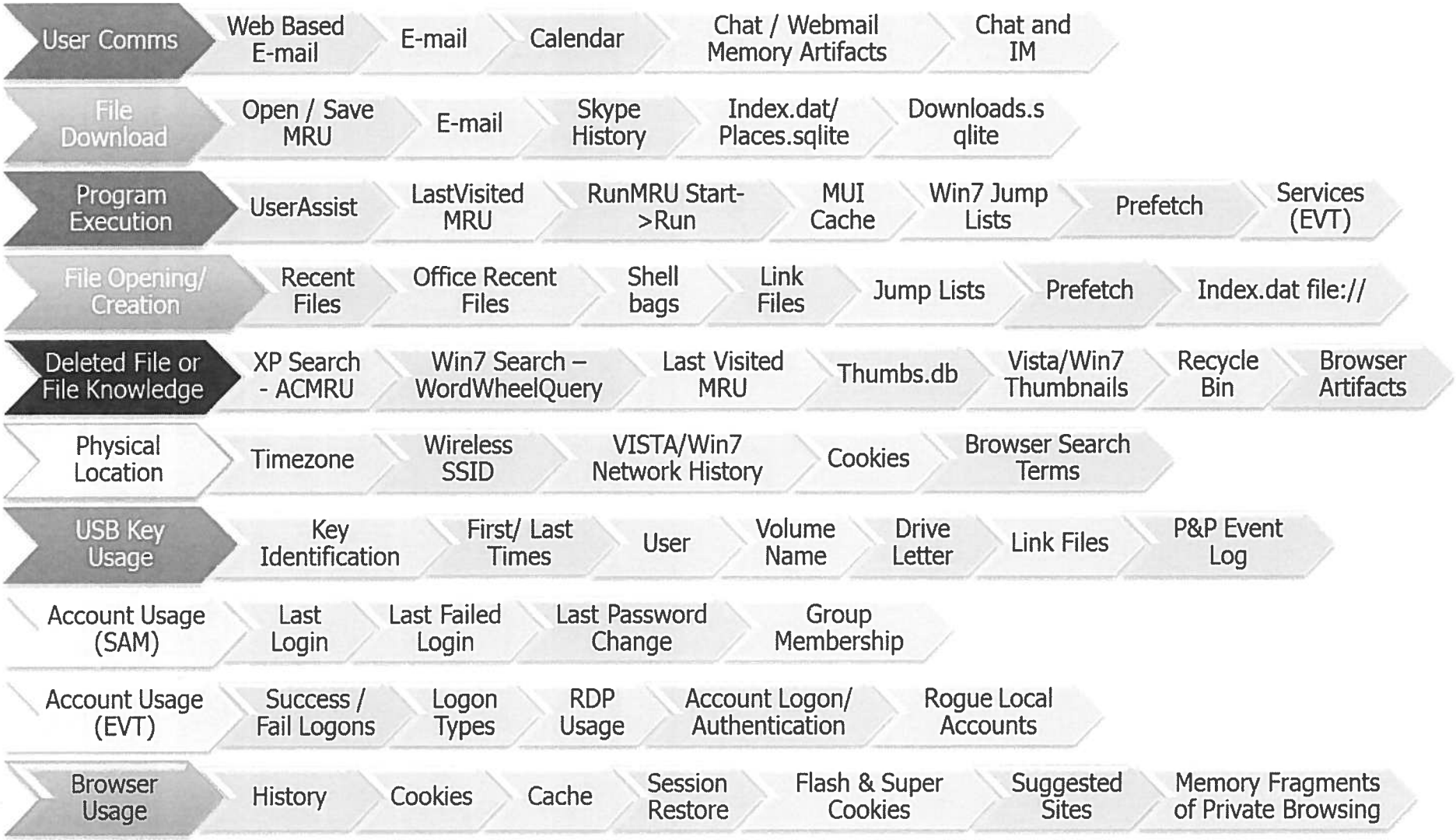
Detecting Compromise Without Malware Artifacts

From the previous page, the *Mandiant M-Trends* report discusses the utilization of common forensic techniques to detect systems that were compromised. You might notice that many of those techniques are discussed in the first class. We cover the rest in FOR508 to give you a full picture view to determine systems that are compromised.

As detailed, it is much easier to detect systems with active malware on them than systems that were compromised with residue that was left over. Because it mainly requires deep-dive forensics, the enormity of the scope of the analysis across an enterprise is daunting. It is recommended that once you determine malware traces or some fragments you find on one machine, you scan the enterprise looking for those same fragments. Hackers change their profile, but not often enough so that their profile would be completely unique on each system. Use the capability discussed in this next section to help you build an intrusion profile against your adversaries to give you the best chance to identify systems that were compromised that currently do not have malware on them.

It's a mistake to assume that the skills for intrusion scenarios and tracking advanced individuals require a different set of skills beyond tracking regular criminals. In intrusion cases, many adversaries are detected on systems using the previous techniques. Take a look at the *Mandiant M-Trends* report from 2012 for some details. We will show you how to combine the skills into a methodology that will truly make you a lethal forensicator against our adversaries. But if you are not familiar with what I consider "conversational forensics," then this class will be a challenge.

For example, if we are determining whether malware was executed on a machine, I might mention, "Did we detect anything during prefetch analysis?" I noticed when I parsed the Userassist, that I didn't find any fragments. However, I did find some evidence found in the Shellbags that might be useful to verify that we know something happened. "Conversational forensics" means that you are familiar with most of the terms, and although some of it might not be memorized yet, all it will take the instructor to do is a quick mention of what each artifact is to remind you of it. The locations of the artifacts, their purpose, and how they can help you are all taught back in FOR408.

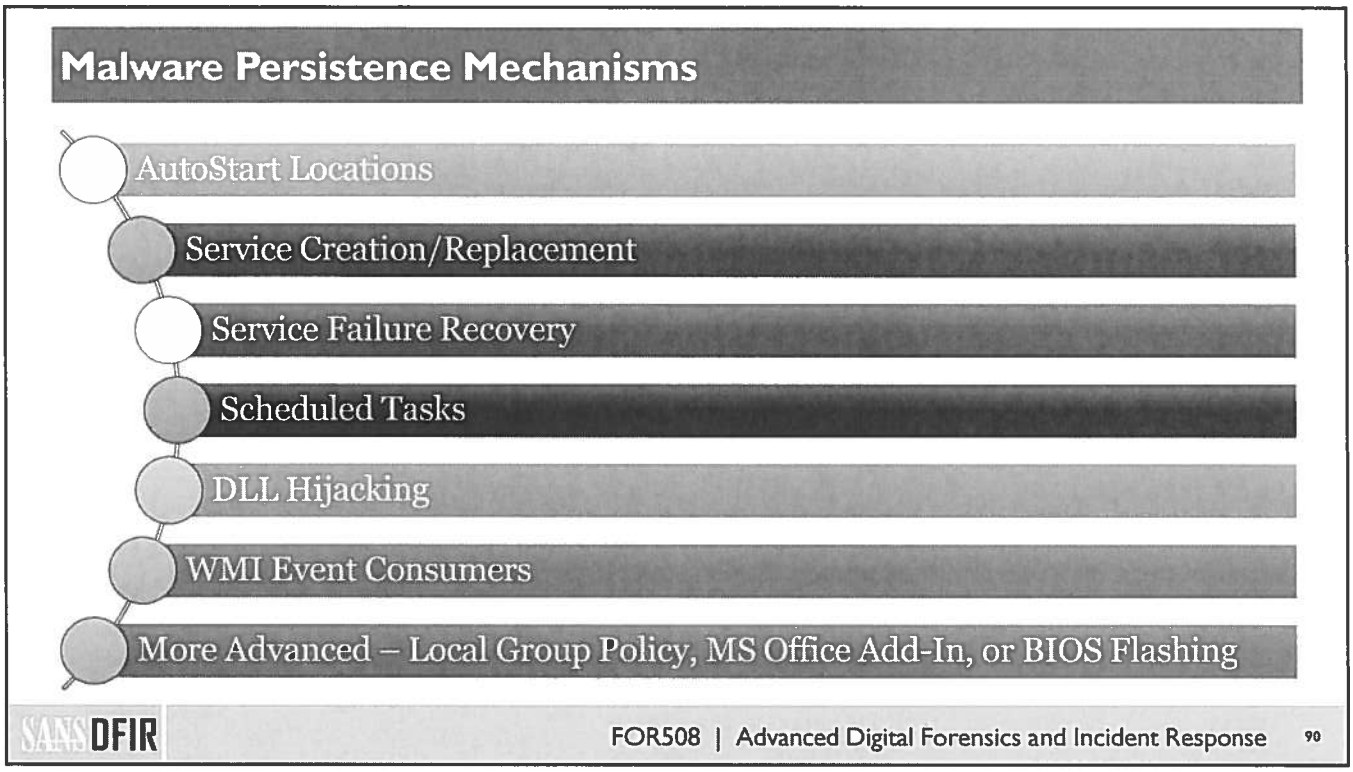

User Comms
Web Based E-mail
E-mail
Calendar
Chat / Webmail Memory Artifacts
Chat and IM
File Download
Open / Save MRU
E-mail
Skype History
Index.dat/ Places.sqlite
Downloads.sqlite
Program Execution
UserAssist
LastVisited MRU
RunMRU Start->Run
MUI Cache
Win7 Jump Lists
Prefetch
Services (EVT)
File Opening/ Creation
Recent Files
Office Recent Files
Shell bags
Link Files
Jump Lists
Prefetch
Index.dat file://
Deleted File or File Knowledge
XP Search - ACMRU
Win7 Search – WordWheelQuery
Last Visited MRU
Thumbs.db
Vista/Win7 Thumbnails
Recycle Bin
Browser Artifacts
Physical Location
Timezone
Wireless SSID
VISTA/Win7 Network History
Cookies
Browser Search Terms
USB Key Usage
Key Identification
First/ Last Times
User
Volume Name
Drive Letter
Link Files
P&P Event Log
Account Usage (SAM)
Last Login
Last Failed Login
Last Password Change
Group Membership
Account Usage (EVT)
Success / Fail Logons
Logon Types
RDP Usage
Account Logon/ Authentication
Rogue Local Accounts
Browser Usage
History
Cookies
Cache
Session Restore
Flash & Super Cookies
Suggested Sites
Memory Fragments of Private Browsing

This page intentionally left blank.

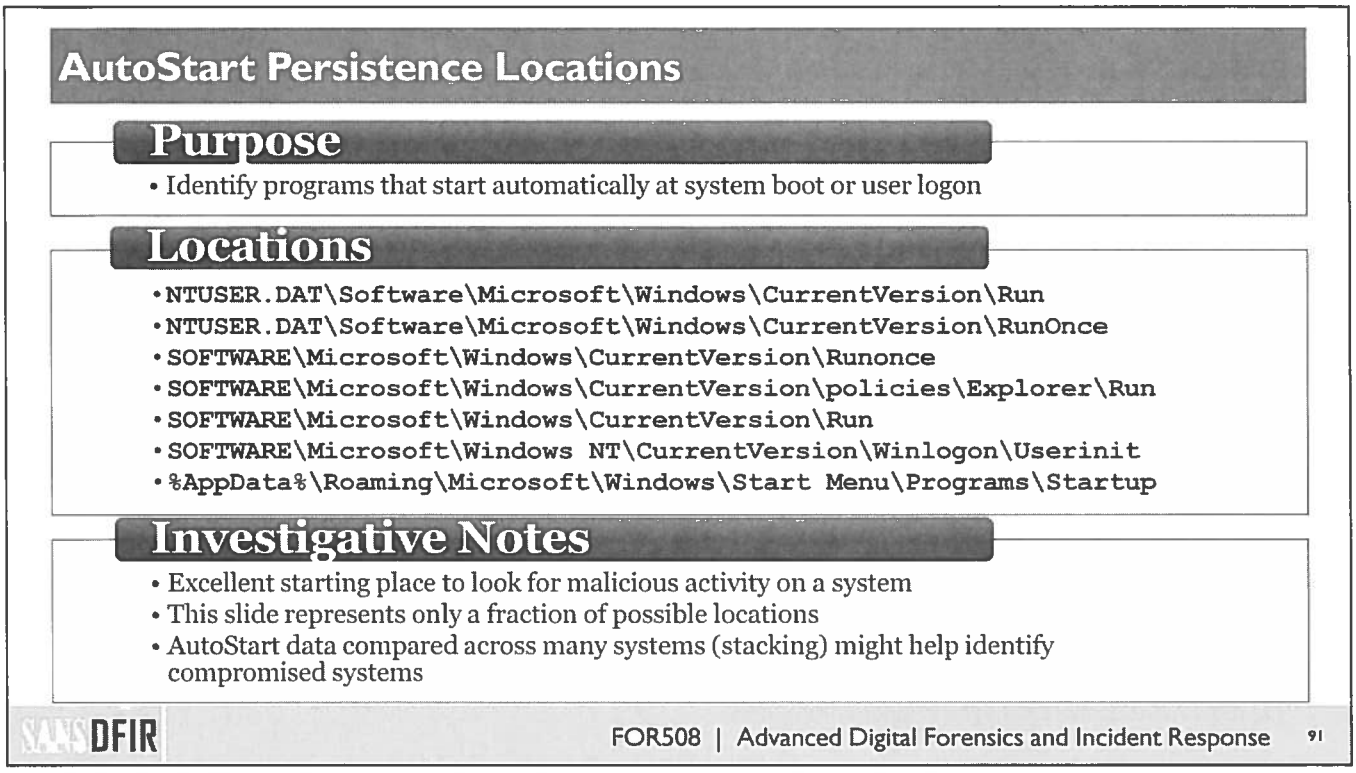
Malware Persistence

Malware wants to hide, but must survive a reboot

This page intentionally left blank.



This page intentionally left blank.



AutoStart Persistence Locations

There are a daunting number of "autorun" locations available in Windows. In Microsoft-speak, these are also known as AutoStart Extension Points (ASEPs) and they are one of the key reasons why Windows is so hard to secure. A quick look at any number of blogs shows well over 50 ASEP locations that a malicious file can place a reference to itself to ensure it survives a reboot. [1] Luckily, many of the most common ASEP locations are in the registry, at least giving us a single place to look (even if there are over 500,000 registry keys on a standard system). A sampling of some common ASEPS are seen on this slide.

By far the most popular ASEPs on the planet are the "run" registry keys:

```
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Run
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\RunOnce
Software\Microsoft\Windows\CurrentVersion\Runonce
Software\Microsoft\Windows\CurrentVersion\policies\Explorer\Run
Software\Microsoft\Windows\CurrentVersion\Run
```

Items listed in these keys are executed when a user logs on—not at boot like other ASEPs. There is no specific order to the startup, and multiple "run" keys exist in both the NTUSER.DAT and the SOFTWARE hives.

Less common, but equally lethal, is the Userinit key.

```
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit\
```

Typically, we would expect this key to contain only a reference to userinit.exe. By default, Winlogon runs Userinit.exe, which runs logon scripts, reestablishes network connections, and then starts Explorer.exe, the Windows user interface. However, the key can be modified to include a reference like C:\Windows\system32\userinit.exe,C:\Temp\winsvchost.exe and the malicious binary will also be executed at boot.

A final location to highlight is in the file system, not the registry. This can actually be advantageous to an attacker because creating persistence here does not require Administrator rights. Even advanced attackers often use this location as an early stage persistence mechanism via phishing attacks.

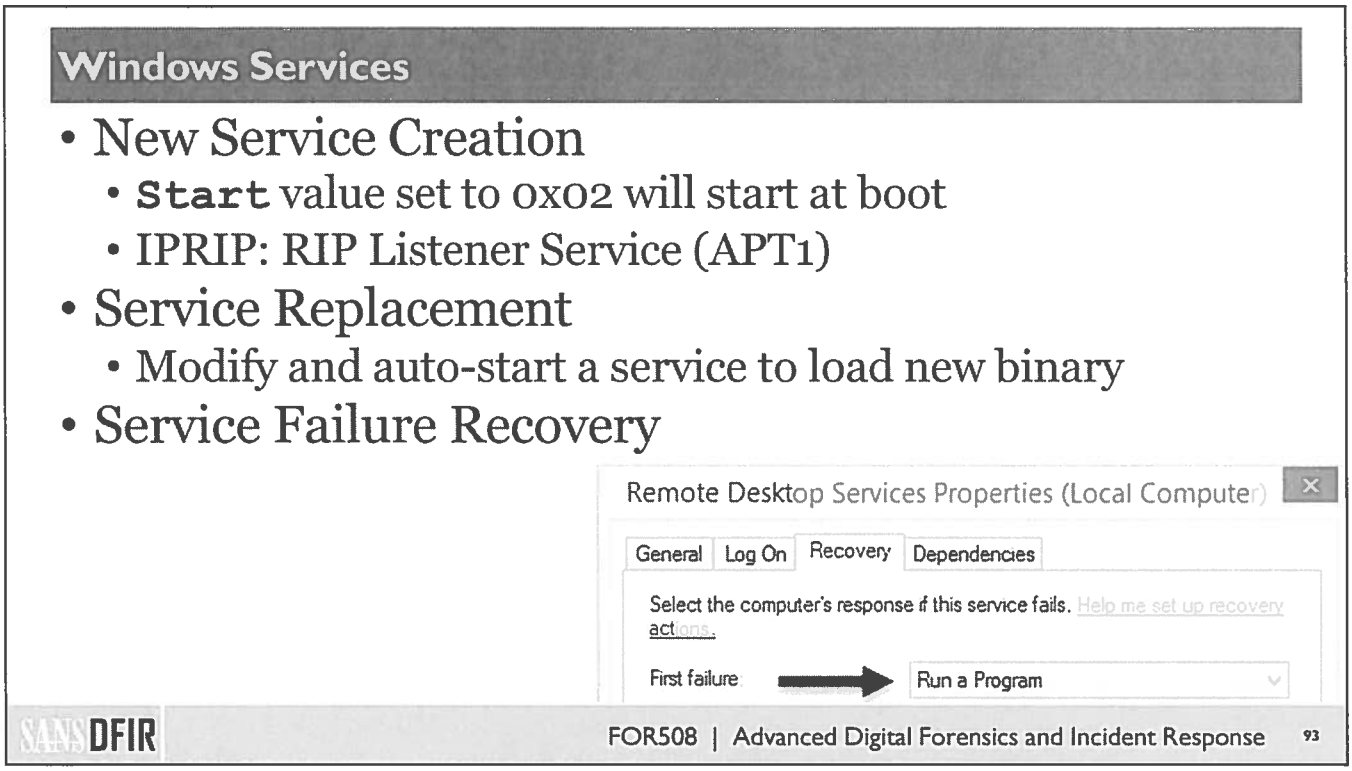
```
%AppData%\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
```

Any shortcuts created in this folder will execute the representative binary upon user logon—easy and effective!

Although these locations are very common for ASEPs, there are many, many more. RegRipper can retrieve many of them from registry hives. [2] We will shortly introduce tools like Autoruns and Kansa that provide the ability to collect this information at scale. Further, by collecting this data across many systems, frequency analysis can be conducted to help identify outliers that can lead us to compromised systems.

[1] http://gladiator-antivirus.com/forum/index.php?showtopic=24610

[2] https://code.google.com/p/regripper/wiki/ASEPs



Windows Services

Windows services are designed to run applications in the background without user interaction. Many services are required at system boot, including the DHCP Client, Windows Event Log, Server, and Workstation services. These services provide critical functionality for the OS and must be started immediately without requiring user input. Services can be implemented as standalone executables or loaded as DLLs. To conserve resources, many service DLLs are grouped together and run under a smaller set of svchost.exe instances. svchost.exe is a Windows-generic service host process, and it is typical to see several running instances of svchost.exe (5 or more is common). Service configurations, as well as device driver configurations, are stored in the registry under HKLM\SYSTEM\CurrentControlSet\Services. The keys here provide the parameters for each service, including the service name, display name, path to the service's executable image file, the start value, required privileges, dependencies, and more. Each service has a start type value configured to start at boot, by manual intervention, or on trigger events such as obtaining an IP address or hardware device connections. Windows services provide great flexibility to developers, and similarly malware authors, for automatically running code on a Windows host.

Because services can be configured to reliably start at boot (often before the loading of anti-virus), they are a very popular persistence vector. It also helps that the average Windows system can easily have over one hundred services registered, making it very easy to hide in plain sight. With administrator rights, it is trivial to either modify the Services registry key or use the built-in "sc" command to create a service that auto-loads a malicious DLL or executable. One of the classic APT examples of this technique is to add (or replace) the rarely used RIP Listener Service (IPRIP) and use it to load a malicious executable. To give an example of how prevalent this technique is, out of the 44 malware families enumerated in the Mandiant APT1 report, 14 used Services for persistence.[1]

Service replacement is similar to service creation, but instead finds a current service that is unused or unneeded and replaces the existing binary with a malicious one. If the service is not already set to auto-start, it is trivial to modify the start type value for the service. This can be more stealthy because it is normal for that service to be on the system, but requires finding an unimportant service to replace. This technique is not as common as simple service creation due to the increased complexity.

Even less common, but potentially more stealthy, is using the service recovery mode option to load a malicious binary when a specific service crashes. The example on this slide shows the Remote Desktop Services recovery options set to run a program upon failure (typically, it would default to restarting the service). As Mark Baggett points out in his post on the topic, this service might be particularly interesting because there are known vulnerabilities for reliably crashing the RDP service.[2]

The Autoruns tools from SysInternals provides an easy means to collect and analyze services on a system. Alternatively, on live systems use the built-in "sc" command to query installed services, using parameters such as "queryex," "qc," "qprivs," and "qtriggerinfo" to get detailed information on service configurations. For offline analysis, investigate service configurations within the registry. Few tools collect service failure recovery information, but the Kansa PowerShell framework has a script named **`Get-SvcFail.ps1`** to collect it within its default ASEP modules. Investigating unusual service crashes in the event logs might also provide clues.

[1] http://contagiodump.blogspot.com/2013/03/mandiant-apt1-samples-categorized-by.html

[2] https://isc.sans.edu/diary/Wipe+the+drive+Stealthy+Malware+Persistence+-+Part+2/15406

Scheduled Tasks

- **`at.exe`**
  - Deprecated, but still present in WinXP and Win7+
  - Use recorded in at*.job files and Schdlgu.txt (XP)
- **`schtasks.exe`**
  - Activity logged in Task Scheduler and Security logs
- Remote capabilities are commonly used for lateral movement and credential dumping

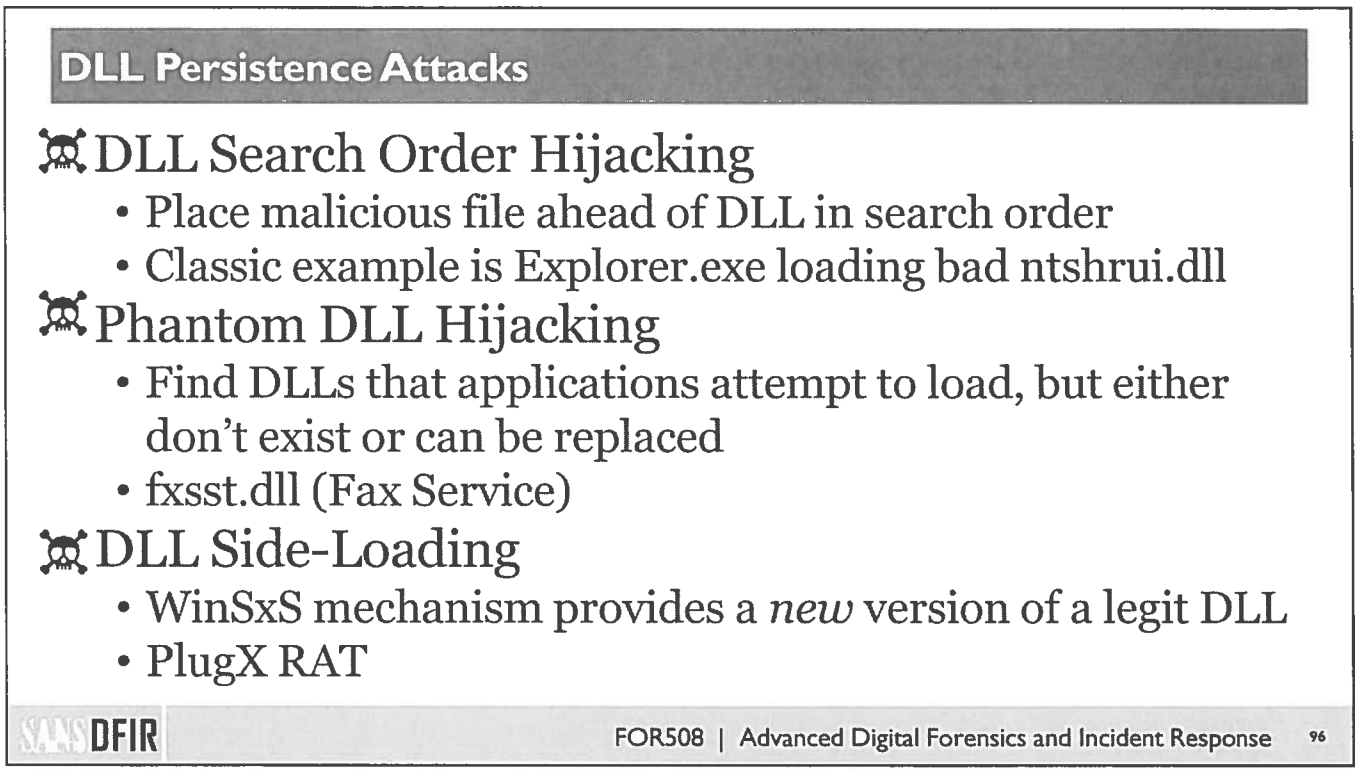
Scheduled Tasks

Scheduled tasks provide an extremely granular means to create persistence in Windows. The at.exe command has long been a core part of the hacker lexicon, most notably because in WinXP, it provided a very reliable privilege escalation attack ("at" jobs originally ran as SYSTEM regardless of the user's privilege level). Even with that vulnerability patched, they are still commonly used by adversaries in Windows 7 environments, likely due to familiarity or laziness (schtasks.exe requires more typing). When an "at" job is created, corresponding ".job" files are created in the \Windows\Tasks and \Windows\System32\Tasks folders (the latter directory was added in Windows7 and records duplicate task information in XML format). These files are named sequentially starting with "at1.job" and record details about what was scheduled. In XP, task information is also stored in the C:\Windows\Schedlgu.txt log. A simple command might look like: **`at.exe 22:01:00 c:\temp\winsvchost.exe`**

The schtasks.exe tool is an upgraded version of at.exe and has an immense number of features allowing tasks to be set and finely controlled. Tasks can even be set for specific Windows events such as when a specific user logs on, allowing much more creativity for persistence over just using specific times. New logging for scheduled tasks appeared in Windows Vista, including a dedicated event log named "Task Scheduler Operational." A simple command might look like: **`schtasks.exe /create /sc daily /tn winsvchost /tr c:\temp\winsvchost.exe /st 22:01:00`**

Interestingly, both commands have the ability to schedule tasks on remote systems. As you might imagine, this opens interesting possibilities for attackers. Remote scheduled tasks are routinely used to spread malware (including backdoors), execute batch scripts, and perform routine actions like credential dumping across many systems. Most forensic artifacts for these remote tasks will be present on the systems they were executed on, not the originating system.

The Autoruns tool from SysInternals will collect currently scheduled jobs from the task scheduler service.



DLL Persistence Attacks

DLL persistence hijacks largely attack legitimate and legacy features of the Windows operating system. Search order hijacking is an excellent example. It turns out that when an executable runs in Windows, it is not required to hardcode the location of any required dynamically loaded libraries (DLLs). Instead, a specific search order is often used to find the required DLL, starting with the local directory the .EXE is run from and eventually ending up in a folder like C:\Windows\System32 where most standard DLLs exist. The only real exception is for DLLs present in the KnownDLLs registry key that does effectively hardcode a small number of specific system DLL locations. If adversaries can find an executable that is not located in the System32 folder and loads a DLL not present in the KnownDLLs registry key, they can place a malicious DLL in the same folder as the target executable and trump the search order, ensuring their bad code is loaded whenever that application starts. Amazingly, there are lots of these susceptible locations going back all the way to Windows2000. A classic example was documented in the wild by Mandiant where Explorer.exe (the Windows GUI desktop) loaded a vulnerable DLL (not protected by KnownDLLs) named "ntshrui.dll."[1] Because Explorer.exe is located in the \Windows folder and the legitimate ntshrui.dll is located in the \Windows\System32 folder, all the attackers had to do is find a way to drop their malicious dll (named "ntshrui.dll") into the \Windows folder, and it would be executed every time the desktop was started. Winning! Because this is directly related to backwards compatibility, there is no likely fix on the horizon. File system forensic analysis looking for newly created DLLs in unusual locations is usually the best way to discover it.

Phantom DLL hijacking is a similar attack, but uses the fact that some very old DLLs are still attempted to be loaded by applications even when they are completely unnecessary. In fact, some applications try to load very old DLLs that no longer even exist on modern Windows operating systems! If attackers can find such a DLL (and many are already documented), all they have to do is provide a malicious file with the same name of that long forgotten DLL in the search path and code will be executed.[2] Similarly, even if the DLL exists, it might be able to be easily replaced with a trojanized version. A great example is the replacement of the fxsst.dll (Fax Service) DLL in the System32 folder documented by Mandiant and still being used by attackers in the wild. [3]

Finally, we have DLL side-loading, which has a similar sounding name, but is actually quite different than the previous two attacks. This attack uses the Windows side-by-side (SxS) DLL loading mechanism to introduce an "updated" version of a DLL. SxS functionality is a legitimate feature of Windows and is used by many applications to prevent problems that can arise due to updated and duplicate versions of DLLs. SxS gives the ability to load updated DLLs, but has few validity checks for these new DLLs and thus the loading mechanism can be abused due to missing DLLs, use of relative paths, and other shortcuts not taken into account by the application developer. This attack is often used to circumvent AV protections and provides an opportunity for a known good, even digitally signed, executable to be used as the persistence mechanism. The most notable example is the very popular PlugX RAT, which drops a legitimate executable (with a hash present in the NSRL known good hash database) and then uses SxS during runtime to load and construct a malicious DLL in memory. Amanda Stewart at FireEye wrote an excellent whitepaper describing the process.[4] Similar to other DLL persistence attacks, the best way to discover this behavior is to identify new executables and helper files added to the system during the attack (PlugX often creates three new files on the system).

[1] https://www.mandiant.com/blog/malware-persistence-windows-registry/

[2] Phantom DLL Hijacking: http://www.hexacorn.com/blog/2013/12/08/beyond-good-ol-run-key-part-5/

[3] https://www.mandiant.com/blog/fxsst/

[4] https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-dll-sideloading.pdf

WMIEvtConsumer Backdoors

WMI allows triggers (filters) to be set than when satisfied will run scripts or executables

- Event Filter → Trigger condition
- Event Consumer → Script or executable to run
- Binding → Tie together Filter + Consumer

- Filters can be based on time (every 20 sec), service start, user auth, file creation, etc.
- The PowerShell cmdlet "Get-WmiObject" can identify and help remove suspicious entries

WMIEvtConsumer Backdoors

WMI is often overlooked by security professionals, but it contains very powerful capabilities that have not gone unnoticed in the hacker community. One of the more recent persistence methods identified in the wild has been the use of WMI Event Consumers. WMI provides the ability to monitor for specific events and when triggered, alert event consumers that can then do things like run scripts and execute code.[1] Administrative privileges are necessary, but once achieved, attackers can use WMI to create a backdoor that is difficult to detect without the proper tools. A WMI consumer runs with SYSTEM privileges in XP and with LOCAL_SERVICE privileges in Win7+.

The technique requires three discreet steps:

1) An event filter must be created describing a specific trigger to detect (for example, trigger every twenty seconds).
2) An event consumer is added to the system with a script and/or executable to run (run a PowerShell script to beacon to a command and control server).
3) Finally, the event and consumer are tied together via a binding, and the persistence mechanism is loaded into the WMI repository.

The three steps are often written inside a managed object format (MOF) file that is used to register new classes into the WMI repository. Event filters can be set up to trigger immediately upon being registered or via virtually any other windows event such as a specific time, the existence of a file or folder, service starting or stopping, a specific user being authenticated, etc.

This type of attack is not theoretical. Stuxnet was perhaps the first sample in the wild to use the attack. [2] It used a zero-day vulnerability in the print spooler (MS10-061) to allow the transfer of two files to remote systems—an .EXE and a .MOF file. The .MOF file was auto-compiled by the system, creating a WMI event filter and consumer to immediately execute the .exe file. Wow!

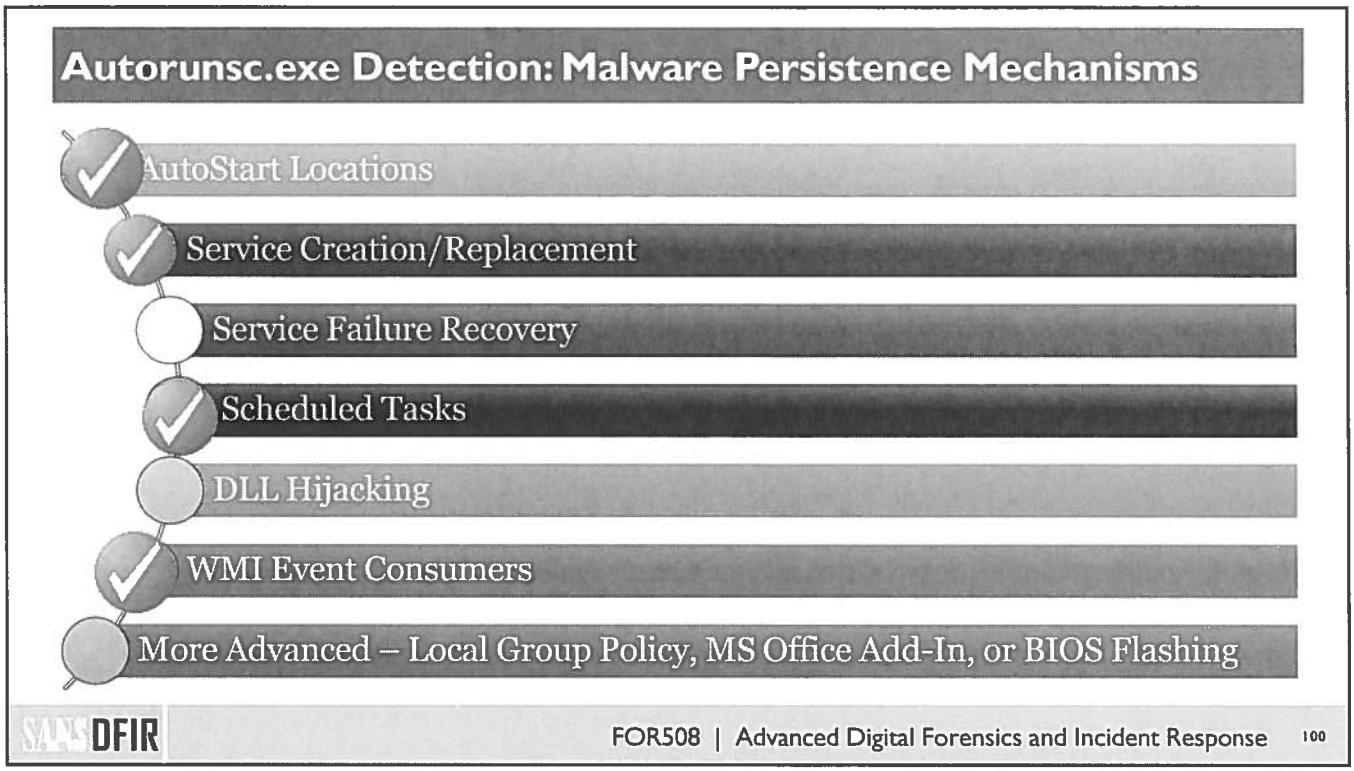
Chris Glyer gave an excellent presentation in 2014 describing the (limited) forensic artifacts left behind by WMI Event Consumers.[3] The SysInternals tool Autoruns and the Kansa PowerShell framework both identify WMI event filters and consumers.[4] The PowerShell cmdlet "Get-WmiObject" can also be used as a native means to identify and help remove suspicious entries.

[1] Monitoring Events - https://msdn.microsoft.com/en-us/library/aa392396(v=vs.85).aspx

[2] http://www.esetnod32.ru/company/viruslab/analytics/doc/Stuxnet_Under_the_Microscope.pdf

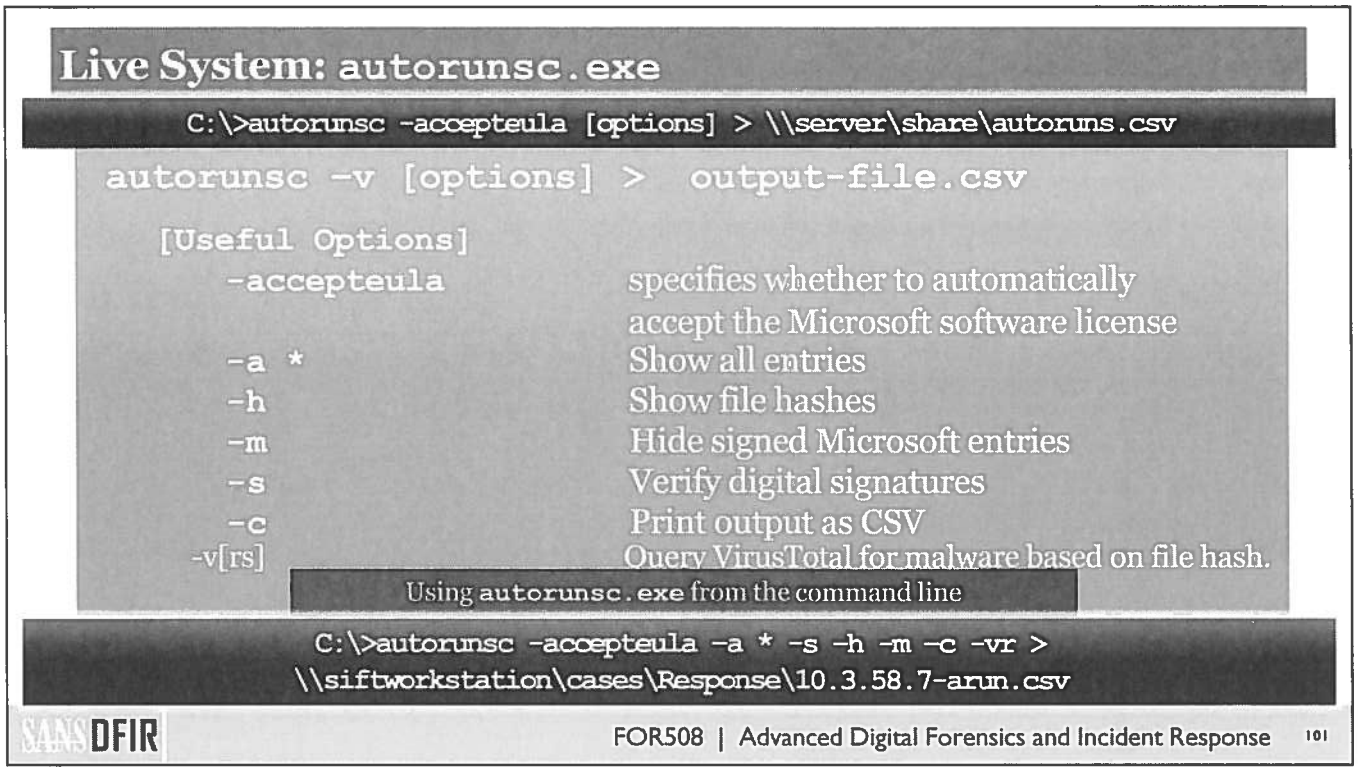
[3] There's Something About WMI - https://files.sans.org/summit/Digital_Forensics_and_Incident_Response_Summit_2015/PDFs/TheresSomethingAboutWMIDevonKerr.pdf

[4] Kansa and WMI Event Consumers - http://trustedsignal.blogspot.com/2014/05/kansa-collecting-wmi-event-consumer.html



Autoruns Detection: Malware Persistence Mechanisms

Similar to application vulnerabilities, we will likely never reach the end of new persistence mechanism discoveries. The top six items on this list comprise probably 98% of those you are likely to find, with the other 2% being rare and esoteric examples of firmware implants, new forms of DLL hijacking, etc. Although some persistence mechanisms are most often identified via timelining and disk forensics (DLL Hijacking and group policy scripts come to mind), the Autoruns tool from SysInternals has the ability to collect data from the vast majority of other ASEPs. It is a go-to tool for incident responders and is often one of the first items reviewed in an investigation.



Live System: `autorunsc.exe`

This utility, which has the most comprehensive knowledge of auto-starting locations of any startup monitor, shows you what programs are configured to run during system bootup or login, and shows you the entries in the order Windows processes them. These programs include ones in your startup folder, Run, RunOnce, and other Registry keys. You can configure Autoruns to show other locations, including Explorer shell extensions, toolbars, browser helper objects, Winlogon notifications, auto-start services, and much more. Autoruns goes way beyond the MSConfig utility bundled with Windows XP.

Autorunsc Usage

Autorunsc is the command-line version of Autoruns. Its usage syntax is:

Usage: autorunsc [-a] | [-c] [-b] [-d] [-e] [-g] [-h] [-i] [-l] [-m] [-n] [-p] [-r] [-s] [-v] [-w] [-x] [user]

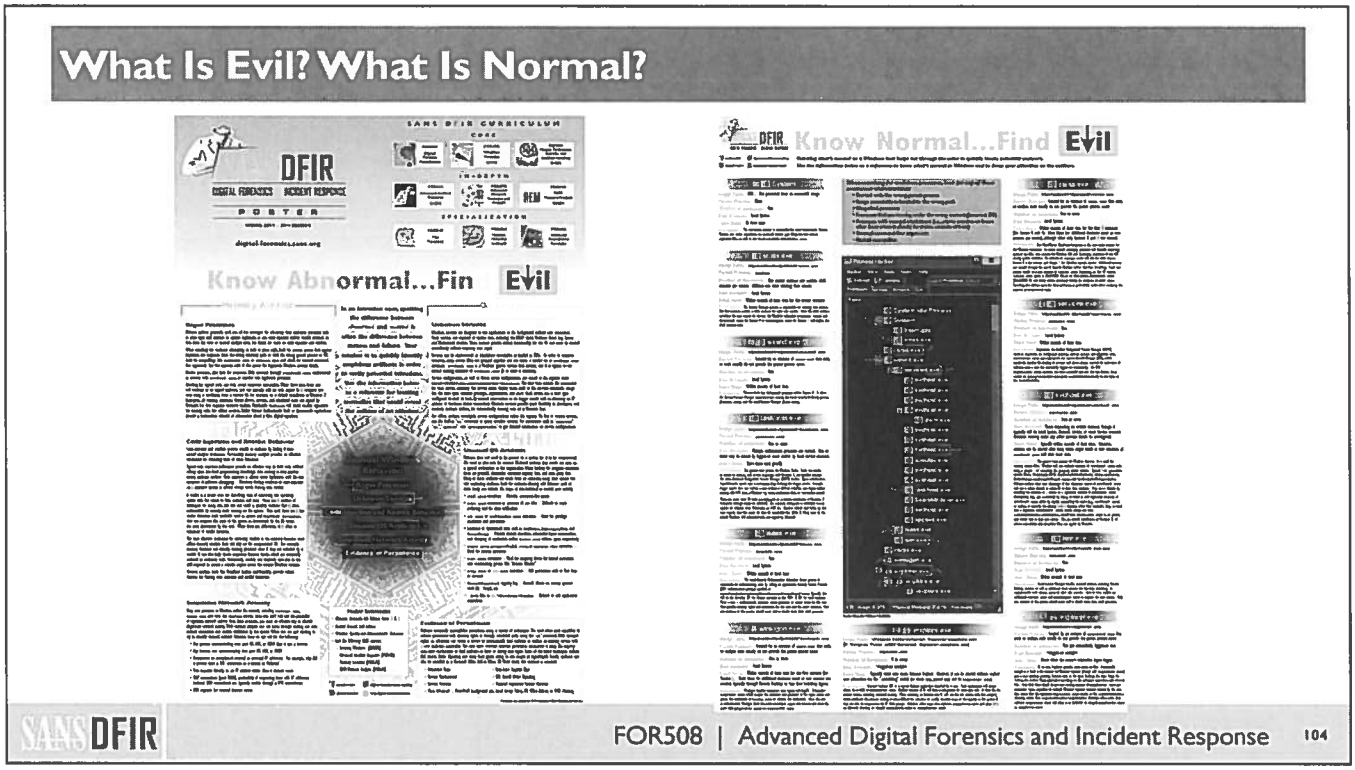
- **-a** Show all entries.
- **-b** Boot execute.
- **-c** Print output as CSV.
- **-d** Appinit DLLs.
- **-e** Explorer addons.
- **-g** Sidebar gadgets (Vista and higher).
- **-h** Image hijacks.
- **-I** Internet Explorer addons.
- **-l** Logon startups (this is the default).
- **-m** Hide signed Microsoft entries.
- **-n** Winsock protocol and network providers.
- **-p** Printer monitor drivers.
- **-r** LSA providers.

-s	AutoStart services and non-disabled drivers.
-t	Scheduled tasks.
-v	Verify digital signatures.
-w	Winlogon entries.
-x	Print output as XML.
user	Specifies the name of the user account for which Autorun items will be shown.

Exercise 1.4

AutoStart Persistence Analysis

This page intentionally left blank.



What Is Evil? What Is Normal?

In an intrusion case, spotting the difference between abnormal and normal is often the difference between success and failure. Your mission is to quickly identify suspicious artifacts in order to verify potential intrusions.

Use the following information as a reference for locating anomalies that could reveal the actions of an attacker. You should receive the poster with the FOR508 course. If you did not, one can be downloaded from our website: http://digital-forensics.sans.org/media/poster_2014_find_evil.pdf.

Rogue Processes

Malware authors generally pick one of two strategies for obscuring their malicious processes: hide in plain sight and attempt to appear legitimate, or use code injection and/or rootkit methods to hide from the view of normal analysis tools.

When searching for malware attempting to hide in plain sight, look for process names that appear legitimate but originate from the wrong directory path or with the wrong parent process or SID. Look for misspellings like scvhost.exe or lssass.exe and check for unusual command-line arguments. See the opposite side of this poster for legitimate Windows process details.

Besides processes, also look for suspicious DLLs executed through rundll32.exe, implemented as services with svchost.exe, or injected into legitimate processes.

Checking for signed code can help reveal suspicious executables. Although there have been and will continue to be signed malware, you can typically rely on code signed by a company you trust using a certificate from a trusted CA. For example, on a default installation of Windows 7 Enterprise, all running processes, device drivers, services, and scheduled tasks are signed by Microsoft. For live response memory analysis, Mandiant's Redline will check on-disk signatures for running code. For offline analysis, Didier Stevens' Authenticode Tools or Sysinternal's sigcheck.exe provide a tremendous amount of information about a file's digital signature.

Code Injection and Rootkit Behavior

Code injection and rootkits provide stealth to malware by hiding it from normal analysis techniques. Fortunately, memory analysis provides an effective mechanism for detecting both of these behaviors.

Typical code injection techniques provide an effective way to hide code without relying upon low-level programming knowledge, thus making it very popular among malware authors. Code injection is almost never legitimate, with the one exception of software debugging. Therefore, finding evidence of code injection on a standard system is almost always worth looking into further.

A rootkit is a broad term for describing ways of subverting the operating system with the intent to hide activities and data. There are a number of techniques for doing this, but the end result is stealthy malware that is often undetectable by security tools running on the system. That said, there are a few rootkit detection tools available, such as GMER and Rootkit Revealer, which can compare the state of the system as determined by the OS versus the state determined by the tool. When there are differences, it is often an indication of rootkit behavior.

The most effective technique for detecting rootkits is via memory forensics, because offline memory analysis does not rely on the compromised OS. For example, memory forensics can identify running processes even if they are unlinked by a rootkit. It can also help locate suspicious function hooks, which are essentially redirects to malicious code. Fortunately, rootkits are relatively rare due to the skill required to create a reliable exploit across the various Windows versions. Memory analysis tools like Mandiant Redline and Volatility provide robust features for finding code injection and rootkit behaviors.

Unknown Services

Windows services are designed to run applications in the background without user interaction. Many services are required at system boot, including the DHCP Client, Windows Event Log, Server, and Workstation services. These services provide critical functionality for the OS and must be started immediately without requiring user input.

Services can be implemented as standalone executables or loaded as DLLs. In order to conserve resources, many service DLLs are grouped together and run under a smaller set of svchost.exe instances. svchost.exe is a Windows generic service host process, and it is typical to see several running instances of svchost.exe (5 or more is common).

Service configurations, as well as device driver configurations, are stored in the registry under HKLM\SYSTEM\CurrentControlSet\Services. The keys here provide the parameters for each service, including the service name, display name, path to the service's executable image file, the start type, required privileges, dependencies, and more. Each service has a start type configured to start at boot, by manual intervention, or on trigger events such as obtaining an IP address or hardware device connections. Windows services provide great flexibility to developers, and similarly malware authors, for automatically running code on a Windows host.

For offline analysis, investigate service configurations within the registry. On live or remote systems, use the built-in "sc" command to query installed services. Try parameters such as "queryex," "qc," "qprivs," and "qtriggerinfo" to get detailed information on service configurations.

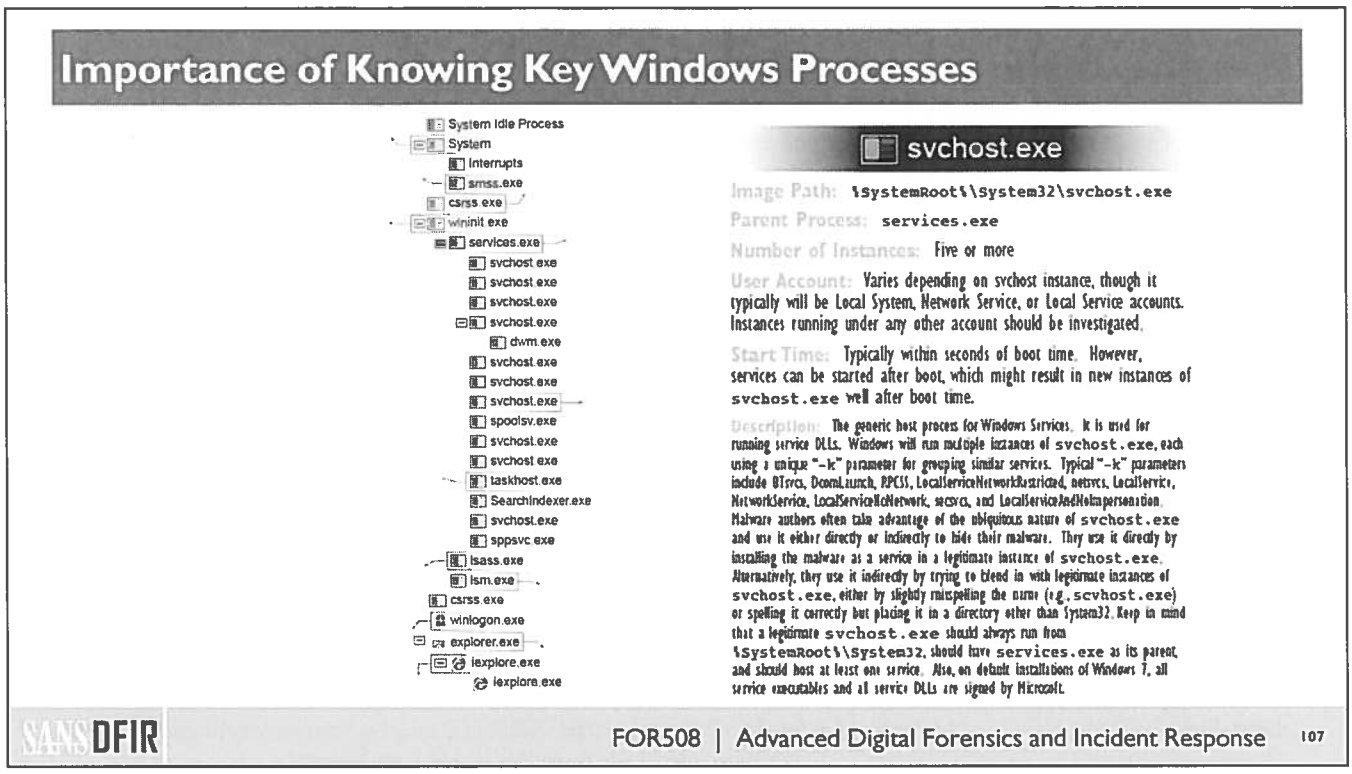
Unusual OS Artifacts

Malware does not need to be present on a system for it to be compromised. We need to also look for unusual OS-based artifacts that would not exist on a typical workstation in the organization. When looking for program execution, focus on prefetch, shimcache, userassist registry keys, and even jump lists.

Many of these artifacts can result from an adversary using your system but not implanting malware. Look for evidence showing odd behavior such as tools being run outside the scope of non-technical or normal user activity:

- **cmd.exe execution**: Provides command-line access.
- **rar.exe execution or presence of .rar files**: Difficult to crack archiving tool for data exfiltration.
- **at.exe or schtasks.exe execution**: Used for privilege escalation and persistence.

- **Existence of Sysinternals tools such as PsExec, PsLoggedOn, and ProcDump**: Provide remote execution, interactive logon enumeration, and dumping of credentials within lsass.exe address space, respectively.
- **wmic.exe, powershell.exe, or winrm.vbs execution**: Used for remote execution.
- **net.exe execution**: Used for mapping drives for lateral movement and enumerating groups like "Domain Admins."
- **reg.exe or sc.exe execution**: Add persistence such as Run keys or services.
- **MountPoints2 registry key**: Records shares on remote systems such as C$, Temp$, etc.
- **.job files in C:\Windows\Tasks**: Related to odd application executions.



Importance of Knowing Key Windows Processes

Knowing what's normal on a Windows host helps cut through the noise to quickly locate potential malware. Use the following information as a reference to know what's normal in Windows and to focus your attention on the outliers.

When searching for malicious processes, look for any of these anomalous characteristics:

- Started with the wrong parent process
- Image executable is located in the wrong path
- Misspelled processes
- Processes that are running under the wrong account (incorrect SID)
- Processes with unusual start times (for example, starts minutes or hours after boot when it should be within seconds of boot)
- Unusual command-line arguments
- Packed executables

System

Image Path: N/A—Not generated from an executable image

Parent Process: None Number of Instances: One User Account: Local System Start Time: At boot time

Description: The System process is responsible for most kernel-mode threads. Modules run under System are primarily drivers (.sys files), but also several important DLLs as well as the kernel executable, ntoskrnl.exe.

smss.exe

Image Path: %SystemRoot%\System32\smss.exe

Parent Process: System

Number of Instances: One master instance and another child instance per session. Children exit after creating their session.

User Account: Local System

Start Time: within seconds of boot time for the master instance

Description: The Session Manager process is responsible for creating new sessions. The first instance creates a child instance for each new session. Once the child instance initializes the new session by starting the Windows subsystem (csrss.exe) and wininit.exe for Session 0 or winlogon.exe for Session 1 and higher, the child instance exits.

csrss.exe

Image Path: %SystemRoot%\System32\csrss.exe

Parent Process: Created by an instance of smss.exe that exits, so analysis tools usually do not provide the parent process name.

Number of Instances: Two or more

User Account: Local System

Start Time: Within seconds of boot time for the first 2 instances (for Session 0 and 1). Start times for additional instances occur as new sessions are created, although often only Sessions 0 and 1 are created.

Description: The Client/Server Run-Time Subsystem is the user-mode process for the Windows subsystem. Its duties include managing processes and threads, importing most of the DLLs that provide the Windows API, and facilitating shutdown of the GUI during system shutdown. An instance of csrss.exe will run for each session. Session 0 is for services and Session 1 for the local console session. Additional sessions are created through the use of Remote Desktop and/or Fast User Switching. Each new session results in a new instance of csrss.exe. Depending on the OS version, csrss.exe (prior to Win7/2008 R2) or its child process conhost.exe ((Win7/2008 R2 and later) contain command history for instances of cmd.exe. Searching the address space for these processes is particularly useful when analyzing the memory of compromised hosts.

services.exe

Image Path: %SystemRoot%\System32\services.exe

Parent Process: wininit.exe Number of Instances: One User Account: Local System

Start Time: Within seconds of boot time

Description: Implements the Unified Background Process Manager (UBPM), which is responsible for background activities such as services and scheduled tasks. Services.exe also implements the Service Control Manager (SCM), which specifically handles the loading of services and device drivers marked for auto-start. In addition, once a user has successfully logged on interactively, the SCM (services.exe) considers the boot successful and sets the Last Known Good control set (HKLM\SYSTEM\Select\LastKnownGood) to the value of the CurrentControlSet.

svchost.exe

Image Path: %SystemRoot%\System32\svchost.exe

Parent Process: services.exe

Number of Instances: Five or more

User Account: Varies depending on svchost instance, though it typically will be Local System, Network Service, or Local Service accounts. Instances running under any other account should be investigated.

Start Time: Typically within seconds of boot time. However, services can be started after boot, which might result in new instances of svchost.exe well after boot time.

Description: The generic host process for Windows Services. It is used for running service DLLs. Windows will run multiple instances of svchost.exe, each using a unique "-k" parameter for grouping similar services. Typical "-k" parameters include BTsvcs, DcomLaunch, RPCSS, LocalServiceNetworkRestricted, netsvcs, LocalService, NetworkService, LocalServiceNoNetwork, secsvcs, and LocalServiceAndNoImpersonation. Malware authors often take advantage of the ubiquitous nature of svchost.exe and use it either directly or indirectly to hide their malware. They use it directly by installing the malware as a service in a legitimate instance of svchost.exe.

Alternatively, they use it indirectly by trying to blend in with legitimate instances of svchost.exe, either by slightly misspelling the name (e.g., scvhost.exe) or spelling it correctly but placing it in a directory other than System32. Keep in mind that a legitimate svchost.exe should always run from %SystemRoot%\System32, should have services.exe as its parent, and should host at least one service. Also, on default installations of Windows 7, all service executables and all service DLLs are signed by Microsoft.

lsm.exe

Image Path: %SystemRoot%\System32\lsm.exe

Parent Process: wininit.exe Number of Instances: One User Account: Local System

Start Time: Within seconds of boot time

Description: Local Session Manager handles terminal services, including Remote Desktop sessions as well as additional local sessions via Fast User Switching. It communicates with smss.exe to start new sessions. Smss in turn creates an additional csrss.exe and winlogon.exe to support the new session. Only one instance of this process should occur and it should never have child processes.

explorer.exe

Image Path: %SystemRoot%\explorer.exe

Parent Process: Created by an instance of userinit.exe that exits, so analysis tools usually do not provide the parent process name.

Number of Instances: One per interactively logged-on user

User Account: <logged-on user(s)>

Start Time: Starts when the owner's interactive logon begins

Description: At its core, Explorer provides users access to files. Functionally though, it is both a file browser via Windows Explorer (though still explorer.exe) and a user interface providing features such as the user's Desktop, the Start Menu, the Taskbar, the Control Panel, application launching via file extension association, and shortcut files. Note that there should be just one running instance of explorer.exe per interactive logon, regardless of multiple Windows Explorer windows opened by the user. Also notice that the legitimate explorer.exe resides in the %SystemRoot% directory rather than %SystemRoot%\System32. Attackers often name their malware explorer.exe and place it in System32 or misspell explorer.exe as explore.exe.

iexplore.exe

Image Path: \Program Files\Internet Explorer\iexplore.exe

[or \Program Files (x86)\Internet Explorer\iexplore.exe]

Parent Process: explorer.exe Number of Instances: 0 to many User Account: <logged-on user(s)>

Start Time: Typically when user starts Internet Explorer. However, it can be started without explicit user interaction via the "-embedding" switch (in which case, parent may not be explorer.exe).

Description: Internet Explorer (IE) is a typical desktop application launched by a user. Such applications will almost always be a child of explorer.exe. Modern versions of IE will have a sub-process for each open tab. It does this for several reasons, including enhanced security. When accessing an Internet site, IE will run the tab process with low integrity, which sandboxes the process, making it more difficult for attackers to modify sensitive areas of the registry or file system if they are able to compromise the IE child process. Attackers often name their malware iexplore.exe and place it in an alternate directory or misspell iexplore.exe as iexplorer.exe.

winlogon.exe

Image Path: %SystemRoot%\System32\winlogon.exe

Parent Process: Created by an instance of smss.exe that exits, so analysis tools usually do not provide the parent process name.

Number of Instances: One or more

User Account: Local System

Start Time: Within seconds of boot time for the first instance (for Session 1). Start times for additional instances occur as new sessions are created, typically through Remote Desktop or Fast User Switching logons.

Description: Winlogon handles interactive user logons and logoffs. It launches LogonUI.exe, which accepts the username and password at the logon screen and passes the credentials to lsass.exe to validate the credentials. Once the user is authenticated, Winlogon loads the user's NTUSER.DAT into HKCU and starts the user's shell (explorer.exe) via Userinit.exe.

lsass.exe

Image Path: %SystemRoot%\System32\lsass.exe

Parent Process: wininit.exe Number of Instances: One User Account: Local System

Start Time: Within seconds of boot time

Description: The Local Security Authentication Subsystem Server process is responsible for authenticating users by calling an appropriate Security Service Provider (SSP) authentication package specified in HKLM\SYSTEM\CurrentControlSet\Control\Lsa. Typically, this will be the Kerberos SSP for domain accounts or the MSV1_0 SSP for local accounts. Once a user is authenticated, lsass.exe generates an access token for the user that specifies security rights and constraints for the user and the user's processes. Only one instance of this process should occur and it should never have child processes.

taskhost.exe

Image Path: %SystemRoot%\System32\taskhost.exe

Parent Process: services.exe

Number of Instances: One or more

User Account: Multiple taskhost.exe processes are normal. One or more may be owned by logged-on users and/or by local service accounts.

Start Time: Start times vary greatly

Description: The generic host process for Windows Tasks. Tasks are similar in nature to services, and in fact beginning with Windows 7, are handled through the same Universal Background Process Manager (UBPM) facility. Upon initialization, taskhost.exe runs a continuous loop listening for trigger events. Example trigger events that can initiate a task include a defined schedule, user logon, system startup, idle CPU time, a Windows log event, workstation lock, or workstation unlock.

There are more than 70 tasks preconfigured on a default installation of Windows 7 Enterprise (though many are disabled). For example, defrag.exe is scheduled to run against all volumes every Wednesday at 1:00 am. Another default task backs up the core registry hive files every 10 days. All executable files (DLLs & EXEs) used by the default Windows 7/8 scheduled tasks are signed by Microsoft.

wininit.exe

Image Path: %SystemRoot%\System32\wininit.exe

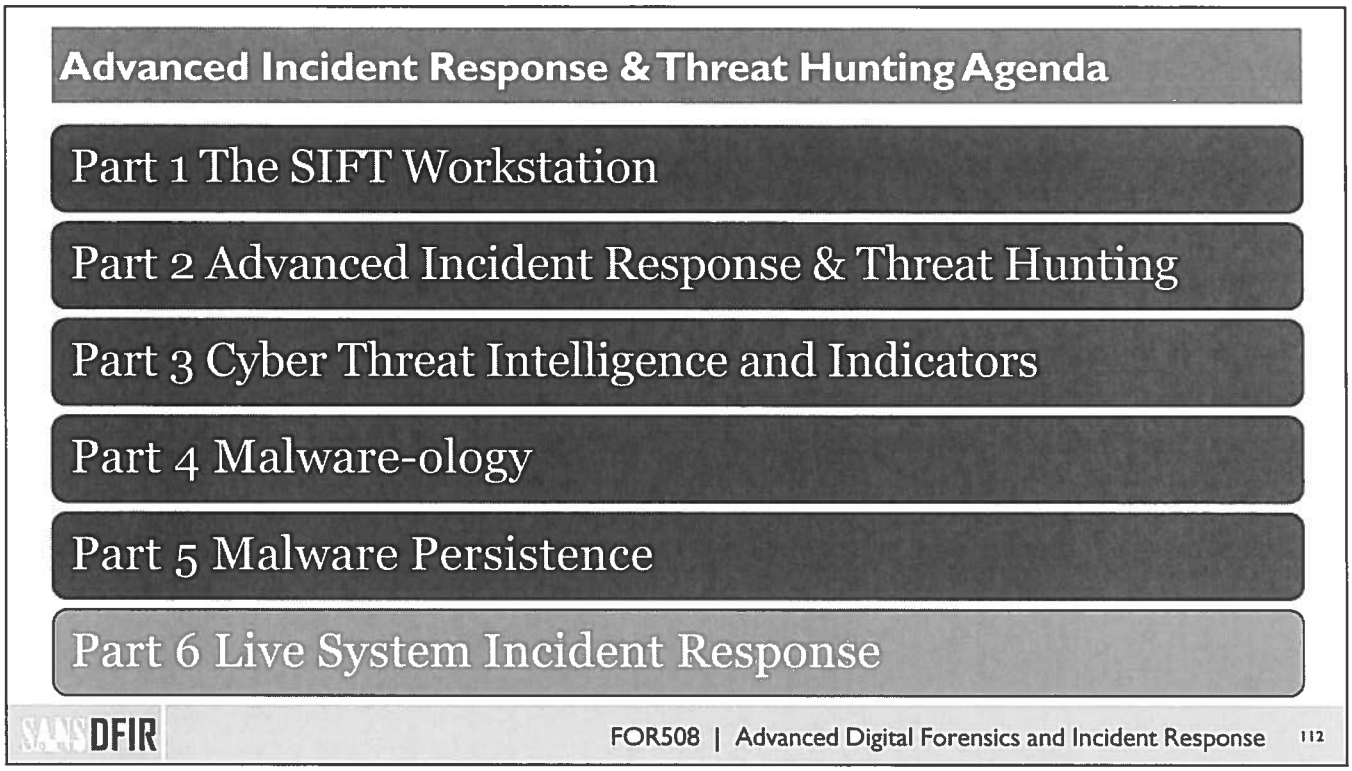
Parent Process: Created by an instance of smss.exe that exits, so tools usually do not provide the parent process name.

Number of Instances: One

User Account: Local System

Start Time: Within seconds of boot time

Description: Wininit starts key background processes within Session 0. It starts the Service Control Manager (services.exe), the Local Security Authority process (lsass.exe), and the Local Session Manager (lsm.exe).



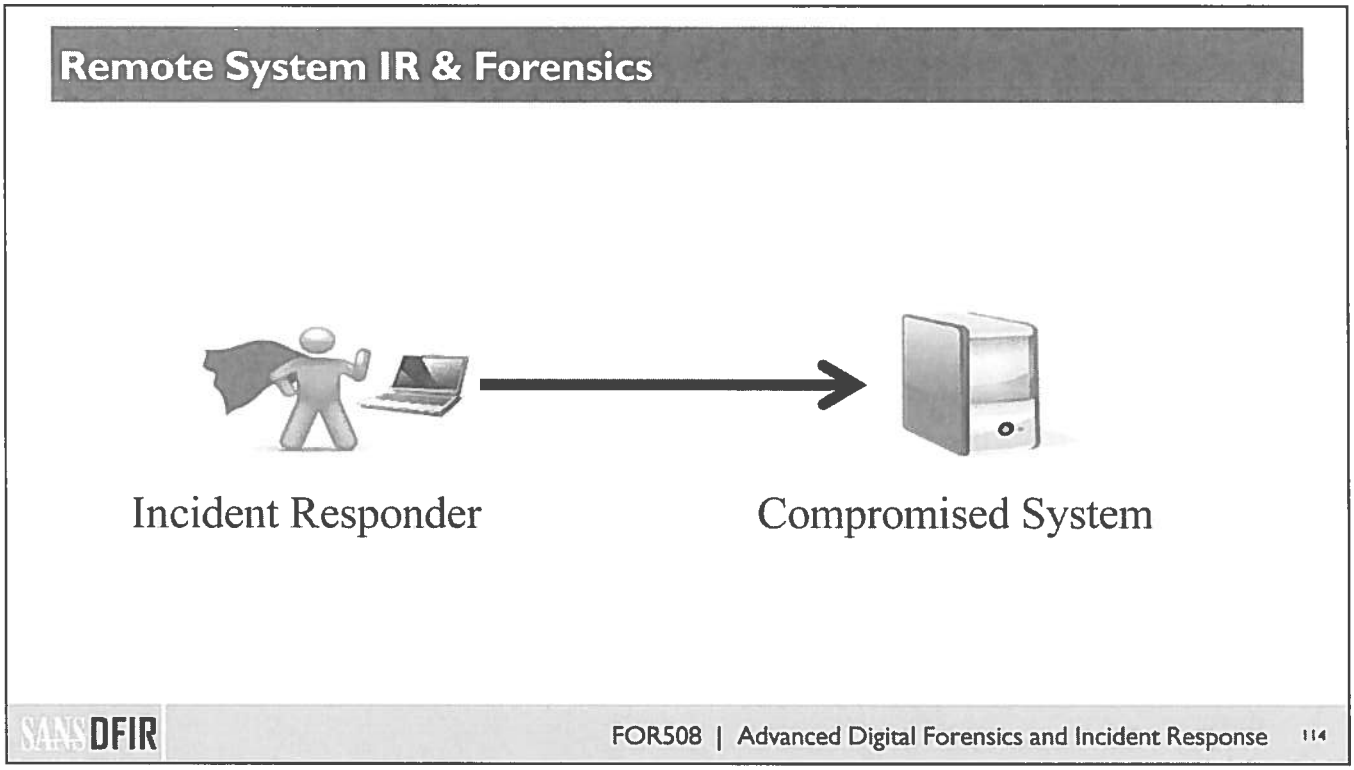
This page intentionally left blank.

Agent Based: Enterprise and Remote System Analysis

"There's only one thing that will make them stop hating you. And that's being so good at what you do that they can't ignore you." —Orson Scott Card, *Ender's Game*

SANS DFIR
DIGITAL FORENSICS & INCIDENT RESPONSE

This page intentionally left blank.



Remote System IR & Forensics

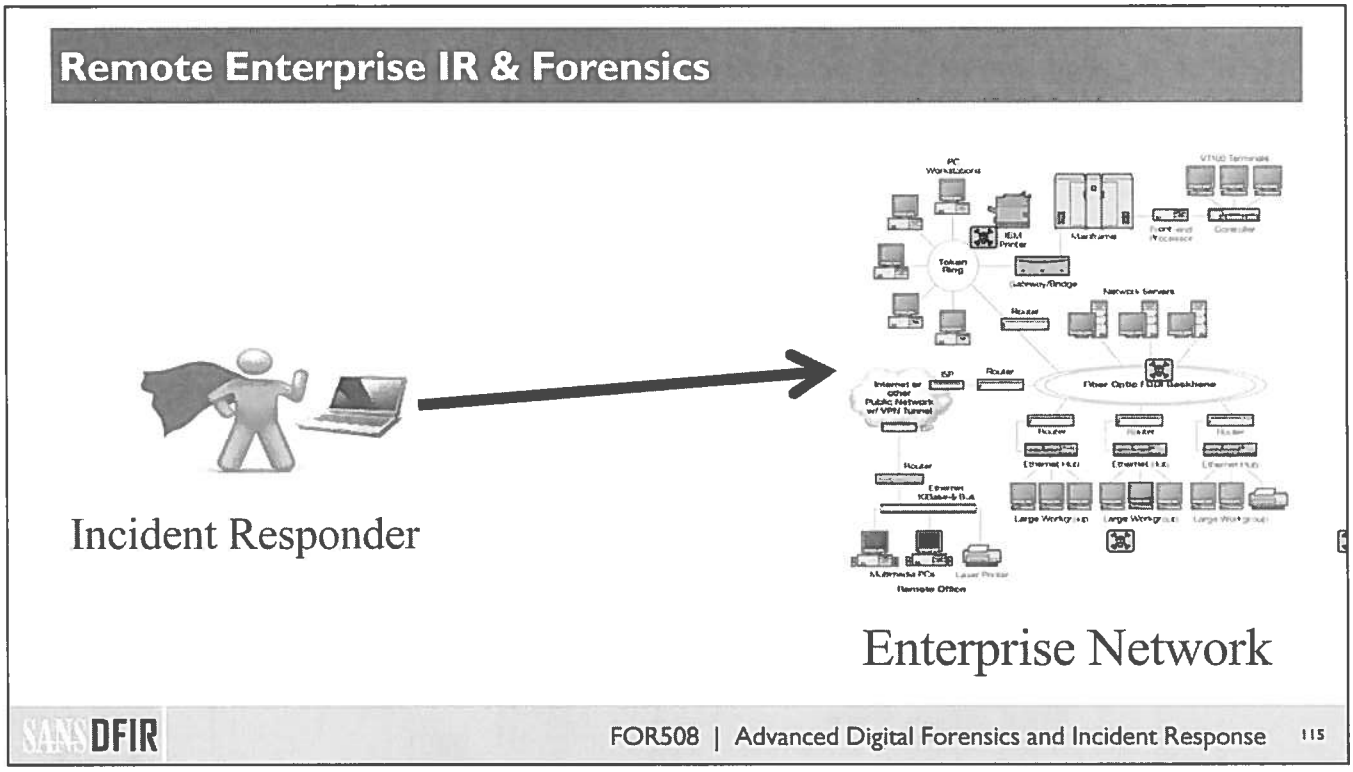
One of the challenges of digital forensics is that you need to be extremely patient because many tasks we perform take a very long time to complete. Imaging for example, could take up to an entire day. A key idea that is now becoming more popular is performing triage by examining live machines for key data before they are pulled offline and then imaged.

Being able to preview remote systems, until recently, has been limited to logging on the machine and using the OS and tools on a CDROM/USB drive to directly interact with the machine. We expanded that capability a bit by introducing a way to execute commands on a remote machine using psexec. However, that is still limiting as a perfect solution and prevents low-level raw access without compromising the integrity of the evidence.

This next section will introduce you to the concept of being able to remotely access the memory and raw disks from your examiner (SIFT Workstation) machine. This will allow you to remotely investigate, recover, and analyze multiple artifacts without compromising the integrity of the evidence or the system. This concept is key because many are moving away from the default policy of "image everything" and moving more toward triage examinations.

How fast would you be able to move if you could easily remotely access 10 machines without imaging them to run a registry examination tool against the SYSTEM hive? Without a remote capability, you would need to export the file and then examine it or image the entire machine and then examine the specific files.

For efficiency purposes, remote forensics is one of the paths forward to help limit the amount of data you must collect off of a machine. Once data is found to be critical, it is the best practice to obtain a copy of the key data—or as a fall back, image the entire hard drive of the system. In either situation, your capabilities as a forensic professional will be greater with new remote access capabilities combined with the SIFT Workstation in this class or using the Win7/8/10 SIFT Workstation from FOR408 - Windows In-Depth.



Remote Enterprise IR & Forensics

Once you understand the unique capabilities of remote forensics, you quickly realize that what you can accomplish against one machine could be utilized across hundreds of systems. A single investigator now has too many systems to potentially analyze in a network. Once you have gained an appreciation of remote forensics, you tend to want to move even further into that capability.

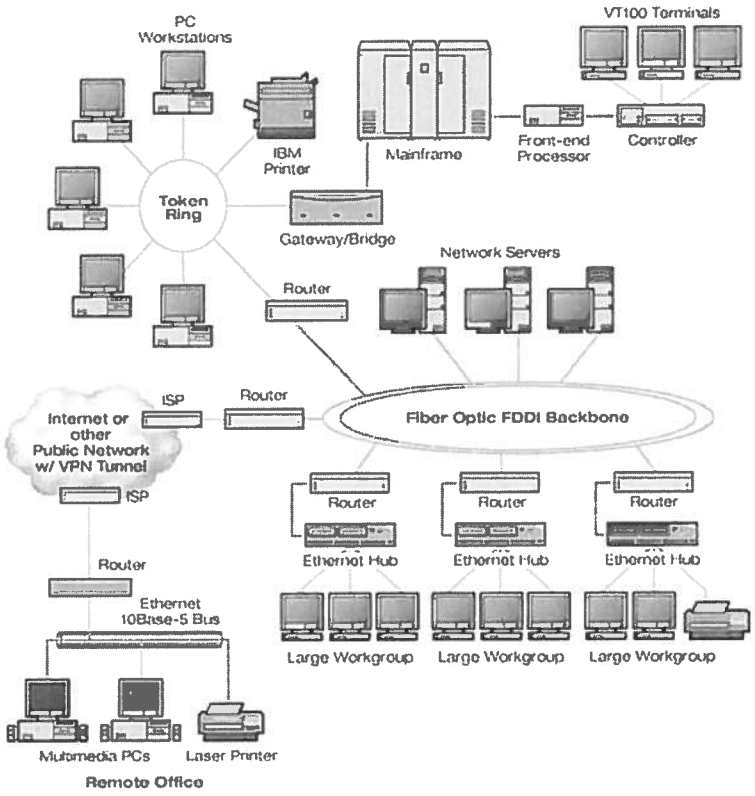
One of the ways this can be accomplished is deploying agents in your infrastructure that have similar remote capabilities. A controller would communicate with these agents. You would tell the controller you are looking for a specific artifact found across the enterprise. The controller would communicate with the agents to begin looking across thousands of machines simultaneously for evidence of that artifact.

Although a single system could be forensicated manually, enterprise forensics leans specifically toward automation. It also leans toward scalability for the investigator. It is not hard to imagine an investigator being able to investigate thousands of machines over a couple of days looking for a unique registry key or keyword on one system. Enterprise forensic tools aim to aid that investigator in just this task.

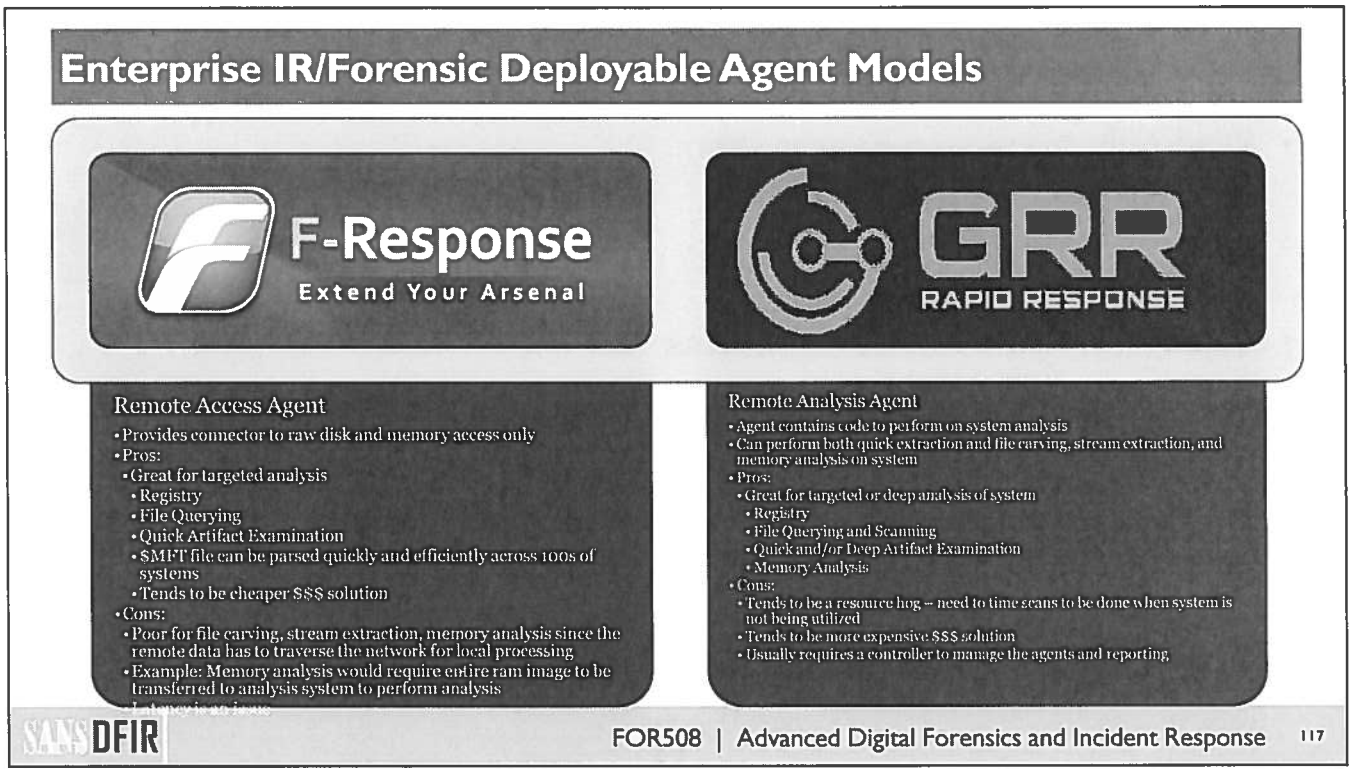
In this next section, we will look at how remote forensics can be accomplished against a single machine, but keep in the back of your mind of how this type of capability could be automated, scripted, and executed across thousands of machines in your enterprise.

The days of imaging every hard drive from each system will soon be over. This is truly the future of digital forensics against end-points in your infrastructure.

Lethal Forensicator



Enterprise Network



Enterprise IR/Forensic Deployable Agent Models

Remote Access Agent

Provides connector to raw disk and memory access only.

Pros:

Great for targeted analysis:

Registry
File querying
Quick artifact examination
Tends to be cheaper $$$ solution

Cons:

Poor for file carving, stream extraction, and memory analysis because the remote data has to traverse the network for local processing.
Example: Memory analysis would require entire ram image to be transferred to analysis system to perform analysis.

Remote Analysis Agent

Agent contains code to perform on system analysis.
Can perform both quick extraction and file carving, stream extraction, and memory analysis on system.
Pros:

Great for targeted or deep analysis of system:

Registry
File querying and scanning
Quick and/or deep artifact examination
Memory analysis

Cons:

Tends to be a resource hog—need to time scans to be done when system is not being utilized.
Tends to be more expensive $$$ solution.
Usually requires a controller to manage the agents and reporting.

F-Response Enterprise

F-Response
Extend Your Arsenal

- Read-only access to remote system:
  - RAID disks
  - Physical drives
  - Logical volumes
  - Physical memory (32 & 64 bit)
- Deployable agent to remote systems
- Does not require a reboot
- Vendor neutral: Works with just about any tool
- Number of simultaneous examiners = Unlimited
- Number of simultaneous agents deployed = Unlimited

- Note: If you register a dongle, you get additional months on your license. Look in your kit for additional details.

F-Response Enterprise

F-Response Enterprise was designed from the ground up to for simplicity of operation. The F-Response Enterprise Management Console (FEMC for short) enables investigators to perform any size network deployment to a virtually limitless number of remote target machines. F-Response Enterprise allows you the examiner to obtain completely vendor neutral, write protected access to remote physical disks, logical volumes, and in some cases physical memory from over ten different remote operating system environments.

F-Response Enterprise also includes access to the F-Response Accelerator, a secondary connectivity tool allowing for an unlimited number of remote examiners as well as optional HIPAA compliant and industry standard AES 256-bit Encryption for connections to almost all supported target platforms.

F-Response Enterprise deployment and connectivity was designed to be covert and efficient allowing the investigator to access multiple machines quickly without concern for alerting the end user.

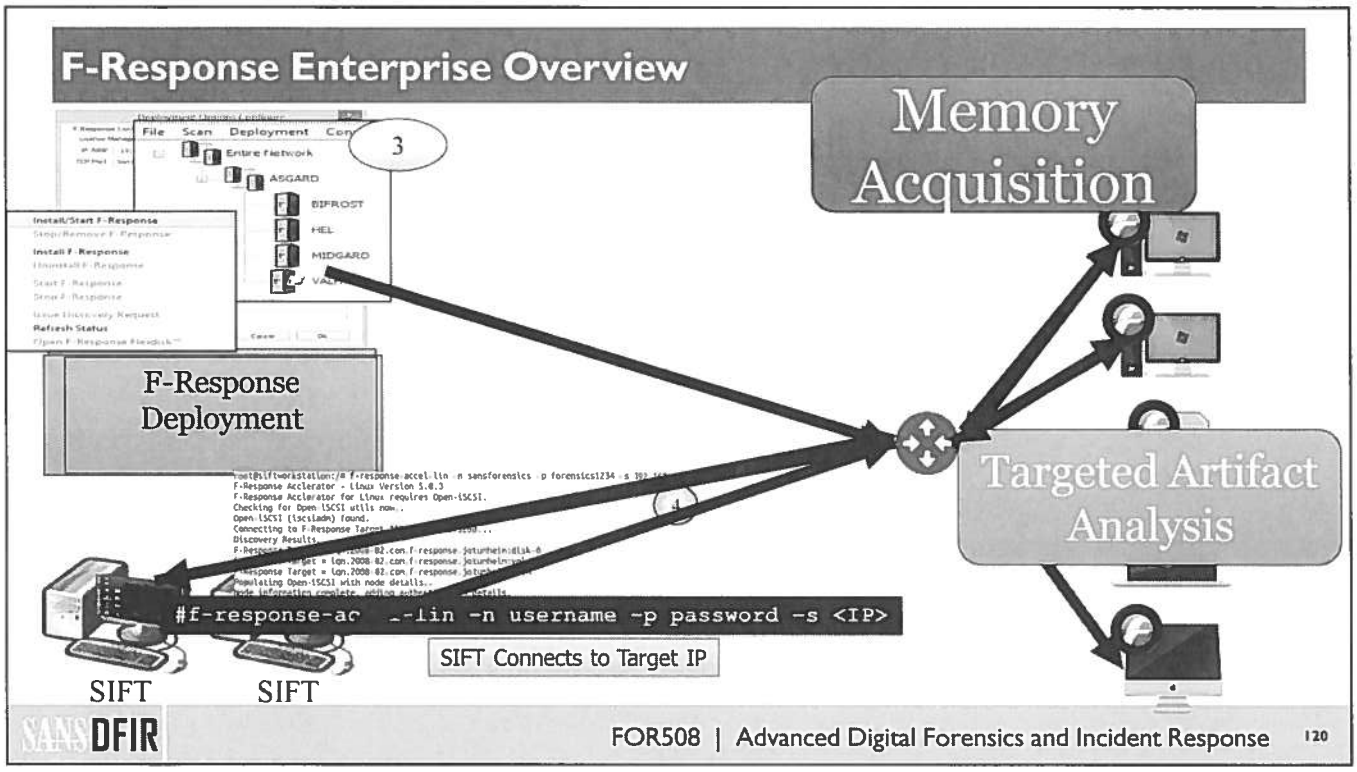
Best of all, F-Response Enterprise is not licensed on a seat basis, one license of F-Response Enterprise provides unlimited client installations, unlimited target connections, and unlimited examiner connections.

F-Response provides direct, live, read-only access to the remote target computer's disks, volumes, and in certain cases physical memory. Because all access is at the physical level, there is no file level locking. F-Response gives you access to any and all content on the remote target, including protected system content (registry files, e-mail PSTs, database files, etc.).

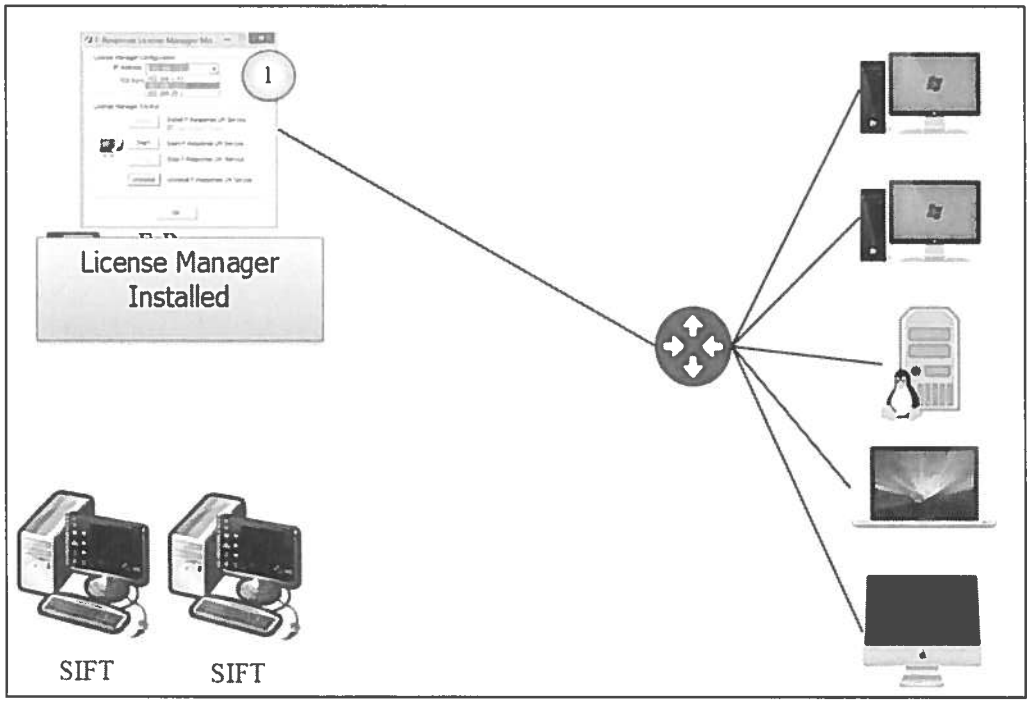
F-Response Enterprise supports the largest array of remote target platforms including the following:

- Windows includes Windows 2000, XP, 2003, Vista, 2008, 7, 8, 10, 2012, 2016 ; physical memory supported only on 32-bit and 64-bit Windows.

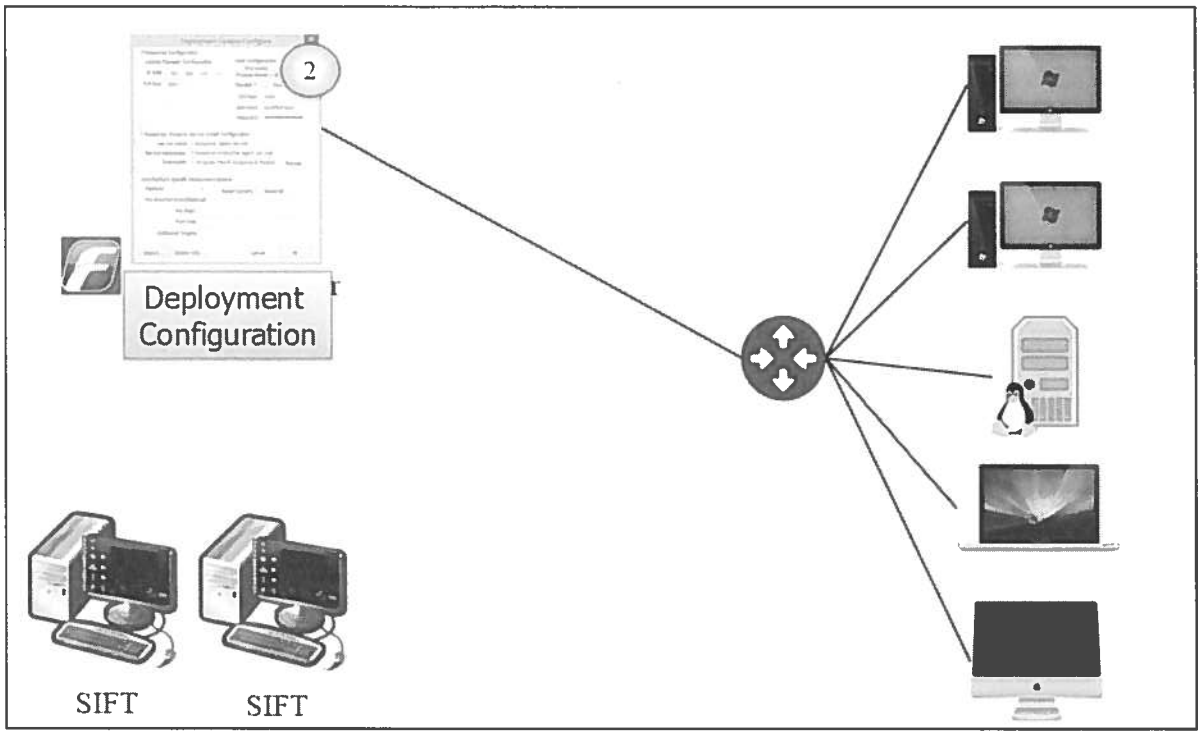
- Apple OSX includes OSX 10.3, 10.4, 10.5, 10.6, 10.7, and 10.8 Universal Binary; FK supports only Intel Apple OSX.
- Linux includes most Linux distributions builds on Glibc 2.3.5 and higher, Android on ARM, and Embedded Linux (Netgear ReadyNAS).
- Solaris includes Solaris 8, 9, and 10 on SPARC and OpenSolaris and Oracle Solaris on Intel.
- IBM AIX includes AIX 5.1, 5.2, 5.3, and 6.1 on the Power processor.
- HPUX includes HP_UX11iv2,11iv3 on the Itanium processor.
- FreeBSD includes FreeBSD 7 on the Intel/i386 processor.
- SCO includes SCO OpenServer 6 and Unixware 7 on the Intel/i386 processor.



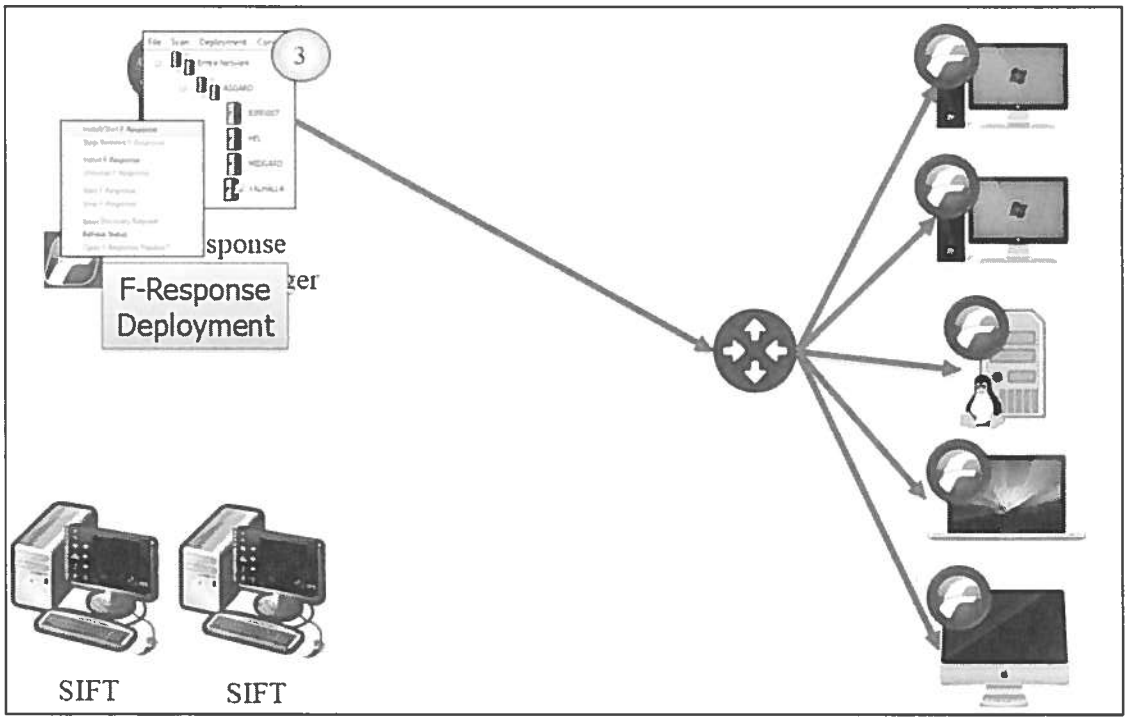
Step 1 – License Manager Installed



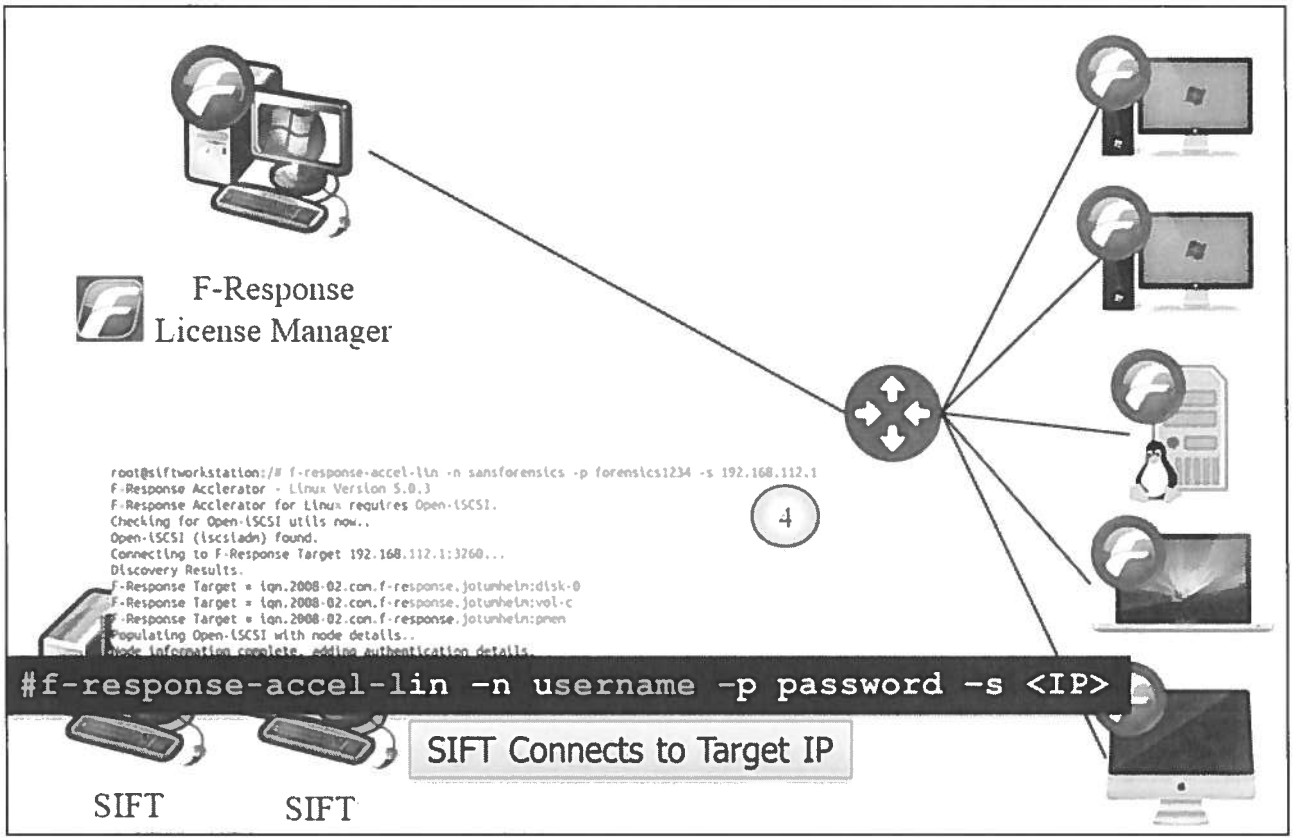
Step 2 – Deployment Configuration



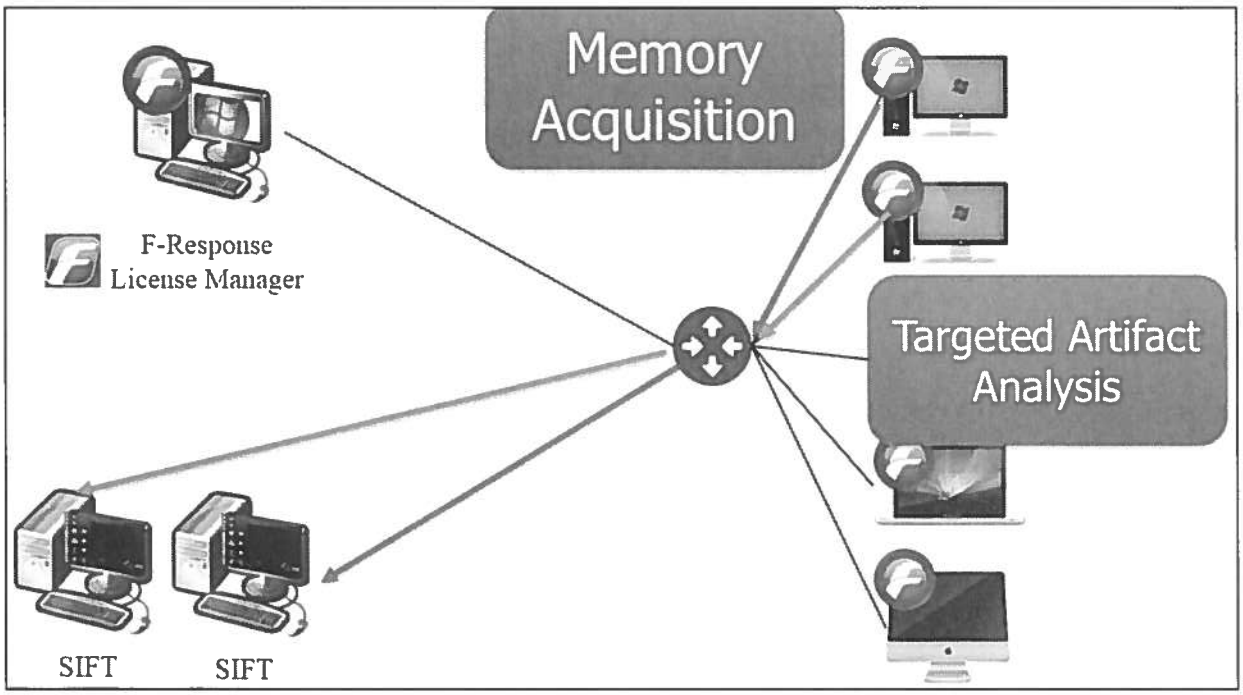
Step 3 – F-Response Deployment

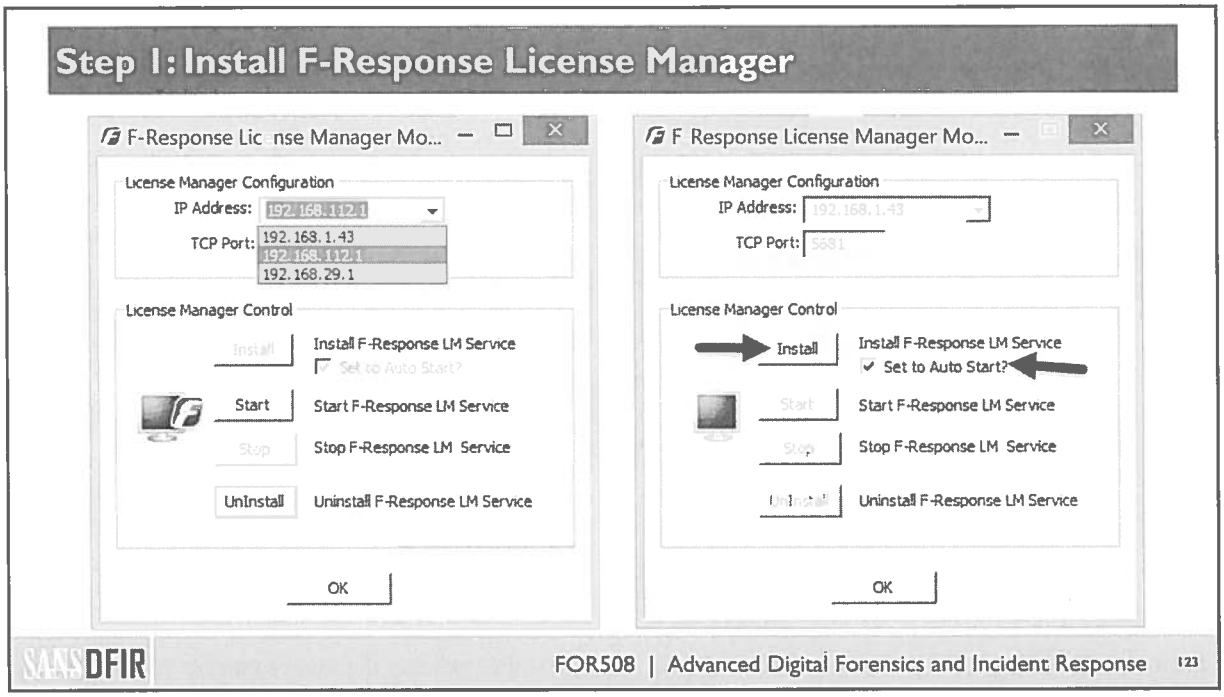


Step 4 – Connect with SIFT Workstation



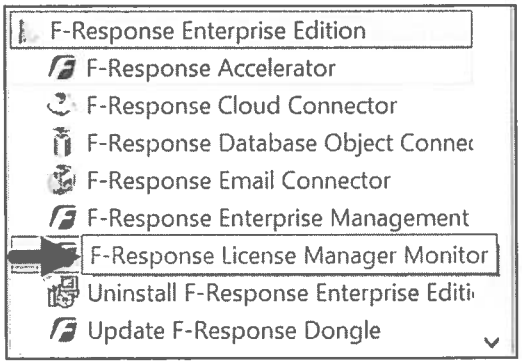
Step 5 – Perform Memory Acquisition or Targeted Artifact Analysis



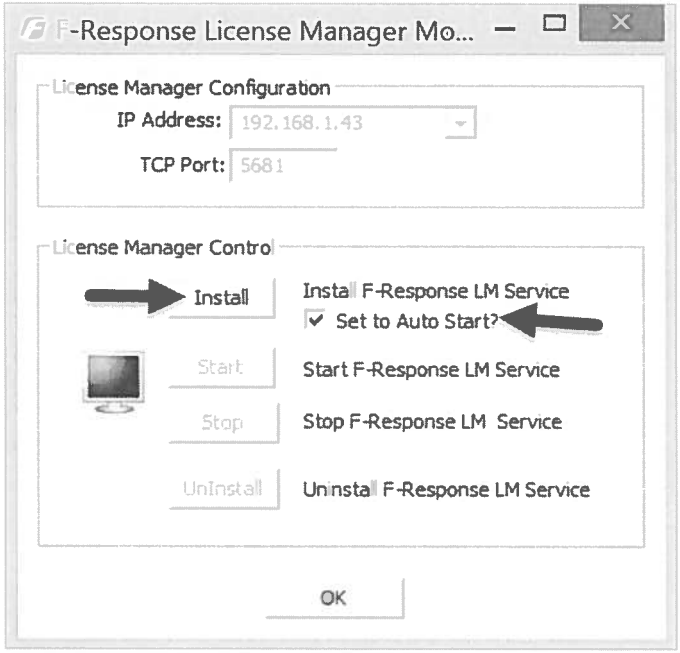


Insert F-Response Enterprise Dongle into a Windows operating system. The F-Response Enterprise software you just installed allows you to create remote agents. It will also serve as your license manager for remote workstations connecting to other system targets in your environment.

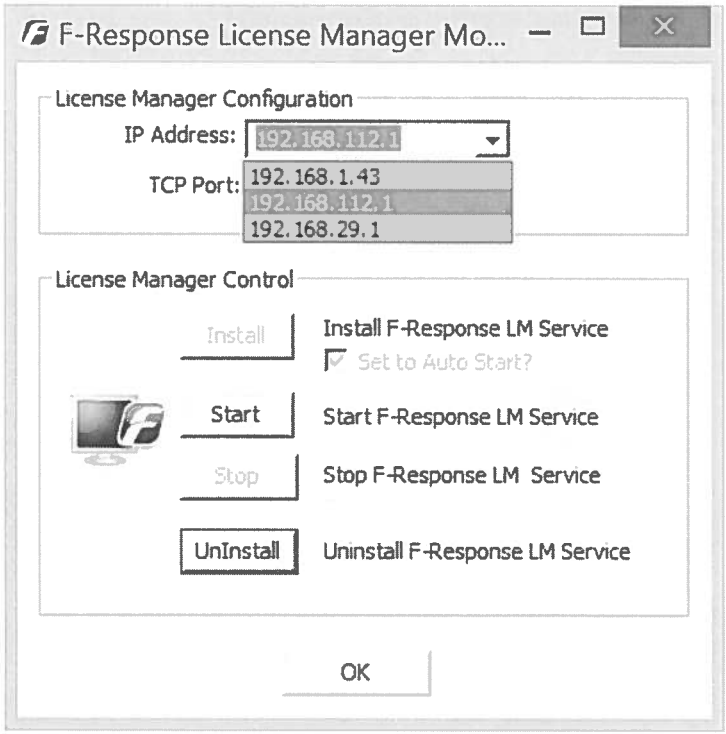
Launch **LICENSE MANAGER MONITOR**.



Install **F-Response LM Service** and check **"Set to Auto Start."**

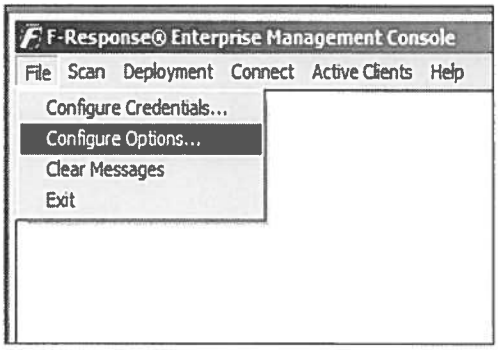


From the "IP Address:" drop-down menu, select the IP Address that matches the same subnet of your SIFT Workstation IP Address documented in step 4 during your Exercise Preparation. NOTE: In the example listed here, our SIFT IP Address was "192.168.112.138," thus the IP Address of the License Manager will need to be on the same subnet "192.168.112.1".



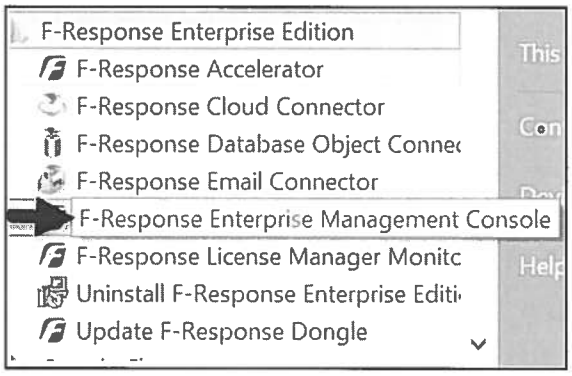
Start the **F-Response LM Service** and select **OK**.

Step 2: Create F-Response Agent

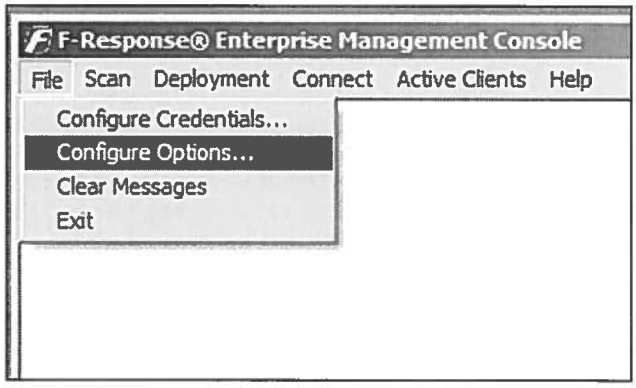


License Manager IP -> **DO NOT CHANGE**
License Manager Port -> **5681 – DO NOT CHANGE**
Host Configuration-> **Physical Memory-> CHECKED**
Username -> **sansforensics**
Password -> **forensics1234**
Service Name -> **F-Response Agent Service**
Service Desc-> **F-Response Enterprise Agent Service**
Executable -> **C:\program files\f-response\f-response enterprise edition\f-response-ent.exe**

Launch **F-Response Enterprise Management Console**.



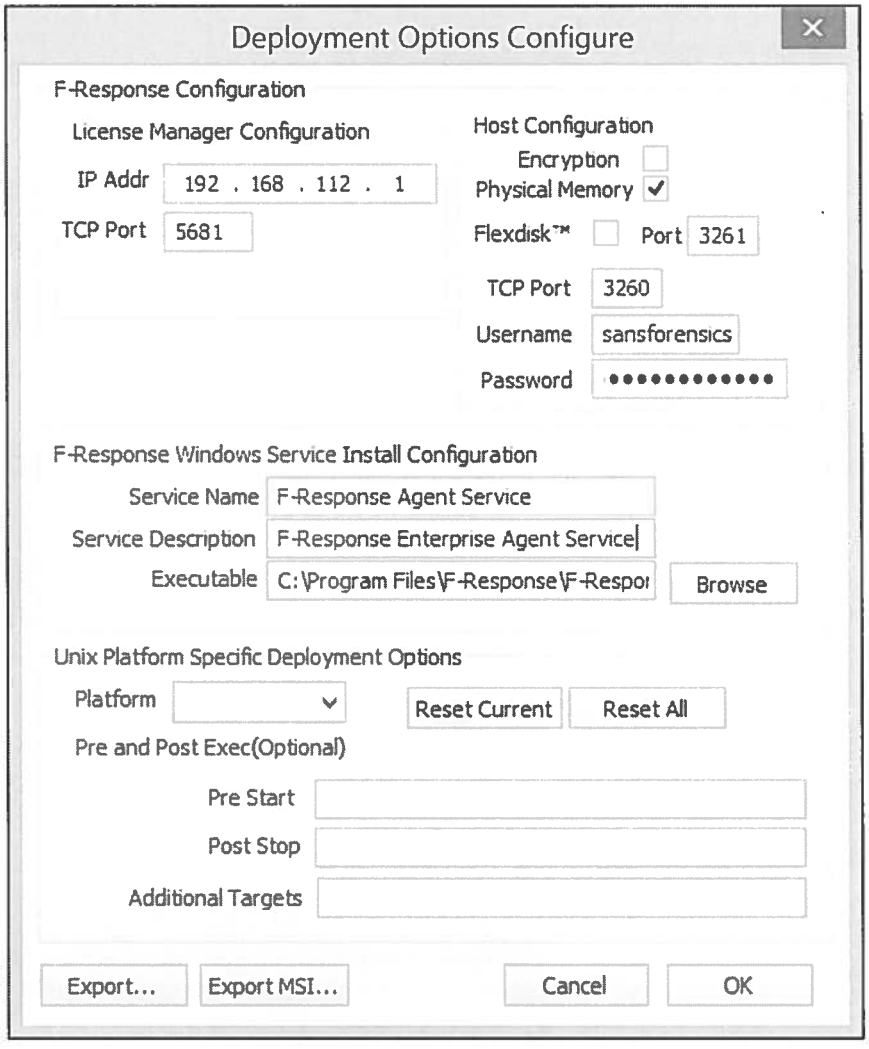
Select **File -> Configure Options**.



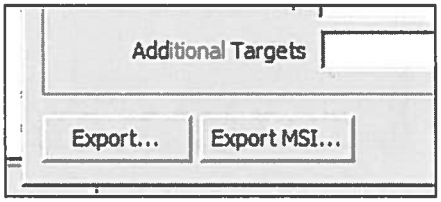
You will need to configure the following:

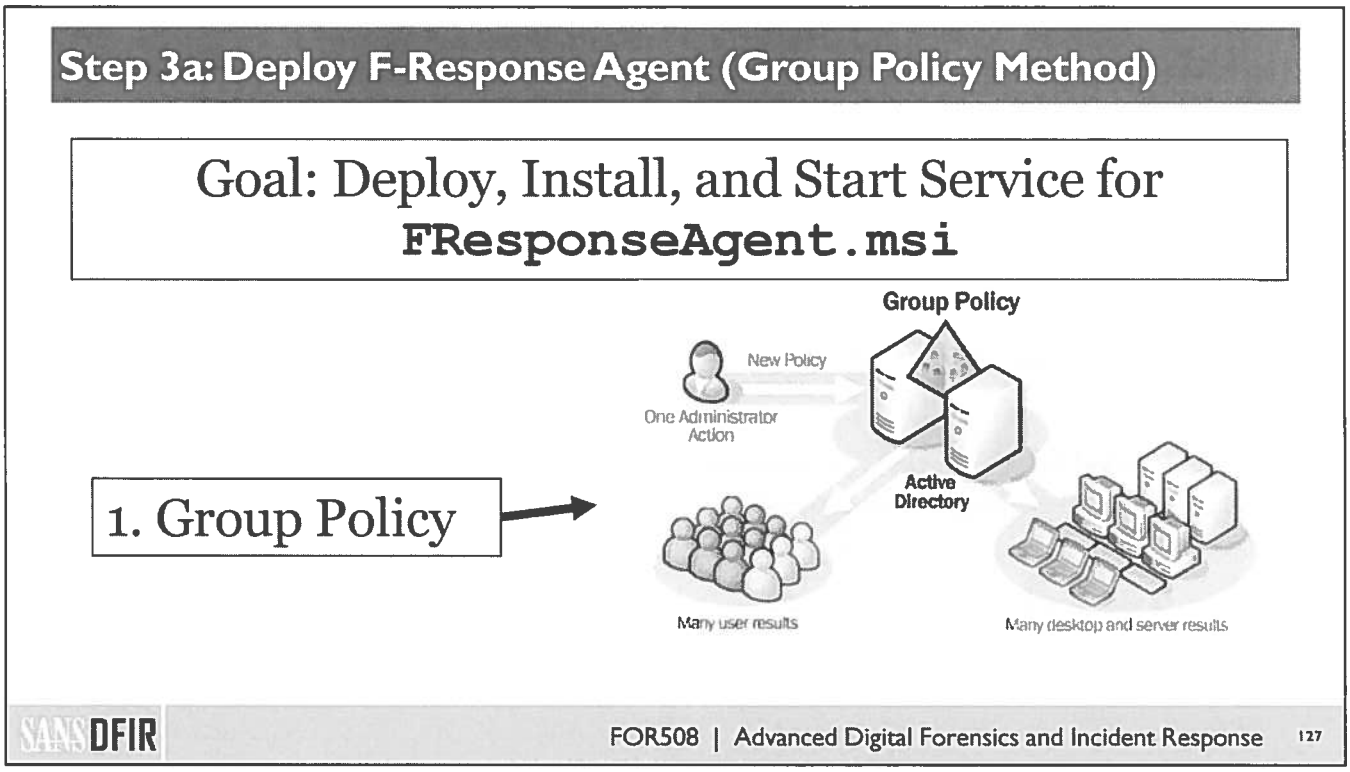
License Manager IP ->	**DO NOT CHANGE**
License Manager Port ->	**5681 – DO NOT CHANGE**
Host Configuration->	**Physical Memory-> CHECKED**
Username ->	**sansforensics** (or choose your own username)
Password ->	**forensics1234** (or choose your own password)
Service Name ->	**F-Response Agent Service**
Service Desc->	**F-Response Enterprise Agent Service**
Executable -> **ent.exe**	**C:\program files\f-response\f-response enterprise edition\f-response-**

Click **OK** to save the changes.



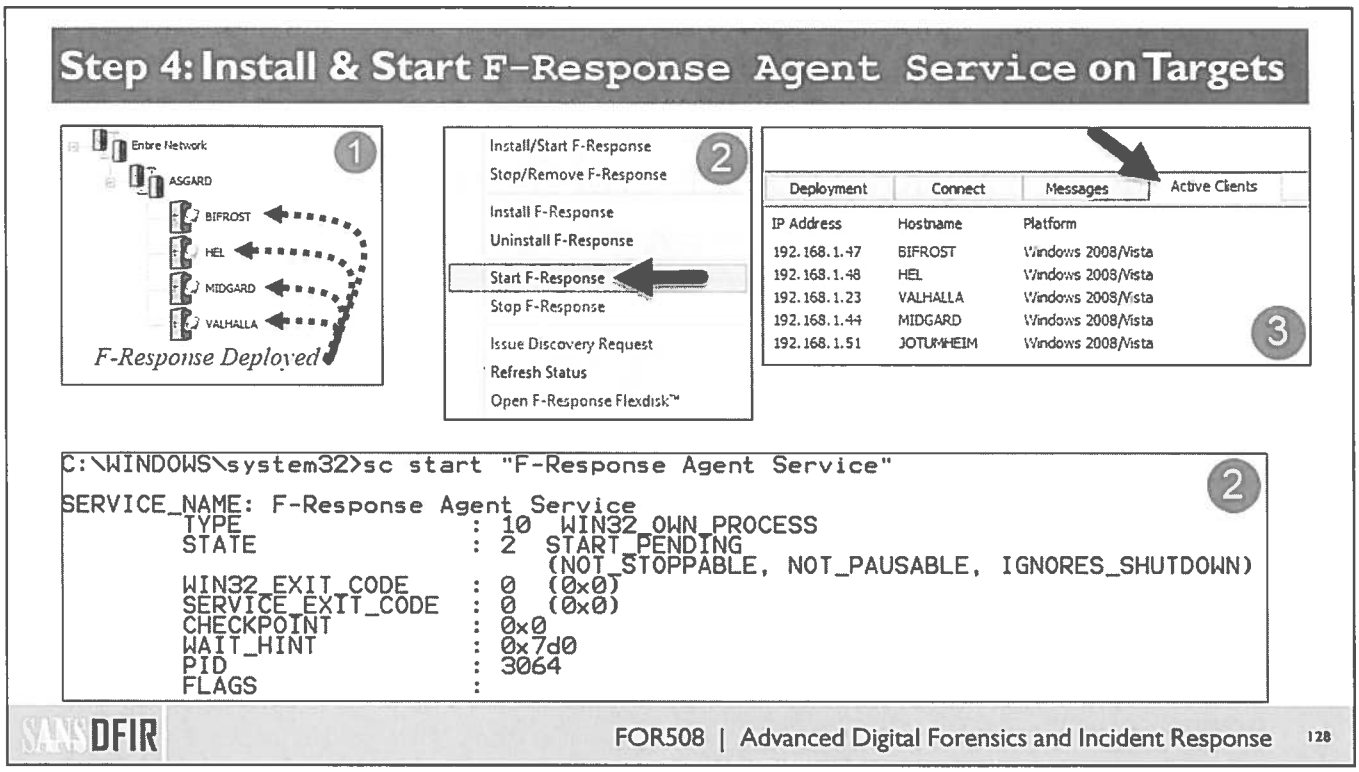
Re-open the **Launch F-Response Enterprise Management Console** dialog box and select **File -> Configure Options**. Export the deployable **F-ResponseAgent.MSI**. Once created, this .msi file can be distributed to thousands of systems in your environment.





Step 3a. Deploy F-Response Agent (Group Policy Method)
This is an example of an enterprise deployment of the collection agent across all systems in your domain. To demonstrate how an IR team could accomplish this, we have two examples: Group Policy and F-Response Management.

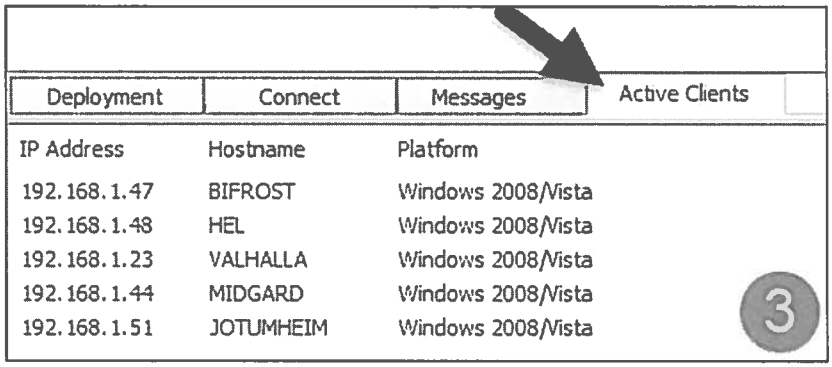
If you have an Active Directory domain, you can use Group Policy to do remote command execution with an "immediate task," which is just a scheduled job that runs a script just once and then deletes the job. You can find this setting in a Group Policy Object (GPO) by opening the Group Policy Management tool in the Administrative Tools folder, editing a GPO, and then navigating inside the GPO to Computer Configuration > Preferences > Control Panel Settings > Scheduled Tasks (right-click **Scheduled Tasks** and select **Immediate Task**). The immediate task should run a script that 1) copies fresponse.msi from a shared folder on a server to the local computer's drive, 2) installs the service 3) executes the following command -> **`sc start "F-Response Agent Service,"`** and 4) deletes fresponse.msi from the local drive.



Step 4: Install & Start F-Response Agent Service on Targets

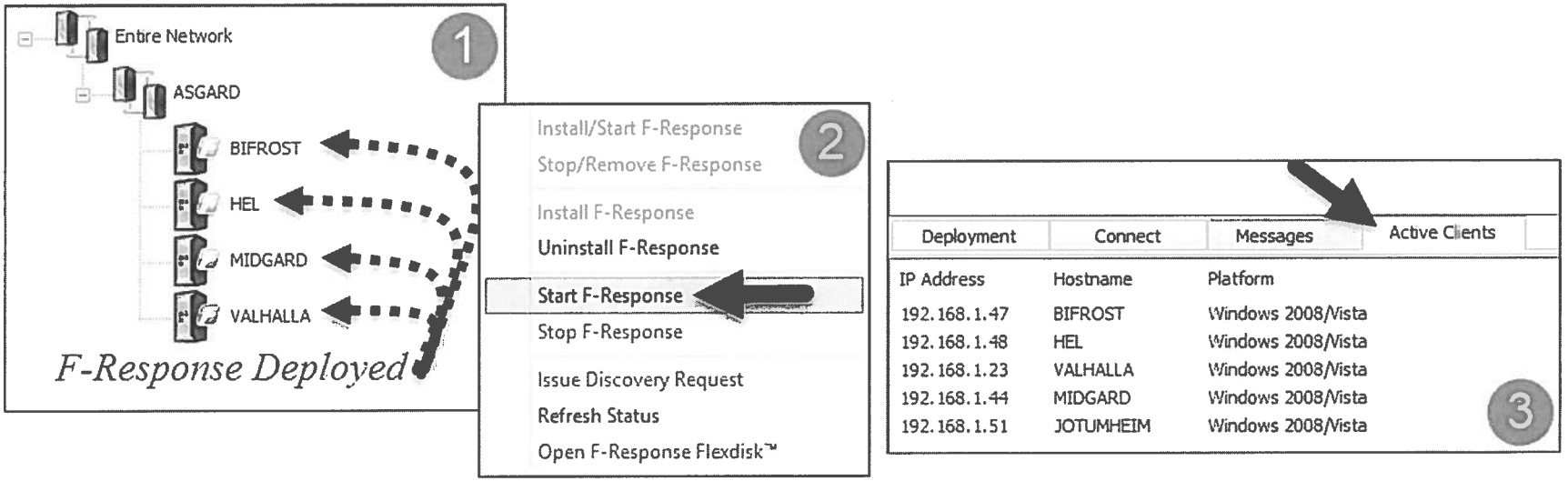
Following a successful scanning/enumeration process, the F-Response Enterprise Management Console can then be used to install, start, stop, and uninstall F-Response Enterprise from accessible computers on the network. The following is a step-by-step progression for using the FEMC to install, start, connect to, disconnect from, stop, and uninstall F-Response Enterprise on remote computers. Installation can also be performed on multiple targets by selecting them in the Deployment panel. Select individual targets or multiple targets and select **Start F-Response** to start the remote F-Response Enterprise service.

Icon badges indicate F-Response has been successfully started on the target computer.



The Active Clients tab will also show more information about the remote F-Response Enterprise targets currently connected to your license dongle, including platform, hostname, and IP address.

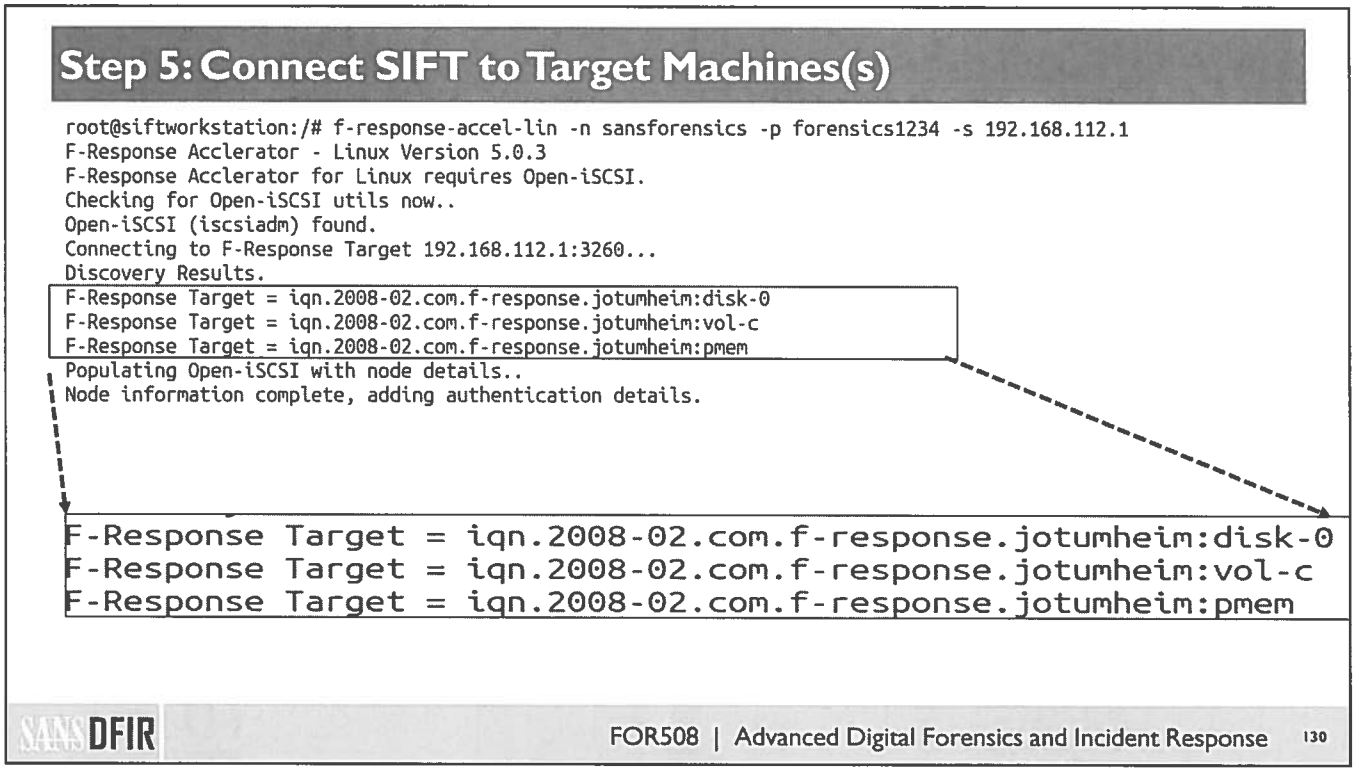
Install & Service on Targets




```
C:\WINDOWS\system32>sc start "F-Response Agent Service"

SERVICE_NAME: F-Response Agent Service
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 3064
        FLAGS              :
```

2



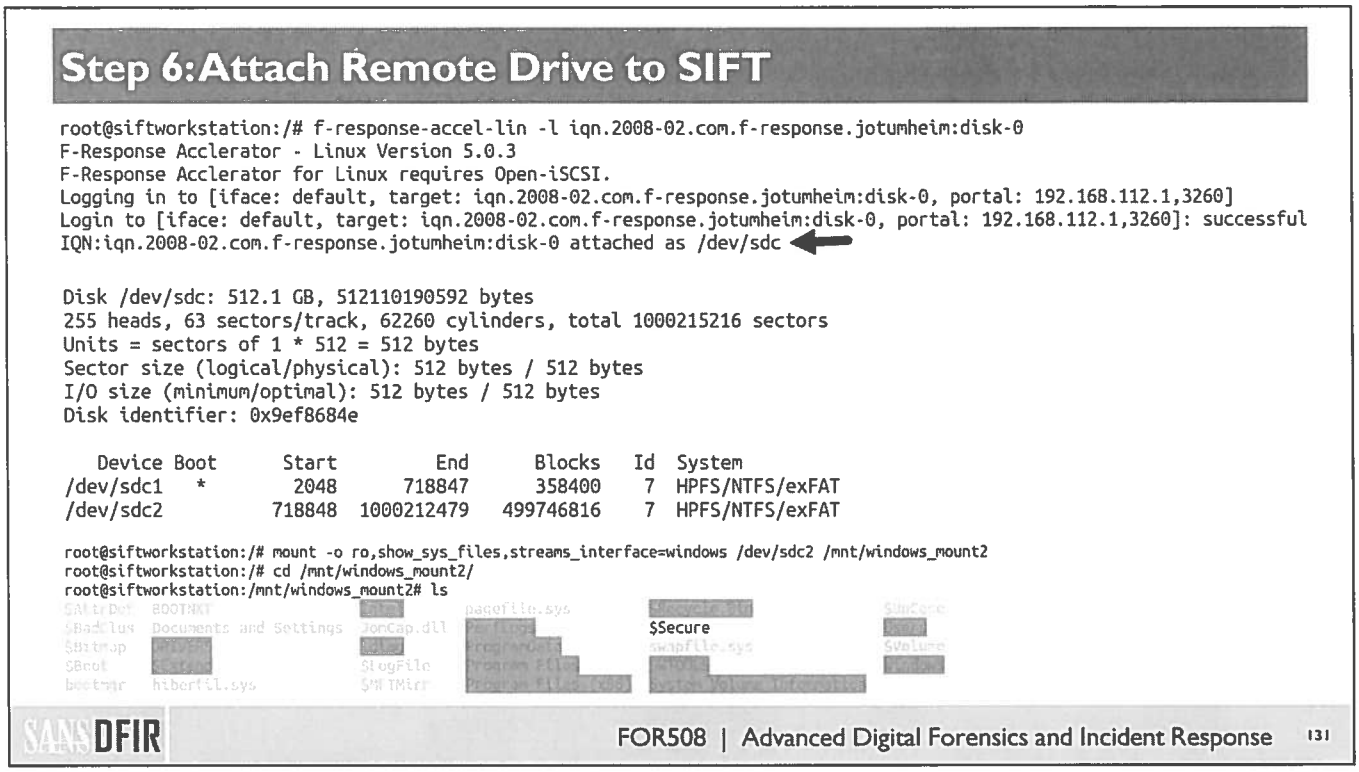
Step 6: Connect SIFT to Target Machines(s)

Switch to your SIFT Workstation.

Log in to see the available nodes.

 # f-response-accel-lin –n sansforensics –p forensics1234 –s 192.168.112.1

```
root@siftworkstation:/# f-response-accel-lin -n sansforensics -p forensics1234 -s 192.168.112.1
F-Response Acclerator - Linux Version 5.0.3
F-Response Acclerator for Linux requires Open-iSCSI.
Checking for Open-iSCSI utils now..
Open-iSCSI (iscsiadm) found.
Connecting to F-Response Target 192.168.112.1:3260...
Discovery Results.
F-Response Target = iqn.2008-02.com.f-response.jotumheim:disk-0
F-Response Target = iqn.2008-02.com.f-response.jotumheim:vol-c
F-Response Target = iqn.2008-02.com.f-response.jotumheim:pmem
Populating Open-iSCSI with node details..
Node information complete, adding authentication details.
```



Step 7: Attach Remote Drive to SIFT

Log in to Remote Hard Drive Target—usually Disk-0. Note in the previous example that my system name was **jotumheim**. The exact name of the system you are logging into will change for each system you are logging into. **Please note that it is a lowercase "L" in the f-response-accel-lin command. ?????? = your system name.**

 # f-response-accel-lin –l iqn.2008-02.com.f-response.??????:disk-0

```
root@siftworkstation:/# f-response-accel-lin -l iqn.2008-02.com.f-response.jotumheim:disk-0
F-Response Acclerator - Linux Version 5.0.3
F-Response Acclerator for Linux requires Open-iSCSI.
Logging in to [iface: default, target: iqn.2008-02.com.f-response.jotumheim:disk-0, portal: 192.168.112.1,3260]
Login to [iface: default, target: iqn.2008-02.com.f-response.jotumheim:disk-0, portal: 192.168.112.1,3260]: successful
IQN:iqn.2008-02.com.f-response.jotumheim:disk-0 attached as /dev/sdc
```

Run **fdisk -l** and examine the output locating the new drive that is attached at /dev/sdc.

 # fdisk –l

```
Disk /dev/sdc: 512.1 GB, 512110190592 bytes
255 heads, 63 sectors/track, 62260 cylinders, total 1000215216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x9ef8684e

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *        2048      718847      358400    7  HPFS/NTFS/exFAT
/dev/sdc2          718848  1000212479   499746816    7  HPFS/NTFS/exFAT
```

Mount a larger partition via the SIFT Workstation using the mount command. (Note: **/dev/sdc2** is what it is on the example; your actual results will vary depending on your system.) You can determine the largest partition by examining the partition with the largest number of blocks. Also, usually on Win7+ and later systems, the second partition is the C:\ of the system.

 # mount –o ro,show_sys_files,streams_interface=windows /dev/sdc2 /mnt/windows_mount2 (note that on your system, /dev/sdc2 could be different)

Change Directories to the **/mnt/windows_mount2** directory and examine files. You should now be able to see the files from the remote system with the F-Response Agent installed.

 # cd /mnt/windows_mount2

 # ls -la

Step 8:Acquire Remote Memory to SIFT

```
root@siftworkstation:/# f-response-accel-lin -l iqn.2008-02.com.f-response.jotumheim:pmem
F-Response Acclerator - Linux Version 5.0.3
F-Response Acclerator for Linux requires Open-iSCSI.
Logging in to [iface: default, target: iqn.2008-02.com.f-response.jotumheim:pmem, portal: 192.168.112.1,3260]
Login to [iface: default, target: iqn.2008-02.com.f-response.jotumheim:pmem, portal: 192.168.112.1,3260]: successful
IQN:iqn.2008-02.com.f-response.jotumheim:pmem attached as /dev/sdd

Disk /dev/sdd: 9636 MB, 9636413440 bytes
64 heads, 32 sectors/track, 1148 cylinders, total 2352640 sectors
Units = sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk identifier: 0x00000000

Disk /dev/sdd doesn't contain a valid partition table
root@siftworkstation:/# dc3dd if=/dev/sdd of=/cases/remote-system-memory.img hash=md5 hlog=/cases/remote-system-memory.md5

dc3dd 7.1.614 started at 2014-05-23 17:00:48 +0100
compiled options:
command line: dc3dd if=/dev/sdd of=/cases/remote-system-memory.img hash=md5 hlog=/cases/remote-system-memory.md5
device size: 2352640 sectors (probed)
sector size: 4096 bytes (probed)
392626176 bytes (374 M) copied ( 4%), 10.0541 s, 37 M/s
```


Step 8: Acquire Remote Memory to SIFT

Log in to Appropriate Target—usually PMEM. **Please note that it is a lowercase "L" in the f-response-accel-lin command. ?????? = your target system name.**

Example (**Memory**) = **iqn.2008-02.com.f-response.?????:pmem**

 # f-response-accel-lin –l iqn.2008-02.com.f-response.?????????:pmem

```
root@siftworkstation:/# f-response-accel-lin -l iqn.2008-02.com.f-response.jotumheim:pmem
F-Response Acclerator - Linux Version 5.0.3
F-Response Acclerator for Linux requires Open-iSCSI.
Logging in to [iface: default, target: iqn.2008-02.com.f-response.jotumheim:pmem, portal: 192.168.112.1,3260]
Login to [iface: default, target: iqn.2008-02.com.f-response.jotumheim:pmem, portal: 192.168.112.1,3260]: successful
IQN:iqn.2008-02.com.f-response.jotumheim:pmem attached as /dev/sdd
```

Run **fdisk -l** and examine output locating the memory image. It should not show a valid partition table at **/dev/sdd.**

 # fdisk –l

```
Disk /dev/sdd: 9636 MB, 9636413440 bytes
64 heads, 32 sectors/track, 1148 cylinders, total 2352640 sectors
Units = sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk identifier: 0x00000000

Disk /dev/sdd doesn't contain a valid partition table
```

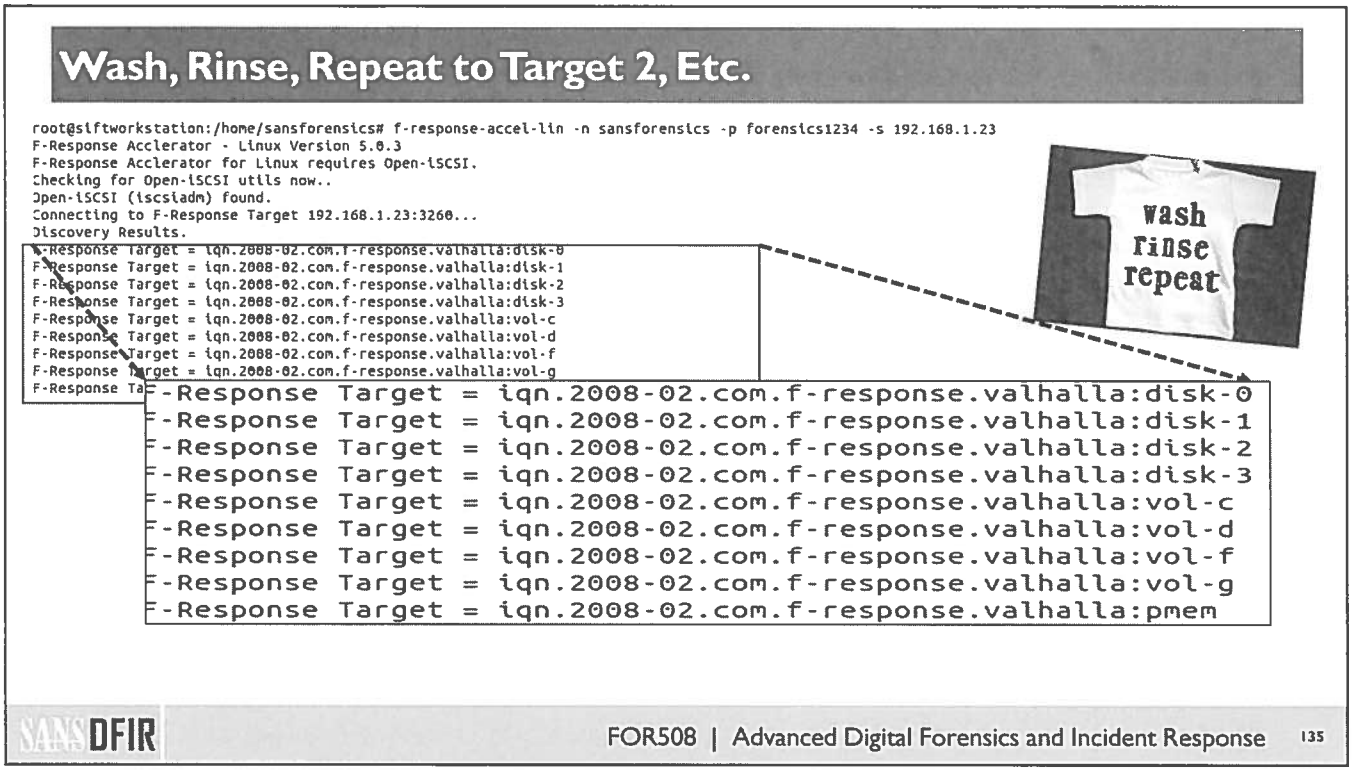
Image remote system memory using dc3dd:

Example (**Memory**) = **iqn.2008-02.com.f-response.?????:pmem = /dev/sdd** (note on your system, /dev/sdd could be different)

 # dc3dd if=/dev/sdd of=/cases/remote-system-memory.img hash=md5 hlog=/cases/remote-system-memory.md5

```
root@siftworkstation:/# dc3dd if=/dev/sdd of=/cases/remote-system-memory.img hash=md5 hlog=/cases/remote-system-memory.md5

dc3dd 7.1.614 started at 2014-05-23 17:00:48 +0100
compiled options:
command line: dc3dd if=/dev/sdd of=/cases/remote-system-memory.img hash=md5 hlog=/cases/remote-system-memory.md5
device size: 2352640 sectors (probed)
sector size: 4096 bytes (probed)
392626176 bytes (374 M) copied ( 4%), 10.0541 s, 37 M/s
```



Wash, Rinse, Repeat to Target 2, Etc.

Once you have completed the analysis of target 1, you can move to target 2.

The key to F-Response Enterprise and SIFT is understanding that you now use it for scripting specific analysis techniques across 100s of systems in your environment automatically. The true benefit to F-Response and SIFT is that you have a capability that can allow you remote access via hundreds of smaller tools that are available via the SIFT command-line interface.

The following are examples of malware detection artifacts to look for using F-Response and SIFT across 100s of systems.

Malware does not need to be present on a system for it to be compromised. We need to also look for unusual OS-based artifacts that would not exist on a typical workstation in the organization. When looking for program execution, focus on prefetch, shimcache, userassist registry keys, and even jump lists. Many of these artifacts can result from an adversary using your system but not implanting malware. Look for evidence showing odd behavior such as tools being run outside the scope of non-technical or normal user activity:

- **cmd.exe** execution: Provides command-line access.
- **rar.exe** execution or presence of .rar files: Difficult to crackarchiving tool for data exfiltration.
- **at.exe** or **schtasks.exe** execution: Used for privilege escalation and persistence
- Existence of Sysinternals tools such as **PsExec**, **PsLoggedOn**, and **ProcDump**: Provide remote execution, interactive logon enumeration, and dumping of credentials within **lsass.exe** address space, respectively.
- **wmic.exe**, **powershell.exe**, or **winrm.vbs** execution: Used for remote execution.
- **net.exe** execution: Used for mapping drives for lateral movement and enumerating groups like "Domain Admins."
- **reg.exe** or **sc.exe** execution: Adds persistence such as Run keys or services.
- **MountPoints2** registry key: Records shares on remote systems such C$, Temp$, etc.

Optional - Exercise 1.5a OR 1.5b

Enterprise Response, Hunting, & Forensics Using F-Response

This page intentionally left blank.

Post-AutoStart Analysis & Agent Deployment: Breach Status

Known Hosts Compromised

Name	IP	Function
nromanoff	10.3.58.5	Workstation

Initial IRT Call & Agent deployment

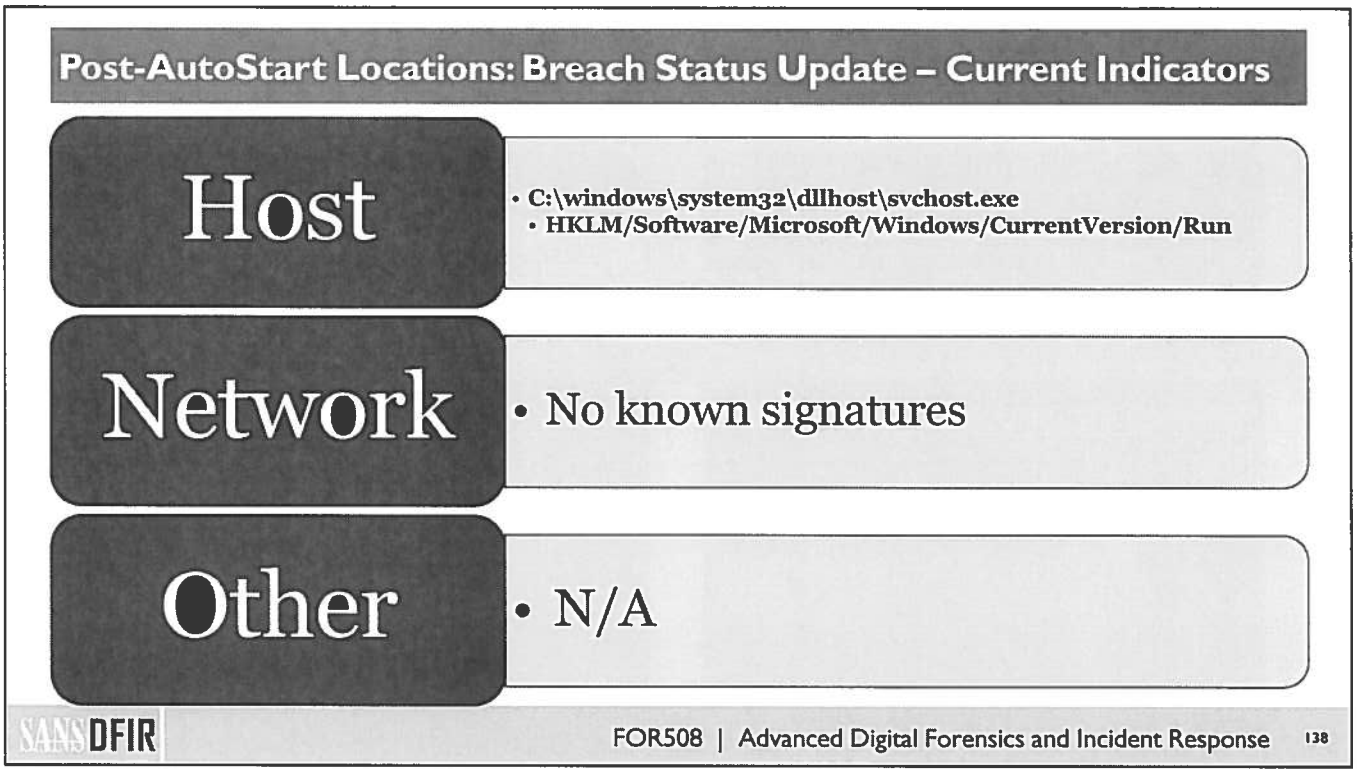
- Access Host
  - Memory
  - C-Drive
  - AutoStart Locations Examination
- Time: ~60 min

Current Spreadsheet o' Doom

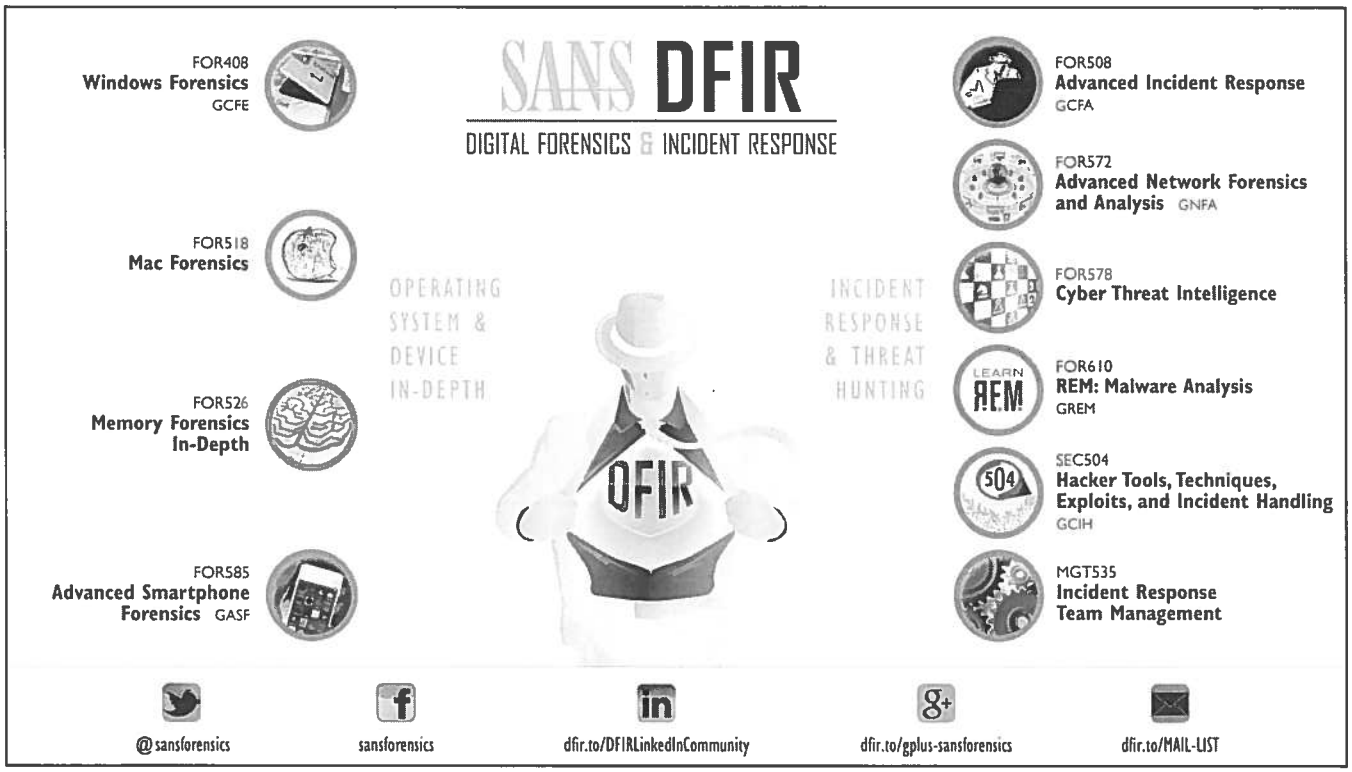
- 10.3.58.5 – nromanoff

Incident Response Total Time Elapsed: ~60 Min

This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

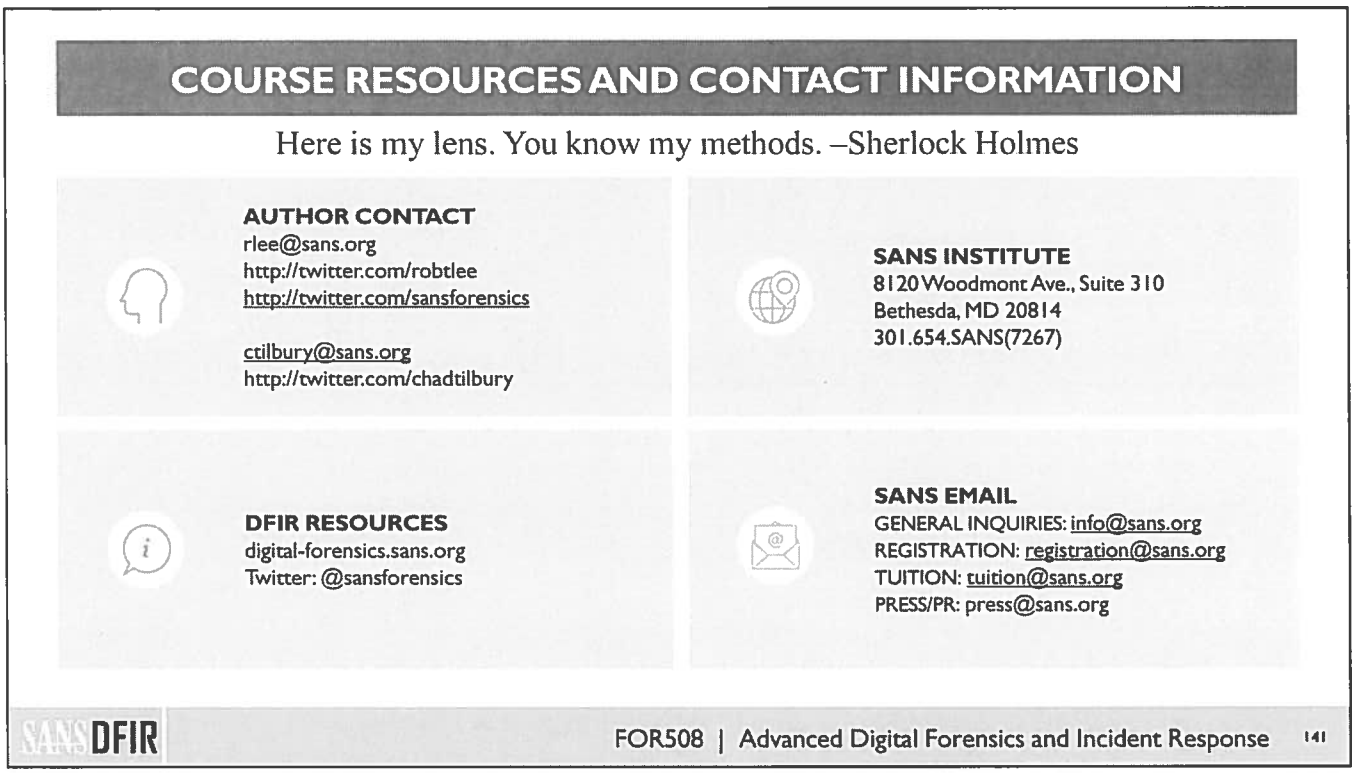
OPERATING SYSTEM & DEVICE IN-DEPTH

- FOR408 **Windows Forensics**
- FOR518 **Mac Forensics**
- FOR526 **Memory Forensics In-Depth**
- FOR585 **Advanced Smartphone Forensics**

INCIDENT RESPONSE & ADVERSARY HUNTING

- FOR508 **Advanced Incident Response**
- FOR572 **Advanced Network Forensics and Analysis**
- FOR578 **Cyber Threat Intelligence**
- FOR610 **REM: Malware Analysis**
- SEC504 **Hacker Tools, Techniques, Exploits, and Incident Handling**
- MGT535 **Incident Response Team Management**

@sansforensics | sansforensics | dfir.to/DFIRLinkedInCommunity | dfir.to/gplus-sansforensics | dfir.to/MAIL-LIST



This page intentionally left blank.