

450.2

Understanding Your Network

To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



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 SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, 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 SANS Institute to 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 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 SANS INSTITUTE, AND THAT 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 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 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 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.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

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

SEC450.2

Blue Team Fundamentals: Security Operations and Analysis

SANS

Understanding Your Network

© 2020 John Hubbard | All Rights Reserved | F01_02

Welcome to SANS Security 450.2: Understanding Your Network

TABLE OF CONTENTS	PAGE
Network Architecture	05
Traffic Capture and Analysis	22
Understanding DNS	35
DNS Analysis and Attacks	58
EXERCISE 2.1: Exploring DNS	90
Understanding HTTP(S)	92
HTTP Analysis and Attacks	117
EXERCISE 2.2: HTTP and HTTPS Analysis	147
Understanding SMTP and Email	149
Additional Network Protocols	171
Day 2 Summary	184
EXERCISE 2.3: SMTP and Email Analysis	186

450.2 Table of Contents

This table of contents outlines the plan for 450.2.

Course Outline

Day 1: Blue Team Tools and Operations

Day 2: Understanding Your Network

Day 3: Understanding Endpoints, Logs, and Files

Day 4: Triage and Analysis

Day 5: Continuous Improvement, Analytics, and Automation

This page intentionally left blank.

Day 2 Overview

Day 2: Network monitoring concepts and attack detection

- How enterprise networks are architected
- Network monitoring methods and data types
- Use and abuse of the most important protocols
 - DNS
 - HTTP/HTTPS
 - SMTP
 - SMB, ICMP, DHCP, FTP, SSH, PowerShell Remoting

Day 2 Overview

In this book, we will start out by taking a defender's view of our network and examining how a modern network is architected and monitored as well as the protocols that are most commonly used and abused by attackers.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

Network Architecture Intro

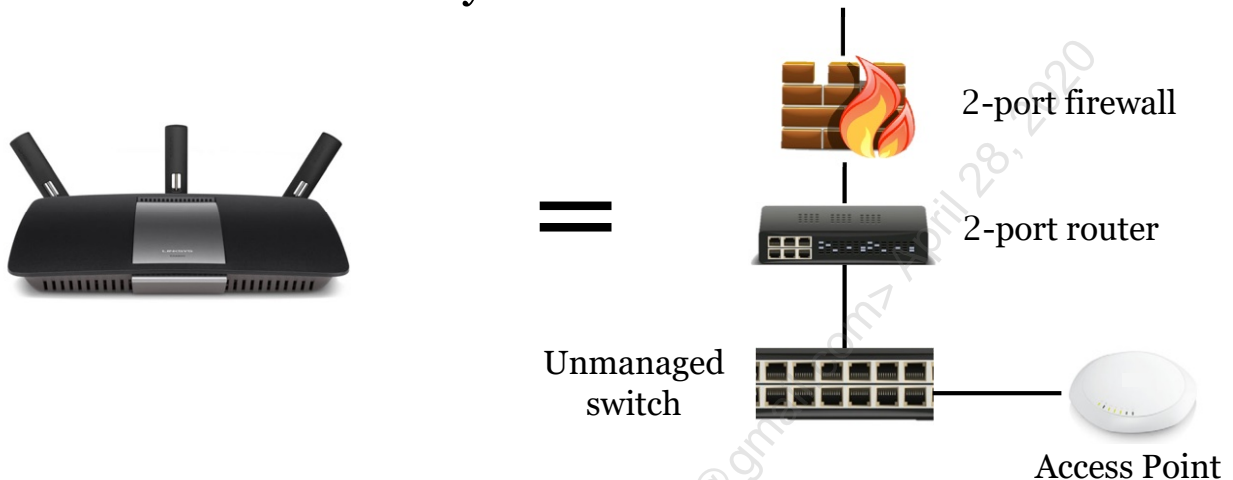
- Understanding our networks: Required for catching evil
- **Traffic flow** and **visibility points** are crucial information for the blue team
- To understand traffic visibility and flow, we must
 - Understand **security zones**
 - How to implement them with **access control / firewalling**
 - Understand how **routers, switches, and firewalls** assist us with visibility and traffic control
- We'll start from home networks
 - Will build up from there to understand the differences

Network Architecture Intro

In order to have a clear understanding of where we can see traffic traversing the network, it is important to understand the way a modern network is architected. Modern networks are generally split up into at least a few zones consisting of rules that will restrict or allow traffic to flow between them based on routing and firewall rules. Understanding these rules, the physical connections that govern the zones, and the points of visibility available on your organization's network is crucial for the defender. Without it, it is impossible to derive where traffic we are interested in can be captured and inspected.

Home Router/Switch Combo: Deconstructed

What's really inside a home router?



Home Router/Switch Combo: Deconstructed

Let's take your home router. It is a single piece of physical equipment, but what would happen if we were to explode it into its functional pieces? You would find something very different indeed. A home router performs several functions:

- Firewalling traffic from the outside from getting into your home connection
- Acts as a Network Address Translation (NAT) router to control whether traffic stays within your local network or goes to the internet
- Contains multiple ports to act as a switch
- Has an embedded access point to allow clients to wirelessly connect as though they were plugged into the switch

A corporate network is not, in fact, all that different from your home network in terms of functions. There are still switches, routers, firewalls, and wireless access points. What IS different, however, is that each of these items is split into a separate physical device, and each one of them likely does a more complex job. We'll take a look at each of these pieces and talk about how they are meaningful to us as a Blue Team that wants to separate the network into segments, as well as monitor traffic between those segments.

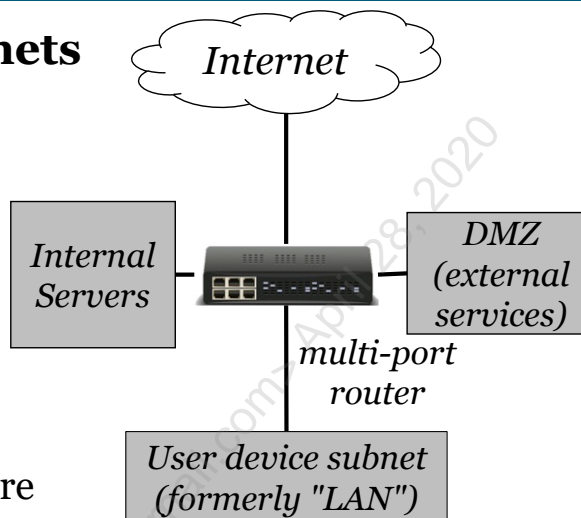
Enterprise Routers

Divides the network into **subnets**

- Likely multiple routers in use

Provide security features:

- **Network flow visibility**
- ACLs for traffic flow
- 802.1X
- VPN and IPsec connections
- Content filtering, IPS and more



Enterprise Routers

Let's discuss the router portion of the network first. The first and most obvious distinction between the home and corporate router is the number of ports available. While the average home user only needs a 2-port router (a LAN side, and WAN side for the internet connection), business networks will need much more flexibility. This means that the router in your business network will be doing the job of not only dividing the internet from an inside LAN but will be breaking the "inside" network up into many different pieces. In corporate networks there will almost always be more than one router as well—border routers, internal routers, and beyond. Large, complex networks will have multiple sections, and these additional routers provide additional divisions, connectivity, and redundancy to keep the network up in the case of a device outage. You are likely familiar with the subnets that are commonly made—a DMZ, an internal server subnet, and as many others as your organization has a need for. As a Blue Team, we are interested in these divisions because they often group machines of similar purpose and are paired with firewall rules that restrict traffic between them.

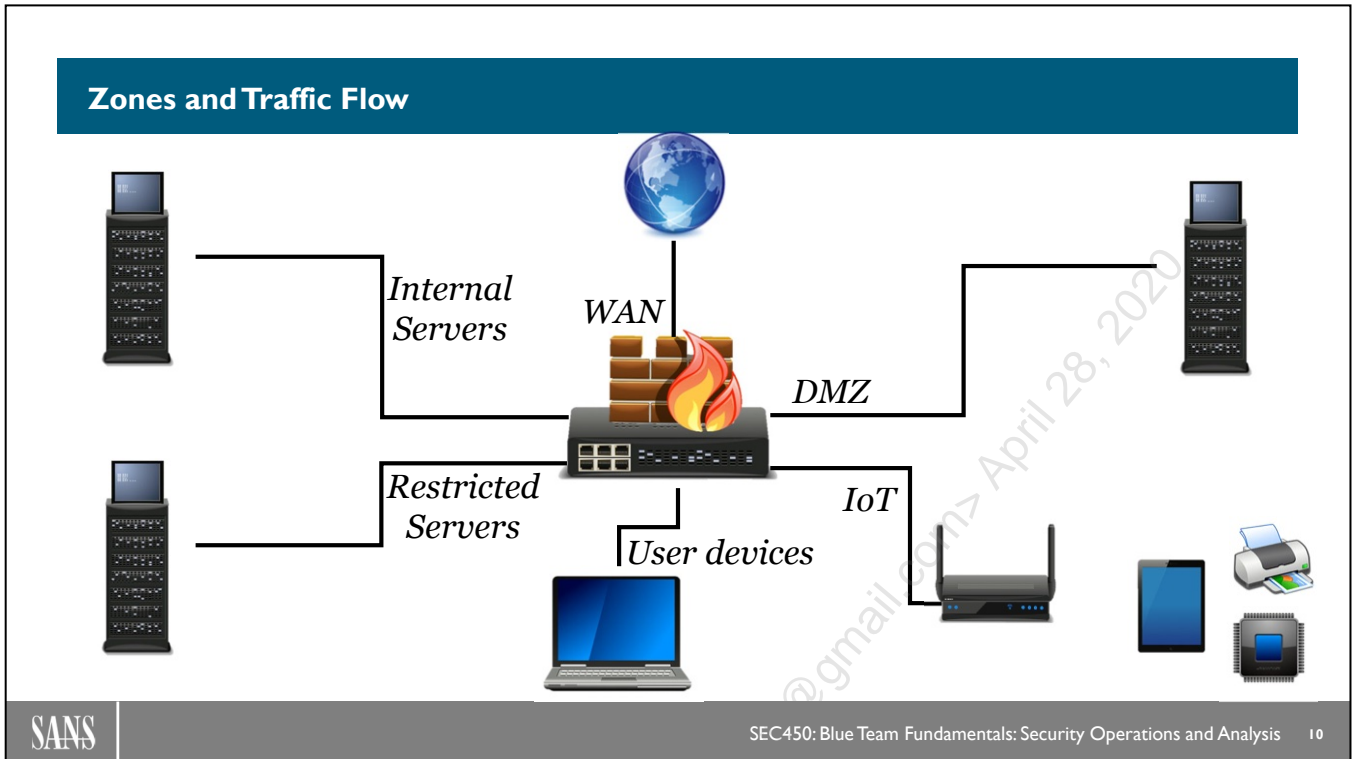
The second difference is that in your corporate network, although a router is not primarily a security device, your router is likely utilizing features to help restrict traffic flow as well as provide visibility for traffic flowing through it. NetFlow, for example, is a protocol Cisco router use to give administrators or the security team visibility into the traffic passing through the router. Access control lists (ACLs) allow router admins to blackhole traffic, or restrict it by source, destination, port, and more. Some routers can even connect remote networks through VPN connections, use 802.1x to authenticate endpoints, provide content filtering, intrusion prevention, and beyond.

Network Zones

- Corporate networks are typically segmented into zones
- Zones enforce security boundaries, group similar systems
- Typical zones types could include:
 - External
 - DMZ
 - Servers
 - Desktops
 - IoT
 - Guest access
 - Business-to-Business / VPN links
 - Cloud
 - Restricted/Sensitive
 - Air-gapped, high-risk networks
- Each subnet has traffic flow rules enforced by firewall / ACL

Network Zones

Network zones are designed to break up a network into multiple sections containing systems of similar types. Zones that you will likely hear discussed will range from the internet zone, which is everything outside the network, down to subnets where desktops and laptops sit, one for IoT devices, and others for DMZs, Cloud, and perhaps even restricted and air-gapped sections of the network. The idea is to segment the network into pieces where access rules can be defined and attacker movement can be disrupted. Although, ideally, each system would have its own set of rules specific to its individual configuration. Often, it is the case that things are not this tightly locked down and many similar systems will share a switch without any specific network firewall rules between them.

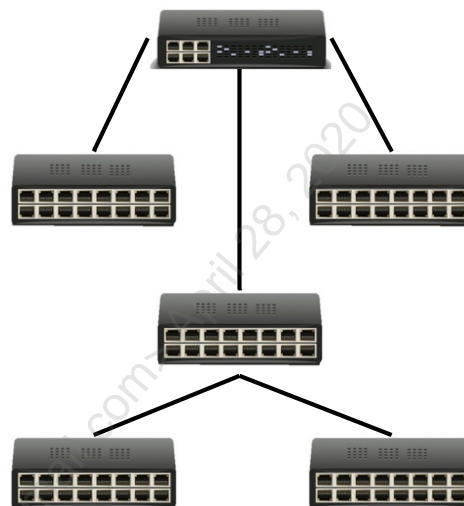


Zones and Traffic Flow

Here is a basic map of a network showing only the logical connections from each zone to the main firewall/router where all rules between them are implemented. We can see there is a DMZ, Server, Desktop, IoT, and restricted zone. The firewall rules determine which type of traffic can flow from one zone to another, with ideally a set of rules that represents the least privilege governing the flow of traffic. Traffic from inside a zone to the same zone may or may not be subjected to another set of rules. This would be determined by the specific implementation details and how the connections of the hardware are physically set up.

Enterprise Switches

- *Managed* enterprise switches provide much more capability
 - Provide useful security features
- Key features for the Blue Team:
 - **Network flow visibility**
 - **Mirror ports to tap traffic**
 - **ACLs for device isolation**
 - **VLANs** separate traffic

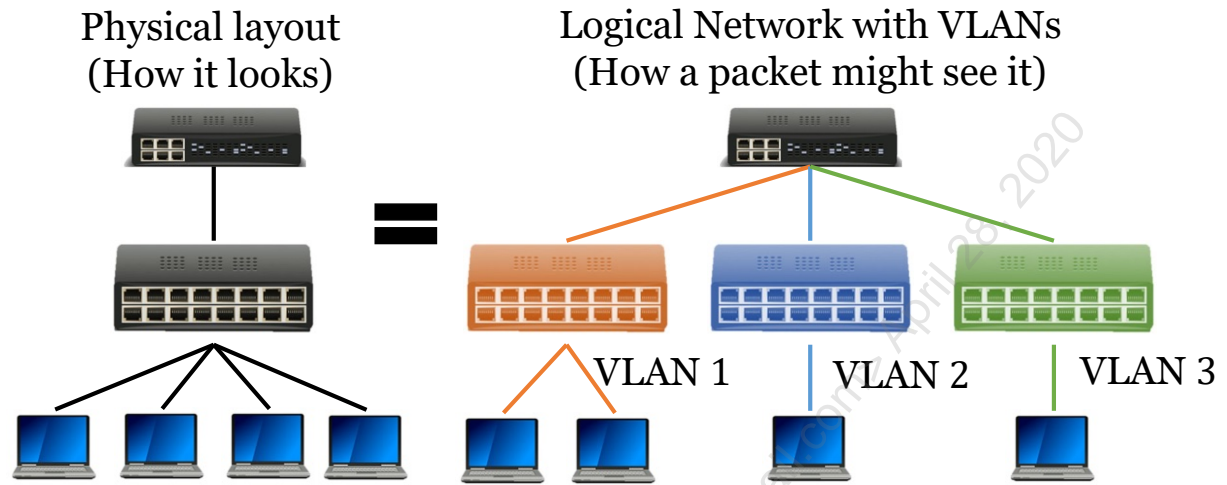


Enterprise Switches

At your organization, *managed* switches (as opposed to *unmanaged* switches) provide a wealth of features not found in the average home device. Like routers, these devices are not often considered security devices, but do provide some key capabilities to the Blue Team.

- Network flow visibility via the ability to record network flow record (such as NetFlow) like a router
- Mirror (SPAN) ports can be defined that will create a copy of all traffic traversing the switch allowing the Blue Team to use it for traffic capture and visibility
- ACLs can be implemented to isolate devices plugged into the switch from talking to each other, preventing lateral movement by attackers
- Virtual LANs (VLANs) can separate traffic from multiple devices physically connected to the same switch

The Physical Network vs Logical Network



To answer “What subnet is that device on?” requires knowing your VLANs

The Physical Network vs. Logical Network

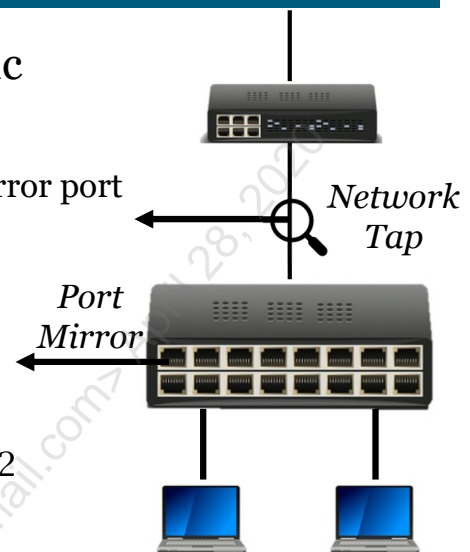
VLANs are an important concept to understand. With an enterprise grade managed switch (and even some cheaper “smart” switches you can buy for home use), VLANs can be defined, which, in effect, make a single physical switch act like it is several different independent switches from the perspective of a packet. This is one key technology that organizations use to implement traffic to multiple *zones* using the same physical switch without introducing the risk of traffic being mixed. In other words, VLANs allow a device from the DMZ subnet, user subnet, and internal subnet to all be plugged into the same physical switch without the risk of these devices being able to talk directly to each other. This gives network designers much more flexibility, but also means traffic may flow in complex ways that are not apparent by only looking at the physical devices being used.

In the slide above, on the left there is a picture of what you might see looking at your switches. What you cannot see is the fact that this one switch is dividing traffic into three different VLANs, and that traffic from a device on VLAN 1 cannot go directly to a device on VLAN 2 without traveling through the router first. While this is convenient for network operations, for the Blue Team, VLANs mean that understanding what subnet a device is on is no longer as simple as looking at which switch it is connected to. Since switches allow devices to connect to multiple VLANs, you now must know the config of the router and which ports connect to which VLAN to understand what subnet a plugged-in device is placed on. Being familiar with the different VLANs in use within your organization will help you picture the different subnets being used, regardless of the hardware used to connect them.

Visibility Points

The most common ways to access traffic

- Mirror (SPAN) ports on a switch
 - Sends all traffic from one port/VLAN out mirror port
 - Active packet duplication, takes CPU
 - Not transparent – drop errors, VLAN tags
 - Potential packet loss if over-subscribed
- Network tap on the wire
 - No device impact, passive duplication of L1/2
 - Extra cost, need to reaggregate traffic



Visibility Points

Understanding your network zones, VLANs, and switches is important because it drives where you must place a network tap or mirror port to duplicate traffic to the network. Whether you use tap, or a switch mirror port will have large implications on whether you will be able to see traffic to and from certain destinations, and the blue team needs to understand these scenarios. Consider the two laptops in the picture. If they are on the same subnet and send traffic directly to each other, the network tap will be blind to it since the traffic never goes upstream from the switch, while the network mirror port would be able to pick it up. This means an attempt at lateral movement could be missed by an upstream network tap!

Why would anyone use a tap then? They have their own benefits as well—consider bandwidth limitations. If one machine is downloading at the full speed of the switch while the other uploads at full speed, the switch will be saturated in full duplex, and a single port mirror port (trying to aggregate 1Gbps of traffic in both directions to one outbound mirror port) will not be able to carry the 2Gbps out a single port. The network tap, which would have a copy for each direction, would not have an issue with it (assuming it wasn't trying to aggregate the traffic to one port).

The interplay between your chosen monitoring method and location is important to understand as a member of the blue team. When an alert has fired, the analyst often must answer the question "what sensor can I gather the data for this alert from?" Understanding the traffic flow and capture capabilities of your network allows you to answer this question and quickly move to the data required or determine that it may not be available.

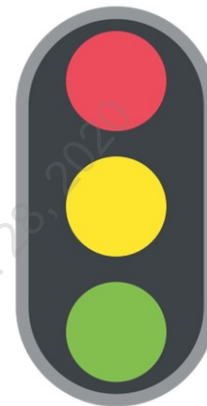
For additional details on the SPAN vs. network tap discussion, see the whitepaper from IXIA listed in the references.¹

[1] <https://support.ixiacom.com/sites/default/files/resources/whitepaper/915-3534-01-tap-vs-span-ltr.pdf>

Enterprise Firewalls

What about enterprise firewalls?

- At home:
 - Blocks traffic from WAN to LAN, maybe outbound, too
 - Functions at Layer 3 and 4 – IP and TCP/UDP port based
 - Doesn't understand users or applications
- Enterprise "**next-gen firewalls**" do much more!
 - Define access between all network segments
 - Controls traffic Layer 2 through **Layer 7**
 - Understand **user identity** and expected **application** behavior
- Firewalls are your centralized traffic control system!



Enterprise Firewalls

We haven't yet discussed one of the most common tools for restricting movement through a network—firewalls. At home, your firewall is likely very basic. Again, it has 2 ports, an inside and outside, and likely has a very simple setup: Block all traffic that originates outside and let everything inside talk to each other or reach outbound. Corporate networks have the same style of rules but are much more granularly defined due to the split of the network into multiple segments and the more locked down configuration that is generally desired.

Corporate firewalls will likely have a ruleset that is similar to our home ruleset. The difference is that they will need to define what is allowed in and out of their multiple ports. They may potentially block outbound traffic, prevent traffic from internal sources to other internal destinations, and make decisions on what's allowed all the way to the application layer, not just IP addresses and ports. They can also be much more detailed in rule logic, such as making user identity-based decisions. If we were going to make a basic ruleset based on Layer 3 and 4 for the model network we've been drawing, what would it look like?

Example Enterprise Firewall Segmentation Rules

		<i>Traffic To</i>					
		WAN	DMZ	IoT	Users	Servers	Restricted
<i>Traffic From</i>	WAN	NA	Allowed	Blocked	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>
	DMZ	<i>Blocked</i>	Partial	Blocked	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>
	IoT	Allowed	<i>Blocked</i>	Allowed	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>
	Users	Through Proxy	Allowed	Allowed	Partial	Allowed	Partial
	Servers	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>	Partial	<i>Blocked</i>
	Restricted	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>	<i>Blocked</i>	Partial

Example Enterprise Firewall Example

Back to our zone map—if we had a centralized firewall, we might have rules similar to what is shown in the table above (note: do not take this table too literally, in a real firewall there should be much more nuance than this, it is meant to show the rough idea only).

- Traffic initiated from the internet is only allowed to go to the DMZ servers, nowhere else.
- Traffic from the DMZ may go to other DMZ services, but, in general, these servers should not be reaching back out to the internet.
- Traffic from the IoT subnet is allowed to go to the internet, and to other IoT devices, but not to the rest of the network.
- Traffic from users is allowed to go to the DMZ services, the internet (through the proxy) IoT devices, and servers. It is only *partially* allowed user-to-user and restricted servers (for IT admins and other special cases—shown in yellow with dashed line).
- Traffic from servers and restricted servers is blocked from going anywhere, servers, in general, should not need to reach out and talk to anything else other than maybe other servers.

Although this is a basic ruleset, you can see how this helps to define what is allowed and, therefore, would slow an attacker down that is trying to bounce around in the network. If an IoT device were to be compromised, in this setup, it would pose no threat to the rest of the subnets. If a normal user device were compromised, attackers wouldn't immediately be able to jump to any of the restricted access servers. Any server that became compromised wouldn't be immediately able to connect anywhere else on the network since no traffic is allowed outbound, including to the internet, which means direct exfiltration of sensitive data from servers becomes impossible! This is a great starting setup, but in production, we would want to go even further and get as granular as possible while still being maintainable. One way to get more granular is to base your blocking rules on Layer 7 characteristics and user identities as well—not just subnet identities. This is something a "next-gen" firewall can do!

Enterprise Layer 7 Firewall and User Filtering

- Define specific **content** and **protocols** allowed, regardless of port!
- Specific site **features** can be blocked
- Understand who you are as a user, not just source IP
- Examples:
 - Can any average user make SSH connections outbound? NO!
 - Can Sally in IT use SSH? Yes, but only her
 - Can Bob in accounting use FTP? Yes, but only him
 - Can people on the manufacturing floor use the internet? No
 - Can VIPs use Facebook, YouTube, and Gmail? Sure...if they really have to

Enterprise Layer 7 Firewall and User Filtering

The whole idea of much of information security is "least privilege"—if something is not explicitly required, don't allow it in the first place. While traditional firewalls work great for filtering a huge volume of our traffic, they aren't a surgical enough instrument for many situations. What if you want to allow access to HTTP sites, but only ones that are used on the well-known ports? What if you want to restrict access to a certain protocol, regardless of port, except for the IT administrators that need it. What if your VIPs make a special request for Facebook, YouTube, or webmail; can you accommodate this with a traditional firewall regardless of where they are in the world? This is where the "next-gen" firewall comes in. Situations like these are very difficult to implement at a TCP/IP level, but next-gen firewalls (NGFW) have no problem implementing it since they understand people and protocols regardless of their source IP or port numbers used.

How do firewalls do that? Deep packet inspection and much higher-powered hardware allow them to understand and classify traffic down to the application level. Signatures for each protocol are written by the vendor and will match against traffic that flows through the firewall, regardless of port. For identity, there are a number of ways of the firewall keeping track of who is coming from where. Sometimes, this is done via an agent installed on each computer that authenticates the machine and user to the firewall. Other times, the central authentication source (such as Windows Active Directory domain controller) logs can be used to feed it as well.

As a defender, the ability to implement least privilege at this fine level of detail is like a superpower!

Network Design Ideals

Ideal: Every system locked down to "least privilege" and only able to talk to other systems that are required, in encrypted/authenticated transactions, damage contained

- "Zero-Trust Architecture" – Filtering at every device



Reality: Flat networks (no filtering at Layer 3, 4, 7), VLANs without filtering, shared administrator passwords, poor configuration, catastrophic failure



Future: Zero trust networks?

Network Design Ideals

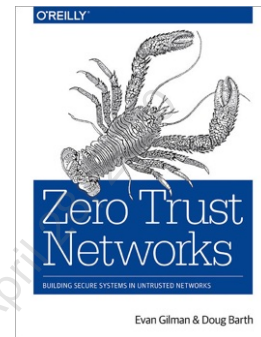
Though our networks have (hopefully!) been broken down into subnets of similar security level systems, the ideal setup goes even further than that. You may have heard the term "zero-trust" architecture or micro segmentation. These ideas take the best practice of segmenting the network to its logical limit and essentially make every system an island. If every machine could only talk to other machines that we knew it absolutely had to talk to and was able to authenticate those systems, it would be extremely difficult for attackers to move! If one desktop couldn't talk to all other desktops, if a single server couldn't reach all the other servers it had no business speaking to, breaches like Equifax where the attacker was able to turn access to one system in to a compromise of 51 databases might not be possible!

While it is true that the industry is moving in this direction, most organizations aren't there yet. The tools to design and manage networks like this are still maturing so micro-segmented and zero-trust networks have been extremely difficult to implement in reality. What we do find in most companies is a much more dire situation, poorly segmented or even completely flat (minimal segmentation beyond perimeter and DMZ) networks that allow attackers to run through them at ease. Many companies have implemented VLANs as discussed yesterday, but often use it for operational purposes and don't realize it is not actually improving security. Remember that VLANs don't, by nature, keep two machines from talking. They just keep those packets from going directly across a switch from source to destination, additional VLAN Access Control Lists (ACLs) and filtering must be implemented *between* VLANs to secure you from internal network pivoting across 2 VLANs. To make matters worse, host configuration can also lend to the problem—shared passwords, slow patching, and non-hardened host builds lead to catastrophic failure.

One solution has been proposed to these woes—moving to "zero trust" networks. It's an ideal that has been around for a long time but is rarely seen in reality, let alone in the truest pure form. We can still learn from the ideals, however, as a way to guide ourselves down the right path. Let's take a look at zero-trust architecture for a moment and see how it can help prevent and slow incidents.

Zero-Trust Architecture Guidelines

- Most networks are *far* from ideal
- But we can learn from the principles...
- Zero-trust architecture assumes¹:
 - The network is always assumed to be **hostile**
 - External and internal **threats** exist on the network at all times
 - Network **locality is not sufficient** for deciding trust in a network
 - **Every device, user, and network flow is authenticated and authorized**
 - **Policies** must be **dynamic** and **calculated** from as many sources of data as possible
- Makes security asset-centric



Zero-Trust Architecture Guidelines

While we are unlikely to see a zero-trust architecture implemented at most organizations, we can still learn from the principles. In the book *Zero Trust Networks* by Doug Barth and Evan Gilman, the principles on the slide above are laid out as an assumption of the modern operating environment and the steps that must be taken to fix it.¹ Even if we have not implemented any pieces of zero trust, these ideas are good guidance on how you should view your own traffic. They are a reminder of the reality of network defense—compromise *will* occur and that means we should assume the network is always hostile and that includes things inside the perimeter. Do not get caught in the trap of thinking that because a network flow came from an internal machine that it was not malicious! Often, attacks will break your implicit assumptions about who is controlling a device, or whether something generating traffic is actually the device it is expected. With credential theft, man-in-the-middle attacks, and non-authenticated protocols, it's entirely possible that a user or device is not what it appears to be.

For an interesting look at what is probably the best implementation of the zero-trust network design ideals, check out BeyondCorp by Google.²

[1] <https://www.oreilly.com/library/view/zero-trust-networks/9781491962183/>

[2] <https://cloud.google.com/beyondcorp/>

Why Network Architecture Matters

- Assumption of compromise
 - Under this assumption, is it justifiable to have a flat network?
 - Initial compromise is not the end goal...
 - But once they get in, initial exploit to goal is FAST in a flat network
 - **More segmentation = slow progression**
- #1 Example: NotPetya and WannaCry
 - Matched APT quality exploit with pen test-style pivoting
 - Segmented networks = slower, contained spread
 - Flat network under this malware = HUGE disaster

Why Network Architecture Matters

Why does all of this matter? Because the nature of attacks today mean that exploitation and internal pivoting will play a large part, and internal pivoting by definition comes from inside your network. Under the assumption of compromise, is it justifiable any longer to have networks that will let attackers reach out and talk to/compromise whatever they please? Of course not. Once we accept that attackers will get inside the perimeter, it becomes an inevitability that a flat network is an irresponsible way of designing a network.

Although we cannot stop attackers entirely, we can do our best to slow them down, and the further a network has been segmented, the slower they will progress. Slow progression buys the blue team more time to catch up and stop an incident before it becomes a full-on expensive breach that hits the news.

The perfect example of this was the WannaCry and NotPetya malware from 2017. These pieces of malware combined nation-state level quality of exploitation (MS17-010 ETERNALBLUE exploit from the Equation Group) with a worm. As soon as one device was infected, they would try to talk to every other machine on the network, exploit them, and run the malware there as well. Since the exploits were so good, this worked very well, which means one of the only ways to stop them was to limit the machines that were available to talk to. How do we do this? Segmentation! Without it, it is likely a single infected machine on the network often exploded into a full-on company outage, and this did happen to several organizations during this crisis.

If you want to read the story of what it's like to be in an organization when this happens, check out the outstanding Wired article referenced that details the mood and recovery during the NotPetya outage at Maersk. It is a very illuminating read that can draw a more realistic picture of what it's like to find yourself in this unfortunate scenario.

[1] <https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/>

Defending a Flat Network

Your network has no segmentation? You're not alone...

Do realize the implications:

- **Harder to define and detect bad** when all traffic is allowed
- **Incidents will escalate quickly**
 - More exploitable servers to find
 - More exposed credentials
 - Worm-style malware will spread extremely fast
- **Blue team must compensate** with strong host-based controls
 - Firewall, IPS, AV, anti-exploitation, EDR, and thorough monitoring

Defending a Flat Network

Although a flat network does not meet the criteria of "defensible", some of us have them and must deal with them anyway. Flat networks are in a much higher risk position. Once an attacker gets a foothold, they will be able to attempt to exploit any server in the environment and move between any host without impediment. This means any incident that does occur can potentially escalate quickly, so the blue team must be even better and faster at responding. How do we compensate for a lack of network controls? Through a combination of good processes involving faster response tools, lower SLAs, better monitoring, and strong host/endpoint protection. If we can't firewall off segments of the network from each other, a focus should be placed on ensuring endpoints can defend themselves with a host-based firewall, anti-exploitation features (like EMET or Windows Exploit Guard), intrusion prevention systems, and antivirus. Host defense should also be paired with a strong monitoring policy to help ensure that if something does happen, at least the SOC will know about it as quickly as possible.

Network Architecture Summary

As an analyst, your interest mainly lies in:

- Routers
- Switches
- Firewalls
- Physical vs. Logical setup
- Monitoring locations (for IDS/IPS, PCAP, NetFlow, etc.)
 - Taps
 - Mirror ports
- The location of your "crown jewels"

Network Architecture Summary

In summary, as defenders we need to understand the fundamental building blocks of our networks, how we've laid them out in the network, and where our points of visibility are. For our job, it is not always necessary to see the network engineer level detailed diagram of *exactly* what is going on, but we do need to get a sense of the traffic flow in the environment. This will come into play over and over as you triage incidents and need to locate or understand if you even have a copy of what happened. As a general rule, your SOC should have a copy of a generalized diagram that shows how traffic is able to flow between your different network segments, as well as where any taps or span ports are located, and whether routers and other devices are sending NetFlow or other network analytics info. If you have this information close at hand, any time a potential incident comes with attempted internal pivoting, you'll be able to go right to the correct sources of data to verify whether or not the attack progressed or was stopped by a firewall.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. **Traffic Capture and Analysis**
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Network Data Capture Formats

NetFlow: Layer 3 and 4 only

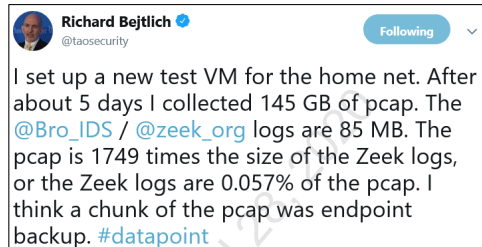
- Alternatively: Jflow, NetStream, sFlow
- Smallest of all options, least useful

Service Logs: Zeek, Suricata, etc.

- Layer 7 metadata only
- Not that much larger than NetFlow, MUCH more useful

PCAP: Complete information

- Contains the actual data transferred, will have your answer
- Huge storage space required



Network Data Capture Formats

When it comes to capturing data about traffic traversing the network, there are 3 common ways that are used. Your organization is likely to have a combination of all of them, so it will be useful to understand what is included in each data source, and when you may have to use one over the other.

NetFlow is the first option. It is the lightest weight capture format meant to describe all sessions at a Layer 3 and 4 level (IP and TCP/UDP). You'll receive session timing, how much data was transferred in each direction, the high-level protocol used, and the source and destination IP and port, but nothing much more specific than that. The good news about NetFlow is that it is still very useful for catching obvious badness—huge volumes, top talkers, most frequent connections, and odd port numbers/bad IP addresses can all be used to catch evil based on NetFlow only. One thing to be aware of is sFlow, which stands for *sampled* flow. sFlow is used for high-speed networks where even NetFlow capture cannot be done. If you have sFlow, just be cognizant that you are working with partial data.

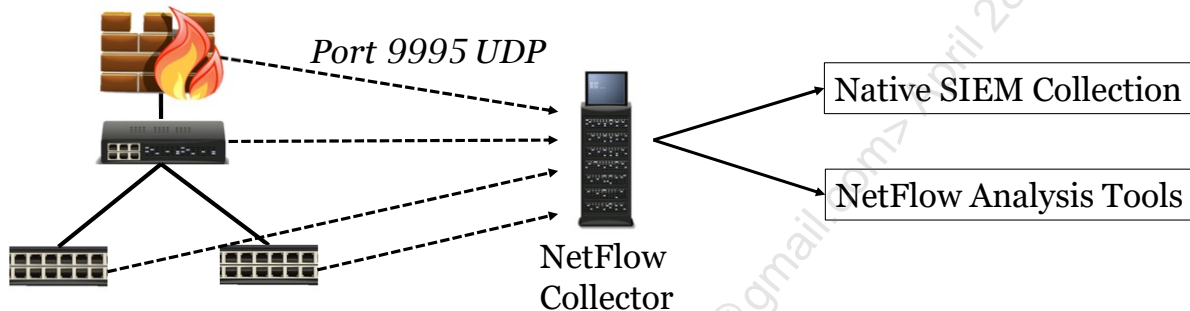
Service logs captured by a tool like Zeek (Bro) or Suricata are another option. These tools will look deeper than NetFlow and capture information about Layer 7 metadata as well. They will cut logs for HTTP methods, DNS requests types, hostnames, SMTP connection details and more, plus everything from the NetFlow tools. This detail gives much more context around a connection that is identified as suspicious. The good news is, the data is still comparatively small, but gives an outstanding amount of detail.

Full PCAP is the elephant-sized solution of the bunch. The good thing about full PCAP capture is no matter what happens (assuming a connection isn't encrypted) you will have an exact copy of the network traffic and be able to figure out the exact truth. The downside of PCAP is that it takes enormous amounts of storage space compared to the other options. Although it's only a single datapoint, Richard Bejtlich Tweeted the math on his Zeek vs. PCAP data recently and found that Zeek data was 0.057% of the full PCAP—a rather astounding difference!¹

[1] <https://twitter.com/taosecurity/status/1066377040256004096>

NetFlow

- Captured by agent on routers, firewalls, or switches
- Can be synthesized via network monitoring or PCAP file
- Contains: Interfaces, src/dest IP and port, protocol, volume



NetFlow

NetFlow itself is technically a Cisco proprietary protocol and comes in 3 common versions, each of which can be customized to a further degree – v5, v9 and v10, also known as IPFIX. Other vendors have their own protocols such as Juniper's Jflow, and HP's NetStream, but even these protocols are commonly referred to colloquially as "NetFlow", so for the purpose of this class, we will group all of them under the name NetFlow.

Remember, NetFlow is not a log. It is a network protocol that sends data about *other* network information the sending device has seen passing through it. Although RFC 3954 for Cisco-style NetFlow does not specify a port, it is typically sent over UDP port 2055, 9995, or 9996. Tools like nfdump are required to convert NetFlow data into a human readable text log such as what is shown in the following slide. For this reason, some SIEMs can take in and interpret NetFlow data directly, a useful feature when available. If your SIEM cannot do this, there's always manual analysis, but an alternate method is to synthesize the same data using other tools (like Zeek), which *will* create normal log files that can be imported into the SIEM as a plaintext log the same as all others.

NetFlow Synthesized via TShark

	<-		->		Total		Relative	Duration
	Frames	Bytes	Frames	Bytes	Frames	Bytes	Start	
24.39.21.194:1311 <-> 144.76.86.115:80	234	349580	120	7289	354	356869	5.717833000	66.9429
24.39.21.194:1509 <-> 46.4.115.211:25	114	163336	63	3570	177	166906	13.913458000	62.8729
24.39.21.194:1897 <-> 198.20.248.102:80	57	81137	34	2598	91	83735	34.521170000	9.5996
24.39.21.194:2149 <-> 119.245.179.77:80	49	68498	30	1958	79	70456	51.380231000	6.5449
24.39.21.194:1781 <-> 77.55.53.147:80	47	63958	30	2620	77	66578	28.379888000	22.6184
24.39.21.194:2148 <-> 95.129.200.2:80	44	59711	27	2222	71	61933	51.379811000	10.5841
24.39.21.194:2159 <-> 64.22.85.186:80	44	61212	25	1708	69	62920	51.624389000	0.8514
24.39.21.194:2178 <-> 213.186.33.4:80	39	39470	29	1955	68	41425	53.755190000	2.7963
24.39.21.194:1280 <-> 173.201.140.128:80	42	58372	26	2226	68	60598	4.734575000	8.8046
24.39.21.194:1460 <-> 212.55.198.50:80	39	50805	24	2002	63	52807	12.292507000	11.2975
24.39.21.194:1313 <-> 184.106.119.164:80	39	52050	24	2918	63	54968	5.777983000	7.7600
24.39.21.194:1308 <-> 186.202.149.17:80	39	21751	23	2074	62	23825	5.671920000	1.2544
24.39.21.194:1541 <-> 49.50.8.93:80	29	39533	30	1832	59	41365	14.708050000	11.6454
24.39.21.194:1254 <-> 70.32.102.108:80	36	45882	22	2878	58	48760	4.193346000	68.3558
24.39.21.194:1267 <-> 213.186.33.4:80	35	36892	22	1434	57	38326	4.500042000	72.3843
24.39.21.194:1715 <-> 67.199.15.24:80	35	45987	21	1930	56	47917	25.037557000	11.9735
24.39.21.194:1711 <-> 206.223.152.85:80	34	47161	22	2004	56	49165	24.990666000	19.0679
24.39.21.194:2142 <-> 216.240.189.6:80	33	44228	22	1545	55	45773	51.163120000	6.7633
24.39.21.194:2069 <-> 216.240.189.6:80	34	44325	21	2039	55	46364	44.927426000	3.0764
24.39.21.194:2110 <-> 216.240.189.6:80	34	44372	20	1567	54	45939	47.910707000	3.3125
24.39.21.194:1968 <-> 216.240.189.6:80	34	44603	20	1460	54	46063	38.521365000	6.4726
24.39.21.194:1752 <-> 66.29.219.64:80	34	45837	20	1742	54	47579	26.895393000	9.4222
24.39.21.194:1596 <-> 46.165.204.101:80	32	42786	22	1502	54	44288	16.839837000	17.3715

NetFlow Synthesized via TShark

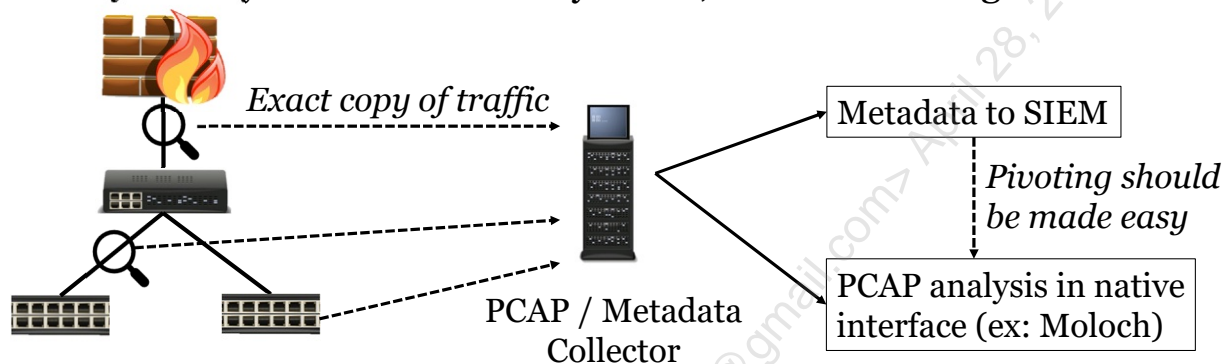
This slide shows an example of the same type of data you would see with NetFlow but was created from a PCAP file instead. It was made by taking a sample traffic capture from the "cutwail" botnet and running it through Tshark (the command line version of Wireshark) with the command documented below.¹ This just goes to show that if you do not collect NetFlow information natively but would prefer having it for a particular investigation, you can easily recreate it from PCAPs or other metadata that was captured.

Notice that NetFlow data is indeed high-level. It shows information on IP addresses, ports, frame and byte counts, how long and when the session started, and how long it lasted. This is obviously lacking some detail, but don't count it out as not useful. There are a shocking number of breaches that have been caught using NetFlow alone.

[1] \$ tshark -r [pcap file] -z conv,tcp -q -n

PCAP and Layer 7 Metadata Collection

- Captured directly off the wire, or from switch mirror port
- Sends an *exact copy* of traffic for PCAP
- May or may not be collected by SIEM, but should integrate with it



PCAP and Layer 7 Metadata Collection

Acquiring a full copy of traffic for a PCAP or performing network extraction of Layer 7 metadata requires a different type of collection mechanism than NetFlow. Whereas NetFlow information is generated natively by agents built into your router, firewall, or switch, full PCAP and Layer 7 metadata requires a duplicate copy of the traffic to either record in full or generate the service logs from. In practice, this means that we must either use a network tap or span port to get a copy of the traffic to the PCAP or metadata collector.

Full PCAP capture devices will have a *much* larger job than a NetFlow collector since they are potentially recording *all* traffic that goes in and out of your organization. If you have a 1Gbps connection for your business that is 50% utilized on average, you will need to record 37.8TB per week to keep that data. This is why many organizations collect full PCAP only at one or two select locations, such as on the perimeter, and leave other sensors only to collect Layer 7 metadata. The size difference, as referenced earlier in Richard Bejtlich's Tweet, can be enormous. Of course, the actual size difference will depend on the nature of your traffic and what service logs are in use, but vendor Corelight, which essentially makes commercialized Zeek appliances, says in their promotional literature that the data is usually 1/100th the size of PCAP or less.¹

[1] <https://www.corelight.com/cases/why-corelight/>

Layer 7 Metadata: Network Service Logs

- A middle ground between NetFlow and PCAP
- Best value for the dollar – small size, extremely useful
- Many free tools for collection and deployment

src_ip	src_port	dest_ip	dest_port	method	host	uri	user_agent
24.39.21.194	1218	199.83.128.93	80	POST	www.traderush.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1210	208.97.174.44	80	POST	graceweb.net	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1212	192.64.112.193	80	POST	naijagurus.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1228	108.162.196.90	80	GET	www.graceweb.net	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1225	85.233.160.22	80	POST	eygggroup.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1221	204.11.237.35	80	POST	denville.ca	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1208	5.9.122.172	80	POST	justconnect.co.za	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1211	91.121.66.183	80	POST	e-storming.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1219	81.209.182.37	80	POST	rueggeberg.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1214	66.49.139.143	80	POST	christybarry.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1227	54.229.116.65	80	POST	eurasia.it	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1229	162.159.250.145	80	POST	cbsprinting.com.au	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1220	188.121.45.218	80	POST	avant-ime.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1231	141.101.116.108	80	POST	glmghotels.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1216	109.74.242.160	80	POST	marcusgrimes.co.uk	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
24.39.21.194	1230	12.158.190.246	80	POST	mojacar-vacaciones.com	/	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

Layer 7 Metadata: Network Service Logs

Imagine having the information contained in NetFlow, but also including the transaction detail for every session of HTTP, DNS, SMTP, SMB, SSL, and more. These are what the Layer 7 metadata network service logs give you. These tools likely represent the best bang for the buck in terms of ease of collection and ability to detect intrusion. There's just one problem—many companies do not collect all, if any of them! There's good news, though. Changing that situation is very easy. To start collecting these items, all that's needed is hardware that can keep up with the rate of traffic you'd like to sample, a network tap or switch span port, and a copy of the free and open source Zeek or Suricata IDS! Alternatively, if network extraction of this data is not an option, the actual service logs should be available from all endpoints in the environment and a log agent can be used to collect it. This is more complex to manage and keep up to date but is a theoretically equivalent solution.

The slide above shows an example set of logs as recorded by Zeek for a set of HTTP traffic for the "cutwail" botnet. For each network protocol supported, Zeek will write tab-separated value files of everything it sees on the wire, and this information can be sent to a SIEM for easy analysis and anomaly detection.

The protocols Zeek currently supports are Conn (NetFlow equivalent), DCE_RPC, DNP3, DNS, FTP, HTTP, IRC, Kerberos, Modbus, MySQL, NTLM, RADIUS, RDP, RFB(VNC), SIP, SMB, SMTP, SOCKS, SSH, SSL, Syslog, and Tunneling protocols. It also will cut metadata for files seen on the network, threat intel matches, and known devices and hosts on the network.¹

Suricata currently supports NetFlow style logging, HTTP, SSL, TLS, SMB, DCERPC, SMTP, FTP, SSH, DNS, Modbus, ENIP/CIP, DNP3, NFS, NTP, DHCP, TFTP, KRB5, and IKEv2.²

[1] <https://docs.zeek.org/en/stable/script-reference/log-files.html>

[2] <https://suricata-ids.org/features/all-features/>

PCAP

The de facto standard format for packet capture

- Stores every single byte transferred over the wire
- Usually captured using winpcap/libpcap
- 2 versions: .pcap, .pcapng
 - NG version includes higher precision, interface info, comments, more

Tools for capture and analysis:

GUI:

- Viewing: **Wireshark**, PacketTotal.com,
- FPC & Analysis: **Moloch**
- Extraction: NetworkMiner

CLI:

- Capture, filtering: **tcpdump**, **Tshark**
- Metadata: Zeek, CapTipper
- IDS: Snort, Suricata
- Term matching: Ngrep, strings
- Extraction: tcpextract, chaosreader, scalpel, foremost, xplico

PCAP

The PCAP file format is the de facto standard for recording full packet captures. If you have ever used Wireshark, tcpdump, or other common tools, you've already run into this format. Capturing PCAP live off the wire usually involves the usage of the libpcap software from the tcpdump project, or the winpcap library equivalent for Windows and the files written can be of either pcap, or pcap-ng format. Almost all tools support the traditional pcap format, while newer tools may also support the pcap-ng standard, which makes room for extended information in the capture file.

There are numerous excellent, free tools available for full packet capture, analysis, metadata extraction, file extraction, and threat detection, some of the most common of which are named on the slide above. Most analysts have their first taste of packet capture with tcpdump or Wireshark, but newer packet capture and analysis suites like Moloch are making headway fast in the security landscape. Since full packet capture is an exact copy of what was transferred over the network, analysts who have pcap available can run these tools after the fact that can replay interactions, dump files that were transferred, or run detection signatures over the traffic. Let's take a closer look at Wireshark since it will be used numerous times throughout this class.

The screenshot displays the Wireshark interface with the following sections:

- Filter:** http or dns
- Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
39	60.533245	10.5.11.102	72.247.8.136	HTTP	543	GET /qsm1.aspx?query=http%3A%2F%2F10.5.11.10&maxwidth=32765&rowheight=31&...
41	60.663576	72.247.8.136	10.5.11.102	HTTP/XML	428	HTTP/1.1 200 OK
43	60.927417	10.5.11.102	72.247.8.136	HTTP	544	GET /qsm1.aspx?query=http%3A%2F%2F10.5.11.103&maxwidth=32765&rowheight=31&...
49	61.059121	10.5.11.102	10.5.11.103	HTTP	308	GET / HTTP/1.1
51	61.060031	10.5.11.103	10.5.11.102	HTTP	537	HTTP/1.1 200 OK (text/html)
- Network Layers:**
 - Frame 39: 543 bytes on wire (4344 bits), 543 bytes captured (4344 bits)
 - Ethernet II, Src: Vmware bd:c2:53 (00:0c:29:bd:c2:53), Dst: Vmware f6:83:50 (00:50:56:f6:83:50)
 - Internet Protocol Version 4, Src: 10.5.11.102 (10.5.11.102), Dst: 72.247.8.136 (72.247.8.136)
 - Transmission Control Protocol, Src Port: 49319 (49319), Dst Port: http (80), Seq: 978, Ack: 749, Len: 489
 - Hypertext Transfer Protocol
 - GET /qsm1.aspx?query=http%3A%2F%2F10.5.11.10&maxwidth=32765&rowheight=31§ionHeight=310&FORM=IE11SS&market=en-US HTTP/1.1\r\n
 - [Expert Info (Chat/Sequence): GET /qsm1.aspx?query=http%3A%2F%2F10.5.11.10&maxwidth=32765&rowheight=31§ionHeight=310&FORM=IE
- Packet Details:**
 - Request Method: GET
 - Request URI: /qsm1.aspx?query=http%3A%2F%2F10.5.11.10&maxwidth=32765&rowheight=31§ionHeight=310&FORM=IE11SS&market=en-US
 - Request Version: HTTP/1.1
 - Accept: */*\r\n
 - Accept-Language: en-US\r\n
 - User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n
 - Accept-Encoding: gzip, deflate\r\n
- Packet Bytes:**

```

0000  00 50 56 f6 83 50 00 0c 29 bd c2 53 08 00 45 00  .PV..P..)S..E.
0010  02 11 21 57 40 00 80 06 70 a6 0a 05 0b 66 48 f7  ..!W@...p....fH.
0020  08 88 c0 a7 00 50 aa 0c c2 1e c8 fb 89 9e 50 18  ....P.....P.
0030  f8 04 cf 60 00 00 47 45 54 20 2f 71 73 6d 6c 2e  ....GET/qsm1.
0040  61 73 70 78 3f 71 75 65 72 79 3d 68 74 74 70 25  asp?que ry=http
0050  33 41 25 32 46 25 32 46 31 30 2e 35 2e 31 31 2e  3A%2F%2F10.5.11.

```

Wireshark

Wireshark is the de facto standard for taking a look around PCAP files. To show the traffic, the display is intuitively laid out into 3 panels by default. If you haven't had a chance to use Wireshark for analysis before, there will be multiple chances to learn in labs throughout the week. In Wireshark, the top panel shows the packet list where there is one line for each packet present in the PCAP. Columns can be added, removed or changed, and typically show IP addresses, the highest protocol that can be identified for a given packet, the length of the packet, and a bit of metadata where it can be usefully determined.

The middle section is the packet details window, which highlights the packet selected from the above list section and gives additional information broken down by the network layer. In the screenshot, you can see that the top line is a folded-up section concerned with Layer 1 information—frame number, bytes on the wire, and other top-level information. Under that is the Layer 2 info about the ethernet frame, showing source and destination MAC addresses. Under that is the Layer 3 info telling us what source and destination IP addresses were present in the packet, followed by the Layer 4 information in the line below giving the TCP source and destination port info. The final line, which is unfolded to show additional detail, is the Application layer information and it shows the contents of the HTTP protocol. We can see the request method, URI, and the HTTP version number. Each of these layers can be unfolded as the HTTP layer is above, giving the user an interpretation of what the bytes mean at each piece of the network stack.

The final panel on the bottom is the actual packet bytes. This view is useful if you need to see exactly what data was sent over the wire. Clicking any of the fields in the packet details section will highlight the bytes that pertain to that area, as shown in the screenshot with the GET request method that is highlighted in the details and bytes pane.

Wireshark is an incredibly powerful tool for analysis and understanding the contents of a PCAP file. The thing is, it is not meant for looking at enormous amounts of continuously recorded traffic. It is used when there is a limited scope PCAP file that needs to be opened as a one off. For mass analysis of PCAP traffic recorded for an entire network over time, Wireshark is not the tool. Something like Moloch is needed for a job like that.

Wireshark Search and Demo

Protocol-based display filters:

- `http` – filter to all http traffic
- `dns` – filter to dns traffic
- `http or dns` – filter to http or dns traffic

Network-based display filters:

- `ip.addr eq 1.2.3.4` – filter to source/dest. IP = 1.2.3.4
- `tcp.port eq 80` – filter to source or dest. port = 80
- `ip.addr eq 1.2.3.4 && tcp.port eq 80`
- `frame contains search` – filter to any packet with "search"

Wireshark Search and Demo

Here are some terms that may help you more quickly navigate packets in Wireshark. Your instructor will give a demo of the basic use of Wireshark at this point. We do not have to become pros for this class but will need a basic ability to navigate and investigate packets.

Example Scenario: Layer 3 & 4 View

Here's a NetFlow-style view of some traffic:

```

=====
TCP Conversations
Filter:<No Filter>

```

		<-		->		Total		Relative Start	Duration
		Frames	Bytes	Frames	Bytes	Frames	Bytes		
10.0.2.15:1031	<-> 83.69.233.156:80	177	194892	60	3797	237	198689	52.329302000	28.9207
10.0.2.15:1032	<-> 212.98.162.62:80	158	210051	36	2545	194	212596	54.902852000	6.4497
10.0.2.15:1030	<-> 83.69.233.156:80	43	52307	20	2110	63	54417	20.025252000	30.5373
10.0.2.15:1035	<-> 91.230.147.222:80	5	465	6	833	11	1298	142.488360000	62.0023
10.0.2.15:1034	<-> 91.230.147.222:80	5	465	6	833	11	1298	80.987344000	61.4957
10.0.2.15:1029	<-> 212.98.162.62:80	4	720	6	596	10	1316	18.363194000	3.0005
10.0.2.15:1036	<-> 91.230.147.222:80	4	411	5	773	9	1184	204.541612000	15.8497
91.230.147.222:8888	<-> 10.0.2.15:1033	5	578	4	220	9	798	79.938763000	1.0005

```

=====

```

Is this traffic malicious?

Example Scenario: Layer 3 & 4 View

Let's say, for example, we have a NetFlow view of our network that shows the traffic above. This view gives us details at only Layers 3 and 4 of the network stack, IP addresses and ports used, as well as bytes and duration of the sessions, but can we tell exactly what happened? If we're trying to identify potentially malicious traffic, would we easily be able to do it from the data shown above?

In this case, it's not very clear if anything malicious has happened or not. We merely have a list of IP addresses, ports, bytes exchanged, and the associated times. We may be able to infer from the use of an odd port that there is the traffic of interest—there is nothing that clearly sticks out as bad. If we were able to enrich this data, or correlate it with other logs to show what websites were visited at those IP addresses, we might have a better idea of what happened. If your only source of data is NetFlow from a switch or router, or traditional (meaning not a "next-gen") firewall logs, this might be all the information you have to go on, and monitoring capabilities will be seriously hindered. Let's look at the same traffic with some of the Layer 7 HTTP transaction data listed as well and see if the situation gets any better.

Example Scenario: The Layer 7 View

A little more detail ...

```

10.0.2.15 -> 83.69.233.156 HTTP 524 fewfewfewfew.biz.cc GET
/main.php?page=95fc4549d83b0486 HTTP/1.1
83.69.233.156 -> 10.0.2.15 HTTP 1088 HTTP/1.1 200 OK (text/html)
10.0.2.15 -> 83.69.233.156 HTTP 504 fewfewfewfew.biz.cc GET /data/ap2.php HTTP/1.1
83.69.233.156 -> 10.0.2.15 HTTP 629 HTTP/1.1 200 OK (application/pdf)
10.0.2.15 -> 83.69.233.156 HTTP 255 fewfewfewfew.biz.cc GET /w.php?f=9603e&e=4 HTTP/1.1
83.69.233.156 -> 10.0.2.15 HTTP 1219 HTTP/1.1 200 OK (application/x-msdownload)
10.0.2.15 -> 91.230.147.222 HTTP 531 google-analytics-sv1.com POST /aaa.php HTTP/1.1
(application/x-www-form-urlencoded)
91.230.147.222 -> 10.0.2.15 HTTP 245 HTTP/1.1 200 OK
10.0.2.15 -> 91.230.147.222 HTTP 531 google-analytics-sv1.com POST /aaa.php HTTP/1.1
(application/x-www-form-urlencoded)
91.230.147.222 -> 10.0.2.15 HTTP 245 HTTP/1.1 200 OK
10.0.2.15 -> 91.230.147.222 HTTP 531 google-analytics-sv1.com POST /aaa.php HTTP/1.1
(application/x-www-form-urlencoded)
91.230.147.222 -> 10.0.2.15 HTTP 245 HTTP/1.1 200 OK

```

Can we tell now?

Example Scenario: Layer 7 View

This view shows the same traffic as the previous slide, but in a view with information that you might receive from either a proxy log, next-gen firewall, IDS, or other appliance capable of reading HTTP headers and URIs. Given this information, we can start to tell that things may indeed be suspect. We see the domain name "fewfewfewfew.biz.cc" listed as the host that was contacted—not exactly confidence-inspiring. We can also see the URIs that were requested and which HTTP method (GET/POST) was used for each request. Looking at the responses, we can also tell the types of files that were returned for each request. This view highlights the necessity of at least basic Layer 7 data visibility for network security monitoring. Without this information, it would be impossible to tell whether the user was visiting "fewfewfewfew.biz.cc" or any other website that may be hosted on the same server.

At this point, we can probably say this is traffic we should investigate. We have a highly-suspect domain name and on the sixth line of the log, we can see the server responded with a potential executable file that would've been downloaded to the endpoint. Is it truly an executable? Is the executable malware? These are some of the questions we would likely need full packet capture in order to solve.

Example Scenario: Layer 7 Full Payload View

```

Stream Content
GET /w.php?f=9603e&e=4 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: fewfewfew.ibiz.cc
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.2.0
Date: Sat, 28 Apr 2012 06:53:17 GMT
Content-Type: application/x-msdownload
Transfer-Encoding: chunked
Connection: keep-alive
Pragma: public
Expires: Sat, 28 Apr 2012 06:55:00 GMT
Cache-Control: must-revalidate, post-check=0, pre-check=0
Cache-Control: private
Content-Disposition: attachment; filename="calc.exe"
Content-Transfer-Encoding: binary

e9a
MZ.....@.....!..L!This
program cannot be run in DOS mode.

$.6.S.r{.-r{=.r{=.p|=..v~<.X{=.\..~w{...2..}...s...t|N.n...s{=.i...s
{=.....PE..L.....M.....
    
```

Request

Response

Example Scenario: Layer 7 Full Payload View

If we had a full packet capture solution installed on the network in a location that would record the whole session byte by byte, we could inspect the exact packets that were transferred and what truly happened. This view shows the request and response from the previous slide where the server claimed an executable file was returned. From this view, we can see everything—that is, every single byte in the session. The client first requested a URI ending in w.php with a User-Agent that claims to be Microsoft IE 6.0. This is highly questionable, as no one should be using that software at this point. We then can see that the server did indeed return an executable and that it was apparently called "calc.exe"—another clue that something bad is happening here. We can tell an executable was downloaded because the full payload contains the "magic bytes" that signify an executable file—the "MZ" followed by the string, "This program cannot be run in DOS mode."

Don't worry if you're not familiar with identifying executables on the wire in this way. We will discuss it in more detail later in the class. At this stage, the idea is to point out that sometimes it's just impossible to tell what happened without the full packet payload. Given this complete information, we can start to develop the theory that this was highly likely a malicious download, and that it may be trying to disguise the virus itself by naming it calc.exe to blend in with the built-in Windows tool. With the PCAP from this session loaded in Wireshark, this executable file could be extracted, hashed and checked against VirusTotal, or scanned with antivirus to continue the investigation.

Traffic Capture and Analysis Summary

Traffic capture happens on 3 levels:

- NetFlow: IP, port, protocol, size and timing info only
 - Uses a proprietary protocol, *might* go to your SIEM
- Layer 7 Metadata: The best value for collection
 - Centrally creates log files for services, *should* go to the SIEM
- Full PCAP: Exact copy of the traffic
 - Huge, but contains everything
 - Analysis will likely occur in other tools, SIEM should link to it

Traffic Capture and Analysis Summary

When it comes to traffic capture, you will likely have at least one of these sources of data. Ideally, you have all 3. There are certain questions that are the fastest and easiest to answer with NetFlow—Who's using the most bandwidth for upload and download? Where is most of my traffic going? Who has the longest running or most network sessions? All these questions can and will point you in the direction of potential compromise, especially if you can pull the data into a SIEM dashboard.

Layer 7 metadata can get you even further. It's the best for looking for anomalies based on the field data from each specific protocol. Tools like Zeek are capable of parsing out hundreds of fields for typical application layer protocols, and with a SIEM, will enable you to use that data for dashboards, anomaly detection, and alerting. Full PCAP is the heaviest to collect by far, but when you absolutely have to know exactly what happened, it's the only way to do it. Data captures by your full PCAP solution will likely not be integrated with your SIEM, but it should be easy to bounce from one tool to the other during investigations.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
- 3. Understanding DNS**
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

In This Module

In this module:

- What details of DNS are important to analysts
- How DNS records are set up and maintained
- DNS request types and their uses
- DNS operation in-depth
- How DNS looks on the wire



In This Module

This module is a whirlwind tour of DNS. You are likely already somewhat familiar with how DNS works—that it is used to resolve hostnames to IPs, but we must go further than this. DNS is a protocol that is often abused by attackers, and undoubtedly there will be a scenario you will encounter that will require a deeper understanding. This module takes us through the reasons to study DNS, the important record types, and the details of operation that are most relevant to intrusion detection.

Why Do We Care About DNS?

Determines info about remote host, usually IP:

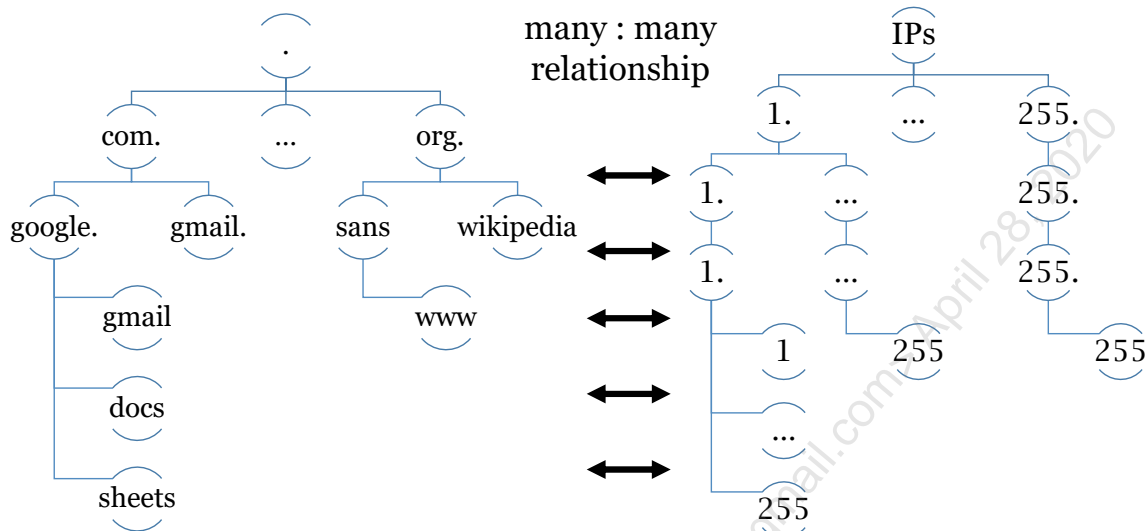
- Like an internet phone book
 - Maps a hard-to-remember number with an easy name
 - Facilitates connecting to something



Why Do We Care About DNS?

As analysts, we care about DNS because it is the primary way we can arrive at any given site on the internet. Similar to a phone book, it maps names we know and can remember against a somewhat random set of numbers that is much more complicated to recall. The internet is reliant upon DNS is a huge way. Every site you browse to, and service you visit likely leverages the global service to look up the IP address any given computer is actually located. Almost every single incident you work will require you to record IP addresses and the respective domain name that was queried that pointed to it. Since DNS queries can be used and misused in multiple ways, it is critical that we have a clear understanding of what the protocol does normally, how it can affect our investigations, and how it can be used against us.

DNS: Mapping Trees Together



DNS: Mapping Trees Together

Let's take a step back now and look at, on a grand scale, what DNS is doing. DNS's job is, in a general sense, largely about mapping two trees onto each other. One tree is made of all the IP addresses that exist (right side), the other is all the hostnames to reference them by (left). Some hosts will be mail servers, some will run websites, some do more than one thing. Some hostnames will point to other hostnames, and some IPs won't have hostnames, while other hostnames won't map to IPs. It's a complicated issue no doubt.

Ultimately, DNS is designed to help us find something in a tree of information given a second piece of information related to it, and it must handle it in as elegant of a way as possible. To do this, each of these arrows can be thought of as a different type of record in the DNS system, depending on where side they start, and where they end. Some could be A records, PTR records, or CNAME records, or a whole host of other options that may link to other data, such as bits of arbitrary text.

DNS Server and Client Types

- **Stub Resolver:** Your computer, as you browse
 - May or may not locally cache answers
- **Forwarding Server:** Caches answers but does not recurse
 - Your home router / domain controller / organization's local DNS Server
- **Caching/Recursive Server:** The server that will recurse up to the root nameservers to find the answer caches answers
 - ISP DNS server, OpenDNS, 8.8.8.8, 1.1.1.1, pi-hole, etc.
- **Authoritative nameservers:** Do not need to recurse to find an answer, know all names in their zone. No caching/recursing

DNS Server and Client Types

There are multiple different types of DNS servers and clients that get used in the process of resolving a hostname. In general, there are 2 types: Recursive and authoritative. Recursive nameservers offer resolution services but are not authoritative for any zone. Authoritative nameservers will not recurse as they are the source of truth for answers about resources in their zone. Stub resolvers, forwarding servers, and caching servers are all recursive nameservers.

Stub Resolver: This is the basic DNS client run by the host operating system that will initiate the lookup for a hostname. DNS query results can be cached at this level for hosts that have already been looked up and queries are passed up the chain for those answers that are not already known. The next place the DNS query will be sent is set in your network configuration settings. To see your own DNS cache in Windows, type "ipconfig/displaydns" in a cmd.exe window. Note that while Windows keeps a list of recently resolved hostnames, Linux does not cache answers by default.

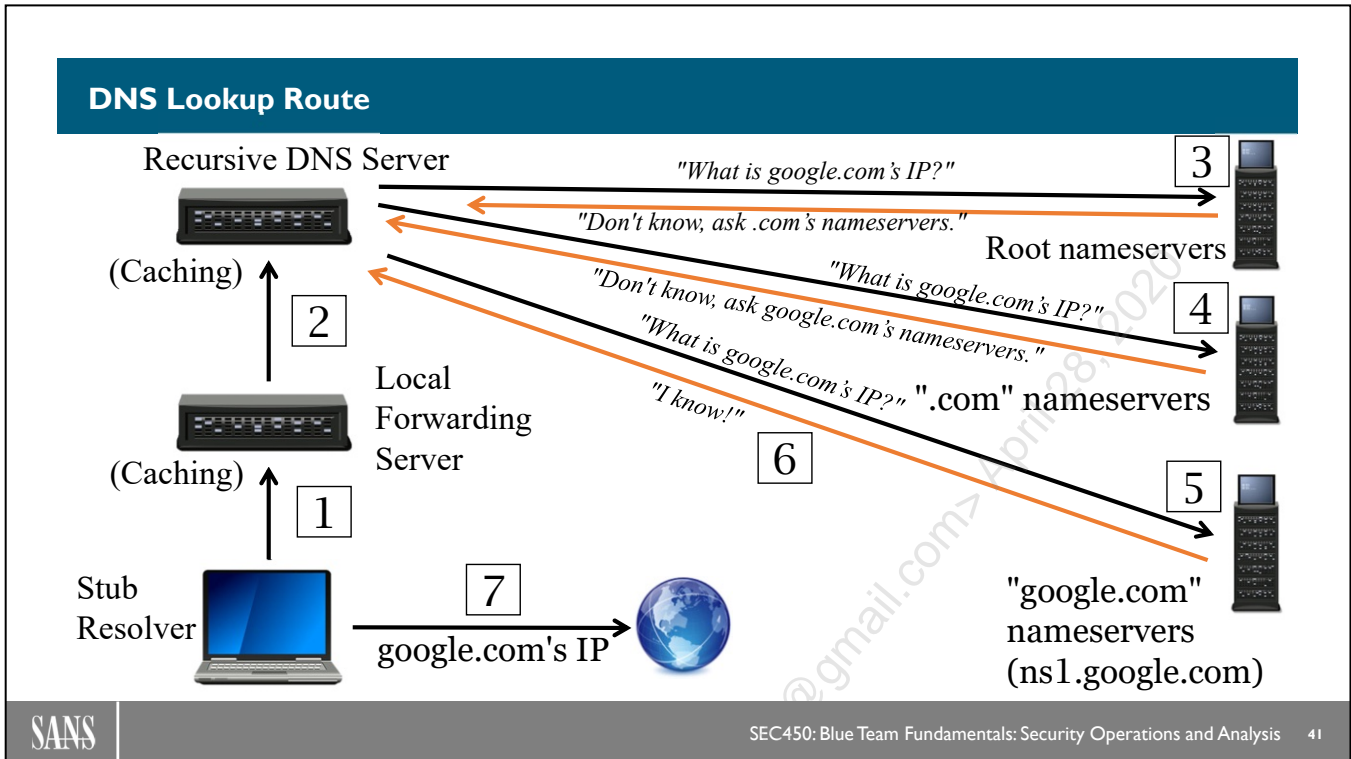
Forwarding Server: These servers typically sit local to the clients so that queries with known answers in the cache can be quickly answered from a local source. They will not perform recursion to the root servers. Merely toss the request on up the chain as well if it does not know the answer. They can also be used to filter queries that should be answered internally and redirect the query in the direction of an internal server.

Caching Server: These servers typically receive queries from forwarding type DNS servers and will return the answer from the cache if it is known. If it is not known, it will recurse all the way to the root servers to find the authoritative nameservers where it can get an answer (described on the next two slides). This is the role that DNS services like OpenDNS, Google's 8.8.8.8 or Cloudflare's 1.1.1.1 play. If you want to set up your own recursing and caching server at home, you can easily set one up with the excellent ad-blocking Pi-Hole project.¹

Authoritative Server: These servers do not recurse or cache. They hold the actual answers for the zones where they are authoritative zone. For example, Google's authoritative nameservers such as ns1.google.com have the definitive answers for all the records pertaining to Google such as mail.google.com, doc.google.com, and www.google.com.

[1] <https://pi-hole.net/>

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



DNS Lookup Route

What happens when you perform a DNS query? The answer depends on whether anyone else in your network using the same DNS servers has recently looked up that hostname. DNS, as a protocol, is meant to provide quick answers to queries that can be cached at various levels. If the servers you use, or any servers upstream of them have recently seen the answer to a query, they will cache the answer and provide it directly to the client. The length of time an answer is cached for is determined by the TTL value in the original response, a number representing the number of seconds the resolver should cache that response for. In this picture, if the Local Forwarding Server or the ISP Recursive server knew the answer, it could provide them without any of the other steps.

If the answer is unknown, however, a recursive resolver will need to reach out to the authoritative nameserver designated by the organization that owns the domain to ask for the answer. Finding that server requires queries to be made in multiple steps. The first query is to the root server "." to find the top level ".com." nameservers. The next one goes to the ".com." nameservers to find the "google.com." nameservers, and then finally the A record query is sent to the official Google nameservers to get the answer. DNS is designed this way such that every nameserver only needs to know where to point to find the next level down the chain, not the entirety of all domains on the internet. The root servers at "." know where to find ".com.", and the ".com." servers know where to find the servers for every [something].com, and from there, each site, like google.com, knows how to answer questions for its own domain such as www.google.com, or mail.google.com.

Once the authoritative nameserver for a domain is known, the DNS query can be sent to the domain's official nameservers and passed back down the chain from the ISP to the local forwarding server (where it is cached again) and ultimately back to the client that asked for it. This walkthrough hides some of the fine detail of all the queries that must be made for this process to work. A more complete explanation, as well as how you can do this on your own, is on the following slide.

DNS Request Types

DNS records come in different types¹:

<u>TYPE</u>	<u>Type Code</u>	<u>Meaning</u>
A/AAAA	1/28	A host address
NS	2	An authoritative name server
CNAME	5	The canonical name for an alias
SOA	6	Marks the start of a zone of authority
NULL	10	A null RR (EXPERIMENTAL)
PTR	12	A domain name pointer
MX	15	Mail exchange
TXT	16	Text strings
SRV	33	Specifies location of server(s)

DNS Request Types

Continuing the analogy from the previous slide, what if you wanted to send a fax to John instead of calling him? You'd need to look up his fax number. In the same way people have multiple phone numbers for different purposes, DNS has multiple record types for different uses. This slide shows the most common lookup types as well as the description of what they are used for (there are many more than this defined, but these are the ones that will matter most to you as an analyst). Technically, each type has a numbered type code that corresponds with it as well (A=1, TXT = 16), although you may see these from time to time, it's not as important to remember the types by number as it is to understand them by the name A, or TXT.

Let's walk through some of these lookups in more detail. We don't need to know everything about DNS, but there are some pieces of it that will be crucial to understand. In this section, we'll walk through those pieces. We start here because DNS is one of the most fundamental pieces of networking, and understanding the various request types, what they're used for, and how they can indicate suspicious behavior is a critical skill for any analyst to build.

[1] <https://tools.ietf.org/html/rfc1035>

A and AAAA Records

Source	Destination	Info
192.168.1.156	192.168.1.1	Standard query 0x0004 A youtube.com
192.168.1.1	192.168.1.156	Standard query response 0x0004 A youtube.com A 216.58.217.110
192.168.1.156	192.168.1.1	Standard query 0x0005 AAAA youtube.com
192.168.1.1	192.168.1.156	Standard query response 0x0005 AAAA youtube.com AAAA 2607:f8b0:4004:80d::200e

Translate hostname to an IPv4 (A) or IPv6 (AAAA) address

Important A & AAAA record fields:

- **Question:** What domain is being looked up?
- **Answer:** What is the IP address to the question (May contain multiple responses!)
- **Transaction ID:** Ties request to response
- **TTL:** How long the answer should be cached (in seconds)
- **Source / Destination IP:** Who asked? Who was asked?

A and AAAA Records

A and AAAA records are, without a doubt, the most common DNS request types that you will encounter. They're used every time someone visits a website to look up the IP for all domains and subdomains that are referenced in any of the page's resources. You might think when you go to a site like aol.com that most DNS lookups would be like `____.aol.com`. In this case you'd be wrong. Testing showed a single visit to aol.com generated 60 different A record queries alone to 58 unique hostnames, only 4 of which were `*.aol.com` and `*.aolcdn.com`. The rest were advertisers and sponsors. You can probably see how DNS traffic can be voluminous, and that's ignoring the 95 different AAAA record requests that were generated, too!

There are several important fields to be able to pick out for an A record lookup. One is that there is a query packet that contains the hostname being looked up, and an associated response with the IP address of that host (or an error). The first two lines above show a request for youtube.com and a response saying it's located at 216.58.217.110. Note that, for investigations, you will want to collect both the hostname and the IP in the response from the network. If you find that your organization only collects the hostnames in the request, know that you are denying yourself crucial information and should work to remedy that situation.

If you've sent a bunch of lookups at once, how do you map each request to a particular response? Easy, the transaction ID number that is attached to the request will match in the corresponding response. The youtube.com A record request and response above uses transaction ID 0x0004 and the AAAA record request uses 0x0005. How long will that response be cached for by the DNS client on the host that did the lookup (and the servers in between that facilitated getting the answer)? DNS provides an answer in the time-to-live (TTL) field (not shown) represented in seconds. If this is set to 5 minutes, for example, any client that asks this resolver for youtube.com for the next 5 minutes will receive an immediate cached response. Once this time is up, if another client asks for the same hostname, the request will need to be performed by the resolver again.

The final pieces of info that might be useful from the average A record lookup are the destination IP and TTL. The destination IP is Layer 3 info (not application layer) and tells you which DNS resolver the request was sent to. In the example above, the request is sent to 192.168.1.1 from 192.168.1.156. This doesn't necessarily mean 192.168.1.1 knew the answer already. It may need to turn around and look it up itself. DNS is a recursive system as we'll see in a moment, so sometimes one request to be resolved causes multiple others to be sent out. This also means that if you are monitoring DNS "upstream" of user devices, between one server and another, the source IP itself may not be the actual device that sent the request. This is an important detail to keep in mind. but it is the name of the server that caused the answer to be found for you.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

A Record Mapping

You look up hostname *x.com*, you...

- Might get no answer (NXDOMAIN)
- Might get one answer
- Might get multiple answers

You also...

- Might get a new answer on the second resolution
- Might look up *y.com* and find it has the same IP address
- Might find the PTR record does or does not point back to *x.com*

Google.com A record lookup

74.125.21.100
74.125.21.139
74.125.21.102
74.125.21.101
74.125.21.113
74.125.21.138

Passive DNS lookup

Country
Autonomous system 15169 (Google Inc.)

Passive DNS Replication

Date resolved	Domain
2018-10-19	safebrowsing.clients.google.com
2018-10-16	shigjeta.net
2018-10-09	books.google.com
2018-09-26	staging-pledgebrite.org
2018-09-26	accountchooser.biz
2018-09-26	dronesnearby.com
2018-09-20	encuestadecomunidad.com
2018-09-19	tools.google.com
2018-09-13	nvitmedia.com

Also resolved to this IP!

A Record Mapping

There are several other important things to know about A records:

- Answers may shift over time, or with each individual lookup.
 - As we saw in the section on setting up records, what might be set to the IP address could easily change tomorrow, and this is very much true of malware. This is why recording DNS responses is important. You want to know the IP address a site mapped to at the time of the suspicious activity, not what it is right now, because it may have changed.
 - Shifting for each individual lookup is also done for load balancing and is called "round-robin DNS". Person A gets the IP address of server one, person B gets the IP address of server two. If both servers are identical, you have effectively load-balanced a website using DNS.
- There may be multiple answers for one hostname. This is usually done for load balancing purposes. A web browser that looks up a hostname and receives four IP addresses in response will try all four in turn until one is successful. This is similar to the previous approach but gives all the potential answers at once. When doing this, the order of the four IP addresses given out may change per request as well to get the best of both worlds.
- Logically following from the previous item, there also may be many different hostnames that point to the same IP address. If you wanted to host a site on IP address 9.9.9.9, you could buy as many domains as you wanted for it and create A records for all of them that resolve to that IP address.—subdomains as well. There is no limit to the number of domains that can point to a single IP address. This means the A record mapping of hostnames to an IP address is many-to-many. To find this out, there are two ways to go backward from an IP address to domain names. The "official" is a PTR record lookup (covered next), but there are also sites that record all A record lookups over time and let you search by what was seen in the past. This is called "Passive DNS" because it is not making an active query, but giving you an answer based on lookup history from their database instead.
 - VirusTotal.com is a great example of this. In the screenshot above, google.com is looked up, which resolves to multiple IP addresses, but when you take one of those IPs and put it back into VirusTotal,

it gives you back all domains it ever looked up that mapped to that IP instead of doing a PTR lookup. As you can see, the results are not quite what you might expect. Other sites that aren't Google have pointed to Google's IP as well! This demonstrates the "many-to-many" mapping nature of A records and IP addresses.

- There may be no answer for a hostname. If a hostname A record does not exist, an "NXDOMAIN" response will be returned to indicate there is no record matching the query.

If you want to see a visual display of DNS records, type almost any well-known domain into the site Robtex.com and look at the graph output. You'll get an idea just how many different records are in use on a complicated site.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

PTR Records

- Pointer records, "reverse phone number lookup" of DNS
- Maps an IP address back to a domain...maybe
- Slightly more confusing than other types

```
>nslookup 8.8.4.4
Server:  one.one.one.one
Address:  1.1.1.1
```

```
Name:      google-public-dns-b.google.com
Address:   8.8.4.4
```

No.	Time	Source	Destination	Info
4980	08:33:39.108333	192.168.1.156	1.1.1.1	Standard query 0x0011 PTR 4.4.8.8.in-addr.arpa
4983	08:33:39.207215	1.1.1.1	192.168.1.156	Standard query response 0x0011 PTR 4.4.8.8.in-addr.arpa PTR google-public-dns-b.google.com

PTR Records

PTR records are quite a bit different compared to other resource record types. Back to our phone analogy—have you ever done a reverse phone number lookup? Perhaps you had a prank phone call and wanted to know who it was and put the number into Google to find out if it can be mapped back to a name. The PTR record is the same thing, only for DNS.¹

PTR requests send an IP address in the request and the resolver attempts to find a hostname associated with the IP. The thing is, there are some "gotchas" involved in PTR lookups. One issue is that if you are looking at the traffic for a PTR lookup, the IP address is not displayed as you would expect. In fact, it's backwards! Look at the nslookup command used to look up 8.8.4.4 above vs. the Wireshark screenshot of the actual request that was made. See how it says "4.4.8.8.in-addr.arpa"—that is the actual lookup format for a PTR reverse lookup. It doesn't merely send something like "PTR 8.8.4.4" the way an A record request looks. As you can see, PTR records must use a special domain—in-addr.arpa, a separate namespace created for this functionality, which also means these records must be kept in a different zone file. These lookups involve the clever use of CNAME records (covered in a moment) in the in-addr.arpa domain as described in RFC 2317 to make this work.²

The important thing you must take away here is that if you are looking at a PTR lookup, the address that you are seeing is actually the reverse of the IP that was being checked, and you will need to flip it before you use it for analysis! In this case, our PTR record lookup worked great, but that won't always be the case.

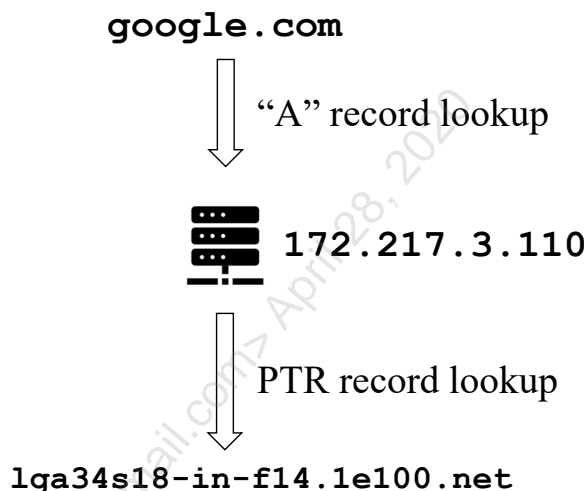
[1] <https://tools.ietf.org/html/rfc1912>

[2] <https://tools.ietf.org/html/rfc2317>

When PTR Records Don't Work

Remember the many:many relationship of A records?

- Records may not resolve the same forward/backward
- IP may not resolve to anything
- Backward lookups may also change over time
- **Conclusion:** Reverse IP lookups may not be useful, good for single point in time



When PTR Records Don't Work

RFC 1912 recommends that "every internet-reachable host should have a name," and that "for every IP address, there should be a matching PTR record." You may not always find this to be true.¹ Clearly, PTR record lookups have some issues you need to be aware of as well. One is that if you take a hostname like Google.com and resolve it to an IP, then try to take the IP and resolve it back to a hostname, you may not get google.com back. Why is this? Remember the many:many mapping of A records to IP addresses—this is a two-way street. Since many hostnames can map to one IP, which domain should the IP return when a PTR request is made? The answer is whatever name or names the owner chooses. Often, this will match, and when it does, it is called "forward confirmed reverse DNS." This is considered best practice, but it certainly will not always be the case. That means when you do a reverse lookup, unless there is a relationship from hostname to IP and the owner has set up the PTR record to map backward as well, you may not receive the same answer going forward and backward. This is shown on the slide for a google.com A and PTR lookup.

The conclusion here is that PTR record lookups on IP addresses are mediocre at best and won't *necessarily* be helpful. If you were to do a reverse lookup on 172.217.3.100, which is pointed to from an A record for google.com, you would receive lga34s18-in-f14.1e100.net in response. What the heck is that? It's most likely the specific hostname of the device with that IP address in the google datacenter. Why doesn't it even end with google.com? Because Google didn't set it up to, although it does end in 1e100—Google's subtle way of referring to the number they conceived their name from, a googolplex or 1×10^{100} , or "1e100" in scientific notation ... clever.

[1] <https://tools.ietf.org/html/rfc1912>

TXT Records

No.	Time	Source	Destination	TXT	Info
1358	08:16:24.335156	192.168.1.156	1.1.1.1		Standard query 0x000d TXT sec450.com
1366	08:16:24.346397	1.1.1.1	192.168.1.156	Hello world!...	Standard query response 0x000d TXT sec450.com TXT TXT

- Used to associate **arbitrary text** with a hostname
 - Only printable ASCII allowed, 255 character limit
- Multiple TXT records allowed for each hostname
- RFC 1464 suggests <key>=<value> pair formatting
- Often used for spam prevention
 - Sender Policy Framework (SPF)
 - Domain Keys Identified Mail (DKIM)
- Can also be used for evil...

TXT Records

TXT records are a commonly used resource record type. They are designed to hold an arbitrary text value containing printable ascii up to 255 characters in length. TXT records are used whenever there is some bit of information about a domain that needs to be passed to a client over DNS protocol. As you might imagine, this ability leads to potential abuse of this functionality, but we'll hold off that discussion for now. TXT records are most often presented in unquoted key value pairs which is suggested in RFC 1464. This is not a requirement, as you can see in the screenshot of the text placed in sec450.com's TXT record.

TXT records are most often used for activities such as spam filtering and proving that you own a domain. For spam filtering, the SPF framework is used. SPF TXT records start with "spf=" and what follows is a whitelist of IP addresses or machines that are allowed to send email from that domain. DKIM is another popular spam filtering technique that is implemented through TXT records. The short story on DKIM is that you post a public key in a TXT record that will be used to sign all outgoing mail. Then, anytime another organization receives mail from you, they can do a TXT record lookup to pull your public key and verify the mail was indeed signed by you, meaning it was not spoofed. For proof of domain ownership, sometimes companies, such as those that issue SSL certificates, will have you prove ownership of a domain by placing a customized TXT record in DNS. Here's an example of an ownership verification to Facebook line from Apple's DNS record.

```
"apple.com IN TXT facebook-domain-
verification=n6cqjfucq6plswmtfbwnbbeulqiq3v"
```

The screenshot above shows a TXT record query to sec450.com where there is a corresponding record set up with the value "Hello world!", which can be seen in the TXT column (a non-standard Wireshark column, but one that was applied for this demo).

CNAME Records

No.	Time	Source	Destination	Info
44	07:54:38.292300	192.168.1.156	1.1.1.1	Standard query 0x0003 A www.sec450.com
45	07:54:38.300453	1.1.1.1	192.168.1.156	Standard query response 0x0003 A www.sec450.com CNAME sec450.com A 1.2.3.4

A pointer from one host name to another

- A redirect from one hostname to another
 - If you type "sans.org", can send you to www.sans.org
 - A record request results in CNAME response + A record
 - May perform a second query to the canonical name (CNAME) in response
- **If one host runs multiple services**, allows multiple names
 - sec450.com pointed to by www.sec450.com, ftp.sec450.com, etc.
- Can, but shouldn't nest CNAME records, might make loops!

CNAME Records

CNAME records are one of the easier types of records to understand. They simply map one hostname to another true name for that host (the canonical name). In the slide above, www.sec450.com returns a CNAME response saying the canonical name of that host is actually just sec450.com, and that the IP of THAT host is 1.2.3.4. This is most often used as a convenient way of making multiple names for a single host that may run multiple services.

Although you can have CNAME records that do multiple levels of redirection, it is not considered best practice since it can lead to accidental loops such as what would be created if the following entries were used on sec450.com. The request would never resolve since each item points to the other.

a.sec450.com. CNAME b.sec450.com.

b.sec450.com. CNAME a. sec450.com.

MX Records

Points to the mail server for a domain

When sending mail:

1. MX record of the domain is queried
Example: MX record for `sec450.com` = `mail.sec450.com`
 2. DNS “A” record lookup performed for the hostname in the answer
 3. SMTP connection established to that IP address
- **MX record must contain an FQDN**
 - FQDN must map directly to A record (CNAME not allowed)
 - Lists a numerical **priority** for server selection
 - Lowest number is the most preferred server



MX Records

The MX record is the next resource record in the list of the most common. This record will likely not play a part in your day-to-day life, but it is important to know what it's for in case you do see it. MX records are used when an email is sent to a given domain. If you send email to `sec450.com`, for example, a DNS MX record lookup is performed by the server sending the mail to find out which host handles email for the remote domain. The answer it will receive is the hostname “`mail.sec450.com`”. The server must then turn around and do another request for the A record of `mail.sec450.com`. The answer to that query is the IP address the sending mail server would connect to using SMTP to send the mail. According to RFC 974, an MX record must contain an FQDN that has an A record associated with it, not an IP address or a hostname with a CNAME record.¹

[1] <https://tools.ietf.org/html/rfc974>

SRV Records

Help clients find a service by name and protocol

- Use an `_` prefix for each to not collide with hostnames
 - Uses the format `_service._proto.name`
 - `_ldap._tcp.sec450.com` = LDAP TCP service lookup
 - SRV record holds the hostname and port number of server
- Most likely will see in Active Directory environments
- Won't be as useful as other types, still good to know
- Should only be visible to internal network

SRV Records

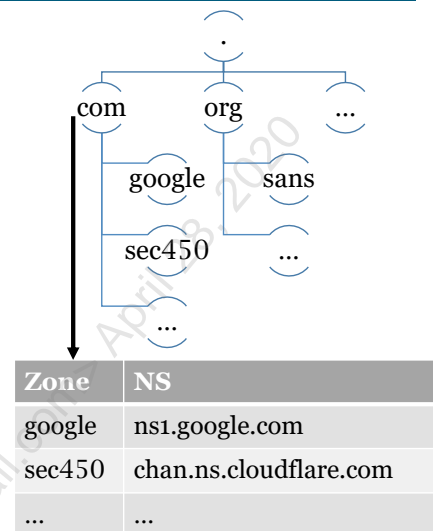
The SRV record is another DNS query type that probably won't play a role in your day-to-day security analysis. Given their odd-looking nature, however, it is still useful to understand what they are for and how to interpret them. This type of record is used by clients to locate services on a network via DNS. To differentiate themselves from overlapping with actual hostnames, the resource record names they use in the request include an underscore as the first character in the form of `_service._proto.name`. These two fields will explain what service the client is looking for by name, as well as the protocol. The slide shows a lookup for an LDAP service running over TCP using the `_ldap._tcp.[domain]` format. The answer will include, among the more common fields like TTL, class, priority, and weight, the name of the server that hosts that service, as well as the port that it runs on.

NS Records

Identifies *name* of DNS server, not IP address

Can be confusing, stored in 2 locations

- In **TLD zone file**, used for **delegation**
- In your own DNS server – authority records
- Hierarchical nature means:
 - "." root zone holds NS records for all TLDs (.com, .net, etc.)
 - ".com." zone holds NS records for all [anything].com sites (see table)



NS Records

NS records designate the name of the authoritative DNS server with all the records about a given domain resides. NS records are actually specified in 2 locations, however, one "authoritative" location—on the nameservers specified in the NS record (the authoritative answer)—and one copy one level up on the global top-level domain (TLD) nameserver (for delegation).

One of Google's nameservers, for example, is at ns1.google.com, so you can ask the server at ns1.google.com for A records for google.com, mail.google.com, docs.google.com, etc., and you will get the authoritative answer. How do we know these are the nameservers for google? We can do an NS record DNS query for google.com and we will see it listed as a result. Think about where this response came from for a second. If we don't know ahead of time that ns1.google.com exists as the nameserver, which we presumably don't otherwise we wouldn't be asking for it, then we must be able to source the answer from some other location. The other location is the second copy of the NS records on the .com TLD zone nameserver (illustrated in the slide above). The .com nameserver knows about all registered domains through communicating with domain registrars, and those who purchase a domain must separately enter the nameserver information so that it can be placed into the NS record at the TLD nameserver to populate its zone file. In other words, when you register a domain, you must fill in the NS records for your domain, (a process separate from making NS records on your own nameserver), and these records are then sent to the TLD zone above it to bootstrap the process of finding your nameservers.

Since this can be confusing, let's show an example. Running the query below asks the .com zone nameserver (one of which is located at a.gtld-servers.net) to list all the nameservers it knows about for the google zone, which returns with ns1.google.com, ns2.google.com, etc., as shown on the next page.

```
$ dig google.com NS +norec //note this query is being sent to a.gtld-servers.net, the .com zone nameserver
```

```
;; flags: qr ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 9
```

```
;; ANSWER SECTION:
```

```
google.com.          5          IN         NS         ns2.google.com.
google.com.          5          IN         NS         ns4.google.com.
google.com.          5          IN         NS         ns3.google.com.
google.com.          5          IN         NS         ns1.google.com.
```

```
//We can then query that server directly for the docs.google.com subdomain IP address
```

```
$ dig docs.google.com @ns1.google.com A
```

```
;; ANSWER SECTION:
```

```
docs.google.com.    300        IN         A          172.217.10.78
```

Note in the flags section in the NS query above we do not see the "aa" flag. This means that the request did not receive an "authoritative answer" for the nameserver records. Why was it not authoritative? Because the .com nameserver is not the authoritative server for the google.com zone, ns1.google.com is. Now that we know it, we can re-ask the question directly to ns1.google.com by specifying it on the command line and see how the response differs and now includes the aa flag.

```
$ dig google.com @ns1.google.com NS +norec
```

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> google.com @ns1.google.com NS
```

```
;; flags: qr aa; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 8
```

```
;; ANSWER SECTION:
```

```
google.com.          345600     IN         NS         ns2.google.com.
google.com.          345600     IN         NS         ns4.google.com.
google.com.          345600     IN         NS         ns3.google.com.
google.com.          345600     IN         NS         ns1.google.com.
```

Again, this is why NS records can be a bit confusing, there are actually 2 zone files with NS records concerning google.com in use here (the TLD nameserver copy used for delegation and Google's own nameserver copy as an authoritative answer). The .com TLD nameservers must be told separately about Google's nameservers at ns1.google.com because they redirect all questions they receive about google to those servers. They must do this because they are not authoritative for the google.com zone and do not have the rest of the records the authoritative nameserver has. This is DNS delegation in action.

The information that google.com uses ns1.google.com as an authoritative nameserver *also* exists in the zone file on ns1.google.com, which is the authority record, and is the source of truth for all DNS records inside the google.com zone. The NS records on ns1.google.com are not where the nameserver location is sourced when a recursive DNS resolver is trying to resolve a google.com address, however, as shown in the first example above.

When an A record query is made for google.com, the nameservers listed in the .com zone are the ones that are used to direct the A record to. This flow of information will be explained further in the next few slides.

Quiz: DNS On The Wire

No.	Time	Source	Destination	Info
1	11:15:32.126353	192.168.1.156	1.1.1.1	Standard query 0x001a A sec450.com
2	11:15:32.216490	1.1.1.1	192.168.1.156	Standard query response 0x001a A sec450.com A 1.2.3.4 A 162.255.119.240
3	11:15:43.124817	192.168.1.156	1.1.1.1	Standard query 0x001c CNAME mail.google.com
4	11:15:43.133356	1.1.1.1	192.168.1.156	Standard query response 0x001c CNAME mail.google.com CNAME googlemail.l.google.com
5	11:16:27.600721	192.168.1.156	1.1.1.1	Standard query 0x001d PTR 27.208.194.173.in-addr.arpa
6	11:16:27.642776	1.1.1.1	192.168.1.156	Standard query response 0x001d PTR 27.208.194.173.in-addr.arpa PTR ql-in-f27.1e100.net

Questions:

1. What domain was resolved via A record request?
2. What IP(s) did it resolve to?
3. What is the canonical name for the Google mail host?
4. Which IP address was looked up via PTR record?
5. What DNS server was used for these queries?

Quiz: DNS On The Wire

Here's a brief quiz to check if you understood the types that were discussed in the previous slides. Can you answer these questions? The answers are on the following page.

Answers: DNS On The Wire

3 DNS Queries were made – A, CNAME, PTR

1. sec450.com was resolved to 2 different IP addresses
 - 1.2.3.4 and 162.255.119.240
2. CNAME query showed that mail.google.com's canonical name was googlemail.l.google.com
3. PTR query performed reverse lookup of 173.194.208.27
 - Resolved to a 1e100.net domain – Google owned
4. Cloudflare's DNS service at 1.1.1.1 serviced the requests

Answers: DNS On the Wire

Here are the answers to the previous slide's questions:

1. In packet 1, an A record query was made for sec450.com.
2. In packet 2, the answer was received, which was actually 2 different IP addresses (remember this is possible). The answers were 1.2.3.4 and 162.255.119.240.
3. In packet 3, a CNAME query for mail.google.com was made. Packet 4 returned the response saying the canonical name of mail.google.com was googlemail.l.google.com. Remember, the canonical name is the name in the response, not the name that was sent.
4. In packet 5, a PTR request was sent. Recall that the order of octets is reversed for a PTR lookup, therefore, the person who issued this query was attempting to reverse resolve the IP 173.194.208.27. In packet 6, the answer came back, which was ql-in-f27.1e100.net, a Google-owned host. This IP address was actually sourced from doing a MX lookup for gmail.com, which returned gmail-smtp-in.l.google.com as the mail server host. Looking up this hostname with an A record request gave the IP 173.194.208.27, so in actuality, this was a reverse lookup for the host that accepts messages bound for gmail.com addresses.
5. The destination IP of all the outbound queries was 1.1.1.1, this is a public DNS server run by Cloudflare similar to Google's 8.8.8.8 service.

Understanding DNS Summary

DNS is extremely important to understand

- Resource records set in a zone file, hierarchical nature
- A records: More than one IP per hostname possible
- PTR records: More than one possible, might not go backwards and forwards the same
- TXT records: Arbitrary text, used for spam reduction
- CNAME: Pointer to another hostname
- NS: Name of the authoritative DNS server

Understanding DNS Summary

Although DNS may seem like a rather basic idea, as you can see, the details can get fairly complex and there are multiple nuances that analysts need to be aware of to analyze DNS traffic. In this module, we went over why we care about DNS, how it functions, how it is set up, and the meaning of the different types of resource records, as well as how to interpret them. In the next section, we'll dive into how to look at DNS traffic with a critical eye to pick out malicious traffic and understand how attackers use it to their advantage. Remember, DNS is important to us because when we browse the internet, we don't connect to a domain name, we connect to an IP, and that IP is determined by DNS.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. **DNS Analysis and Attacks**
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

In This Module

- Detecting evil DNS requests
- How DNS can be used and abused by attackers
 - Custom servers, DNS Tunneling, IDNs
- Detecting when DNS plays a part in an attack
- Key data/fields to use and store
- A records vs. PTR records vs. passive DNS lookups
- How DNS can become a vulnerability and how we hope to fix it

In This Module

In the previous module, we explained how DNS is *supposed* to function. In this module, we will discuss how attackers use it for their own purposes and how to spot when they are doing so.

DNS Gone Bad

Identifying signs of evil with DNS:

- Requests for malicious sites
- DNS infrastructure compromise (shadowing)
- DNS itself used in malicious ways
 - Discovery: Brute forcing, zone transfers
 - Unauthorized external server use
 - Tunneling
 - Exfiltration



DNS Gone Bad

What way can DNS be used for evil? In this section, we will cover 2 categories. The first is normal usage of DNS, but looking up malicious sites. In other words, how to identify potentially compromised machines purely from a DNS request. And the second, DNS itself used for carrying out malicious activities such as reconnaissance before an attack or data exfiltration.

Detecting Requests for Malicious Sites

TLDs: Some are just more likely evil

Get one of these domains. They are free!

sec450 .tk	• FREE	USD 0.00
sec450 .ml	• FREE	USD 0.00
sec450 .ga	• FREE	USD 0.00
sec450 .cf	• FREE	USD 0.00
sec450 .gq	• FREE	USD 0.00

Free domains

Rank	TLD	Percentage of Shady Sites* (All Time)
1	.country	99.94%
2	.stream	99.79%
3	.download	99.58%
4	.xin	99.41%
5	.gdn	99.40%
6	.racing	99.30%
7	.jetzt	99.16%
8	.win	98.92%
9	.bid	98.87%
10	.vip	98.76%

New TLDs

Detecting Requests for Malicious Sites

Before we dive into the ways that DNS can be used maliciously as a protocol, let's first discuss some of the basic analysis techniques for evaluating whether any given DNS request is potentially malicious.

One item that can initially be used to up the risk factor is the top-level domain (TLD). Is the site a .com, or is it .tk, .ml, or other free domain – malware authors love to use these domains because they're completely free. Seriously, if you want a free domain name, head on over to dot.tk and you can register anything you like for no cost at all.¹ The screenshot above shows sec450 as a parent domain is available for free with 5 different TLDs. The fact that some TLDs are statistically more dangerous than others is a well-known fact. In the second photo is a copy of Symantec's 2017 study of the "Top 20 Shady Top-Level Domains"² (note these will surely change over time; this is just a snapshot). For domains like this, you can merely alert on *anything* at all using these TLDs, and with no other logic, have a pretty low false positive rate!

There are other ways to identify malicious sites via DNS queries, though some take extra work that your SIEM can perform automatically for you. Nonetheless, having merely a domain name can lead to surprisingly high-fidelity detections. In the next few slides, we'll explain how to use parent domains, subdomains, and IP reputation to determine whether a site may or may not be malicious.

[1] <http://www.dot.tk>

[2] <https://www.symantec.com/blogs/feature-stories/top-20-shady-top-level-domains>

Domain Reputation and Age

Reputation / Category

Both **category** and **reputation**

Lots of lookup options:

- virustotal.com
- community.riskiq.com
- talosintelligence.com/reputation_center
- urlquery.net
- threatcrowd.org
- trustedsource.org

URL	Status	Categorization	Reputation
http://opm-learning.org	Categorized URL	Malicious Sites	High Risk

Age

New domains = more likely bad

Tools:

- `$ whois [domain]`
`$ whois sec450.com | grep -i creation`
 Creation Date: **2018-06-29T09:38:23Z**
- Mark Baggett's domains_stats.py¹

DNS Research Sites:

- Centralops.net
- DNSStuff.com
- Who.is

Domain Reputation and Age

Reputation:

There are many services available both on the internet and through vendors that offer reputation services for domains. A simple search of one of the many options on the slide or beyond may give you a hint whether a domain is categorized in one category or another (business, travel, etc.), and whether it is known to be malicious. The information in the table is an example lookup of opm-learning.org—a site associated with the OPM breach. Trustedsource.org, a site that is associated with McAfee's Web Gateway proxy service, labels this site as high risk and in a category of malicious sites. Since this is a known APT site, this is about as clear of a warning as it gets, but other responses may be less obvious, such as a category of "business" and medium risk. These services are not a perfect solution, but another source that can be considered for determining the truth about a domain.

Age:

The age of a website can be a great additional piece of data to decide whether a site is malicious or not. Many times, attackers will register malicious sites very close to their use in an attack. As a defender, if you aren't sure about a domain and find it was created days or weeks ago, that is another vote for the "potentially malicious" category. Although this obviously isn't a perfect test, reputable sites often have been around for a longer period. To learn the creation date of a website, you can either run a query with the whois tool on Linux or use one of the many online services that will look up the information for you via a web interface. To get even fancier, tools like Mark Baggett's domain_stats.py¹ can be used either as a command line, or even as a server to implement these checks as a built-in enrichment for SIEM data!

[1] https://github.com/MarkBaggett/domain_stats

A Quick Note on Whois

- Maps a domain name to a person/organization
- For defenders, it's a useful source of data to
 - "**Connect the dots**" when a malicious actor registers multiple domains
 - Find the **creation date** of a domain name
- Works a bit like DNS
 - Each registrar keeps info of their own customers
 - **Details often hidden** (especially after GDPR)
- Information recorded depends on the TLD
- Lookups are rate limited

A Quick Note on Whois

Whois¹ is a system for tracking the contact information for the owner of a domain name. When purchasing a domain, the owner is supposed to fill in their mailing address and other details, which the registrar keeps in case they need to get in touch. This information has also traditionally been made available to the public and served as a good source of data for tracking the same person or company (or malicious actor) registering groups of domain names. Unfortunately, most domain owners now opt to have their registrars hide the information so that spammers and OSINT researchers can't get a hold of it, and the GDPR regulations that recently have been enacted in Europe continued to push that trend.

As defenders, whois information is useful because when an adversary registers multiple malicious domains over time, they are often lazy and use similar contact info or other information that can help us connect the dots, assuming the information isn't hidden. Whois info also helps us scope compromises because if we see a connection to evilsite.com, and whois information tells us the site was created 1 week ago, there's no point in wasting time searching for command and control before that date since we can assume it would've been impossible. When performing research on domain names, do not forget to use whois information to fill out gaps in threat intelligence.

[1] <https://www.namecheap.com/domains/how-does-whois-work/>

Domain Randomness and Length

Easy rule: *Parent* domain **randomness is almost always bad**

- Malware uses domain generation algorithms to avoid takedowns
 - `aafgcvjyvxslosy.com`, `aarbvsrdnhhidhwk.com`, `absqvhpldvsmclt.com`¹
 - False positives exist, but are relatively rare

Length of domain:

- What is the longest domain you've ever typed?
- Longer domains = more likely procedurally generated, bad



Domain Randomness and Length

One easy heuristic to apply to domains is the apparent randomness and length of the name. Businesses buy domain names so that people can easily type them, and with few exceptions that are only used for programs on the backend, domains tend to be human-rememberable and easily typable. Malware often uses random domains to avoid signatures and implements this through what is called a "domain generation algorithm." Each day, the malware will try to communicate with a new, but deterministically generated hostname. The day the attacker wants to send commands to the malware, they will pre-register that domain. Using a domain generation algorithm (DGA) makes it extremely difficult to block the malware by domain or predict when the next command will come or where it will come from.

Therefore, any domain you run into, short or long that appears to be randomly generated, is highly likely to be malicious. SIEMs and other tools that are capable of measuring and detecting apparent randomness do well at finding active malware with this tactic alone. Note that this doesn't necessarily apply to subdomains, just parent domains. There are some reasons we will want to pay attention to long subdomains as well, but there are many content delivery networks and other reasons out there to create long subdomains. We'll come back to that idea in a bit...

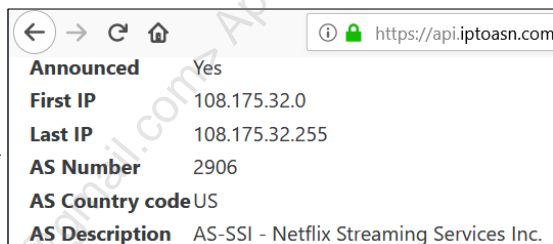
One humorous exception to this rule is the town of *Llanfairpwllgwyngyllgogerychwyrndrobwlllantysiliogogoch* in Wales, which does happen to have the distinction of owning the world's longest single word domain name: `llanfairpwllgwyngyllgogerychwyrndrobwlllantysiliogogoch.co.uk`.¹

[1] <https://en.wikipedia.org/wiki/Llanfairpwllgwyngyll>

Detecting Based on DNS Response: IP-Based Reverse Lookups

Questions to ask when you have the DNS A record response:

1. Is there a **PTR** record? Does it help at all?
2. What domains do **passive DNS** sources say point to it?
3. **OSINT**: Google, VirusTotal, ThreatMiner, etc.
4. **Autonomous System Number (ASN)**: Who owns the IP?
 - Example: Alert for IP 108.175.32.110
 - No OSINT data, who owns it?
 - iptoasn.com shows Netflix!
5. Shared hosting?



Field	Value
Announced	Yes
First IP	108.175.32.0
Last IP	108.175.32.255
AS Number	2906
AS Country code	US
AS Description	AS-SSI - Netflix Streaming Services Inc.

Detecting Based on DNS Response: IP-Based Reverse Lookups

Now let's say you know the IP address of a domain from a DNS log and want to start there. Can you get an idea if traffic is malicious or not based purely on the IP? Maybe...

Depending on what the IP is and how well it is known on threat information sites, you may be able to look up a PTR record that definitively points the IP as being associated with a malicious domain. That won't always work, so the other option is reverse DNS lookups, which are like passive DNS, but instead of starting with a hostname, they start with the IP address and consult a database of what domains have been previously looked up that point to it. Of course, the typical "Google it" advice applies here, as well as putting it in to open source threat intel tools (including your own). You might be able to gain some perspective if someone has previously classified it.

Let's say you have an alert for IP address 108.175.32.110. Currently, a search for this domain on VirusTotal gives no results and there are no PTR records available, either. OSINT has turned up nothing useful. Where else can you go? Another option is to look at the Autonomous System Number (ASN). What is an ASN? It associates an IP with the actual organization that owns the IP space, and can be found with tools like centralops.net's Domain Dossier, or any other ASN lookup tool. In the case of the IP mentioned, an ASN lookup would identify it as part of Netflix's infrastructure, something that is highly unlikely to attack us.

ASN lookups will not necessarily be able to tell you that something *is* evil but can be useful to give context and say something is *less* likely to be evil given its association with a known good organization. Be careful with this. Just because you know the company who owns the ASN doesn't mean it's not bad. All Amazon AWS servers will show up as an Amazon ASN, but people use those for attacks and penetration tests all the time. If an organization offers private servers to the public, be wary of "good" sounding ASN organizations. Attacks may still originate from them.

Shared Hosting

- There are 2 main ways to establish a website
 - Buy yourself a private server
 - Share a webserver with many others, get access to one folder only
- **Takeaway:** Check domains for shared hosting before blocking
 - Use a reverse DNS lookup tool to map IP to domains
- **Analysis implications:**
 - If shared hosting, consider **blocking by domain, not IP address**
 - Unlikely, but may block something else that was needed
 - Threat intel list may have had IP due to a different domain being bad on same IP – **frequent false positives**

Shared Hosting

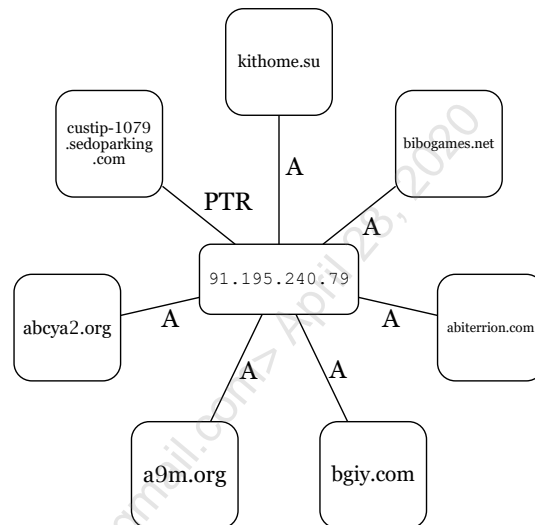
One concept we should explain before going into detail on identifying malicious websites is that of shared hosting. If you want to put up a website, there are 2 main options for doing so. If you are running a professional operation and know your site will need a considerable amount of resources, then you would likely buy your own dedicated server. If you are only running a very small site without much traffic, or an attacker hosting a small amount of malicious content, it's easier to share a server with others. When sharing a server, the attacker's site, as well as the site of many others, will all have the same IP address. This means that if you see an alert for a malicious IP being contacted and you do not know the context, it is worth checking if that IP is a shared web hosting environment. Since DNS is all about resolving hostnames to IPs, and many hostnames can point to one IP, you may find that many websites use that same IP address.

Shared Hosting Example

Example: You see an alert for a DNS lookup for `cuzb.com`

- A record = `91.195.240.79`
- What else maps to `91.195.240.79`?
- **5,857 other sites**, according to rev. IP lookup on `hackertarget.com`!!

a9m.org	better-mankind.com	kisahviral.org
aaron-herford.org	bettingthrill.com	kithome.us
aarpr.com	betvui.com	kizi100.games
abajarm3.com	bevrani.com	kizi2017.net
abcya2.org	beyond2.org	kizi2017online.org
abidjanjeux.com	bfpbc1.r04.org	kizi2.games
abiterrion.com	bgiy.com	kizi4school.org
abrimicorreo.com	bhairavayoga.com	kizlarinegitimi.org
abtoa.r04.org	bharatproperti.com	kkkc.u.com
abusecenter.org	bibogames.net	kksj.pw
acesetm.net	bigjfencinginc.com	kleez.com
acg18.me	bigsoviet.org	klik3r.mwb.im
achingao.net	bigtdallas.com	klipovito.com



Shared Hosting Example

To demonstrate an example of this, pretend you had an alert fire for someone going to domain `cuzb.com` (a spammy looking site). This type of alert, based purely on a domain or IP that was on a blacklist somewhere, can be a very common occurrence. What steps should we take when analyzing this? In general, we'll want to see if the domain is truly bad or not, and if it is, ideally block both the domain *and* the IP address. The theory behind this is that if this is a private server owned by the attacker and there is more than one domain pointing to that IP address, then it would be more effective to block it at the IP level instead of just the domain. When we look at the IP of the domain, we see `91.195.240.79`, taking this information back to a site like `virustotal.com`, `hackertarget.com`,

On the flip side, the opposite may occur. You may have an alert for a "known bad" IP address that was accessed. Upon looking at the actual traffic, you find it is one of the many domains that seem to be pointing to an IP address where many other sites are hosted. Should you block the entire IP? Doing so will effectively stop everyone in the organization from reaching all those websites. Most times, shared hosting sites are not important for business (if they were that crucial, the site wouldn't be on shared hosting), so you may be able to get away with it. But consider the fact that if you block the whole IP, you will in effect be potentially blocking "good" sites as well if they are co-located.

Attack: DNS Discovery

Put yourself in the shoes of an attacker...

- You can't attack a host you don't know exists
 - How can we discover hosts? Nameservers!
 - Goal: Find A records, but how do we get **all** of them?
- Options, in order of sneakiness:
 1. Passive DNS & OSINT: Use Google, VirusTotal, Shodan (sneaky, but possibly incomplete)
 2. Zone transfer: Try to get the server to give them to you (unlikely)
 3. Brute Forcing: Run a dictionary of words against nameserver as subdomains – mail.x.com, vpn.x.com, ftp.x.com, ... , etc.

Attack: DNS Discovery

In the early stages of an attack, recon must be done to understand what hosts are present in a network. It may be an attacker trying to understand what systems a company has on the internet, or something that is done once the attacker is inside, but regardless, you can't attack something you don't know exists. One great way for an attacker to find out if something exists is to discover the A record for it in the nameserver. The problem is their options for doing so aren't necessarily great and can be detected if you know what to look for.

Options:

- (Least obvious) There is a wealth of data available on the internet about active hostnames in a domain that can be found without ever actually talking to the organization's nameservers at all. For example, a google search for "site:microsoft.com" or putting Microsoft.com into VirusTotal.com will deliver a wealth of hostnames that are known. In this method, the attacker is very difficult to detect as there is no direct interaction with them at all until they try to talk to one of these hosts.
- Another bolder option is the zone transfer. We haven't described zone transfers in detail, but they can be viewed as the primary nameserver sending a copy of the zone file to a backup nameserver to ensure the two match in case the primary goes down. The timing of this is controlled with the SOA records. Hosts allowed to perform a zone transfer are generally whitelisted to only other known nameservers, at least in theory. If the nameserver had been misconfigured for sec450.com, the attacker could merely use the command "dig AXFR sec450.com @dns1.registrar-servers.net" and it would respond with a copy of the entire zone file, giving the attacker the full list of hosts on the network (that had DNS entries at least). Detecting this method is easy. Simply making an alert for any zone transfer requests coming from unexpected places would alert you to this activity and the source of the attempt could be further investigated.
- (most obvious) The loudest option is hostname brute forcing. If zone transfer attempts don't work (and most of the time they shouldn't), and the attacker cannot find anything of interest through open-source

research, one option is simply to throw a bunch of A record requests at the nameserver for different names and see which ones don't return an error. Many times, companies run predictably named services, so they can be quite effective. Consider your own organization. Do you use a VPN., webmail., ftp., or another similar subdomain? If so, this technique would work to discover them. The downside of this method for the attacker is that it generates a LOT of DNS requests, depending on the size of the dictionary used. We will focus on detecting this method, since it's the one you are most likely to encounter.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

Detecting DNS Discovery Attempts

Mainly looking for sudden odd bursts of DNS traffic

Alerting on DNS discovery:

- **High volume** of requests from one source IP
- Spike in **A record request rate**
- Spike in **NXDOMAIN response rate**
- Request for **known non-existent** name
 - Example: You have no ftp.site.com, but alert on requests for it assuming attackers will guess that, may provide A record or not

Detecting DNS Discovery Attempts

Detecting DNS Discovery is actually rather easy. Since the attacker will, by definition, be attempting to resolve domains that mostly don't exist, at a suspicious rate of speed, from (likely) a single IP address, this leaves us several options.

One option is to merely have a SIEM alert for a high volume of DNS requests from a single source. If an alert for this fires, reviewing the traffic from that source and verifying that it is looking up many names that don't exist can validate the attack.

Another option is to simply look for a spike in A record request rate coming inbound from the internet. Note this is different than looking at the A record request rate for internal clients. A spike in internet-sourced DNS requests is another possible way to identify DNS brute forcing.

What about if the attackers are already inside and performing this on the internal network? In that case, verifying a single source IP is making many A record requests for mostly non-existent hosts can be enough to verify this is going on.

When a domain fails to resolve, the NXDOMAIN reply is sent back to the client. An alternative to looking for spikes in A record requests could be looking for NXDOMAIN responses being returned by destination IP. If you see an alert for this, again, verifying that the source IP receiving the responses was looking up mostly nonexistent domains can verify the situation.

A final method is to employ blacklisting of sorts and proactively put an alert out to trigger any time someone queries for a subdomain that an attacker might think would exist, but doesn't. The assumption is that even if you don't have an ftp.site.com, most attackers will attempt to resolve this, and that attempt can give you immediate warning of the brute force attack. To take this even further, you could set up a fake A record for it and wait for an IP to try to connect to the fake IP address. This way, if the attacker was using one IP to perform the brute force and another IP to connect to the hosts, you would now see the true attacking machine and not just the DNS brute forcing machine.

Unauthorized DNS Server Use

Operating systems have a default DNS server to use


- Often handed out by DHCP, can be manually set
- In your organization, most will be the same or of a finite set

In Linux:

```
$ cat /etc/resolv.conf  
nameserver 127.0.0.53
```

In Windows:

```
> ipconfig /all
```



The screenshot shows a Windows network settings dialog box. It has two radio buttons: "Obtain DNS server address automatically" (which is selected) and "Use the following DNS server addresses:". Below the second option are two text input fields: "Preferred DNS server:" and "Alternate DNS server:", each with a dotted placeholder for an IP address.

Unauthorized DNS Server Use

In an organization, most devices will be using the same internal caching DNS resolver (or one of a small set, depending on organization size). These settings are either set manually for static IP machines, or obtained through DHCP for dynamic IP address hosts. Using an internal DNS server allows the organization to offer resolution for internal hosts as well as control website access through DNS.

In many distributions of Linux, the nameserver can be found in the /etc/resolve.conf file. In Windows, the command ipconfig /all will show the settings for each separate adapter, or the DNS server settings can be changed for each interface through the networking settings in control panel. This is set at the operating system level so that programs that need to resolve hosts can call the API for the operating system (gethostbyaddr for Windows for example) and receive the result after the OS does the lookup with the required settings. This doesn't mean, however, that a program couldn't just come with its own DNS server settings and the ability to do DNS resolution on its own. In fact, this is exactly what happens in many cases with malware specifically to avoid using the organization's DNS server, which may log the requests or block them. Therefore, understanding when a single rogue device is using an odd external server for DNS resolution is a key ability for analysts.

Discussion: Find the Suspicious Request

Source	Destination	Protocol	Info
192.168.42.2	192.168.42.139	DNS	Standard query response 0x81a5 AAAA ssl.gstatic.com AAAA 2607:f8b0:4006:8
192.168.42.139	192.168.42.2	DNS	Standard query 0x2f46 A id.google.com OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0xae1f1 AAAA id.google.com OPT
192.168.42.2	192.168.42.139	DNS	Standard query response 0x2f46 A id.google.com CNAME id.l.google.com A 21
192.168.42.2	192.168.42.139	DNS	Standard query response 0xae1f1 AAAA id.google.com CNAME id.l.google.com A
192.168.42.139	192.168.42.2	DNS	Standard query 0x7962 A www.gstatic.com OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0xcf97 AAAA www.gstatic.com OPT
192.168.42.2	192.168.42.139	DNS	Standard query response 0x7962 A www.gstatic.com A 172.217.10.99 OPT
192.168.42.2	192.168.42.139	DNS	Standard query response 0xcf97 AAAA www.gstatic.com AAAA 2607:f8b0:4006:8
192.168.42.139	192.168.42.2	DNS	Standard query 0x21b9 A apis.google.com OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0x91fd AAAA apis.google.com OPT
192.168.42.2	192.168.42.139	DNS	Standard query response 0x21b9 A apis.google.com CNAME plus.l.google.com
192.168.42.2	192.168.42.139	DNS	Standard query response 0x91fd AAAA apis.google.com CNAME plus.l.google.c
192.168.42.139	192.168.42.2	DNS	Standard query 0x6d7a A adservice.google.com OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0x164c AAAA adservice.google.com OPT
192.168.42.2	192.168.42.139	DNS	Standard query response 0x6d7a A adservice.google.com CNAME pagead46.l.do
192.168.42.2	192.168.42.139	DNS	Standard query response 0x164c AAAA adservice.google.com CNAME pagead46.l
192.168.42.139	8.8.8.8	DNS	Standard query 0x7430 A google.com OPT
8.8.8.8	192.168.42.139	DNS	Standard query response 0x7430 A google.com A 172.217.10.14 OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0x6d7a A daisy.ubuntu.com OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0xbef2 AAAA daisy.ubuntu.com OPT
192.168.42.2	192.168.42.139	DNS	Standard query response 0x6d7a A daisy.ubuntu.com A 162.213.33.132 A 162.
192.168.42.2	192.168.42.139	DNS	Standard query response 0xbef2 AAAA daisy.ubuntu.com SOA ns1.canonical.co
192.168.42.139	192.168.42.2	DNS	Standard query 0x1c6b A www.google.com OPT
192.168.42.139	192.168.42.2	DNS	Standard query 0xfb62 AAAA www.google.com OPT

Discussion: Find the Suspicious Request

Let's say you are looking at the traffic capture above from your network. How might we find suspicious DNS A record requests? Think about the information in the fields we reviewed—we have a source and destination IP, the hostname that was queried, and a response from the nameserver to that query. Which one of these would highlight the use of a rogue external DNS server? The destination IP!

In the traffic above, there are many requests, mostly to Google-owned domains. If you are looking for external DNS resolver use, the hostname being looked up is unlikely to be of help. Many times, malware will do connectivity checks by resolving Google first before sending traffic to suspicious domains. In these packets, most DNS traffic has been sent to 192.168.42.2, but one request in packet 708 was sent to 8.8.8.8 (Google's public DNS servers). If your network doesn't allow outside resolvers (and it shouldn't), this is a dead giveaway that something is amiss. Why is a device using a non-standard DNS server for lookups? The reason is either it is a misconfiguration, or malware that came with its own settings to ignore what is set up in Windows and use the Google server for some reason. This is one very easy way of identifying potentially infected devices!

Detecting Unauthorized DNS Server Use

- Source to find UDP port 53 traffic to external IP address
 - Firewall block for port 53
 - NetFlow / Network metadata
 - Host firewall logs
 - IDS alerts
 - Host integrity checks
 - Endpoint Detection and Response
- Any of these should send an alert!

Detecting Unauthorized DNS Server Use

When looking for signs of external DNS server use, there are multiple places we can find the evidence. On the previous slide, the use of 8.8.8.8 as the resolver is likely to show up in multiple places. If an alert fires for the use of external DNS services, or you'd just like to go hunting for it, there are plenty of places that would record this data.

- Firewall logs: Any logs (block or allow) from the inside going outbound should record the activity.
- NetFlow / Network metadata: Any service that is recording Layer 3 and 4 statistics for outbound traffic can be searched for UDP port 53 packets with an internal source and external destination.
- Host Firewall Logs: If you use a host firewall on your servers or desktops that records outbound traffic, do not forget to search for evidence here.
- IDS Alerts: Ideally, the network is instrumented to highlight when this type of activity is performed, so check historical alerts and see if external DNS activity has been seen over time.
- Host integrity checks: What if malware has changed the system settings for DNS, but the host has not made much or any traffic at all yet? One way we can look for malicious modification of system DNS settings is to monitor them and then alert on any unexpected changes.
- Endpoint Detection and Response: Any EDR tool should have the capability to record DNS traffic for each host. Searching these solutions for all DNS traffic that is NOT bound for the authorized internal DNS server is a very comprehensive way to immediately find all unauthorized external DNS server use.

Attack: Domain Shadowing

Domain Shadowing: Taking over control of a good domain's DNS infrastructure and inserting a malicious A record

Steps:

1. Attacker gets into your registrar account / DNS server
2. Creates new A record – **evil.thing.sec450.com**
3. Points record to *their* attack infrastructure, not your servers
 - Note: This is *not* a breach of your data (except for DNS server access)
 - Abuses others trust in your domain to attack *them*
4. Blue team sees person visited `____.sec450.com`, assumes safe
 - IP is malicious, parent domain name is categorized as “good”



Attack: Domain Shadowing

Domain shadowing¹ is a common technique used to abuse the trust you have in known domains to successfully deliver malicious content. To perform this attack, an attacker must gain the ability to create a new A record in your DNS zone, whether that's by directly accessing the server or hacking into your domain registrar account and making the modification. Once they have access, they create a new record that is a subdomain of your top-level domain. It might be random, it might be one level deep or it might be more. For sec450.com, for example, an attacker might choose newsite.sec450.com, i.like.sandwiches.sec450.com or something random like v9a7j4c59.sec450.com. In this situation, the attackers are abusing the public's trust in sec450.com. If sec450.com is a known site to your organization that will not set off alarm bells when someone visits, they are now free to set the A record for that subdomain to an IP address that they have control of and host malicious content on. This content has a much higher chance of being delivered since proxies and firewalls will not, in general, block a "known good" domain.

This means that as a blue team, detecting this attack will be much more difficult. In order to do so, something else will need to tip you off that something bad happened. It may be that the IP address they use is already on your threat indicator platform as evil, or perhaps an IDS will recognize the payload they deliver and alert you to the attack which can then be investigated and identified as domain shadowing. Note that this attack is not directed against the organization that has their domain "shadowed" but rather against the people they then send to that new subdomain that was created under your zone. You are more likely to be the victim of someone using domain shadowed links to deliver malicious content to you than you are to have someone create the shadowed subdomains on your DNS server. So, although stopping both are important, the former scenario is the primary concern. Detecting that you have become the host of a malicious DNS record is easy. Simply implement change detection on all your zone files or DNS setup and any unauthorized changes will be immediately surfaced.

[1] <https://blogs.cisco.com/security/talos/angler-domain-shadowing>

Identifying Domain Shadowing Attacks

This is hard to do programmatically, but there may be hints

- Is the subdomain found on google/OSINT?
 - Either with "site: [subdomain].site.com or regular search
- Are there any reports for the URL/hostname?
- Is the subdomain random?
- Is the IP address in the same subnet as other subdomains?
 - Cloud makes this less likely :/
- Is it even in the same ASN?
- Is this company likely to get its DNS server compromised?

Identifying Domain Shadowing Attacks

If you see an alert for suspicious content being downloaded from or served from a website that is otherwise known as reputable but notice that it came from a subdomain that doesn't look quite right, you may be a victim of a domain shadowing attack. What factors could you analyze as an analyst to check if this is the case?

- Throw the subdomain into VirusTotal or the RiskIQ community portal and see if anyone else has ever seen the domain and if it is labeled as malicious. This can be a dead giveaway that something is likely bad. If you're the first one to know about the domain, this is another potential vote for domain shadowing.
- Is the domain findable in a Google search or in other passive DNS databases? If you can merely search Google for the subdomain and find its reference, it is highly unlikely that it is a domain shadowing attack. People wouldn't be talking about a non-legitimate domain unless it were malware analysis web pages.
- Is the subdomain random? This will certainly not always be the case, but many companies would not make a random-looking subdomain (with some notable exceptions like content delivery networks and such). If it is a long or random-looking subdomain, this would be a (minor) vote for domain shadowing.
- Check some other subdomains of that parent domain. Is the IP resolution in the same subnet? Is it even in the same ASN? (Unfortunately, with cloud computing becoming more normal, it will be hard to tell who owns what based on subnets and ASNs at times).
- Is the company likely to get their DNS servers compromised? If the domain the victim visited is Google, it's probably much less likely that Google's DNS servers were compromised compared to a small company that may not run as good of a security operation as Google would.

Attack: DNS Tunneling

Encodes data and sends it over DNS

- Uses weaponized custom nameserver
- Data encoded in subdomain, return data in response
- Very sneaky, unlikely to be caught unless watching
- Almost always works – everyone must use DNS!
- Two methods:
 1. Send requests directly to attacker-owned nameserver
 2. Send requests through normal organization's DNS resolvers

Attack: DNS Tunneling

DNS Tunneling is a very interesting and potentially lethal attack. Step back for a second and consider what DNS is capable of—quickly sending arbitrary bits of text and receiving a response of other arbitrary bits of text. What if, instead of that text being related to hostnames and IP addresses, we encoded files or command prompt output and sent the requests out not to a legitimate nameserver but one controlled by an attacker running custom software. That custom software (DNS tunneling server software) would receive the encoded text from the request, decode it, present it to the attacker in its original form, and allow them to respond in a way that will be encoded into the DNS response packet. This is what DNS tunneling is doing. It uses the admittedly limited ability to transfer data across the DNS protocol, and controlling both sides, makes the content it is sending malware output and file exfiltration, instead of record lookups.

The particularly dangerous thing about this attack is that it almost always works. Organizations must allow DNS traffic outbound to the internet from at least one source; otherwise, no one would be able to use the internet. In companies where outbound deny rules are not in place, malware implementing DNS tunneling can even directly use a rogue DNS server, as was previously discussed. Whether or not you implement a block for usage of outside DNS servers, this attack will work, because even going through your company's designated DNS resolver, the message will eventually get through, and the response will get back to the victim. The good news is, if you know what to look for, spotting DNS tunneling is extremely easy.

DNS Tunneling Visualized

CNAME tunneling:

Info
Standard query 0xc5eb CNAME 29b9018040daa4c0c3e34400252a5fb2af.1.eej.me
Standard query response 0xc5eb CNAME bc590180408db3a10758baffffffaf2b1.1.eej.me

TXT record tunneling:

Answers
▾ 9083008040ab1a45582b9d0001963400ba234ad3806d5af297413b735ab1.1af94689.1.eej.me: type TXT, class IN Name: 9083008040ab1a45582b9d0001963400ba234ad3806d5af297413b735ab1.1af94689.1.eej.me Type: TXT (Text strings) Class: IN (0x0001) Time to live: 5 seconds Data length: 35 TXT Length: 34 TXT: 131a008040e5caec9f3b58ffffff517c48

DNS Tunneling Visualized

The top picture on this slide is a screenshot of DNS tunneling being performed through CNAME record lookups. The DNS request is being made by malware installed on the machine, not normal Windows programs, and it is encoding whatever data it is sending into the subdomain (29b90180...). This request gets out to the internet either directly or via recursion through the company's designated nameservers, and eventually arrives at the malicious nameserver, which isn't running normal DNS server software, but the DNS tunneling endpoint server. Once the command is interpreted, a response can be returned that, although encoded, will make sense to the malware on the far side (bc590180...). Keep in mind although these fields seem to be hex encoded, that doesn't mean we can directly convert them back to the original data or text that was sent. There could be a layer of encryption applied by the malware to the data before encoding. These encoded fields have a maximum length that does limit the speed with which a DNS tunnel can carry data, but since attackers are often interested in having the most responsive backdoor possible, they use the longest requests to get the most out of each. This leaves us with multiple detection opportunities.

The bottom photo is essentially the same name as the top. Whereas the top photo shows a CNAME request and response encoding the data, the bottom photo shows a TXT record request with encoded data heading outbound (90830080...), and the TXT record response (131a0080...) as the method of returning data. TXT records make a particularly good vehicle for DNS tunneling as by definition they are meant to carry an arbitrary bit of text as a response. Remember, implementing this does not use normal DNS server software like BIND, but DNS tunneling specific software like iodine¹ or dnscat2² that can take the subdomain from a TXT record request, immediately interpret it, and craft a response that will be unique each time and place it into the TXT record response field.

[1] <https://code.kryo.se/iodine/>

[2] <https://github.com/iagox86/dnscat2>

Discussion: Spot the DNS Tunneling

1. TXT Record Query:

```
3.1o198r00n57o62rrp3743xxxxxxxxn7p8n1p328q8uew742o437qp83152.48s2p9
q4on2q5892367xxxxxx497rs238p3qp2859q3pp2q02nq2.s5034p9o22rpqo2r3r44
n277qoqr6344n79qo70o957n0.883q83532378773.i.04.s.sophosxl.net
```

2.

```
Standard query 0xd9af NULL 1ygbu82\3122hb\276\356\354\3568m\314d\277\276\356iFpqcViom\340dyP\354i7\312tc\357\307\326\33638m\344dbb
Standard query response 0xd9af NULL 1ygbu82\3122hb\276\356\354\3568m\314d\277\276\356iFpqcViom\340dyP\354i7\312tc\357\307\326\3363
Standard query 0xf7de NULL pafycg3y.pirate.sea OPT
Standard query response 0xf7de NULL pafycg3y.pirate.sea NULL pafycg3y.pirate.sea
```

3.

```
Standard query 0xbb43 A dwgyu36up6iuz.cloudfront.net
Standard query response 0xbb43 A dwgyu36up6iuz.cloudfront.net A 52.85.107.38 A 52.85.107.19
```

4.

DNS	Standard query response, No such name
DNS	Standard query A qvshurwrvlg.lan
DNS	Standard query response, No such name
DNS	Standard query A mnogujcphm.infigo.local
DNS	Standard query A rvkhwwicio.infigo.local
DNS	Standard query response, No such name
DNS	Standard query A mnogujcphm.lan

Discussion: Spot the DNS Tunneling

Now that we have some idea what DNS tunneling looks like, let's look at some actual traffic and see if we can pick out what is and what isn't DNS tunneling.

Pretend we have 4 different traffic captures here that triggered an alert for possible DNS tunneling. How could we work through these to determine which is which?

1. We see an extremely long TXT record query being sent out. Normally, long queries like this, especially in excessive numbers, should fire a potential DNS tunneling alert. If we saw this, where could we start? First, look at the parent domain – sophosxl.net. We could take this domain to Google and a simple search would reveal that this is the Sophos Antivirus agent using a feature called "live protection." *It is actually tunneling data over DNS, but in this case, it is being used for good. Sophos uses DNS as a mechanism to send file information up to the cloud for a fast determination on if something is safe to run or not.*¹ So in this case, yes it is DNS tunneling, but not the type we're concerned about—tricky huh? The key here is some open source research quickly gives us the answer, and also that there shouldn't be a high volume of lookups from one source.
2. This one is interesting. It's using the NULL request type, one that we haven't said much about. There's a reason we haven't talked about it—it's almost never used, and that fact alone is highly suspicious considering that it's sending very large queries with non-standard character sets. Investigating the domain that this traffic is going to, pirate.sea, will turn up that it is not associated with any known-good service. In fact, this is indeed DNS tunneling being run with the Iodine tool.² This was a simulated "real" attack from a traffic capture from a hack.lu CTF contest.³ Again, searching the domain the traffic is going to along with the context about the length and type of DNS queries can quickly sort this out.
3. This is an A record request with a random subdomain of the parent domain CloudFront. If you're familiar with content delivery networks, the answer here may be immediately obvious. This is not DNS

tunneling, but simply the benign use of the Amazon CloudFront CDN service. You could separate this from true tunneling again by looking up the parent domain, but also by the fact that there wouldn't necessarily be a large number of requests over time going to the same destination.

4. In this capture, we see several random-looking A record requests, some to .lan TLDs, and some as a subdomain to .infigo.local. The fact that some of these are going to parent domains immediately eliminates likelihood of DNS tunneling because you wouldn't use multiple different parent domains for this technique. Data cannot be encoded in that location since each parent domain has its own DNS nameserver and the parent name must stay static. What about the infigo.local requests? Well, these *could* be DNS tunneling, but since they are going to the .local TLD, one that is reserved by RFC 6762 as not being routable on the global domain name system (something you would have to know from experience), that means these requests were never meant to be used on the internet. So what are they then? Have you ever mistyped a domain name in your browser at home and received a search result page back that was labeled with your ISP's logo? Instead of sending the "failed to find that name" response you should receive, your ISP is trying to monetize your typo by using the mistyped term as a search query, feeding it into a search engine, and giving you back the results. This is called DNS hijacking and, in these packets, this is actually Google's Chrome browser checking for this activity. When you start Chrome up, it purposely issues random named queries like those seen here, that should never resolve. If they still do, this indicates your ISP is hijacking requests. Analysts who see Google Chrome requests for the first time often get very worried and confused where it is coming from, which is why it is pointed out here. The key to identifying this is that it will use whatever local search domain you use for your own environment, which in this case was infigo.local. This sample was sourced from a SANS ISC blog on the subject.⁴

As you can see, sometimes picking out tunneling is not as easy as it seems. Often making the determination simply comes down to looking at the parent domain and seeing if the volume of traffic matches the large amount that would be expected of true malicious DNS tunneling. Once you know the edge cases like AV suites, CDNs, and Google Chrome's DNS hijacking test, eliminating true from false positives becomes much easier.

[1] <https://community.sophos.com/kb/en-us/111334>

[2] <https://github.com/yarrick/iodine>

[3] <http://stalkr.net/files/hack.lu/2010/9/bottle.cap>

[4] <https://isc.sans.edu/diary/Google+Chrome+and+%28weird%29+DNS+requests/10312>

Detecting DNS Tunneling / Exfiltration

- LONG / random looking subdomains
- Unknown parent domain destinations
- Excessive queries for one domain with many subdomains
- Excessive DNS queries from one source
- Excessive amount of odd query types
 - TXT, CNAME, MX, NULL
- Encoded data in TXT responses
- Usage of unauthorized DNS servers
- False positives: CDNs, AV checks, DNS Hijacking tests

Detecting DNS Tunneling/Exfiltration

In summary, here are the attributes you should look for in DNS traffic that can verify whether it is true, malicious DNS tunneling. Keep in mind that not all tunneling is bad, such as the case with Sophos AV. The difference between good and bad uses, however, will often be the volume of traffic produced from a single source. Also, in the parent domain, the traffic is often easy to verify as good when the traffic is legitimate. High-volume DNS requests to an unknown or known malicious parent domain should be investigated as fast as possible.

DNS Traffic Flow and Analysis Considerations

- **How/where** are you monitoring?

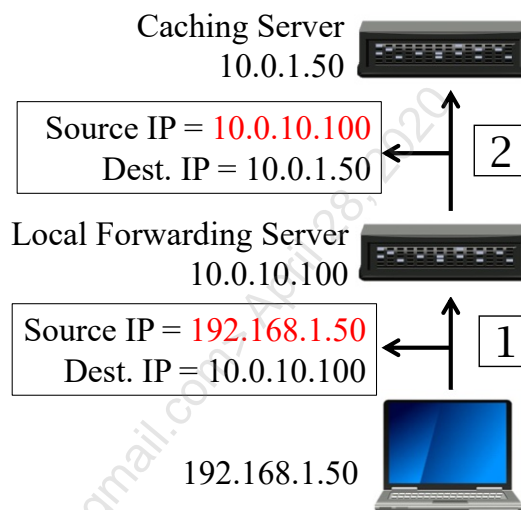
- With DNS's recursive nature, your sensor placement matters
- Log files or network extraction?
- Can you find the source host?

- Are you recording responses?

- Without responses, you will not know the IP resolution

- Are you saving responses?

- Create your own PassiveDNS database!



DNS Traffic Flow and Analysis Considerations

There are a few other considerations when it comes to analyzing DNS traffic recorded on the network.

- As an analyst, you should understand how and where the DNS traffic is being logged. By "how and where", we mean is the information being sourced from the log file generated on a resolver itself or are you using network extraction? If you're using a log file (Windows or BIND DNS text logs), looking at the slide, is it coming from the device one hop away from the client making the request, the local forwarding server, in this case? Or is it coming from somewhere down the line like the caching server? If you're using network extraction, such as a Suricata or Bro (now called Zeek), is it pulling it off the wire at position 1 or position 2? These questions are important because if you're recording logs at the caching server, or only recording the information off the wire at position 2 on the diagram, you will NOT be able to discern where the original DNS request came from. Once 10.0.10.100 receives the request from the device, it will recreate a new UDP packet with itself as the source, and ask the question again, leaving you without a way to figure out which device is potentially infected.
 - **An important point:** Are you stuck on being able to identify the device if you only record the source IP upstream? No, not necessarily! Consider that you now have the site's resolved IP address, and think about what happens next after an IP resolution—the device is likely going to reach out to that IP. If you have the IP the domain resolved to, and it's unique enough that only one device is talking to it (which may be the case when it comes to malware), you can simply search firewall, proxy, NetFlow, and other logs to see who talked to that IP. With any luck, the system that made the request will, immediately after the request was made, have made a connection to that IP identifying itself even though you don't have the direct logs to say so!
- Are you recording just the requests, or the responses as well? And what is happening with those responses that are (hopefully) being recorded? Recording the request is a great first step and is much better than having nothing but using log files for recording DNS requests sometimes makes it more difficult to get the response to the query in the log. If you can't record the response to each request, you are robbing yourself of useful information. Do the responses you receive get thrown out or are you

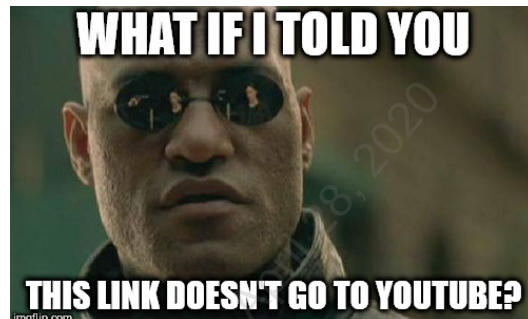
saving them into your own personal passive DNS database? The ideal answer is that you save them into some type of system that will keep them around for a longer period, enabling you to consult that database of past lookups well after they've occurred in case it becomes important to see what IP addresses a site has been seen to resolve to over a period of time. We'll touch on this again later in the class when we discuss passive DNS and OPSEC during investigations.

- If you collect DNS information as a log file from your DNS server only, what happens if someone is using DNS tunneling to directly contact a malicious DNS server? You won't see it! This is an important side effect of collecting DNS information from your server, requests that don't go there won't be logged, which means if you don't block contacting external DNS servers with your firewall, DNS tunneling may be happening *and* you might not see it!

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

Internationalized Domain Names (IDNs)

- We don't all use English...
 - But DNS only works with characters a-z, 0-9, and a dash
 - What now?
- To fix this, IDNs were created
- Enables all languages for DNS
 - Uses multi-byte Unicode encoding
 - Converts back to ASCII character set
 - Therefore, works with nameservers



<http://youtube.com>

(it's not a hovering trick)

<http://youtube.com/>
Ctrl+Click to follow link

Internationalized Domain Names (IDNs)

Obviously, not everyone in the world uses the English alphabet for communications, and computers have no issue dealing with this utilizing multi-byte Unicode encodings for character sets outside ASCII. But domain names cause a special problem because by definition, the only characters allowed are those of the English alphabet. Characters a-z, 0-9, and a dash are all that is allowed in a valid hostname for DNS, so a solution had to be devised that could compensate for this. That solution was internationalized domain names.

Punycode

- Punycode stores IDNs as ASCII strings
- Uses "Bootstring" encoding to map Unicode to ASCII
- Identify by prefix of "xn--" and suffix characters
 - Previous slide's link was actually **<http://xn--youtube-wqf.com>**
- Normal ASCII will be shown, suffix determines Unicode
- Presents a domain spoofing issue
 - Many Unicode characters look like ASCII characters
- Irongeek.com homoglyph generator demonstrates...

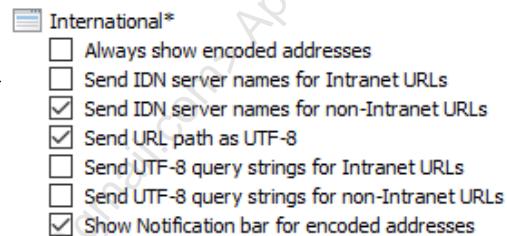
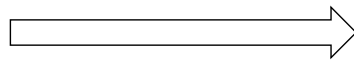
Punycode

To manage storing multi-byte characters within the ASCII space, an encoding method had to be devised. This method is called punycode. It uses what is called "bootstrap" encoding and it's able to map any character sequence in Unicode back into a unique string that can be represented with the traditional (and allowed) ASCII character set. To identify strings that are in punycode format, the prefix "xn—" was chosen. Seeing this prefix is the best clue that a domain you are looking at is made up of a different language's character set, and not just some random crazy-looking domain. In punycode, all ASCII characters will remain the same, but the Unicode piece is determined by a suffix at the end of the Unicode characters. In the last slide, for example, the punycode version of the YouTube link is <http://xn--youtube-wqf.com>. You can see the xn– prefix, the "youtube", but the "o" is missing. That's because the o is the Unicode character that was substituted for a lookalike, and that information is encoded in the –wqf at the end of the domain in a way that is not readily identifiable.

Consider how this can be an easy avenue for attack for phishing and otherwise. If you purchased <http://xn--youtube-wqf.com>, you could display the link in many situations as shown on the previous slide and get people to click on it thinking for sure that they'd be going to YouTube. This is called a "homoglyph attack" and a further example of this is on the next slide.

Browser Display of IDNs

- **Firefox:** Uses a whitelist of domains and algorithm¹
 - Controlled with `network.IDN_show_punycode = true`
- **Chrome:** Similar to Firefox²
- **Safari:** Converts to punycode by default
- **Edge/IE:** Converts to punycode by default unless enabled as system language³
 - IE Settings



Browser Display of IDNs

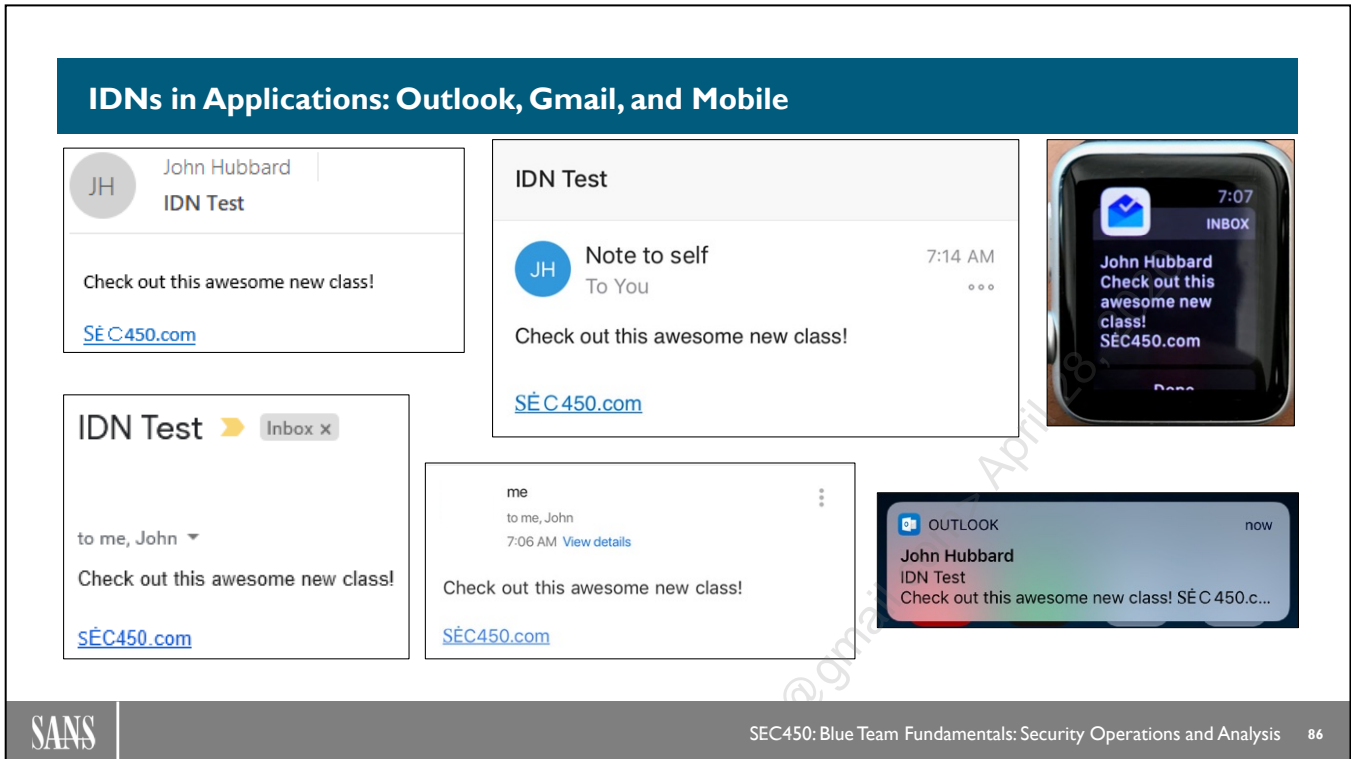
Clearly, any application that chooses to show the non-punycode version of an IDN is potentially susceptible to phishing attacks due to homoglyph attacks. Because of this, some applications have decided to (almost) always show the punycode representation unless the user specifically asks otherwise. When it comes to browsers, each of the major ones approaches this issue differently.

- Firefox has a setting in the `about:config` section called `"network.IDN_show_punycode"` that controls which version of a domain is displayed. There is an algorithm also in effect if you want to dig deeper.¹
- Chrome is similar to Firefox.
- MS Edge does not use punycode by default.
- Internet Explorer has the options shown on the slide that can be changed to match the desired behavior.

[1] https://wiki.mozilla.org/IDN_Display_Algorithm

[2] <https://www.chromium.org/developers/design-documents/idn-in-google-chrome>

[3] <https://docs.microsoft.com/en-us/windows/desktop/intl/handling-internationalized-domain-names--idns>



IDNs in Applications: Outlook, Gmail, and Mobile

Although you may have a chance to spot the domains in punycode form once they are opened in your browser, that assumes you take the time to check once you have clicked. Don't let that trick you into a false sense of security—the danger of IDNs still remains. Outside of browsers, in almost every case, Unicode will be displayed. Applications are simply trying to render the text in the correct way and, therefore, punycode will not be used.

These screenshots were taken showing a link to a homoglyph attack format of sec450.com (with the S, E and C changed). Here is how the links were displayed by the desktop version of Outlook, the Gmail website, Outlook and Gmail mobile, and the mobile notifications shown on iOS. Notice that although some do look slightly odd, the font used in each case can make an IDN look more, or less, deceiving, adding another dimension of complication to spotting the attack. This means at the time the user is making the decision whether to click or not, browser default punycode rendering will happen too late to save you.

Actual Attacks

Phishing with IDNs:

SEC450.com = xn--c450-uva139z.com/

Without punycode:

facebook.com
morganstanley.com
lloydsbankinggroup.com
unitedhealthgroup.com

<http://bankofamerica.com/index.jp>

Passive DNS replication

VirusTotal's passive DNS only stores address records.

2018-08-06	xn--acebook-js3c.com
2018-07-20	xn--moranstanley-0df.com
2018-07-20	xn--lloydsbankinggroup-0ch.com
2018-07-20	xn--unitedhealthroup-8xg.com

Actual Attacks

To further drive home this point and prove that it is not just theoretical, but actually happening, check out these passive DNS results that were found on VirusTotal. Clearly, there is a group out there trying to make lookalike domains for attacks!

The bottom left of this slide shows a slight modification of this attack. In other examples, we have shown the replacement of a letter with a lookalike. This example shows that you can take this another route—find a character that looks like a forward slash and integrate that into the domain name itself! This domain is actually "com/index.jp" with a special Unicode character for the forward slash and "bankofamerica" is actually the subdomain, tricky huh? (For those wondering, this domain is owned by a researcher, not an attacker, but shows that it is possible, even with the restrictions some registrars set, to buy this style of domain!)

The Future of DNS

DNS over TLS (DoT)

- Wraps DNS requests in TLS connections to secure them

DNS over HTTPS (DoH)

- Prevents tampering, and even seeing domain requested!
- **Firefox/Chrome** already support it natively
- Windows coming soon!¹ (announced Nov. 2019)

DNSSEC

- Provides authentication and integrity, not confidentiality
- Signs all responses with a server public key

The Future of DNS

The future of DNS is not yet here, but there are multiple standards proposed to try to make it more secure. Unfortunately, most of those methods also end with the lack of ability for the blue team to monitor queries from the network. Log files will likely become the only way to get information on what domain was requested.

One standard starting to emerge is DNS over TLS or DoT. This standard takes typical DNS queries but wraps them in a TLS connection first so that they cannot be tampered with in flight or read by anyone monitoring the wire. This standard is already supported by multiple big-name public DNS resolvers such as Cloudflare's 1.1.1.1 and Quad9's 9.9.9.9.

Another emerging standard is DNS over HTTPS or DoH. This protocol will effectively wrap a standard UDP-style DNS response as the payload for an HTTP/2 (encrypted) transaction, protecting it from tampering. While no operating systems currently directly support the protocol, some applications do, such as Firefox version 62+ and Chrome as well. An important point to note about this DoH is that while it may encrypt transactions between your machine and your next-hop DNS server, that server may not use it to recursively query the root servers, meaning DoH is *not* end-to-end encrypted communication.

A final acronym to be aware of is DNSSEC. You may have heard DNSSEC is dead throughout the years because the idea has been around for a long time but seems to have had trouble getting industry to adopt it. DNSSEC does not provide confidentiality as the other standards do but addresses authentication of where your DNS data is coming from, and the integrity of that data. It does so by using asymmetric encryption and a public key to sign all records and responses from the nameserver. Through this method, you can be sure that the responses you've received are truly from the nameserver and have not been tampered with.

[1] <https://techcommunity.microsoft.com/t5/Networking-Blog/Windows-will-improve-user-privacy-with-DNS-over-HTTPS/ba-p/1014229>

DNS Summary

Many methods exist to identify evil based purely on DNS

- TLDs – some are just more likely evil
- Reputation, Age, Randomness, Length, Rank, IP Reputation
- IDNs making phishing *very* difficult to catch

Collection considerations:

- DNS is proxied – consider your point of collection carefully
- Network extraction is the easiest method...for now
- Future DNS standards will make network collection difficult

DNS Summary

The unfortunate truth is that many organizations do not collect their DNS logs, believing they either have no value or will be too high-volume to collect. Without filtering and enrichment of the data, this may be true, but organizations that do perform these tasks will find they can routinely catch evil based purely on the domain names contacted by their users.

When collecting DNS data, remember that DNS is a proxied interaction, and while picking DNS traffic up via network extraction is generally the easiest way to do it, the point where the data is collected may lead to difficulty in finding the true source of the query. As the new DNS protocol standards start to become standard and DNS becomes encrypted, network collection will likely need to be replaced with service log collection from the DNS servers.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. **Exercise 2.1: Exploring DNS**
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. **Exercise 2.2: HTTP and HTTPS Analysis**
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. **Exercise 2.3: SMTP and Email Analysis**

This page intentionally left blank.

Exercise 2.1: Exploring DNS



Exercise 2.1: Exploring DNS

Exercise 2.1: Exploring DNS

Please go to Exercise 2.1 in the SEC450 Workbook or virtual wiki.

Course Roadmap

- Day 1: Blue Team Tools & Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. **Understanding HTTP(S)**
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

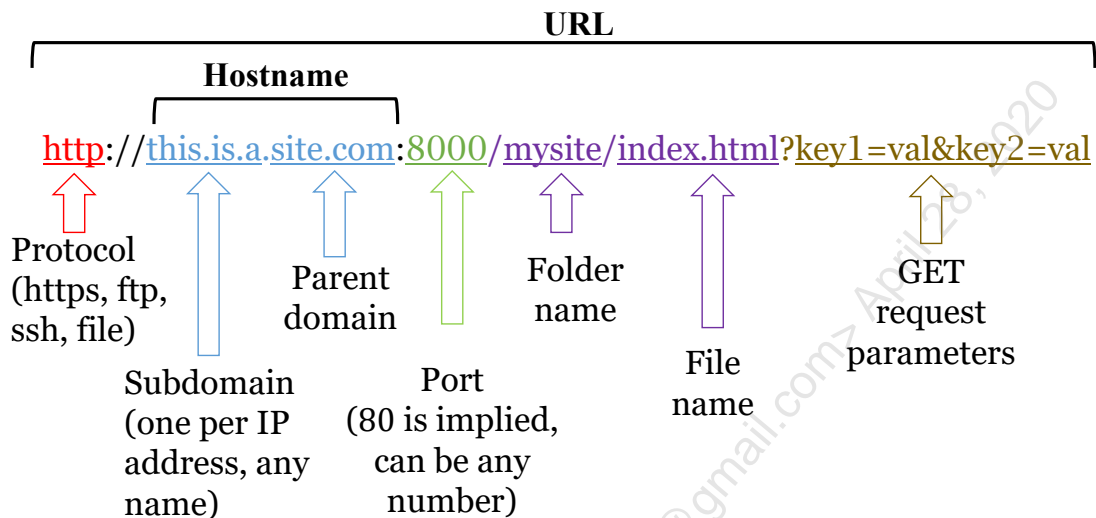
Webservers

- What is a webserver really?
 - At a high level: **File sharing using HTTP protocol**
 - Webserver sends file specified in URL path (/main/index.htm)
- Web pages created using HTML, CSS, JavaScript files
- Scripting can take user input to modify files given back
- Apache, NGINX, IIS are the most common web servers
 - Each has plugins that can modify operation
- Basically – fancy file sharing

Webservers

This module will dive into web servers—how they work, common configuration items and parameters, and the protocol details that analysts need to understand. Interpreting an HTTP connection and its headers will be an extremely common occurrence because so many attacks are HTTP-based. In fact, almost every stage of an attack can be carried out using the HTTP protocol, so it is one protocol that nearly everyone needs to be familiar with.

Decoding a URL

**Decoding a URL**

HTTP analysis requires understanding URLs, so when looking at a URL, you should be able to break it down in your mind to its constituent parts. Let's take a step back for a second and deconstruct an example and point out what each section really means to us.

In this module, we'll be talking about HTTP, so the protocol section and the URLs we are dealing with will always start with HTTP (or HTTPS when encrypted). This is followed by the subdomain and domain, which make up the hostname that will be resolved by DNS. Remember, this is ultimately just a way to refer to one (or more) computer somewhere on the internet. The same way your laptop might have a hostname of dell1234 or Lenovo-abc, the hostname in a URL is just a name for a machine. It just happens to be resolvable across the entire world via DNS.

The next part of the URL, which may or may not be present, is the port number. For a browser, port 80 is implied (or 443 when using HTTPS) when no port is specifically listed. You can run an HTTP server on any port. Other common choices typically involve 8's and 0's such as 8000, 8080, 800, etc. Although using nonstandard ports is not the norm, do not be surprised if you see any of those in use for legitimate sites.

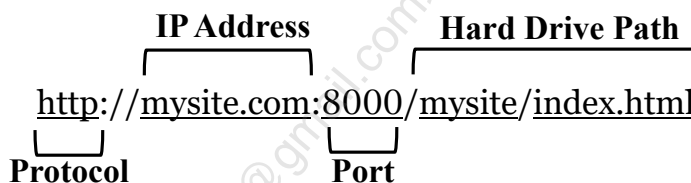
Following the port is the folder name and path. If you've never set up a webserver, you may not have made the connection that this is literally just a folder on the server's hard drive where the various pages for the site are stored. A reductionist but useful way to think of a webserver is just another way of doing file one-way sharing with a specialized protocol (HTTP) to do so. The folder path and filename are just informing you of the current file that is being interpreted by the browser and rendered as a webpage. Of course, there are web-frameworks out there, server-side scripting, and other details that complicate this comparison, but at the base level, that's all HTTP is doing for us.

Speaking of server-side scripting, sometimes the web page content takes input from a user and uses it to render the next page they will see. When this is done via a GET request, the parameters passed are included in the URL as well as key=value pairs with "&" signs between them. In certain cases, analyzing GET parameters can help reveal what a user was doing or the reason they ended up on a certain page, such as a reference from a previous page.

Mapping a URL to an HTTP Server Config

To serve this URL...

- Buy **mysite.com** from registrar
- Set DNS A record of **mysite.com** to point to your IP
- Tell Apache to listen for **HTTP** connections on port **8000**
- Serve (by default with Apache) anything in **/var/www/**
- Make folder called **/mysite/** under **/var/www/** with a file called **index.html** inside it



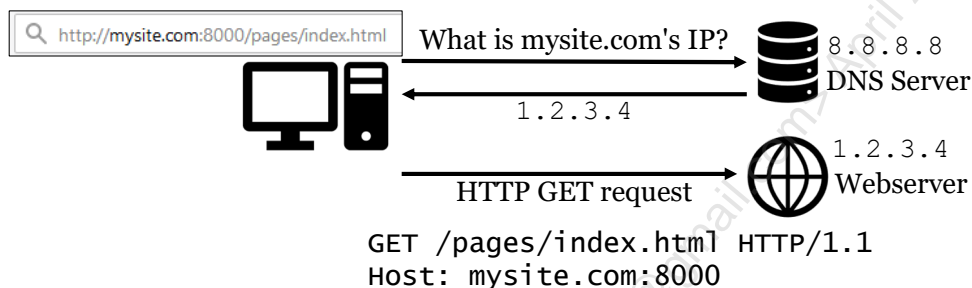
Mapping a URL to an HTTP Server Configuration

Given the previous slide and section on DNS, let's briefly review all the things that need to happen to serve a page on a given URL. Looking at the URL on the slide, `http://mysite.com:8000/mysite/index.html`, how should we mentally parse this?

First, we have the protocol, which is HTTP, so it is likely the remote host is running Apache, Nginx, IIS, or some other webserver. But instead of listening on the traditional port 80, they have decided to listen for connections on port 8000. The folder path and filename are `mysite/index.html`. That means, from whatever folder they decided to share as the root for the webserver (commonly `/var/www` as a default on Linux), they have then made a subfolder called `mysite` and placed the file `index.html` inside it. Are there other files in the `mysite` folder? Yes, probably, and if we navigated to the URL `http://mysite.com:8000/mysite/` we would receive a list of the other files in the folder if the administrator didn't block the webserver from providing the list in that fashion. Finally, to make the site available on `mysite.com`, the owner would first have to purchase the domain `mysite.com` and set up an A record for `mysite.com` to point to the IP address of the webserver. With all these items in place—a domain pointing to the IP, the webserver running and listening on port 8000, and the file `/mysite/index.html` placed into the correct path on the filesystem—we could successfully reach this file over the internet.

When the Client Browser Connects

1. Browser makes A record request for IP of mysite.com
2. TCP handshake using IP from DNS response to dest. port 8000
3. HTTP Request is formatted and sent
4. Server returns file `/var/www/mysite/index.html`



When the Client Browser Connects

From the client side, what does the browser then do to reach this site running on port 8000 from the previous slide? First it must resolve the name mysite.com via DNS and receive back the IP address. Afterwards, a TCP connection is crafted to the webserver's IP address with a destination port of 8000. This is the Layer 3 and 4 data of the packet. Notice the hostname mysite.com from the DNS request plays no part at all here. At this layer, the packet is directed purely by the destination IP address. In the application layer data, Layer 7, the browser must then craft an HTTP compatible request that can be interpreted by the remote server.

The HTTP headers at a minimal level must include an HTTP method, the path to the resource that is being requested, and the version of HTTP the browser is using. It also *may* need to specify the "Host" header as seen above. Why is this? Because the webserver can host more than one site from a single IP address and port number; therefore, addressing the packet to the webserver with only the destination IP is not enough to differentiate how the server should respond. The Host header disambiguates which site should interpret the traffic on the server side. Other headers may be present in this request, such as a Referer, User-Agent, or cookie, but are not strictly required to return a response.

Browser Interpretation

- Code can reference files from any server on the internet
 - CSS, JavaScript, pictures, flash, documents, etc.
- Most resources will come from local server or CDN
- Browser used to interpret files and decide what to do
 - Html, JavaScript, CSS, pictures, flash – render on page
 - Documents, executables, other files – default action or ask user
 - Knows what is what via **content-type** HTTP header
- **Browser enforces sandbox** between code and OS

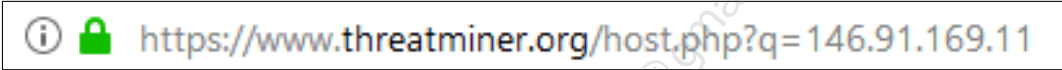
Browser Interpretation

Once a page is requested, an HTML file usually is the first item to load and that HTML will reference a large list of other files that must be downloaded to get the page to render correctly. These resources by convention will all create their own GET method requests, meaning that loading any given webpage is likely to result in a flurry of GET requests as each individual element of what is on the page is requested and downloaded in turn. Many of these elements will be sourced from the webserver that was originally connected to, but there are hosts that will almost certainly be involved. Content delivery networks (CDNs) help websites with capacity by caching content and rehosting it, ads will be served from ad networks, and pictures, flash, or documents could also be pulled from anywhere on the web. Although HTTP is the facility that allows us to download these files, the browser is the true star of the show when it comes to web pages. It knows how to take the files that are being received and interpret them either as visual elements that make up the website, files the user wants to download, or otherwise. It also protects us from malicious website owners that would like to attack us. Remember, the web is our computer taking code and files from untrusted sources and that is usually a recipe for disaster, so keeping each user safe is a job that falls upon the browser.

Since so much of the web these days is scripting, could a malicious website owner put a reference to an executable file on their webpage and use a script to execute it? No, our browser protects us from that, too. It is the ultimate arbiter of what happens to all things that are received, and what a website is or is not allowed to do. It enforces a "sandbox" of what running JavaScript and other code can access in terms of the machine it is running on. Typically, websites are not allowed to enumerate anything about a host other than what is required in order to get the page to render correctly. Browsers allow this through APIs that can be called through scripting languages, and if there is no API for retrieving a piece of information, a malicious website owner will not be able to acquire it (barring an exploit for your browser that breaks this assumption).

Beyond Static Files: Server-Side Coding Languages

- What if we want to modify a page based on user input?
 - PHP, ASP, Java and JSP, Python, Ruby on Rails, etc.
 - Gather info from user: GET parameters / POST data
 - Feed to script on server, produce page, serve it
- How do you know when they are being used?
 - Web requests to .php, .asp, .jsp, etc. files are a clue
- Why do we care?
 - **All requests do not give same response!**



<https://www.threatminer.org/host.php?q=146.91.169.11>

Beyond Static Files: Server-Side Coding Languages

The web would be incredibly boring if all we could see was a bunch of static files being served, so server-side coding languages were invented to modify the page a user saw based on input received from them. This is how a server takes a search query or other user input and modifies the page you see next based upon it. For example, let's say you want to search for an IP address on a site like threatminer.org. After going to the main page and getting the search box, you type in the IP 146.91.169.11 and hit submit. The HTML code on the search page takes the search box contents and crafts a GET request out of it with the q=146.91.169.11 parameter as seen in the picture above. The server then receives the GET request and the PHP code (which we know is PHP because the GET request goes to a .php file) strips off the GET parameters and knows that anything that is in the "q=" field is the term to be searched. The file host.php on the webserver root folder is a PHP script that has some code to the effect of "take whatever is in the q parameter, run a database query using it, and take those results and render them into a response page." This is the basic idea behind taking any user input and creating a webpage based upon it, the data sent from the first page accessed is passed to a script on the server, which receives and interprets it, and renders dynamic output pages based on it in any arbitrary way.

Why do we care about how this works as an analyst? Because it's important to realize that, if server-side coding is being used, you will not always receive the same response from a webserver going to the same URL. If you are trying to examine a potentially infected website that a user has previously visited and there are GET parameters or data sent in a POST request, visiting that site may net you very different results. In the case of phishing links or other potentially custom-created URLs, you may even cause the attacker to know that the link was clicked by the intended victim, tipping them off!

Passing Data as Folder Names: REST APIs

- Data can also be sent as a folder name
- Python, Ruby, and other frameworks implement this
 - Sinatra, Bottle, Flask are simple examples
- Often use to make simple APIs for integration
 - Folder contains operation (ip-address info vs. search below)
- Will play an important part in automation later on...

 <https://www.virustotal.com/#/ip-address/146.91.169.11>

 <https://www.virustotal.com/#/search/146.91.169.11>

Passing Data as Folder Names: REST APIs

Can we then conclude that pages are only dynamic when we see requests to filenames that end in.php, .asp, or other obvious server-side scripting languages? Not necessarily. These languages are far from the only way data is passed back to webservers. Sometimes, the information is sent in what appears to be a folder name. This is usually done to servers following the "REST" (representational state transfer) standard and is equally or more common to see than obvious server-side scripting files receiving requests. The VirusTotal lookup in the slide above is the equivalent of the threatminer.org lookup on the previous page.

Again, we care about this because it's important to realize a site may or may not be serving static information. In the case of REST standard interfaces for security tools specifically, APIs that use the REST format also makes it very easy to create scripts for integration and automation with the service. If you know, for example, that VirusTotal always uses the above format for searching information, it's very simple to then create your own links directly to searches or information of interest by filling in the IP address at the end of the URL. We will revisit this idea later in the automation section of the class.

HTTP Methods: GET

- The **majority of web requests** are the GET method
- Used for retrieving data: Files, data, CSSs, etc.
- Information passed through URL parameters
 - Typically only small bits of information: Search terms, etc.
 - Never use for sensitive data – would show up in logs!

```
GET /favicon.ico HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

HTTP Methods: GET

If all the HTTP methods were in a band, GET would be the lead singer. It's by far the most commonly used method for HTTP requests since it's used for loading most page resources: html, pictures, CSS, JavaScript and all. Most of the HTTP traffic you analyze will consist of GET requests, which are focused on retrieving the file specified in the URL.

One of the key items to remember about GET requests is that if data is going to be sent using the GET method, it is almost universally done with GET parameters. This means that the GET method should only be used for small amounts of text, and never for sensitive usernames or passwords. The reason is that most webserver and proxy logs will include the full URL accessed by a site visitor in the logs, and if names and passwords were passed with GET parameters, this sensitive information would also be recorded in logs—something we definitely don't want. For this reason, login information, files, and other larger information is typically passed using the POST method.

HTTP Methods: POST

- Used for forms, data, and login information
- Can include arbitrary amounts of data
 - Data placed beneath the HTTP headers
- Used when sent info should **not** be included in logs
- Often used by malware

```
POST /aaa.php HTTP/1.1
Host: google-analytics-sv1.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Content-Length: 255
Cache-Control: no-cache

data=vK6yv+bu9er16e3r6/qRLJWaj50Ulfbqm0+e70n67+vj4u3p7Z/9rb6p5urr60/
j/bK+5u316/Xp4uvr9e7u6un9tKjm7vXq9ent6+v9rq/mmr
```

HTTP Methods: POST

The POST method is the second most important HTTP method to be familiar with, as it can be used to send arbitrary amounts of data in the body of an HTTP request. It is commonly used for sending larger amounts of information that wouldn't make sense as a GET parameter, for sensitive information, and for data from web forms. Since the data sent is not part of the URL, it is not recorded in webserver access or proxy logs. Instead the information is located under the request headers as shown in the highlighted section of the image above.

Often, if the HTTP request the contents will be easily read and understood. Data that is sent with further encoding/encryption, such as the "data=" information shown on the slide, should raise the question of why. Malware often uses POST requests to avoid detection and may encode or encrypt the data in the POST request so that if it is found, it still may not be clear what was happening. The screenshot in this slide was, in fact, taken from a blackhole exploit kit malware traffic sample doing exactly that.

Additional HTTP/1.1 Defined Methods

- **CONNECT:** Used with proxies to create an SSL tunnel
- **HEAD:** Exactly like GET, but returns headers only
- **OPTIONS:** Requests a list of valid methods
- **PUT:** Uploading a resource to a site
- **DELETE:** Requests the server deletes the resource
- **TRACE:** For seeing how a request looks to the server (application-layer loopback)

Additional HTTP/1.1 Defined Methods

Here are the 6 additional methods defined by HTTP/1.1 that you may encounter while reading HTTP traffic. Most of them will play a smaller role in your day-to-day life compared to GET and POST. They tend to not be the focus of security incident related traffic unless it involves inbound traffic to organization-owned webservers. Note that not all servers must implement all methods. Technically, only HEAD and GET must be functional to be considered standards compliant.

- **CONNECT:** The connect method is used exclusively when connecting through a proxy and facilitates the setup of the encrypted tunnel with the target website.
- **HEAD:** The HEAD method should produce the exact same response as a GET request minus the body of the response. The reason for its use is to check if the date on a file has changed and therefore must be redownloaded instead of used from cache.
- **OPTIONS:** The options method is used to query the server about which HTTP methods it supports.
- **PUT:** Potentially more likely to be used in attacks since PUT facilitates uploading a file of some sort to the server. If you are responding to an incident involving traffic inbound to a webserver owned by your organization, you may see PUT used by the attackers.
- **DELETE:** The DELETE method is used to request the webserver delete the resource at the specified URL. It is unlikely to show up often in security incidents but is included for completeness.
- **TRACE:** The TRACE method is used to create an application layer loopback of the message sent in the request. The server should reflect the message as received in the body of the response. This is intended to be used for diagnostic and test purposes but may also be used by attackers for the same reasons.¹

Remember, these are just the methods that are part of the HTTP standard. If malware is the client creating the connection and it is talking back to malicious software also programmed by the attacker, the methods have no obligation to follow standard methods. The words are only useful if a typical browser is talking to a standard-compliant server. If the software on both ends is customized, all bets are off!

[1] <https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.2>

HTTP Request Headers

HTTP transactions contain "key: value" **headers**:

- **Host**: IMPORTANT – what domain was visited
- **User-Agent**: What client was used to make request
- **Referer**: URL the user was at previously (yes, it is spelled wrong)
- **Cookie**: Information to identify the user from previous transactions
- **Accept***: What file types, languages, and encoding will be accepted

Header order is arbitrary and should not affect the response received

```
GET /favicon.ico HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

HTTP Request Headers

When sending transactions back and forth to a webserver, additional metadata is specified beyond the method and parameters in the form of HTTP headers. Headers are the key:value pairs seen below the top line with the method, URL, and version number and can be found when looking at the actual text of an HTTP request. Most headers are optional but most times there will be multiple items specified. These headers are used to communicate additional information to the webserver about the hostname being accessed, the client being used to access the page, the URL that the user was previously visiting, and other potentially useful information the server can use to format the best response. Technically speaking, some headers are considered true "request headers" while others are "entity headers." The distinction is that entity headers describe the content of the body of the message while request headers don't relate to the content. Fields such as Cookie, User-Agent, and Referer are true request headers while Content headers are entity headers. When analyzing an HTTP transaction, it is important to understand what these fields mean and how they can be used to identify questionable or malicious transactions.

Some of the most important fields to understand are:

- **Host**: Virtual host field – key field to know what domain was contacted (because more than one website can be served from a single IP)
- **User-Agent**: Describes browser or client being used to access the site
- **Referer**: This field is, in fact, misspelled in the standard and may or may not be present for any given transaction. It tells the server what URL the user was at previously.
- **Accept***: There are multiple types of Accept headers. They describe various items and formats that will be accepted by the client browser—file types, language preferences, and encoding, among others. If more than one option is listed, the first item in the list is the preferred option.
- **Cookie**: This field carries the information that unique identifies as user to the webserver, and allows login state and other info to be preserved
- **X-Forwarded-For**: Identifies true source IP if the client is connecting through a proxy. This field will only be visible on the outgoing side of the proxy, not from the client itself.

Reading HTTP Request Headers

```
GET /PVfXBbR9WRenMmbQzLqrbDvNmen2ConnpC4tVP6U4RFI4HnE HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://k615o5ij7f.skwosh.eu/38104p5h2c
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT
6.1; WOW64; Trident/6.0)
Accept-Encoding: gzip, deflate
Host: k615o5ij7f.skwosh.eu
Connection: Keep-Alive
```

Reading HTTP Request Headers

Here is a slightly more complex set of headers than on the previous slide. This was taken from a PCAP of "angler" malware. Let's examine what we might be able to ascertain from reading this request.

- The **Accept** header says the preferred response is the type text/html, but the client will ultimately accept any format in response (this header is not usually security relevant, but is included for completeness).
- The **Referer** header tells us that the user was previously at the URL `http://k615o5ij7f.skwosh.eu/38104p5h2c`
- The **Accept-Language** header lists en-US (English, United States) as the sole item in the list—also not likely security relevant in most cases.
- The **User-Agent** lists Microsoft Internet Explorer 10.0 running on Windows 7 (NT6.1) as the browser the client was using at the time.
- **Accept-Encoding** shows the response from the server may be sent back using gzip or deflate as a compression algorithm. This is usually done for performance reasons. It is also important because if we wanted to extract the file returned from the server, it would have to be decompressed once the response was captured off the wire.
- The **Host** header tells us that the site `k615o5ij7f.skwosh.eu` is being visited. To know the full URL the user is trying to load requires combining the host header and resource from the first line of the request. For this GET request, the user is loading `http://k615o5ij7f.skwosh.eu/PVfXBbR9WRenMmbQzLqrbDvNmen2ConnpC4tVP6U4RFI4HnE`. If you think this looks like a highly questionable URL, you are correct. This is from the angler exploit kit.
- The final **Connection**: Header tells the browser that it should keep the TCP connection alive to the server for loading additional resources, another performance related header.

More Complicated Headers

```
GET /~forgos/hivatkoz/mediaismeret/common.js HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://translate.google.com/translate_c?
depth=1&hl=en&langpair=en%
7Cen&rurl=translate.google.com&u=http://www.ektf.hu/
~forgos/hivatkoz/mediaismeret/
common.js&usg=ALkJrhhLLHHxbXBDNGnG89v9nEwgVLebRQ
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0;
Windows NT 6.1; WOW64; Trident/6.0)
Accept-Encoding: gzip, deflate
Host: www.ektf.hu
Connection: Keep-Alive
```

More Complicated Headers

Here is an interesting case of a harder to interpret Referer field. The Referer header tells us that the user was previously at translate.google.com. But if you look closely, there are two different URLs contained in the "Referer"—one is translate.google.com, the other is inside of the Google URL as a parameter.

Since translate.google.com was the true referer, why is www.ektf.hu, the site currently being accessed, a parameter within it? Did Google Translate refer the user back to that site? Was Google Translate compromised? No. What likely was happening was that the second URL at ektf.hu was put into Google Translate for whole page translate, which caused it to be in the parameters to translate.google.com, the first URL in the User-Agent, and then Google Translate itself asked for the same resource again. Yes, it is quite a roundabout series of events, but there are plenty of scenarios like this you may encounter.

HTTP Response Headers

HTTP responses contain headers as well:

- **Server**: Webserver software and version number
- **Date**: The date and time the message originated
- **Content-Type**: Type of file being returned from server
 - Very useful for mass searching in SIEM
- **Content-Length**: Length of body content in bytes
- **Last-Modified**: When the resource was last changed
- **Set-Cookie**: Tells the client to set the provided cookie

HTTP Response Headers

The response from an HTTP server contains headers as well, like requests, some are considered **response headers** while others are **entity headers**. The distinction is that entity headers describe the content of the body of the message while response headers don't relate to the content.¹ Fields like content-length, content-type, and content-encoding are entity headers while server, would be considered a response header.

- **Server**: This header contains information about which software was used to provide the response from the server. It usually says something like Apache, nginx, or IIS, and the version number.
- **Date**: Gives the date and time the message was sent back from the server.
- **Content-Type**: The content of this field is also commonly referred to as a "Multipurpose Internet Mail Extensions" or MIME type. It is one of the more important headers returned from the server as it describes the type of file being returned (assuming the server is filling it out and not being dishonest). If the field is not included, some browsers will perform what is called "content sniffing" to try to determine the nature of the response and what to do with it. If this field is parsed for all outgoing HTTP transactions in a centralized log, it allows analysts to easily search for all connections that resulted in a download of, for example, a PDF or executable file.
- **Content-Length**: The length of the body of the returned content in bytes. (Technically, this is not a response header but an entity header.)¹ Using this field could help analysts identify the biggest single downloaded files, or conversely, the largest uploads if the Content-Length of a request is inspected.
- **Last-Modified**: When the HEAD method is used, if the last modified field has not been updated since the last time a page was loaded, this is the sign that it is safe to use cached content instead of downloading a fresh copy.
- **Set-Cookie**: This header tells the client to use the cookie provided. The client will then pass back this value in a cookie header in all subsequent interactions to identify itself.
- For a deep dive on both request and response headers, see Mozilla's HTTP Headers documentation.¹

[1] <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

Most Common HTTP Response Codes

1xx Informational:

- 100 Continue
- 101 Switching Protocols

2xx Successful:

- 200 OK

3xx Redirection:

- 301 Moved Permanently
- 302 Found

4xx Client Error:

- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 407 Proxy Auth Required

5xx Server Error:

- 500 Internal Server Error
- 503 Service Unavailable

Most Common HTTP Response Codes

There are many possible HTTP status codes that could be returned from a webserver. Here's a list of the most common ones and what they mean to you:

- 100: The client can continue with a request and it has not yet been rejected by the server.
- 101: The server is willing to upgrade the connection to a different protocol specified in the Upgrade header (an example of this is coming up).
- 200: Successful request. What is sent in the body depends on the method.
- 301: Resource has been permanently moved to a new URI.
- 302: The resource was temporarily relocated to another destination.
- 400: The user sent a request with an invalid syntax to the server.
- 401: User is trying to access a resource, but they have not yet authenticated.
- 403: Server is not serving the file to the user because they are authenticated but likely not authorized.
- 404: The resource the user requested is not available.
- 407: The user must authenticate to the proxy before accessing a resource.
- 500: The server cannot process the request, reason is undefined.
- 503: This likely means the server does not have enough resources to handle the request and is currently overloaded or undergoing maintenance.

The importance of these codes to an analyst is being able to interpret when something was successfully retrieved, or when the client/server was the cause of the issue, and what that issue might have been. Additionally, you may need to "read between the lines" when it comes to response codes and things like using a proxy. If there is a connection to proxy that receives a 407 code, for example, and then does not send back an authentication attempt, what can we assume? Assuming the authentication was supposed to be automatic through Windows and that most connections immediately follow with an automatic authentication, it may mean that there was malware trying to initiate the connection, and not being able to authenticate properly. This is where understanding the regular flow of traffic to the internet within your environment can come in handy.

Reading HTTP Response Headers

```
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Mon, 28 Apr 2014 03:34:31 GMT
Content-Type: text/html
Content-Length: 62471
Connection: keep-alive
Expires: Sat, 26 Jul 1997 05:00:00 GMT
Last-Modified: Sat, 26 Jul 2040 05:00:00 GMT
Pragma: no-cache
```

Reading HTTP Response Headers

This picture shows a typical set of response headers for an HTTP request. In this case, the Date shows the content was requested on Mon, Apr 28, 2014 at 03:34:31 GMT and successfully returned, which is noted in the 200 OK code on the top line. The server used to return the response is nginx version 1.6.0 and the requested content was text/html 62471 bytes long. This request is interesting in that it expresses in multiple ways that it does not want the response cached. The Expires date was set way in the past; the Last-Modified header is set far into the future, and Pragma: no-cache is used, all of which are ways of saying "please don't save this file in a cache."

For analysts, the most relevant parts of this response are the date and time, that the file was successfully returned, it was roughly 61K bytes in size, and is an HTML file.

Discussion: HTTP Request Header Analysis

```
GET /MJ8BBSp4D3uZDSmbE91prhIGA4FmI6bVPLtYhYn5GFISMda HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg,
application/x-ms-xbap, */*
Referer: http://robertsontrainingsystems.com/forum/showthread.php?95-Lumbar-Flexion-when-squatting
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Accept-Encoding: gzip, deflate
Host: bewarecommadelimited.org:85
Connection: Keep-Alive
```

1. What site is the user accessing in this request? What port? What file?
2. What site brought the user to this new site?
3. What type of content were they looking at?
4. What browser and operating system are being used by the client?
5. What language does the user likely speak?

Discussion: HTTP Request Header Analysis

Read through the headers on this slide and see if you can answer the provided questions. The solutions are provided below:

1. The Host header shows the user is accessing `bewardcommadelimited.org` on port 85, a non-standard HTTP port. The resource they are loading is the long string of random characters in the top line after the "GET."
2. Before linking to `bewardcommadelimited.org`, the user was on site `robertsontrainingsystems.com` as shown in the Referer header.
3. The full URL in the Referer seems to show the user was looking up information about working out (`showthread.php?95Lumbar-Flexion-when-squatting`). Since this is a slightly odd Referer to such a random seeming site, we can interpret this to likely mean that the user was on a workout forum site that was infected and had a malicious automatic redirect, or that the user clicked a link to the new site directly.
4. The browser in the User-Agent is reported to be Microsoft Internet Explorer 8.0 on Windows 7.
5. The header seems to indicate the user speaks English since this was the choice provided in the `AcceptLanguage` field.

Discussion: HTTP Response Header Analysis

```
HTTP/1.1 200 OK
Server: nginx/0.7.67
Date: Tue, 29 Oct 2013 04:05:56 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/5.3.3-7+squeeze15
Set-Cookie: PHPSESSID=fdgeg58hv56enk11r3i45sj4g4; path=/
Expires: Thu, 01 Jan 2000 00:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Last-Modified: Thu, 01 Jan 2000 00:00:00 GMT
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 2182

.....Wk.....4Gjf:..s..3...!.0...TUel.%@.....~.d
```

1. Was the page successfully returned?
2. What type of file is returned?
3. What type of server is this?
4. What server-side coding language is being used?
5. When was the returned file last modified?
6. How big is the file?
7. When was this response sent?

Discussion: HTTP Response Header Analysis

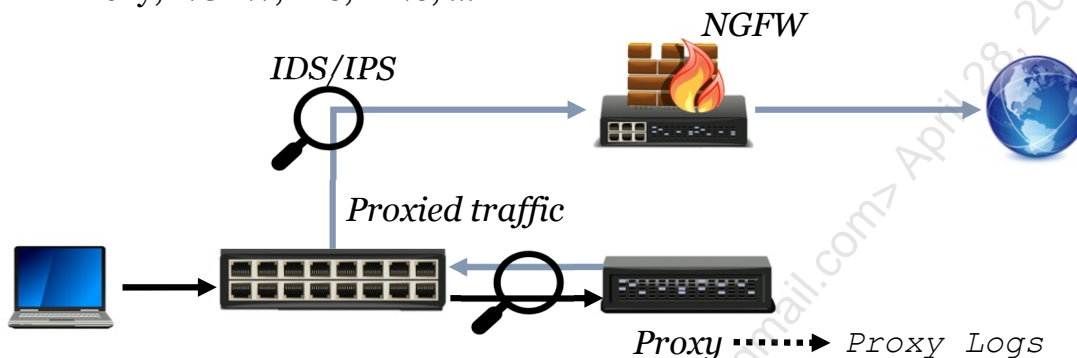
See if you can find the answers to the above questions using the server response. The solutions are provided below.

1. The page was successfully returned. We can see the 200 OK response code at the top.
2. According to the Content-Type header file, the file being returned is an HTML text file.
3. The server is nginx version 0.7.67.
4. This one is slightly tricky. To answer it, we must find the Set-Cookie header and see the PHP Session ID cookie that is being passed back to the user. The answer is PHP.
5. The returned file is marked as last modified Jan 1, 2000.
6. The file returned is 2182 bytes, as seen in the Content-Length header.
7. The Date header indicates this message was sent Tue, 29 Oct. 2013 at 04:05:56 GMT (This message was pulled from an old PCAP, so this is conceivably correct).

HTTP's Path to the Internet

Important: How does HTTP get to the internet in your org?

- Consider path and monitoring / controls that are used
- Proxy, NGFW, IPS, DNS, ...



HTTP's Path to the Internet

One of the most important items to understand as an analyst is the path your traffic takes through your network before it reaches the internet. Think about all the things that must happen before an HTTP transaction makes it out in a modern network that's instrumented with security controls—we'll use the diagram on the slide for an example. A packet must first establish a connection with a proxy on a different network segment (since ideally, we have default outbound deny rule on the firewall and force all traffic through a proxy). This involves traversing the network and crossing the switch, which is a potential collection point. Then, that proxy must agree the user should be able to go to that site and establish its own proxied connection outbound, back through the switch, and this time through an IDS or IPS, which has the chance to identify or stop malicious traffic, as well as through a next-gen firewall, which has its own set of controls. Only if all these devices agree this traffic should reach the internet will the traffic be able to reach the internet.

Your network may or may not be set up exactly like this, but the point is, however it does work, you must understand it. Most compromises have at least some HTTP-based component to them or will at least require frequent outbound communication. Knowing the collection sources available to you along the path that traffic must travel to the internet will allow you to center in on what sensor has the information you require.

WebSockets

- Full-duplex communication channel over single TCP connection
- Starts with HTTP, then upgrades to WebSocket

```
GET / HTTP/1.1
Host: 192.168.43.135:12345
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
Origin: file://
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: bKdPyn3u98cTFZJSh4TNeQ==
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
```

Source	Destination	Info
192.168.43.135	192.168.43.1	WebSocket Text [FIN]
192.168.43.1	192.168.43.135	WebSocket Text [FIN] [MASKED]
192.168.43.135	192.168.43.1	WebSocket Text [FIN]
192.168.43.1	192.168.43.135	WebSocket Text [FIN] [MASKED]
192.168.43.135	192.168.43.1	WebSocket Text [FIN]

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: 4EaeSCku0GBy+rj0SjSMV+VMoC0=
WebSocket-Origin: file://
WebSocket-Location: ws://192.168.43.135:12345/
```

```
> Transmission Control Protocol, Src Port: 12345
> WebSocket
  > Line-based text data (1 lines)
    Welcome, 192.168.43.1 !
```

```
..Welcome, 192.168.43.1 !...~.....221:33:52 ('192.168.43.1', 50999) say: test message....W...;...2...4...521:34:53
('192.168.43.1', 50999) say: hello websocket
```

WebSockets

WebSockets¹ are an interesting alternative to HTTP that allows full-duplex communication back and forth over a single TCP connection. WebSockets can be used for connections that need to have low latency and keep a constant connection to a server that can push the client data at any time without a request. In general, it is a more efficient form of communication and can even be initiated with URIs that start with ws: or wss: for WebSocket Secure. The format can be a little surprising the first time it is seen, but it can easily be read (assuming it's not encrypted) with Wireshark. The protocol will show an HTTP startup with a 101 Switching Protocols response code initiating the upgrade in protocols. The challenge for defenders with WebSockets is verifying that network security monitoring tools understand the protocol and can still catch malicious content transferred with it.

For an example of a malicious use of WebSockets, check out this blog from Black Hills Information Security.²

[1] <https://wiki.wireshark.org/WebSocket?action=AttachFile&do=view&target=websocket.pcap>

[2] <https://www.blackhillsinfosec.com/command-and-control-with-websockets-wsc2/>

HTTP/2

How will Google's SPDY a.k.a. HTTP/2 change the game?

- Faster page loads! But with readability consequences...
- Uses **one TCP connection** with numbered **streams** that contain **frames**
- Sends separate **HEADERS** and **DATA** frames
- Headers are **binary** on wire, not textual
- **Encryption required** for all *browser* traffic, optional for others
- **Server push** means that every file will not have an associated request

No.	Time	Source	Destination	Protocol	Length	Destination Port	Info
4	0.283473	192.168.0.192	106.186.112.116	HTTP	233	80	GET / HTTP/1.1
6	0.567023	106.186.112.116	192.168.0.192	HTTP	140	42895	HTTP/1.1 101 Switching Protocols
8	0.567035	106.186.112.116	192.168.0.192	HTTP2	87	42895	SETTINGS
10	0.567088	192.168.0.192	106.186.112.116	HTTP2	90	80	Magic
11	0.567711	106.186.112.116	192.168.0.192	HTTP2	2962	42895	PUSH_PROMISE, HEADERS
15	0.568137	106.186.112.116	192.168.0.192	HTTP2	1093	42895	DATA
16	0.568272	192.168.0.192	106.186.112.116	HTTP2	109	80	SETTINGS, SETTINGS, RST_STREAM
17	0.568333	106.186.112.116	192.168.0.192	HTTP2	268	42895	HEADERS, DATA

HTTP/2

Up until this point in time, analyzing HTTP version 1.0 and 1.1 connections was easy. We could open the packets in Wireshark and follow a session, seeing all transfer of information, HTTP methods, headers, and URLs. Unfortunately, these days may be numbered. The new standard for HTTP/2 has been released and it is extremely different than previous versions. Most changes have to do with making connections more efficient and enabling privacy with encryption by default. This is great from a user perspective, but unfortunate for network security monitoring as analysts.

HTTP/2 is changing the game for us in that it will no longer require requests for each file to be sent, nor will each request that is made be broken into individual packets. HTTP/2 will make one TCP connection and within each connection use what are called streams, each with individual frames. There are separate frames for HTTP headers and data, and these frames will contain binary data—not plaintext like we are used to seeing traditionally. If that wasn't already hard enough to analyze, the standard mandates that all *browser* based (but not necessarily all) HTTP/2 traffic use encryption. Encryption for non-browser use is optional but will most likely be the norm. In the old versions of HTTP, once a page was loaded, one resource would inevitably reference many others on the same page, pictures, scripts etc., and each one would require the browser to make a separate GET request to acquire. In HTTP/2, the server can assume what you will be asked for and pre-emptively push assets without a request using what is called server push. As analysts, this means tracking down exactly why a file was sent will become much more difficult, if you can even see it in the first place.

The slide includes a screenshot of how a simple page load looks using (unencrypted) HTTP/2. Notice the GET request for / followed by a response saying a protocol switch is occurring. Then there are a bunch of HTTP/2 specific functions that follow instead of the typical "200 OK" response that is easy to understand. The next slide dives deeper into what this transaction would look like when analyzed in Wireshark.

HTTP/2 in Wireshark

```

GET / HTTP/1.1
User-Agent: curl/7.41.0-DEV
Host: nghttp2.org
Accept: */*
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c-14
HTTP2-Settings: AAMAAABkAAQAAP_

HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c-14

.....d.....PRI * HTTP/2.0

SM

.....z.%P.....f..i.....a..Y>...a,j..m@...jb...I|...M..
.q.jl..z....a,j..m@.q..6%1h....[h..Yy`.I..m.....r.(a....r!.....
-...j
.|>.*')SR...J.v..i....T..5.....|.
.R..Jk.....
<!DOCTYPE html>
<!--[if IEMobile 7 ]><html class="no-js iem7"><![endif]-->
<!--[if lt IE 9]><html class="no-js lte-ie8"><![endif]-->
<!--[if (gt IE 8)|(gt IEMobile 7)!!(IEMobile)!!(IE)]><!--><html class="no-js" lang="en"><!--<![endif]-->
                
```

Upgrading connection...

```

HyperText Transfer Protocol 2
> Stream: PUSH_PROMISE, Stream ID: 1, Length 28
v Stream: HEADERS, Stream ID: 1, Length 157
  Length: 157
  Type: HEADERS (1)
  > Flags: 0x04
  0... .. = Reserved: 0x0
  .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
  [Pad Length: 0]
  Header Block Fragment: 886196e4593e940bca612c6a08016d4082e32dd
  [Header Length: 343]
  [Header Count: 10]
  > Header: :status: 200
  > Header: date: Wed, 18 Feb 2015 10:35:15 GMT
  > Header: content-type: text/html
  > Header: content-length: 6607
  > Header: last-modified: Mon, 16 Feb 2015 14:39:52 GMT
                
```

```

.....z.%P.....f..i.....a..Y>...a,j..m@...jb...I|...M..
.q.jl..z....a,j..m@.q..6%1h....[h..Yy`.I..m.....r.(a....r!.....
-...j
.|>.*')SR...J.v..i....T..5.....|.
.R..Jk.....
<!DOCTYPE html>
<!--[if IEMobile 7 ]><html class="no-js iem7"><![endif]-->
<!--[if lt IE 9]><html class="no-js lte-ie8"><![endif]-->
<!--[if (gt IE 8)|(gt IEMobile 7)!!(IEMobile)!!(IE)]><!--><html class="no-js" lang="en"><!--<![endif]-->
                
```

HEADERS frame
(binary)

```

.....z.%P.....f..i.....a..Y>...a,j..m@...jb...I|...M..
.q.jl..z....a,j..m@.q..6%1h....[h..Yy`.I..m.....r.(a....r!.....
-...j
.|>.*')SR...J.v..i....T..5.....|.
.R..Jk.....
<!DOCTYPE html>
<!--[if IEMobile 7 ]><html class="no-js iem7"><![endif]-->
<!--[if lt IE 9]><html class="no-js lte-ie8"><![endif]-->
<!--[if (gt IE 8)|(gt IEMobile 7)!!(IEMobile)!!(IE)]><!--><html class="no-js" lang="en"><!--<![endif]-->
                
```

DATA frame

SANS

SEC450: Blue Team Fundamentals: Security Operations and Analysis

114

HTTP/2 in Wireshark

This is the followed TCP session from the previous slide showing a GET request capable of upgrading to HTTP/2, and the following response from the server. Since the header's frame is now in a binary format, we must use Wireshark to interpret its meaning for use instead of reading it directly from the bytes. The data frame is still human readable, but, unfortunately, these may be broken up over different data frames and different streams. Tools for analysis of HTTP/2 will likely need to play catch-up before this becomes easy.

HTTP/3 is Here, Too!

Google's "QUIC" protocol has already been named HTTP/3!

- More of a revision of Layer 4 than Layer 7
- Currently uses UDP to port 443; goal is to eventually replace UDP/TCP
- Usage on the rise! Check URL below² for current usage

No.	Time	Source	Destination	Info
1	03:12:16.815947	192.168.1.109	216.58.212.101	Client Hello, PKN: 1, CID: 16953050174146338482
2	03:12:16.861947	192.168.1.109	216.58.212.101	Payload (Encrypted), PKN: 2, CID: 16953050174146338482
3	03:12:16.876004	216.58.212.101	192.168.1.109	Payload (Encrypted), PKN: 1
4	03:12:16.876734	192.168.1.109	216.58.212.101	Payload (Encrypted), PKN: 3, CID: 16953050174146338482
5	03:12:16.941384	216.58.212.101	192.168.1.109	Payload (Encrypted), PKN: 2

> Frame 4: 79 bytes on wire (632 bits), 79 bytes captured (632 bits)	0000	64 70 02 8d 3d 39 78 92 9c 0f a8 8e 08 00 45 00	dp...=9x...E-
> Ethernet II, Src: IntelCor_0f:a8:8e (78:92:9c:0f:a8:8e), Dst: Tp-LinkT_B	0010	00 41 1d 92 40 00 40 11 ae 64 c0 a8 01 6d d8 3a	-A-@.@-d...m:
> Internet Protocol Version 4, Src: 192.168.1.109, Dst: 216.58.212.101	0020	d4 65 e1 e9 01 bb 00 2d 28 e2 0c b2 de 5d f1 4f	-e-.....(.....):0
> User Datagram Protocol, Src Port: 57833, Dst Port: 443	0030	55 45 eb 03 ae ef e4 82 ad 86 ee 5d 68 a8 0f 43	UE.....}h...C
▼ GQUIC (Google Quick UDP Internet Connections)	0040	27 1b d4 8f 8f f9 7d 36 cc 3e e7 30 9e 20 66	'.....}6 ->-0- f
> Public Flags: 0x0c			
CID: 16953050174146338482			
Packet Number: 3			
Payload: aeeffe482ad86ee5d68a80f43271bd48f8ff97d36cc3ee730...			

HTTP/3 is Here, Too!

Disappointed by the pain of HTTP/2? Don't worry, HTTP/3 is already here. The thing is that, unfortunately, it doesn't make it any better. Like SPDY for HTTP/2, the Quick UDP Internet Connections (QUIC) standard was also developed by Google and is described in the design document as "very similar to TCP+TLS+HTTP2 but implemented on top of UDP".¹ HTTP/3 isn't so much focused on changing the HTTP transaction language, but implementing HTTP/2 over connectionless UDP packets, and eventually even replacing UDP! The slide shows a sample PCAP of HTTP/3 as seen in Wireshark. Notice there is almost no data of use at all except for the destination IP address. Given the two oncoming standards, NSM will likely look *very* different in the not too distant future.

Fortunately, HTTP/2 and HTTP/3 are not quite yet the majority of traffic. Don't be surprised if these numbers change quickly, though.² At this point, the technology is on a quick path to adoption due to its vast performance and security improvements. That means regardless of our desires, it is important to be aware of how it works and the implications it will have on analysis. When HTTP/2 and HTTP/3 become the norm, we will essentially blind below the IP layer unless tools are able to adapt and decrypt the data.

[1] <https://www.chromium.org/quic>

[2] https://w3techs.com/technologies/overview/site_element

Understanding HTTP Summary

- Understand URL protocol, hostname, path, parameters
- How server-side scripting changes responses
- HTTP methods: GET and POST + others
- HTTP response code interpretation
- Most important request and response headers
 - Virtual Host, User-Agents, Referer, Method, MIME type
- Effects of new HTTP standards and protocols
 - WebSockets, HTTP/2 (SPDY), HTTP/3 (QUIC)

Understanding HTTP Summary

HTTP is one of the most common protocols and is used extensively on almost every network. Everything runs over HTTP—webpages, email, messaging, social networking, calendars, and yes, malware, too. Therefore, understanding and interpreting an HTTP request and response will likely be one of the most common activities you will do in day-to-day work. Analysts must be able to decode a URL into its constituent parts, understand how webservers work and what type of response may be generated, and know the HTTP methods used to exchange different types of input. Being able to view request and response headers and immediately notice if anything is suspicious is an essential analyst skill since so much malware tries to hide running over protocol compliant HTTP transactions. While this is easy now, like DNS, unfortunately the future capabilities for HTTP analysis is somewhat uncertain. New standards with mandatory encryption are coming down the pipe, which will undoubtedly make analysis more difficult and will obscure all but decrypted traffic from view. With any luck, SSL decryption will become more prevalent to compensate, but who knows what the future will hold. At least we get to enjoy it while it lasts!

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. **HTTP Analysis and Attacks**
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

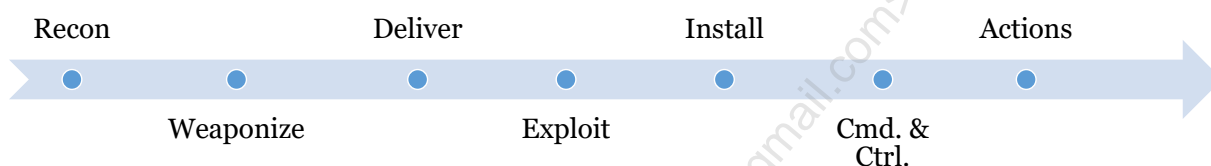
This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

Evil Usage of HTTP

HTTP is used for almost all Cyber Kill Chain phases:

- Deliver/Exploit: Exploit Kits, Phishing, Fake Updates
- Installation: Malicious executables and scripting
- Command and Control: Periodic GET/POST
- Data Exfiltration: Encoded/Encrypted upload



Evil Usage of HTTP

Since HTTP is one of the most ubiquitous protocols, it makes a great choice for attackers to use as well. As shown above, it can be used for almost every phase of an attack (displayed here using the Lockheed Martin Cyber Kill Chain steps—a model we will detail later in the course). It blends in well with typically allowed traffic, allows the delivery of malicious files, facilitates credential theft via phishing sites, and works for command and control and data exfiltration. Therefore, as analysts, we must be able to spot it being used for all these stages of an attack. Throughout this module, we'll discuss some ways of picking out specific evil uses of HTTP as well as some general ways of identifying anomalous HTTP usage.

HTTP Analysis Methods

Automated methods:

- Social Engineering
 - Screenshots
- Delivery / Exploitation
 - Sandboxing
 - URL reputation checks

Manual Methods:

- Installation
 - File extraction
- Command and Control and exfiltration
 - Header analysis
 - Content analysis
 - Anomaly detection

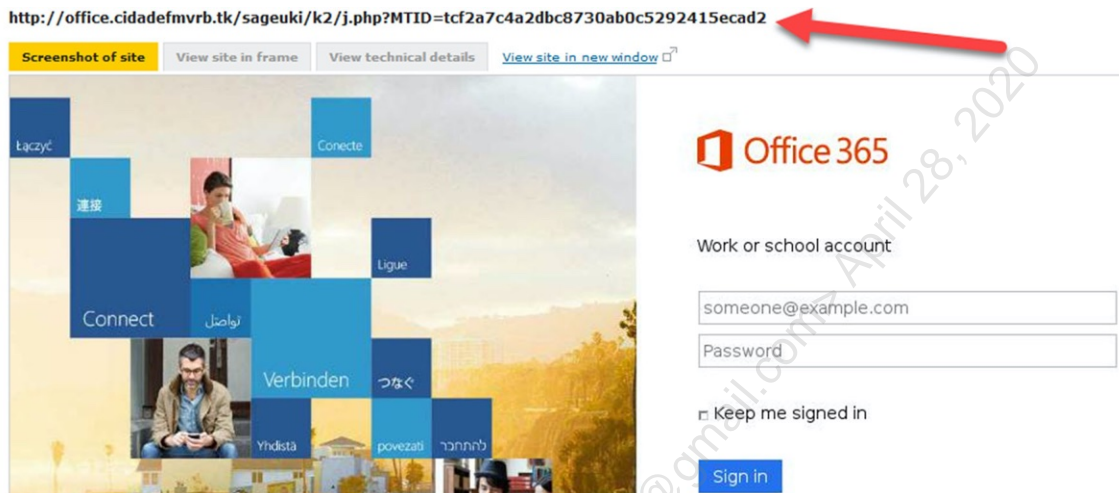
HTTP Analysis Methods

When it comes to HTTP Analysis, there are many tools for the job. The important part is picking out the analysis method that will get you to the correct answer the fastest. Although manual analysis is potentially more likely to produce a correct result, it's not always needed when an automated tool or website lookup might deliver you the answer immediately. Analysts must balance their time and being efficient is often the name of the game, but not to the point where accuracy is compromised.

In this section, we'll review ways to analyze HTTP transactions in both automated and manual ways. For automated analysis, we have tools that range in how comprehensive they are from screenshots, to reputation checks, to full automated sandboxing environments for deep technical analysis. On the manual analysis side, we can always investigate logs or PCAPs and dig into headers and page content to see what was transferred. Sometimes, this level of analysis will be necessary when automated tools cannot produce the answers we need, so understanding all methods will be crucial to your effectiveness in HTTP event triage. Note that certain types of manual and automated analysis methods will work best for identifying different stages of an attack. Screenshots, for example, are often enough to identify attempted phishing websites but will fail to give a useful opinion if a URL is a command and control site. Let's dive in and see how we can best find attacks perpetuated via HTTP.

Checking for Credential Phishing

Screenshots: One of the easiest URL checks¹



SANS

SEC450: Blue Team Fundamentals: Security Operations and Analysis 120

Checking for Credential Phishing

Sometimes, it's incredibly easy to tell a site is being used for credential phishing attempts just by looking at it. Unfortunately, spam reported by users doesn't automatically come with a screenshot; the good news is it's extremely easy to generate one yourself.

Assuming you aren't worried about having a third party load the link to test it, there are many sites that are willing to load a page and present you with a picture. Website screenshot tool capabilities range from a simple picture on sites like page2images.com, to a screenshot that comes with a full sandbox analysis performed by hybrid-analysis.com. Some sites focus on sharing the link as a malicious site with the community—PhishTank.com is a service designed for having the information security community at large view the screenshots and lend a vote as to whether a site is bad or not. Being familiar with each option helps tailor your decision for the given situation. If you do not want to share a link with the public, for example, be sure to pick a tool that doesn't record the link and put it in a database for the world to see.

Tools:

- **Urlscan.io:** Network transaction centric site that shares links and will record tons of information about all files transferred when a link is visited. Great for historical reference on sites that have been taken down.
- **Page2Images.com:** Simple screenshot generator; will not save and share links with others.
- **URLQuery.net:** Generates a screenshot, checks page against IDS rules, lists all page resources loaded, and shares links with the public. Great for historical reference on sites that have been taken down.
- **Phishtank.com:** Designed around crowdsourcing identification of phishing sites. Links can be submitted and users can vote on whether a site is a phishing link (the screenshot above is taken from the site).¹
- **Hybrid-analysis.com:** Sandbox-oriented analysis of URL, makes all submissions public, includes screenshot plus a VERY thorough analysis.

[1] Screenshot and example link provided by PhishTank.com

URL Reputation

One of the easiest methods: Third-party reputation checks!

- Fastest way to get confirmation URL is probably evil
 - Negative result does *not* mean it's clean
 - May or may not comply with operational security standards

• Some of the best free options:



- VirusTotal.com
- URLVoid.com / IPVoid.com
- Trustedsource.org / safeweb.norton.com
- Google Safe Browsing

URL	Status	Categorization	Reputation
http://sans.org	Categorized URL	- Education/Reference	Minimal Risk



URL Reputation

One of the easier automated ways to check if a URL may be malicious is to check its reputation on one of the multitude of websites that track malicious webpages. The tools listed on this slide are some of the bigger and more well-known names for doing reputation checks, but there are numerous options beyond just these. These tools each have their own motivation and purpose for offering URL reputation service and therefore the output from each will have different levels of detail depending on the intended audience.

- VirusTotal: The first stop for most analysts. It will check a URL with several URL scanning suites. It's probably is the best place to go for a breadth of opinions aggregated from many different engines. The downside of the VirusTotal URL check is that it doesn't offer many specifics on why any particular engine thought a URL was bad or not, just the good/bad determination.¹
- URLVoid / IPVoid: Another dedicated site checker that has been around for a long time. URLVoid offers IP resolution details, basic domain info, and shows any matches for your entry from multiple domain blacklists like VirusTotal. The difference between URLVoid and VirusTotal is the blacklist sources, which are significantly different, it may be a good place to go for a second opinion after VirusTotal. The site IPVoid.com is run by the same group and offers similar checks for IP addresses.²
- TrustedSource / Norton Safeweb: These sites are run by major security vendors looking to protect their customers from surfing to malicious websites. They offer these sites as free service and a way of directly checking the databases they build into their products. The benefit of these sites is that they likely have a very large database of both whether a URL is known bad or not, but also categories for a given URL, which may add context to an investigation (such as is this a site created for evil, or a hacked site delivering evil right now?). The slide shows the output of a TrustedSource.org looking for SANS, where the status, categorization, and reputation can be found. Analysts can assume that anything classified as high risk is likely bad, but sites that are uncategorized are just that – unknown.^{3,4}

- Google Safe Browsing: Like TrustedSource, the determination Google has made for a site that would be built into the Chrome browser can be directly queried from the web interface. It unfortunately does not give much detail beyond good/bad, but since it's Google, it's likely they have a very extensive list of known bad sites.⁵

[1] [virustotal.com](https://www.virustotal.com)

[2] urlvoid.com, ipvoid.com

[3] [trustedsource.org](https://www.trustedsource.org)

[4] [safeweb.Norton.com](https://safeweb.norton.com)

[5] <https://transparencyreport.google.com/safe-browsing/search>

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

VirusTotal URL Search vs. Domain Search

URL Search

No engines detected this URL

URL: <http://sans.org/>
 Host: sans.org
 Last analysis: 2018-10-27 10:17:55 UTC
 Community score: +3

0 / 67

Detection	Details	Community
ADMINUSLabs	✓ Clean	
AlienVault	✓ Clean	
Avira	✓ Clean	
BitDefender	✓ Clean	
C-SIRT	✓ Clean	
CLEAN MX	✓ Clean	

Domain Search

sans.org domain information

Categories

Alexa	research
BitDefender	computersandsoftware
Forcepoint ThreatSeeker	information technology
TrendMicro	computers internet,education
Websense ThreatSeeker	information technology

Passive DNS Replication

Date resolved	IP address
2018-11-21	45.60.103.34
2018-11-21	45.60.31.34
2018-09-10	45.60.33.34
2017-08-20	45.60.32.77
2017-08-17	45.60.34.77

VirusTotal URL Search vs. Domain Search

There are two ways of searching for information on VirusTotal—entering a full URL such as <http://sans.org> (left), or the domain name without any protocol specified (right). Each of these will net very different results. On the URL search page, you receive the scan results of multiple URL evaluators, the same as how entering a file will run it through multiple AV scanners. The URL report will also give details on the HTTP headers received in response when loading the site, site categories, the IP address the site is hosted on, and the sha256 hash of the response body (in case a malware sample is served).

Entering the domain gives a much more detailed report. The downside is that it is not focused on the HTTP service; it's more about whether the domain as a whole is evil. Checks like this may fail when there is a single resource that is bad on an otherwise good domain. Checking wordpress.com, for example, as a domain will return that it is a reputable site, but specific URLs on people's blogs can be malicious—something a domain search wouldn't turn up, but a URL search could.

Included in the domain analysis is:

- Domain classification categories
- Passive DNS information on what IP addresses A records have pointed to for the domain and the date
- whois information
- All known observed subdomains
- URLs known associated with the domain
- Files that have been downloaded from the domain
- Submitted files that communicate with the domain
- Submitted files that have some occurrence of the domain inside their text

Notice that the domain search does *not* include a verdict if the domain is good or evil if accessed over HTTP, although that information may be predicted from the other provided information.

URL Sandboxing

Going deeper: Sandboxing

- Loads a URL and logs all activity produced
- Report more complex to interpret
- Some of the best free tools:
 - Hybrid-analysis.com
 - Urlscan.io
 - Urlquery.net
 - Any.run



URL Reputation

If URL reputation services don't offer the detail you need, you can step it up to a sandboxing tool. These sites will render the page in a virtual machine, look for signs of malicious activity, and record what happens on the host that opened the link in case any suspicious changes are made, or processes are spawned. Each tool has its own strengths and weakness, so the key to using sandboxing services is knowing which offers the best output for the answer you're looking for. Are you looking for the name of JavaScript files that run when a site is loaded? There are sites for that. Are you trying to see screenshots of a desktop loading the site? Some do that as well. Do you need a PCAP of the interaction and the availability to download any files loaded from the site? Some sandboxes do that, too. Some sites even give IDS alert matches, MITRE ATT&CK framework tactic hits, and more!

When it comes to sandboxing sites, some of the standards are listed on the site and each should be tested by analysts to investigate their relative strengths and weaknesses. In general, hybrid-analysis.com and any.run are highly detailed and more malware file focused and therefore have great heuristic checks for what is done to the host, but the UI is not as focused on network connections as some other sites. URLscan.io is a bit more network centric and offers more detail on the actual HTTP transactions that occurred, allowing you to see individual requests and responses and download each one. URLQuery.net is good in that it gives a listing of potentially suspect JavaScript that executed, IDS alerts that were triggered, a nice visual representation of all the traffic generated from a given site (shown on slide), and a list of reports generated on the same domain, IP and ASN.

For additional links, Lenny Zeltser, author of SANS FOR610 – Reverse Engineering Malware course, maintains a great list of resources for looking up a malicious websites using URLs or sandbox on his blog.¹

[1] <https://zeltser.com/lookup-malicious-websites/>

Manual Header and Content Analysis

If reputation, screenshots, and sandboxing aren't enough...

- Installation and Delivery
 - File analysis, remote resources
- Header analysis: User-Agents, Cookies, Referers
- Command and Control: GET/POST content analysis
- Anomalous Behavior
 - Naked IP addresses
 - Base64 encoded content

Manual Header and Content Analysis

When quick automated methods like sandboxing, screenshots, and reputation checks don't give you the answer you're looking for, manual analysis might produce the results you seek. Looking at headers for HTTP transactions take a bit of time to get used to, but after enough samples of good and bad, you'll start to get an intuitive sense of what makes a transaction suspicious. Let's review a few of these methods to jumpstart the learning process. We'll discuss how to identify a potentially malicious HTTP transaction based on the virtual host, user-agent, URL, Referer, and body content. No single one of these will always provide a flawless result, but there are often signs that can be picked up on that can start to sway conclusions toward the correct answer.

User-Agent Analysis

Sometimes, User-Agents can identify malware

- Remember, field is optional and **self-reported**
- Still, lots of malware doesn't mask itself
- Even if it does, it may not match the rest of your network
 - Compare OS type, version, 32/64 bit
 - Browser – Version, plugins

```
GET / HTTP/1.1
Host: 10.5.11.103
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/31.0.1650.63 Safari/537.36
```

User-Agent Analysis

Since at compilation time malware authors must select what User-Agent, if any, the malware will use when it communicates, it is likely that eventually this may lead to exposing it. First, many malware authors simply do not care or take the time to choose a User-Agent that matches that of real software. In this case, it is easy to spot a bogus User-Agent with a SIEM and identify the infected device. What if the malware has disguised itself as a "normal" User-Agent, however? Assuming the malware author did not include the ability to change reported User-Agents over time, the OS and browser version numbers used in the malware will either immediately, or at least eventually, stick out against an environment that continues to update. Easy tells can be malware pretending to be Linux in a Windows environment, or claiming to be Firefox when it is against company policy to have it installed. Beyond simple OS and software, however, a statically specified (within the malware, always saying "Chrome 50.0" for example), for example, may hide the malware for a short period of time. However, as that statically programmed version ages compared to the true Chrome, it will start to stick out as a user that hasn't updated for a curiously long time, eventually drawing your attention. Smaller tipoffs like this can be easy to spot but are often ignored by the blue team. A frequency analysis of all the User-Agents on a network can help highlight discrepancies and be a good starting point for generating investigation leads for hunt teaming.

For an example of what some bad User-Agents look like, check out this list on GitHub from the Nginx bad bot and user-agent blocking project.¹

[1] https://github.com/mitchellkrogza/nginx-ultimate-bad-bot-blocker/blob/master/_generator_lists/bad-user-agents.list

User-Agent Deconstruction

- Mozilla/5.0
 - All versions of Firefox, Chrome, Safari, Edge, and Internet Explorer past version 8.0 use this prefix with 5.0 as the number
- (Windows NT 10.0; Win64; x64; rv:63.0)
 - Operating system, 32/64-bit OS, 32/64-bit browser, browser revision
- Gecko/20100101
 - Rendering engine for the browser, usually less interesting unless odd
- Firefox/63.0
 - Browser name and version number – compare to allowed software and if the version indicates it's up to date

User-Agent Deconstruction

Let's take a quick walk through each field in a modern Firefox User-Agent string from Windows 10 to see what the field is actually telling us.

- "Mozilla/5.0" – For historical reasons, almost every browser will now start with the term "Mozilla 5.0" in the User-Agent.¹ If you see anything other than Mozilla, you are likely seeing an old or non-standard browser client (Opera) or other non-browser software making the transaction. In addition, if you see Mozilla but not 5.0 (4.0, or otherwise) it is also suspect, as this would indicate either a very old browser, or potentially malware that did a poor job of replicating what should be in this field. All versions of Chrome and Firefox all the way back to the beginning have used Mozilla/5.0 in their User-Agent, and Microsoft uses it for all versions of Edge and Internet Explorer as well for anything past version 8.0.²
- "Windows NT 10.0; Win64; x64; rv:63.0" – This is the platform field, which should express what operating system the client is running on. In this case, the client was running Windows 10 with a 64-bit browser (you may see WOW64 which means 32-bit browser on 64-bit Windows). The "rv:63" is a tag for the version of the browser being used – note it matches the number at the end of the Firefox tag later in the user-agent.
- "Gecko/20100101" – Firefox uses what is called the "Gecko" rendering engine, IE says Trident, Safari, *Edge*, and *Chrome* (this can be confusing) says AppleWebKit. This field is usually not as useful as the others.
- "Firefox/63.0" – Indicates the browser is Firefox version 63, this field is where you will see Chrome, Safari, Edge, or maybe multiple entries. Unfortunately, for browsers other than Firefox, this field can be a bit confusing.

Here are some additional examples of modern browsers for reference. These also demonstrate how the final browser version field can be a bit confusing (bolding added for emphasis; no these are not typos).²

- Chrome 70.0.3538.77 - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) **Chrome/70.0.3538.77** Safari/537.36

- Safari 7.0.3 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3)
- AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 **Safari/7046A194A**
- Edge 14.14931 - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/51.0.2704.79 Safari/537.36 **Edge/14.14931**

One of the best ways to surface odd user-agents is to use your SIEM to sort unique User-Agents seen on your network by count and look at the ones with very little usage. This is called "long-tail analysis" and is often used during threat hunting activities to identify anomalies in large data sets.

[1] <https://webaim.org/blog/user-agent-string-history/>

[2] <http://www.useragentstring.com/pages/useragentstring.php>

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

Cookies

ChChes Malware Command and Control Callback

```
GET /X4iBJjp/MtD1xyoJMQ.htm HTTP/1.1  
Cookie: uHa5=kXFGd3JqQHMFnmMbi9mFZAJHCGja0ZLs%3D;KQ=yt%2Fe (omitted)  
Accept: */*  
Host: kawasaki.unhamj.com
```

NotPetya Callback

```
GET /last.ver?rnd=0e5ae4fbc9904d81987586e496edf281 HTTP/1.1  
Cookie: EDRPOU=1112222;; un=Admin  
User-Agent: medoc1001189  
Host: upd.me-doc.com.ua
```



Cookies

Cookie headers can be used for hiding evil as well. Using the cookie field for data besides cookies is a clever technique for attackers to use because the sensitive nature of cookies means they do not show up in most logs. That means finding malware command and control or other information inside the cookie field will likely involve having a reason or tipoff to pull up the cookie field data and look at it in the first place. Therefore, data placed inside it is more likely to go unnoticed.

This slide has two examples of malicious data riding along an HTTP transaction in the cookie header, can you spot them? The first example is from the ChChes malware¹ used in targeted attacks in Japan. The cookie field contains base64 encoded data about the infected device and is being used to transfer command and control instructions. We'll discuss how this encoding works in a moment.

The second example is also very sneaky—it's a callback from the NotPetya malware that paralyzed many companies in mid-2017 before it had released its ransomware component.² The malware was bundled into legitimate accounting software and the company that created the software had their own servers hijacked for command and control points. Therefore, this cookie contains not only the real data that would be sent, but a bit of extra info "un=Admin", which is a communication back to the command and control server that the malware was running with administrative credentials on the infected endpoint. Tricks like this are incredibly hard to spot, especially when combined with a supply-chain attack as was used for NotPetya. This is undoubtedly one of the reasons that no one knew the attack was coming until the day the attackers decided to use the command and control to push out the destructive malware that brought down companies across the world.

[1] <https://blogs.jpccert.or.jp/en/2017/02/chches-malware--93d6.html>

[2] <https://www.welivesecurity.com/2017/07/04/analysis-of-telebots-cunning-backdoor/>

Base64 Encoding: What is it?

- A way of mapping ALL byte sequences to printable ASCII
 - Often used for encoding (*not* encryption)
 - Valid characters: A-Z, a-z, 0-9, +, /, = (notice, 65 char. options)
 - Often ends with one or two "=" signs
- Used to encode non-printable characters for channels that can't carry them such as SMTP
- Often used in HTTP for cookies, POST'd values, and more

```
[~]$ echo hello_world | base64
aGVsbG9fd29ybGQK
[~]$ echo aGVsbG9fd29ybGQK | base64 -d
hello_world
```

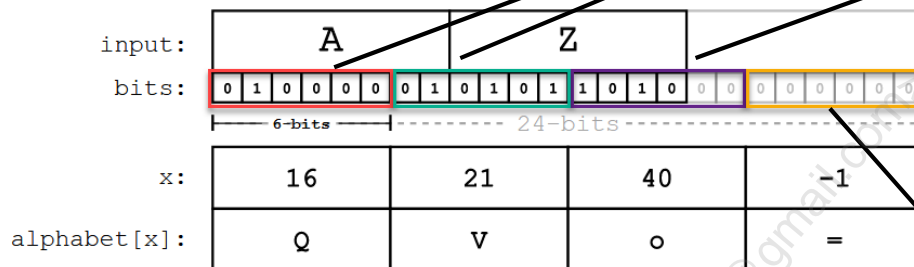
Base64 Encoding: What is it?

Base64 encoding is a concept that must be covered at this point because it is an encoding method that is often seen in HTTP transactions both good and bad. Base64 was first used as a way of encoding all possible 8-bit byte values into a 6-bit ASCII printable format so that content beyond basic text could be sent over email. Why must it be encoded like this? Because the standard specified that the only valid byte sequences that could be transferred over SMTP were values 0-127. Why then was it not base128? Because many values in the ASCII character set are non-printable control characters as well, such as carriage return, line feed, bell, and more.

When it comes to base64, you should be able to know it when you see it, as well as realize when it may be out of place. Many legitimate web applications use it for fields in cookies and other authorization mechanisms, but malware often uses it as well to transfer files and perform command and control. Let's take a closer look at how base64 encoding works, which will help make it easier to spot in the future.

Base64 Conversion Demonstrated

1. Break 8-bit values into 6-bit values
2. Convert 6-bit sequence to B64 char. set
3. Pad with = if required



6-bit B64 Value	Character
0	A
1...24	B...Y
25	Z
26	a
27...50	b...y
51	z
52	0
53...60	1...8
61	9
62	+
63	/
Padding	=

Base64 Conversion Demonstrated

The picture on this slide from lucidchart.com's visual base64 converter makes it easy to see how the 8-bit character values of the string "AZ" turn into "QVo=" when base64 encoded.¹ First, the 8-bit ASCII values are broken into the 6-bit long values shown in the highlighted boxes. Then, the 6-bit long strings are re-encoded using the designated base64 character set. By standard, the base64 character set is the way of mapping all possible 6-bit values to the characters A-Z, a-z, 0-9, +, and /. The decimal value each of those characters represents is shown in the abbreviated conversion chart on the right. The character A would convert to 6 binary zero bits (decimal 0), while the "/" character would convert to 6 binary 1's (63 in decimal). The 64-character options represented in A-Z, a-z, 0-9, +, and /, make up the 64 different values that can be created with a 6-bit number (0-63 in decimal).

This example highlights something important about base64 that will help you identify it when you find it in the wild. Since ASCII characters are 8 bits long and base64 values are 6 bits long, sometimes the base64 value must be padded to the nearest length that is divisible by both 6 and 8 (24 bits in this case) to make a clean conversion. Therefore, sometimes a section of padding bits must be added at the end (the grayed-out byte after the Z in the original string in the slide), and this is represented with a special padding character for base64 encoding, the "=" sign. This means that when you see a sequence of characters somewhere that uses the valid base64 character set, if it ends with an equal sign, you can be fairly confident it is a base64 encoded string.

Since base64 is a method for taking any possible 8-bit value and turning it into the printable character set (which things like SMTP require using for data transfer), it is a very common method used for data encoding. It is important to be able to quickly spot base64 encoding as it is also commonly used by malware to transfer binary data over HTTP, SMTP, and other channels as well. Using base64 encoding both helps attackers avoid signature-based detection and obscure what data is actually being transferred. Therefore, if you see a POST request with a giant blob of base64 encoded text going to a questionable server, your next move should be to attempt to decode that data and see what was being sent out. Note that it is possible to encrypt data, *then* base64 encode it, so decoding a base64 value will not necessarily yield an immediately interpretable result.

[1] <https://www.lucidchart.com/techblog/2017/10/23/base64-encoding-a-visual-explanation/>

Suspicious Base64 Content

APT19 malware beacon:¹

```
GET /IE9CompatViewList.xml HTTP/1.1
Accept: */*
Cookie: Vvi0vVcTyBwKhZda0iMm+Ht+ya5Ko7XkIiI7727uHEukwkqXtAUnaJcX7exCJGWDNB/horP9S09x9cyfhSI8RG6RKM
+9HmGh4ApI096Ie2EFbSZ0Dh108TXG2C1YeHXjhAmxuHMJfSrB8fPRv99EAfEGPDrZgnimJIItemdu6lpM=
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; FunWebProducts; IE0006_ver1;EN_GB)
Host: autodiscover.2bunny.com
Connection: Keep-Alive
Cache-Control: no-cache
```

Ixshe malware beacon:²

```
POST /bbs/search.asp HTTP/1.1
Accept: */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; windows NT 5.0)
Host: info.xxuz.com

BARLAGAKWABAbDCR/QE=
```

Suspicious Base64 Content

Now that we know what base64 encoding looks like, where should it appear? The problem is it's hard to say. On this slide, there are two examples of HTTP traffic, one GET request and one POST request, both from state-sponsored APT group malware.^{1 2} The GET request has base64 content in the cookie field, which although not necessarily typical, is conceivably a way that someone has decided to encode cookie values. The POST request has base64 content in the request body, which again is not necessarily evil, but the question that should be raised in your head is, "why?" Data could be sent, in most cases, as part of a POST request unencoded—are they trying to hide something?

Neither of these examples clearly screams "evil" when only the base64 encoding pieces are considered but combined with other header information, they should raise red flags. If you found requests such as this, your thinking process should be something along the lines of "I see base64 in a place I don't suspect it needs to be used, AND it seems these are going to some questionable hostnames (2bunny.com, info.xxuz.com), AND they contain User-Agent strings that are very much non-traditional. This needs to be looked into further!" The next step in the analysis process here could be attempting to decode these base64 strings. As mentioned before, you won't necessarily be able to interpret the output, but you would at least be able to verify it was legitimate, convertible base64, and if you're lucky, may get an idea of the nature of the data transferred.

A glance at the MITRE ATT&CK "Data Encoding" technique³ shows a long list of attackers and tools that used base64 as a way of encoding data with malware. The usefulness and prevalence of base64 encoding go way beyond HTTP as well. We will see it again when we discuss endpoint processes and questionable command line arguments.

[1] <https://www.fireeye.com/blog/threat-research/2017/06/phished-at-the-request-of-counsel.html>

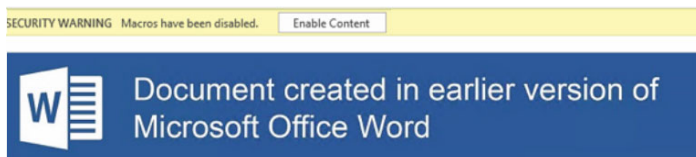
[2] <https://www.fireeye.com/blog/threat-research/2013/08/survival-of-the-fittest-new-york-times-attackers-evolve-quickly.html>

[3] <https://attack.mitre.org/techniques/T1132/>

File Analysis

Common tactic: Serve malicious files from HTTP(S) sites

- **Example:** Jan 2018 "Lazarus Group" delivers targeted phishing campaign via Dropbox share¹



- Consider: How to identify this?
 - Good User-Agent, Referer, Host, method, just an **evil file**
- **Files can easily be carved from HTTP traffic** (if decoded)

File Analysis

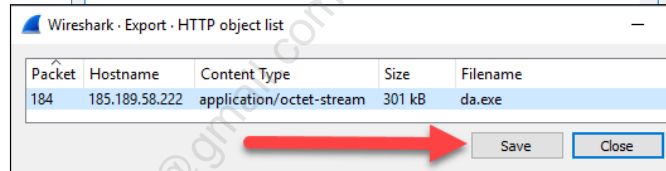
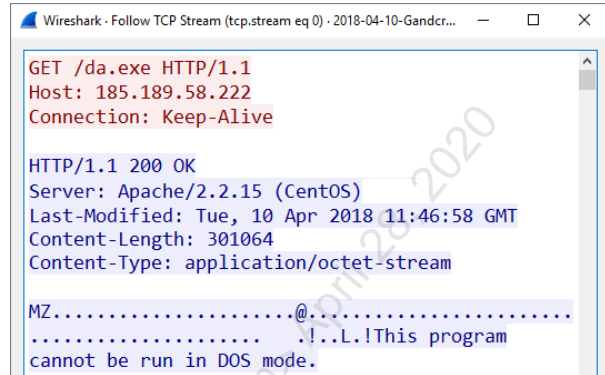
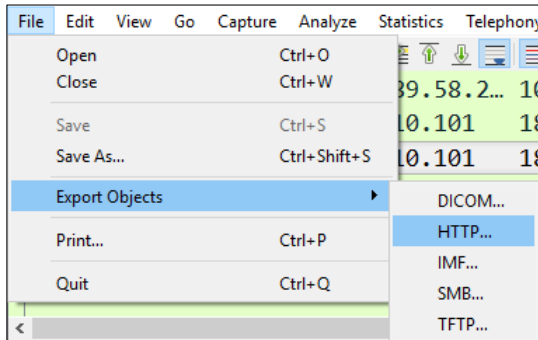
Attackers love serving malicious files over HTTP. Since nearly every organization on earth must allow the protocol, it's a great way of delivering attacks to victims. How do we deal with this when we suspect someone downloaded a malicious file? Fortunately, if we have a copy of the HTTP traffic, it is easy to carve the file out (shown on the following page).

Don't forget, a file that came from a well-known and respected site can still be evil. When it comes to cloud hosting services such as Dropbox, OneDrive, Google Drive and the like, any file can be stored for download by the public. In many cases, we have seen attack groups use these services to spread malware by staging it and sharing links to it over email. This is a particularly tricky situation since the interaction will have a standard user-agent, no Referer to key off of, a Host field of a "known good" service (although they may not be allowed by policy), and a normal looking set of request methods to download the file. What is left to go on? The file itself. As long as the connection has been decrypted, we can extract the file out of the network stream or have our tools do the extraction for us. Unfortunately, sometimes it's the only way.

[1] <https://securingtomorrow.mcafee.com/mcafee-labs/lazarus-resurfaces-targets-global-banks-bitcoin-users/>

File Extraction

- Interesting download?
- If not encrypted, it's easy to dump it out in Wireshark!



File Extraction

If you come across the PCAP of an HTTP transaction where a file was downloaded, it's easy to extract the file with Wireshark for further analysis. To carve a file out to your hard drive, simply go to File, then Export Objects and select HTTP (this also works for TFTP, SMB, IMF, and DICOM files). In the HTTP object list, select the transaction for the download then hit Save. After the file is carved out, it's easy to run a virus check, signature verification, or hash calculation that can be run through VirusTotal.

Warning: If you're going to do this with executable files types, **extract them on a machine where they will not pose a threat** i.e. dump any potentially malicious Windows executables, such as shown on this slide, on a Linux OS where it cannot be accidentally double clicked and run. The same caution applies for scripting languages that Windows can execute by double-clicking as well - .js, .vbs, .ps1, etc. also need to be handled with care. You may think you'd never accidentally double click a file and hit enter while it's highlighted, but trust me, I've seen it happen more than once, and the pain of rebuilding a machine because you can't trust it anymore is much worse than using a separate safe machine for analysis.

High Frequency GET/POST Activity

Malware using HTTP for Cmd. & Control must check in

- Repeated GET/POST requests = potentially suspicious
- Especially if:
 - High volume
 - Periodic in nature
 - New/Untrusted domains – great SIEM filter if possible
 - Going to root of a website (GET or POST to /)
 - Contain encoded data in body

High Frequency GET/POST Activity

If a piece of malware is going to use HTTP as a mechanism for command and control and the adversary wants their remote shell to be responsive, there will need to be repeated GET/POST requests that send new data or check in to ask for new commands. Some malware POSTs data multiple times per second; others may only ping out once a day as a connection check. If you can devise a way to pick up repeated and periodic HTTP requests, especially with filtering for the most common domains (Alex Top x lists work great for this), you can write rules like "Alert when more than 100 POSTs to a non-Top 1000 website in under 1 hour". Malware C2 HTTP requests may be fast or slow, so that type of rule won't be perfect, but there are many more options that can be devised depending on the tools you have available. There are tools such as RITA from Black Hills Information security that is designed to find periodic signals, no matter the distance between them. So just because it's infrequent doesn't mean it can't be caught!

GET/POST to Naked IP Address

What's odd about this?

```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1;
Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C)
Host: 23.88.92.15
```



- POST directly to "/" – doesn't seem like scripting
- Sending data directly to an IP address
 - Could be vendor...needs investigation

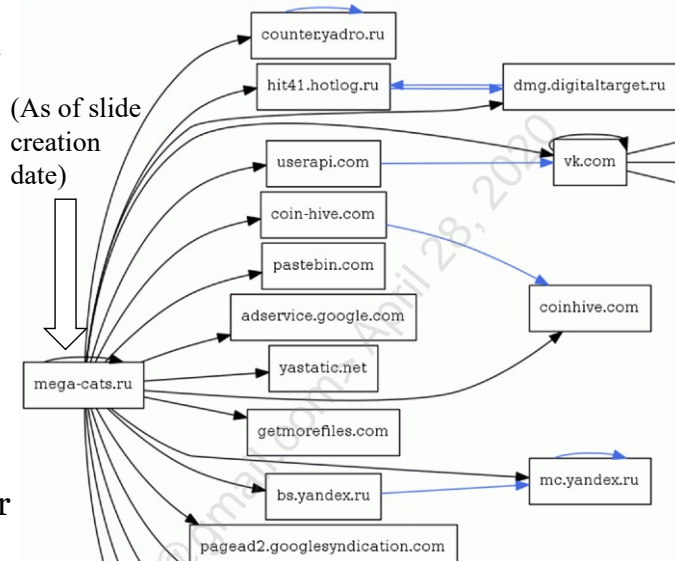
GET/POST to Naked IP Address

Since most internet communications involve the use of hostnames in order to make programs more robust and sites easier to remember for humans, any time a hostname is not used, suspicion should be raised. How do we know when a site was contacted without a corresponding hostname? The Host: field in the HTTP headers can give us that information.

In the screenshot above, malware is using a bare IP address to connect to for command and control traffic. If it were using a hostname, that name would've appeared in the Host header instead of an IP address. There's another red flag here as well—the POST method is being used to send data to the root of the webserver (highlighted at the top of the screenshot). Usually, POST'd data would be sent to a server-side script such as a PHP file, or at least a REST-style API—not the plain root folder of the server.

Redirection to Remote Evil

- First, check the initial domain accessed (and resolved IP)
- Use same hostname checks used for DNS
 - TLD/IP reputation, age, length, randomness, etc.
- If domain is not evil itself, sandboxing still may reveal redirection to evil sites
 - This analysis shows a coin-miner redirection



Redirection to Remote Evil

The "Host:" header is a great place to start for manual analysis since it identifies the site the user was visiting. As mentioned in the DNS section, there are multiple methods of identifying a suspicious domain from the name alone—was it recently created, a TLD that is statistically more likely to be bad, does the hostname seem random, or is it so long that no human would actually type it? These methods can point to if the site is inherently bad in a direct way, but what if the site is in the gray area, or perhaps has been hacked? In the case of the analysis of the domain, it's not always the site the user visited that's technically doing the evil; it may have had a malicious script or item inserted into it that loads malicious code from a different resource.

The above analysis of mega-cats.ru from URLQuery.net shows that when the user visits the site. It also loads code from a number of other domains across the internet. This alone is not odd in the least. Almost every site will be a complex web of remotely loading resources. The interesting item here, however, is that one of those resources comes from coinhive.com. If you aren't familiar with Coinhive, it's an in-browser crypto-coin mining script that many people consider to be malicious or at least annoying because it causes the CPU of the site visitor to jump to 100% while they browse the site, having performance impact computer-wide. Analyzing the domain name mega-cats.ru may not directly reveal that it loads a coin miner script, but having it run through the sandbox will be able to highlight this interaction.

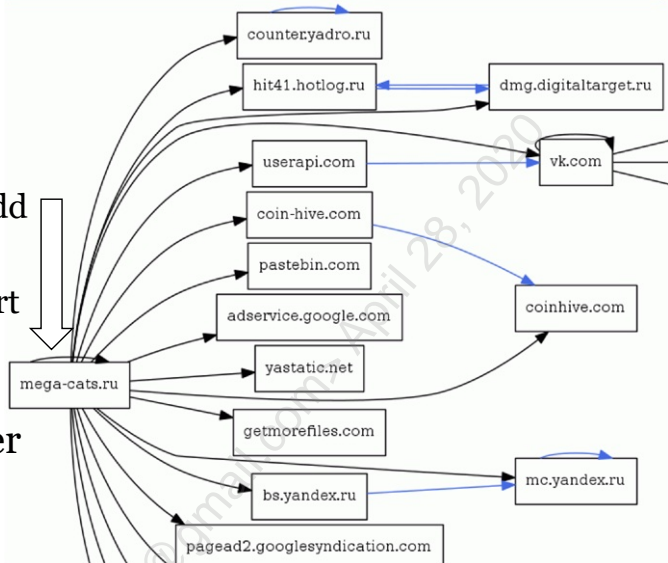
Redirection to Evil

“Good” sites very often will redirect users to malicious ones

Options:

1. **Malvertising** – get your evil add hosted on a good site
2. **Change the sites code** – insert a redirect or hidden iframe that points to malicious site

Evidence may be found in referrer header to lead back to infected site



Redirection to Evil

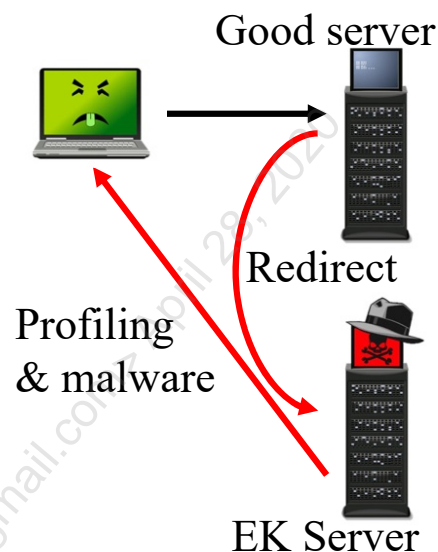
The "Host:" header is a great place to start for manual analysis since it identifies the site the user was visiting. As mentioned in the DNS section, there are multiple methods of identifying a suspicious domain from the name alone—was it recently created, a TLD that is statistically more likely to be bad, does the hostname seem random, or is it so long that no human would actually type it? These methods can point to whether or not the site is inherently bad in a direct way; however, what if the site itself is fine, but sent a user to a malicious place? In the case of the analysis of the domain, it's not always the site the user visited that's technically doing the evil; it may have had a malicious script or advertisement inserted into it that loads malicious code from a different resource or redirects the user to another site.

The above analysis of mega-cats.ru from URLQuery.net shows that when the user visits the site it also loads code from many other domains across the internet. This alone is not odd in the least. Almost every site will be a complex web of remotely loading resources. The interesting item here, however, is that one of those resources comes from coinhive.com. If you aren't familiar with Coinhive, it's an in-browser crypto-coin mining script that many people consider to be malicious or at least annoying because it causes the CPU of the site visitor to jump to 100% while they browse the site having performance impact computer-wide. Analyzing the reputation of the domain name mega-cats.ru may not directly reveal that it loads a coin miner script, but having it run through the sandbox will be able to highlight this interaction as it will follow all the redirects the same way an actual user's browser would.

Exploit Kits

Goal: Delivery tailored exploit to user

- Attacker compromises a good site
 - Or has malvertising placed on it
- User visits site and is redirected to exploit kit server
- Exploit kit enumerates browser
- Delivers custom exploit and payload
- User becomes compromised



Exploit Kits

One of the more ominous attack methods used by attackers is the "exploit kit." The goal of an exploit kit¹ is to compromise an unsuspecting user via their browser by weaponizing a page the user has no suspicion could be malicious. From the user's perspective, they have done "nothing wrong." They merely visited a "normal" webpage, one they may have been to many times before, but if an exploit kit has been injected on it, this time they may find their machine instantly compromised, a scary thought indeed! No social engineering or clicking of anything is required, simply an unpatched browser and being in the wrong place at the wrong time, the process is completely transparent to the user!

Exploit kits run as normal HTTP webservers and function by using user-agents, scripts, and other browser functions to enumerate a victim's browser and installed plugins. The script will then compare these items against a bank of available exploits and find what exploits the victim may be vulnerable to and prepare a custom package for their setup. To do this, exploit kit authors must keep and constantly update a collection of multiple exploits which can, one the fly, be paired with a malware payload (which can also change per delivery). In short, an exploit kit is an automatic exploit selection and arbitrary payload delivery system.

To start infecting users, attackers must get a redirection to their exploit kit server placed on a site the user will visit. They may do this by infecting the site directly and inserting their own code on the actual page or use an advertising service that will inject exploit kit redirects onto the page of multiple sites across the internet. Users are either guided to, or accidentally run into a page that has a link to an exploit kit, and that kicks off the process of infection. Keeping your OS, browser, and all plugins up to date are one of the best defenses against exploit kits, as is turning off running of third-party scripts on webpages with extensions like NoScript. Exploit kit authors have a finite number of exploits that they can choose from, so having a fully patched computer will render nearly all exploit kits harmless except for the rare occasion they are weaponized with a zero-day exploit.

Remember the last time you put off that Java or Adobe Flash upgrade? After this slide, hopefully you'll think twice before hitting the "remind me later" button!

[1] <https://www.zscaler.com/blogs/research/exploit-kits-go-cryptomining-summer-2018-edition>

Spotting Exploit Kits

What does EK traffic look like?

- Typically, lots of traffic to **newly created domain**
 - May use multiple IPs, random subdomains, URLs, or parameters
 - Highly obfuscated pages and scripts used for anti-reversing
- If successful, **followed by file download**¹

Host	Info
freedatingvideo.info	GET / HTTP/1.1
freedatingvideo.info	GET /popunder.php HTTP/1.1
freshechka.info	GET /ads/adv HTTP/1.1
176.57.214.100	POST /?NDE5NjI4&yxwEIBqcKgUDvIp&CNbXSyocGqe=cmVzb3J0&tas4=SwFhyIsLU18Q8638h0LdyBLNiZLUqx
176.57.214.100	GET /?Njg4MzY=&zsACTdJwMYDGe&PHACXRtdwd=cmVzb3J0&PpUfpM=c3BvcnQ=&tas4=IEawL1jkHScgFpnd9U
176.57.214.100	GET /?NjIwMTMx&FunosLLDmuf&wbKkAY=c3BvcnQ=&tas4=yI0LU18Q86z8h0XdyBPNiZPUqxaONQ9NqZedRbVt
217.23.4.201	POST /index.php HTTP/1.1
217.23.4.201	POST /index.php HTTP/1.1
85.143.171.2	GET /fazu.exe HTTP/1.1

Rig EK Traffic

Spotting Exploit Kits

The traffic pattern for a user hitting an exploit kit is not always exactly the same. Each exploit kit implements the profiling and redirection a bit different, but there are some general trends across them. One is that the kit often involves multiple HTTP requests across various domains and IP addresses. These domains are often suspicious looking in that they have non-sensical names, random in either the parent or subdomain section, and also tend to be freshly created. Other times, there are very long URLs with or without extensive encoded parameters on the end. If you were to look at the content of any of the pages loaded during the process, you would likely see highly obfuscated JavaScript code, nonsense text, and other fairly obviously questionable content. The slide shows the progression of a "RIG" exploit kit infection on Aug. 7, 2018, starting from the site freedatingvideo.info as recorded by Brad Duncan, author of the awesome malware-traffic-analysis.net blog and a SANS ISC handler. You can see how the site redirects the victim to several different IP addresses that have odd-looking long URLs and a final transaction of a random-looking executable download. Although this is not what all exploit kits will look like, you can start to get the idea of what to look for from this example. For more outstanding examples of various exploit kits and other malware traffic captures, check out Brad's blog and ISC diaries.

How do we detect these programmatically? Fortunately Snort and other IDSs do a pretty good job at keeping up with the URL patterns for infections and can often identify a user hitting a potential exploit kit. If there is any doubt, a quick glance at the traffic should be able to narrow down whether an alert for exploit kit activity is a false positive or not. When faced with an alert like this, analysts should verify whether it truly was an exploit kit attempt, and whether the delivery was successful or not by looking for any type of executable file or script being pushed near the end of the redirection chain. URL sandboxing sites may be able to help by visiting the site and hitting the exploit kit themselves, which will then be hopefully recognized and display in the graph of site redirections like the coinhive script on the previous slide.

The good news is, over the last few years, the prevalence of exploit kits has decreased immensely. There are multiple reasons for this. The combination of better browser and plugin development practices combined with the reduction of attack surface from less Flash and Java usage has led to a drop in their effectiveness. Since 2012,

multiple exploit kit authors have also been caught and arrested, which lead to their shutdown, further improving the situation. It's far from safe out there though, ZScaler's annual exploit kit report shows there are multiple kits still out delivering malware, "RIG" being one of the main ones that is still alive. Over the years, ZScaler² notes the payload of exploit kits has shifted from traditional malware and banking trojans to crypto-currency mining software, and not a very surprising turn of events.

[1] <http://malware-traffic-analysis.net/2018/08/07/index.html>

[2] <https://www.zscaler.com/blogs/research/exploit-kits-go-cryptomining-summer-2018-edition>

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

HTTPS

HTTP sites are quickly becoming HTTPS!

- Great for us personally, bad for blue team visibility
- Most orgs report at least **65%+ encrypted traffic**
- Many do *not* use SSL decryption
- Browsers now marking HTTP "not secure"
- Let's Encrypt makes SSL easy to set up
- SSL/TLS adoption is only going to go up
- What do we do?



Not secure | example.com

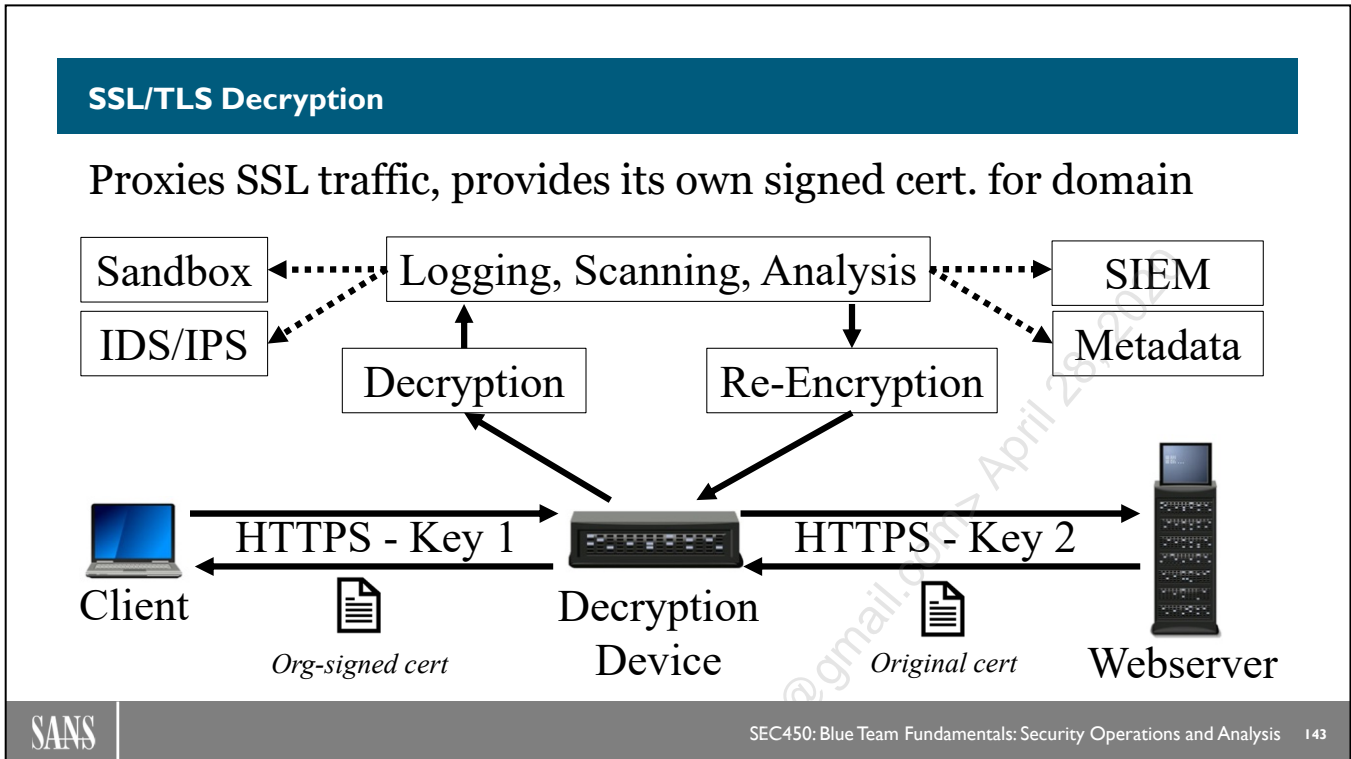
HTTPS

One of the main issues facing many security teams is the ever-increasing adopting encryption. Encryption is great for defending individuals from attackers snooping on their traffic and performing man-in-the-middle attacks to alter traffic, but it unfortunately blinds the security team as well if measures are not taken to deal with it. Most organizations are now seeing well over 50% of their traffic using SSL/TLS connections. This means performing network security monitoring requires either a shift in tactics, or the ability to decrypt the connections.

As free offerings like Let's Encrypt¹ make SSL certificates free and easy to get, and browsers start to mark non-HTTPS sites "not secure" (which several already do²), it's likely the situation will only get worse over time. All is not doom and gloom, however. Remember, even when network traffic becomes completely opaque, that doesn't stop us from gathering information directly from the endpoint. As encryption percentages increase, it is likely that this will be one of the primary answers to the loss of network visibility.

[1] <https://letsencrypt.org/>

[2] <https://www.blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/>



SSL/TLS Decryption

This slide shows a relatively simplified version of how SSL/TLS decryption works. First, recognize that for this entire scheme to function, your organization must have created their own root level certificate authority that can sign certificates for other websites and installed it in your browser. Without that, the browser would reject the certificates and none of this could function. Given that has occurred, what happens is this.

First, the client must connect to the decryption device and let it know what site it would like to connect to. The decryption device will then, on the fly, create and sign a certificate for that site issued by the organization level CA and return it to the client's browser, which will only accept it because the organization's root-level certificate is also present. The decryption device then, on the other end, connects out to the website in the same fashion the client would if the decryptor weren't in the way, establishing a secure connection using the certificate the website normally issues. The decryption device has effectively established itself as a man in the middle with one secure connection on the client side using the dynamically created organization CA-signed certificate, and another on the webserver side using the site's normal certificate.

From this point on, all information sent between the two is then proxied to each other, encrypted once on the left side and once on the right side. It's what happens in between the two different encryption points that allows the security team to break the tunnel and peer into what's inside. During that time of decryption, the SOC can use all its normal analysis tools for sandboxing files, running IDS/IPS signatures, metadata collection, and sending logs to the SIEM. This lets the user effectively keep their secure connection, but still allows the blue team to monitor what's going on!

No TLS Decryption?

Don't give up! You can *still* see some useful info

- IP Layer: Destination IP, ASNs, geo-IP location
- TCP Layer: Suspicious ports for SSL
- Session Layer: Cert. details (subject, issuer, org. details)
- Application Layer: Totally blind 😞

Client

IP: 192.168.0.10

TCP: Source Port xxxxx



→ *GET /virus.exe HTTP*



SSL/TLS Tunnel

Server

IP: 2.2.2.2

TCP: Port 443



SANS

SEC450: Blue Team Fundamentals: Security Operations and Analysis 144

No TLS Decryption?

Don't despair if you don't have SSL decryption. There are still ways to get an idea of whether traffic is malicious or not. Even without decryption tools, you can still see:

- The destination IP the user is connecting to. If you have a solid blacklist of known malicious IP addresses, this may be able to help you. An IP address also can be resolved to an ASN and geographical information as well, and don't forget about passive DNS sources!
- The TCP layer will show what destination port the user is using to create the SSL tunnel. Most of the time, this will not be that interesting as the traffic will be on port 443. But if you were to see SSL/TLS connections on port 4444 (the default Metasploit port), you may still have an opportunity for the occasional detection.
- The Session Layer is where most of the interesting information will lie. Although, you may already be able to tell what site a user is connecting to if you have their DNS traffic from prior to the connection (since this maps a site to the IP they have connected to). The certificate for the SSL/TLS connection is also passed to the user in plaintext, which means that it can be used as a secondary source of seeing what site the user went to. There are other fields beyond that too, such as who issued the certificate and optional organizational details.
- Connection data such as the volume of traffic sent and received is still present, meaning though we may not know what was sent, a large upload could still trigger an anomaly alert based on volume.

Unfortunately, beyond this, you will be blind as everything at the application layer will be encrypted bytes. Let's take a closer look at a cert. to see how we can pull the information that is available.

Interpreting an SSL/TLS Certificate

The screenshot displays a Wireshark capture of an SSL/TLS handshake. The main pane shows the 'Secure Sockets Layer' section expanded to 'TLSv1.2 Record Layer: Handshake Protocol: Certificate'. The details pane on the right shows the certificate for 'www.google.com', issued by 'Google Internet Authority G3', with validity dates from November 7, 2018, to January 30, 2019. The subject organization is 'Google LLC' and the subject locality is 'Mountain View, California'.

Interpreting an SSL/TLS Certificate

Let's look at the Wireshark capture of a system connecting to google.com using SSL/TLS. If you dive down into the SSL/TLS layer and unfold the correct items, you'll eventually be able to interpret the various fields from the Certificate that was passed back from the server when the user connected. Notice that this means all of this information is transmitted in plaintext and can be read even if you aren't doing SSL decryption.

If the user were to bring up the certificate details pane in the browser, they would see the screen on the right side of the slide, which will match the information shown in Wireshark that we picked up over the wire. The details show the subject of the certificate that was passed to the client (and therefore the site the user visited) as www.google.com. From there, we see the chain of signatures up to Google Internet Authority G3's intermediate CA, then to the GlobalSign CA that is ultimately installed in the browser as a trusted root level CA. This information is available in Wireshark as well below the section shown on the screenshot. Additionally, we can also see the details on www.google.com certificate such as the Organization, Locality and Country supplied by Google inside the www.google.com certificate. This is significant because it gives us a little bit more detail about who made the certificate. Note that these are optional and therefore are not always filled out. If you spot a certificate without them present, it may be an indicator that the connection is malware related, as attackers are disinclined to fill out any more information than is absolutely necessary to make the certificate work. They may even create self-signed certificates that are not signed by any authority.

For our purposes, these fields are the ones we are most interested in when looking at raw SSL/TLS traffic to try to determine if it is malicious or not:

- **Subject / CommonName:** The site the certificate was issued for (which also means this is the site the user is accessing). This is a required field.
- **Issuer:** The certificate authority that signed the certificate that was presented, a required field. If it's a self-signed certificate, it will match the subject name.
- **Country / State / Locality / Organization Name:** Optional fields that may or may not be filled out. Malware tends to not fill them out.
- **TCP Port / IP address:** Notice that Layer 3 and Layer 4 information is still available here, which may give us a small chance at detection for blacklisted IP addresses and questionable ports.

TLS 1.3

TLS1.3 brings major changes for encryption:

- Enforces use of **Perfect Forward Secrecy**
 - Cannot merely use copy of the private a key for decryption
- Decryption **must be present the whole session** to work properly, no picking up or dropping out
- Server **certificates will be encrypted**
 - Will no longer be able to tell domain visited and cert. details
- **Protection from downgrade attacks**

TLS 1.3

The next big standard in SSL/TLS is TLS1.3.¹ The new standard will be a large shift in strategy for network security monitoring as it will break many of the tools and methods we currently use for encrypted traffic monitoring. For one, all cipher suites that do not provide "perfect forward secrecy" have been eliminated. That means that we cannot simply decrypt SSL traffic by giving an appliance a copy of the private key used for certificate signing. Each session will get its own ephemeral key.

Another issue is that for a decryption middlebox to function, it must be present for the *entire* encrypted conversation. In TLS1.2 and before, the decryption appliance could step in or out of the conversation as needed and could tell when it needs to do so partly because of certificates that were passed in the clear. In TLS1.3, the certificate exchange is encrypted, and the way the new protocol works means any type of decryption appliance will have to stick with a conversation for the entire duration for it to function correctly, even if it's something a normal TLS1.2 proxy would've ducked out from (banks, medical sites, etc.). Proxies for TLS1.3 will need to be in use for the entire conversation, whether we want them to or not. If they aren't, they will have no idea whether a host is talking to a malicious site or a good one (except for the destination IP address). TLS1.3 also adds provisions to prevent downgrade attacks, so we cannot simply just force a TLS1.2 connect instead to use the old methods.

TLS1.3 adds an outstanding set of security controls that will no doubt make the user experience better for the user, but the ramifications for the blue team are significant. Expect to see major changes in the way organizations attempt to find malicious encrypted traffic as a result.

[1] <https://tools.ietf.org/id/draft-camwinget-tls-use-cases-00.html#rfc.section.2.1.1>

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. **Exercise 2.2: HTTP and HTTPS Analysis**
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

Exercise 2.2: HTTP and HTTPS Analysis



Exercise 2.2: HTTP and HTTPS Analysis

Exercise 2.2: HTTP and HTTPS Analysis

Please go to Exercise 2.2 in the SEC450 Workbook or virtual wiki.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. **Understanding SMTP and Email**
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

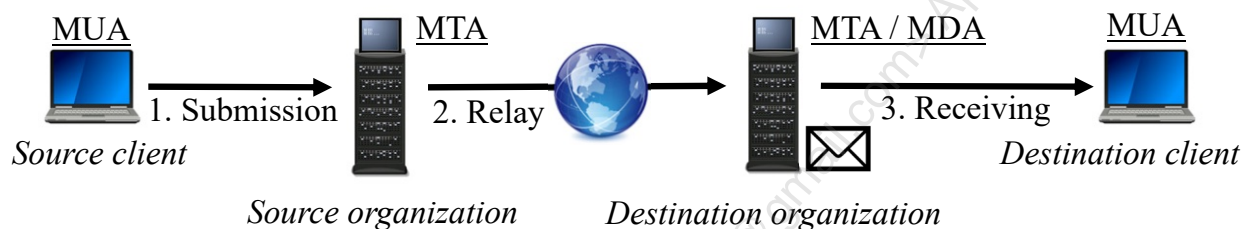
Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

Email Delivery Infrastructure

Email sending and delivery can be thought of in 3 separate pieces

1. **Email submission** from your computer to the SMTP server
2. **Relay** between the source email server and destination
3. **Receiving** the mail from the destination server

Each of these steps can **use different protocols**



Email Delivery Infrastructure

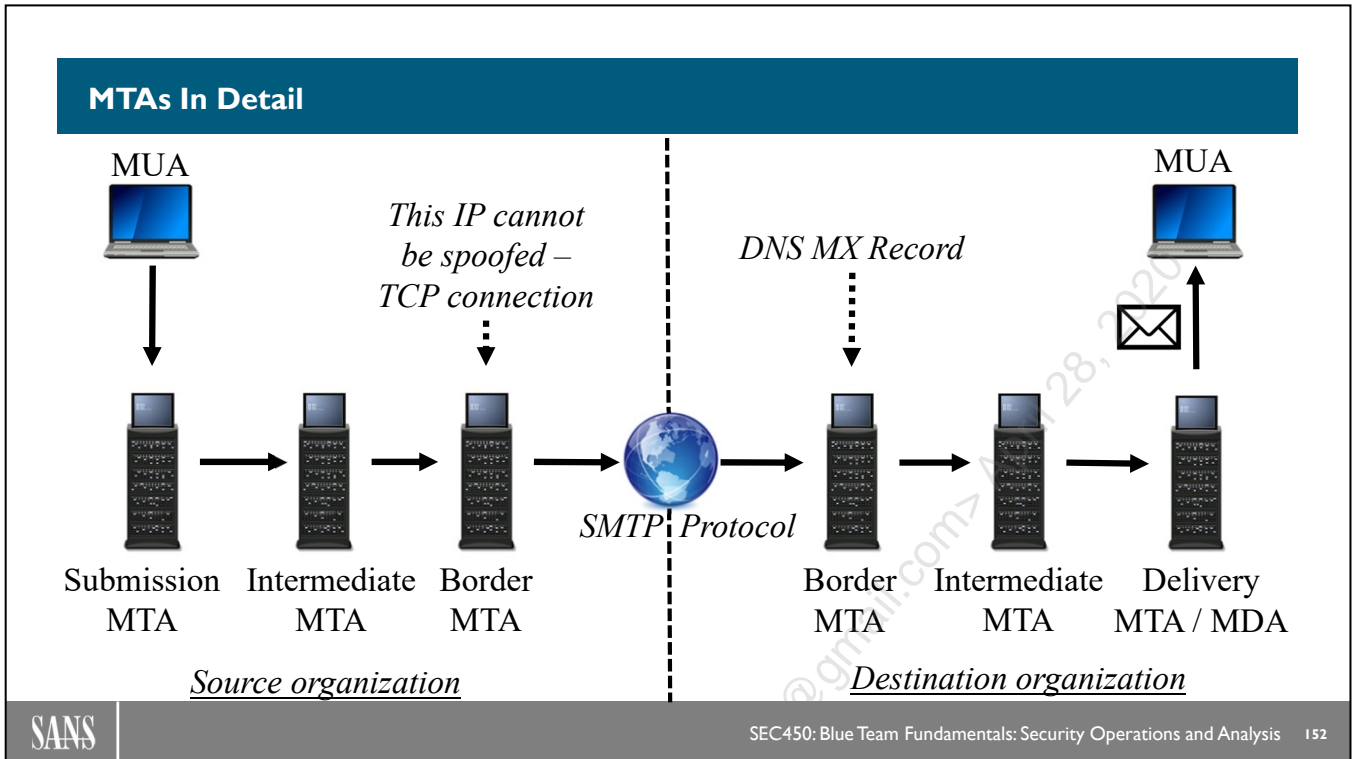
When it comes to sending email, the protocols used and connections made can be broken into three parts:

1. The user's email client, generically referred to as a "Message User Agent" or MUA, is where a user interacts with their mailbox, writing and sending a new email and reading what they have received from others.¹ MUAs are the clients we are all used to such as Gmail's web interface or thick clients like Microsoft Outlook and Thunderbird. They have the job of taking the email from the user and submitting it to the specified SMTP server for the user's email account, as well as displaying email that has been received for the user. MUAs often use SMTP protocol to submit emails to the source server, but not always.
2. The source SMTP server receives the email from the MUA and begins to relay it. These servers are referred to as Message Transfer Agents (MTAs) in the SMTP standard. The source organization may have a series of MTAs the email goes through on its side, with the last one being responsible for transferring the email over the internet to the server listed in the MX record of the receiving organization. These 2 border MTAs are special in that they are the last server the email touches as it leaves the organization, and the first one it encounters on the receiving side (this will come up again in a moment when we discuss SPF records). Once received by the destination MTA, a message may be relayed through additional internal servers before getting to the final step where it can be delivered to a user's inbox.
3. Once the destination MTA receives the email, it must store it for the user who it is intended for. When the destination server performs the email sorting and storage, it is acting as what the standard refers to as a Message Delivery Agent (MDA). Although the final MTA and MDA can be thought of as one step for our purposes, technically each of these jobs can be separated across different servers, which is why each function has a different name. At this stage, the receiving user's MUA reaches out to their MDA and checks for any new mail that has been received, completing the mail delivery.

Why are we discussing the terms MUA, MTA, and MDA? Because the connections between each step are done using different protocols and over different ports, each of which is important to understand. Although the standards are written using the MUA, MTA, and MDA terms, technology like webmail obscures what part each device is playing in the delivery of mail, so it is useful to break them out into their logical components as shown here for discussion in this module. For monitoring purposes, we want to be clear on what type of traffic occurs at each stage, and how the relay of email between these components modifies emails headers, leaving a trail we can analyze.

[1] <https://help.returnpath.com/hc/en-us/articles/220569547-What-is-a-Mail-User-Agent-MUA->

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



MTAs In Detail

To get a more detailed sense of what is truly happening, this slide shows how a typical email might progress through a series of MTAs. In a literal sense, an email may go to an initial MTA designated for mail submission (sometimes called an MSA). From there, it could progress through a series of intermediate MTAs that pass the mail through the source organization until it ultimately hits a "border" MTA, which is the last source organization owned device that touches the email before it is sent to the destination organization. The source's border MTA will be passing the message off to the destination's border MTA, which also happens to be the system defined as the MX record in the DNS logs. To verify the source of the message is legitimate and the email is not spoofed, the destination border MTA may look up the SPF record (covered in a moment) of the source's border MTA. From there, the message progresses through another potential set of internal MTAs at the destination organization until it reaches the message delivery agent, which will pass the message on to the recipient. After the MUA submits the email to the first MTA, all the subsequent interactions, up until the receiving organization's MDA receives the email, will utilize the SMTP protocol. The original user submission of the email and the receiving and reading of the email by the recipient may happen over Exchange ActiveSync, a webmail interface, like Gmail, or IMAP. It is the SMTP portion of the interaction that we are interested in as these are the interactions that leave a record in the message trace headers of the email.

SMTP Protocol

- SMTP is a simple ASCII-based text protocol, usable via telnet
- Uses keywords for addressing
 - EHLO [sending server name]
 - STARTTLS
 - AUTH LOGIN (base64 username/password)
 - MAIL FROM:<sender email>
 - RCPT TO:<recipient email>
 - DATA
 - To, From, Subject
 - [email body], "." to end

```

220 mailhog.example ESMTP MailHog
EHLO sec450.com
250-Hello sec450.com
250-PIPELINING
250 AUTH PLAIN
AUTH LOGIN
334 VXNlcm5hbWU6
am9obgo=
334 UGFzc3dvcmQ6
bXlwYXNzd29yZAo=
235 Authentication successful
MAIL FROM:<user@sec450.com>
250 Sender user@sec450.com ok
RCPT TO:<student@gmail.com>
250 Recipient student@gmail.com ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: user@sec450.com
From: "User 123" <user@sec450.com>
Subject: Look how easy SMTP is!

Sending you an email via telnet!
.
250 Ok: queued as fFzcVRL3CRtaZHdnLZxqnLeVdmRZw
QUIT
221 Bye

```

SMTP Protocol

SMTP is a relatively simple protocol. In fact, it uses plain ASCII-based text commands to define almost everything inside a typical email. The mail client sends keyword and value pairs to the MTA, and the server replies with a 3-digit status code, similar to HTTP. This slide shows what the conversation between an MUA and MTA looks like when sending a basic message from "user@sec450.com" to "someone@outlook.com." Most of the information can be easily interpreted right from the entered text. The first EHLO command is the source server identifying itself as being mail.sec450.com. The user then sends their name and password in base64 encoded format following the AUTH LOGIN command (the server 334 responses say Username and Password in base64 respectively). This is followed by the MAIL section holding the from address, the RCPT (recipient) section containing the To address, and the DATA section where the subject and body of the email is written. According to the standard, capitalization should be ignored for all headers except for the cause of the username of the email receiver.¹ The communication is signified as finished by typing a single "." on a line, which ends the message.

The information entered on these lines will be interpreted by the MTA and determines where the email will go and where it will be labeled as coming from. Some of these fields, such as the from address and subject, will be displayed to the user once the mail is received, and others, such as the EHLO line, will only make an appearance when viewing the email headers. The important thing to realize is that for every MTA, the email must traverse to get to its destination. The same type conversation must be repeated as the message is sent from server to server. It is these repeated conversations that contribute to all the information that ends up in the email header, which can be viewed by the recipient of the email upon arrival.

[1] <https://tools.ietf.org/html/rfc5322>

Email Headers

Trace headers

- Written by each MTA that touches the email
- "Received" headers trace email back through MTAs
- Additional fields add detail for receiving MDA/client to interpret

Message headers

- Written by client sending email or submission MTA
- Contain message metadata

Body

- Shows "plaintext" and HTML view of the email content
- Attachments are usually base64 encoded

Email Headers

If you want to see an email in its native text form, including all the metadata added by the MTAs an email has passed through on its way to your inbox, pull up the view that shows the original email source. In this view, you should see both trace and message headers, as well as the body of the email. The trace and message headers will be at the top of the email source text and contain a wealth of information. Items in this section include the multiple MTAs the message passed through, details that can be used to tell if a message is likely spam or a spoofed message, which mail client was used to send the message, and, in some cases, even the IP address and hostname of the sender. RFCs do not specify how to act on this information, just how it should be formatted so that the receiving organization or client MUA can decide what to do with a message based on the trace header's contents. The body of the email contains all the text, pictures and attachments that are included, and is located underneath the header sections. Let's look at some email headers in detail and see how we can gather useful information from them.

Components of a Received SMTP Message

<pre>Return-Path: <user1@gmail.com> Received: from Lenovo (static-40-30-20-10.phlpa.ftas.verizon.net. [10.20.30.40]) by smtp.gmail.com with ESMTPSA id mlsm3166426qkh.15.2018.11.27.04.59.28 for <user2@gmail.com> (version=TLS1_2 cipher=ECDHE-RSA-AES128-GCM-SHA256); Tue, 27 Nov 2018 04:59:28 -0800 (PST)</pre>	Trace Headers
<pre>From: <user1@gmail.com> To: <user2@gmail.com> Subject: Subject goes here Date: Tue, 27 Nov 2018 07:59:26 -0500 Message-ID: <013501d48651\$07300ed0\$15902c70\$@gmail.com> MIME-Version: 1.0 Content-Type: text/plain; charset="us-ascii" Content-Transfer-Encoding: 7bit X-Mailer: Microsoft Outlook 16.0 Thread-Index: AdSGUQA80mmtR8RQRtO0NYjDMwxUgw== Content-Language: en-us</pre>	Message Headers
<pre>Email body goes here</pre>	Email Body

Components of a Received SMTP Message

A look at the original source text of an email on the receiving side shows there are 3 main sections: The trace headers, the message headers, and the email body. The trace headers are added not by the sending client, but by every server that touches the email in its path from sender to receiver, and they indicate information about the hosts it has passed through. The message headers and body are created by the client MUA or the submission MTA and are created when entering the SMTP data as shown on the previous slide. These are the MAIL FROM, and RCPT TO lines, as well as the DATA section that holds content of the email itself. The order of the fields in the message headers is not specified and will change depending on the email client. Some of the fields are optional and, therefore, may or may not be present for any given message. The trace headers should always be in reverse chronological order (more on that in a second). The body will always be present at the bottom.

Notice that this is a very simple email body content. Email can be sent in either plaintext format or in HTML format, and with or without attachments. The body section of a message must support all of these options.

HTML Email Body and File Attachments

Plaintext email is easy to read

Email with HTML or attachments becomes more difficult...

- Broken up into sections with MIME types
- Nested sections, broken up with boundaries
- Attachments *usually* are base64 encoded

```
Subject: file attachment example
From: <user1@gmail.com>
To: <user2@gmail.com>
Content-Type: multipart/mixed; boundary="0000000000039c0d5057b9d5bb2"
--0000000000039c0d5057b9d5bb2
Content-Type: multipart/alternative; boundary="0000000000039c0d1057b9d5bb0"
--0000000000039c0d1057b9d5bb0
Content-Type: text/plain; charset="UTF-8"

Here is your file!

--0000000000039c0d1057b9d5bb0
Content-Type: text/html; charset="UTF-8"

<div dir="ltr">Here is your file!</div>

--0000000000039c0d1057b9d5bb0--
--0000000000039c0d5057b9d5bb2
Content-Type: text/plain; charset="US-ASCII"; name="attachment.txt"
Content-Disposition: attachment; filename="attachment.txt"
Content-Transfer-Encoding: base64
VGhpcyBpcyBhIHNBbXBsZSBmaWxlIGF0dGFjaG11bnQu
--0000000000039c0d5057b9d5bb2--
```

Base64 attachment

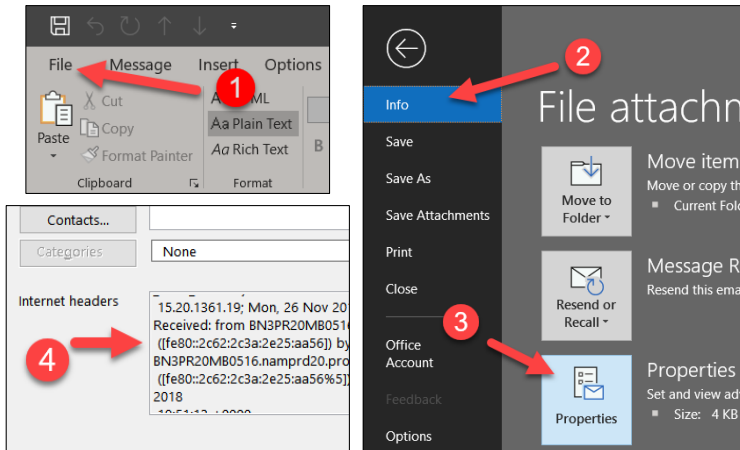
HTML Email Body and File Attachments

When it comes to reading the original source body of an email, plaintext email is easy—there's nothing to see except for the text the author of the email wrote. How does the body of the email represent the content when we want to send email that is formatted with HTML and has a file attached? One thing that must happen is the word content and the file attachments need to be separated, this is done through creating a boundary inside the body of the email and naming each section with the correct Multipurpose Internet Mail Extensions (MIME) type (the same MIME type we saw used in HTTP transactions, but this is what it was actually designed for). In the email on the slide, we can see the top line says Content-Type: multipart/mixed"; boundary="0000000000039c0d5057b9d5bb2". This is telling the receivers MUA that there are multiple pieces to this email that need to be considered separately, and that those sections will be delimited with the string in the boundary section.

In this email, the top is all one section highlighted with this boundary, and the bottom section is another. There's another multipart section, though, nested within the top piece. This one says Content-Type: multipart/alternative; boundary="0000000000039c0d1057b9d5bb0" (note the 0 at the end of the number instead of a 2). Within this subsection, there is a copy of the email in plaintext that is labeled as Content-Type: text/plain, and another section with HTML labeled Content-Type: text/html. The sending MUA has broken the HTML based email up into two different subsections—one with HTML formatted text, and one without HTML.

How To View Email Headers and Source

Outlook:



Gmail:

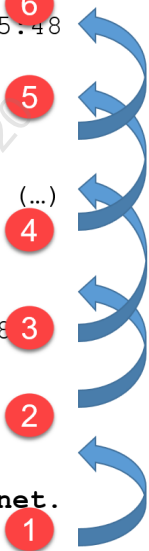


How To View Email Headers and Source

What about the headers? Viewing the headers for a message in Gmail or Outlook is quite easy:

- In Outlook, the message must be opened in its own dedicated window, not viewed through the preview pane. Once open, click on the bottom right hand corner of the Tags part of the ribbon in the Message tab. Doing so will pop up a new window that displays the headers in the "Internet headers" section as shown on the screenshot in the slide. Unfortunately, the window is too small to be of use reading such a large body of text. The best method is to click into the box, type "Ctrl + A" to select all the content and copy and paste it into another text editor for viewing in a large window. To view the HTML source of the email (but not raw message source with MIME types and all, like in Gmail's "Show original"), right click the body of any HTML email and hit "View Source".
- To view the original source of an email in Gmail headers, body, and all, click the 3 dots in the upper right corner of the message and select the "Show Original" option. This will open a new window with the full plaintext view of the message.

Tracing an Email

- **Received:** from **BN3PR20MB0516.namprd20.prod.outlook.com** (...) by **BN3PR20MB0516.namprd20.prod.outlook.com**; Mon, 26 Nov 2018 19:55:48 +0000 6
 - **Received:** from **CY4PR20CA0003.namprd20.prod.outlook.com** (...) by **BN3PR20MB0516.namprd20.prod.outlook.com** (...); Mon, 26 Nov 2018 19:55:44 +0000 5
 - **Received:** from **BY2NAM05FT031.nam05.prod.protection.outlook.com** (...) by **CY4PR20CA0003.outlook.office365.com** (...); Mon, 26 Nov 2018 19:55:43 +0000 4
 - **Received:** from **mail-qt1-f173.google.com** (...) by **BY2NAM05FT031.mail.protection.outlook.com** (...); Mon, 26 Nov 2018 19:55:43 +0000 3
 - **Received:** by **mail-qt1-f173.google.com** with SMTP id e5so19090070qtr.12 for <someone@outlook.com>; Mon, 26 Nov 2018 11:55:43 -0800 (PST) 2
 - **Received:** from **Lenovo** (**static-40-30-20-10.phlapa.ftas.verizon.net** [...]) by **smtp.gmail.com** Mon, 26 Nov 2018 11:55:41 -0800 (PST) 1
- 

Tracing an Email

How can we tell what servers (MTAs) an email has bounced through as it made its way from sender to receiver, or even who sent it? The "Received:" headers give us this information! RFC5322, one of the many regarding SMTP, states the format of a received header should be "Received:" *received-token ";" date-time CRLF. It is the token that we are most interested in for tracing headers. The thing to know about trace headers is that they are written in reverse chronological order—each server adds its own information on the top of what is already present when that server receives the email. This leads to a trace header stack like is shown on this slide where the first MTA is on the bottom, and the final server is on the top. This slide shows a heavily edited version of an email header. All lines except the Received headers were removed since they are not relevant for this slide. After the mail client passes the email on to the submission MTA, the first received header is added on the bottom line.

Received headers are typically laid out in the form of "Received: *from* (source MTA and IP) *by* (receiving MTA and IP writing the header) ; [date]". This means if we read the Received headers from bottom to top, if the email was not spoofed, we should see a continuous chain of matching received "from" and "by" hostnames and IP addresses. This is true of this email except for the received header at #2 which did not include "from" name of the host labeled in #1. This sometimes occurs with intermediate MTAs within an organization and is not cause for concern. The first received header shows the hostname of the device was "Lenovo" and its IP address at the time of sending was 10.20.30.40 (removed for space step #1) which had a DNS PTR record of static-40-30-20-10.phlapa.ftas.verizon.net. In this case, this is indeed the true information because the email was sent from an email client installed on a laptop. Had it been sent from a webmail-based system, the email provider would likely not include this information, so the source IP would be that of a Gmail server somewhere and there would be no meaningful DNS record, IP, or hostname included.

The second stop this email makes is to the submission MTA at smtp.gmail.com. Since the email was sent from a Gmail address using SMTP, this makes sense. To get the mail to its first MTA, the email client is set up to

use smtp.gmail.com as the submission MTA for sending outbound email. From there, we can see in steps 2-6 the email heads to another Google-owned server, and eventually into Microsoft infrastructure at outlook.com in #3. This is the step where the email left Google's border MTA and entered Microsoft's infrastructure. Afterwards, we see several intermediate Microsoft MTAs pass the message on until it finally reaches its last step.

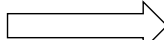
Here's another key point: Assuming this was sent through legitimate email infrastructure without spoofed headers (and it was since it went through Gmail to Outlook), we can also trust the source IP address of the sender is also correct. Why? Because SMTP is a TCP protocol which requires two-way interaction and cannot be spoofed like UDP. Therefore, if we trust that the Gmail submission MTA will create honest connection headers, which we do, there's no way for the client to spoof the IP address they connected from (in the case of this example, the header text was changed to say IP address 10.20.30.40 for this slide). If we did *not* trust the source of the email to not be spoofed, we would then only have confidence in the IP address reported in the "from" section on the line created by the receiving border MTA (step 3). We know our own border MTA must have connected to that IP address as the sender of the email, but everything before that is suspect.

Licensed To: David Owerbach <0mamaloney0@gmail.com>

Email Spoofing

Problem: The SMTP RFC was written in 1982

- **SMTP not designed to question message authenticity**

- So you can do this... 

```
220 17e4569324d8 ESMTX SubEthaSMTP null
HELO mail.microsoft.com
250 17e4569324d8
MAIL FROM:<bill.gates@microsoft.com>
250 Ok
```

- **Email spoofing == easy!**

How do we stop it?

- **Border MTAs could verify email source, but how?**

- Verify source was pre-authorized to send email on domain's behalf
- Have the email gateway sign the email with private key and verify
- Ensure displayed name is not trying to trick user

Email Spoofing

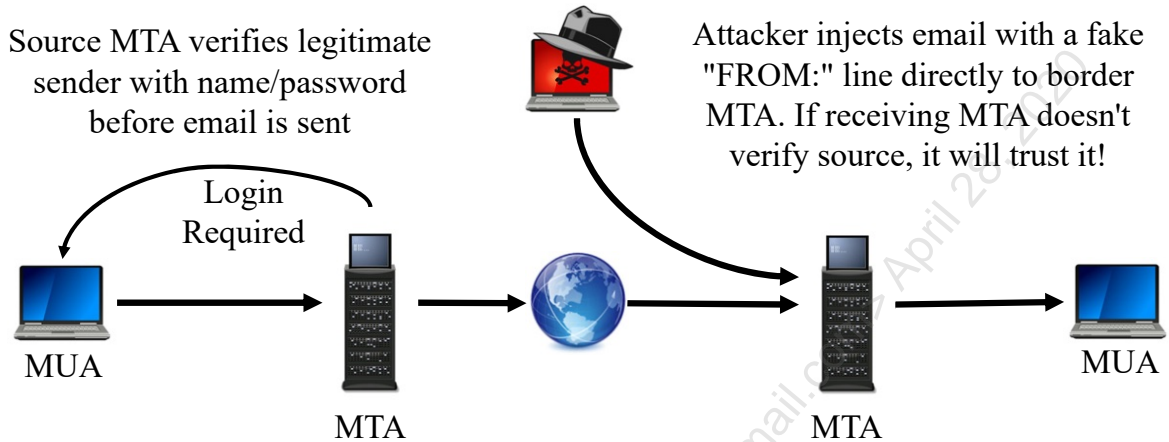
The first SMTP RFC was written in 1982.¹ Email spoofing and cyber attacks weren't a worry yet; therefore, it is not a surprise to know that the original spec did not take security into high consideration. One of the issues with the design of SMTP is that if an attacker uses a poorly configured or malicious SMTP server, they could potentially enter any email address in the "from" as shown above, and the server would accept and pass on the message—no questions asked! As a result of oversights like this, SMTP standards have evolved over the years to bolt on additional security measures to ensure spam and email spoofing are kept under control.

One of the methods invented to reduce spam is the Sender Policy Framework. The Sender Policy Framework has domain owners list in their DNS TXT records a whitelist of hostnames and IP addresses that are authorized to be an email source for that domain.² That way, if an email like the one above was sent across the internet by an attacker, an MTA could look at the FROM line, do a DNS TXT record request, and pull the list of IP addresses allowed to send mail from Microsoft.com. If the email was not sourced from a pre-authorized IP listed by Microsoft, the message headers will indicate an SPF failure, meaning it is likely spoofed. The receiving organization's mail server can then decide to keep the message, deliver it to a spam folder, or just throw the message away. If you've ever had another organization claim to receive email from your company that wasn't actually from it, it is likely you need to define SPF records in your DNS entries to prevent this in the future. Without it, they might not know how to trust that a message truly came from you!

[1] <https://tools.ietf.org/html/rfc821>

[2] <https://tools.ietf.org/html/rfc7208>

Spooled Email Injection



Spooled Email Injection

Here's a depiction of the email authentication problem to make it clearer. Yes, the sending organization's border MTA will verify that an email address is indeed a legitimate sender. It does this by forcing users to authenticate during the SMTP connection using the ESMTP extensions and AUTH LOGIN command, but attackers can bypass that process completely. A malicious email sender talks directly to the receiving organization's border MTA to pass an email that, in a normal scenario, would have already been sent from an authenticated user. Unfortunately, if that receiving MTA has no way of knowing where that email came from, it will believe the assertion in the headers of where and who it came from. Believe it or not, that is how SMTP worked for years! Now that cybersecurity has become a major concern for every organization, multiple standards have risen to help solve this problem.

Verifying the Source of a Message

Three ways to verify an email was not spoofed (RFC 7001 "Authentication" trace header content):

- 1. SPF (Sender Policy Framework)**
 - Mail source from verified source
- 2. DKIM (Domain Keys Identified Mail)**
 - Message content verified via digital signature
- 3. DMARC (Domain-based Message Authentication, Reporting, and Compliance)**
 - Prevents attacks based on different From address and displayed address

Verifying the Source of a Message

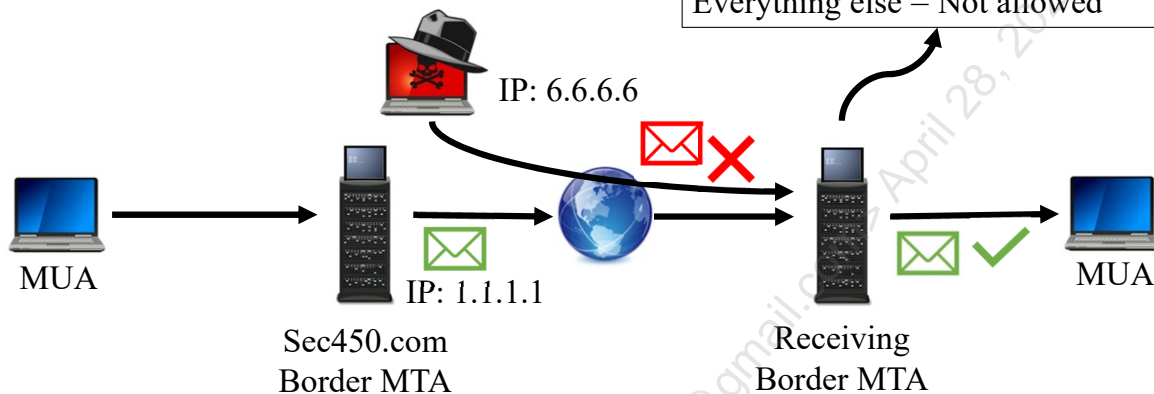
To combat the lack of authentication in email, several standards have emerged over the years. The three standards with the most adoption are SPF, DKIM, and DMARC. SPF is a path-based technology that verifies mail source based on IP addresses and hostnames. DKIM is a signature-based technology that aims to prove where a message came from by having the sender's gateway MTA sign message components and publish a public key in DNS that can be used to verify the contents. DMARC builds on the previous two by taking the verified information and enforcing the claimed identity in a message's from line so that the multiple fields possibly used for a source address in an email cannot be used to fool the user.

SPF in a Nutshell

Receiving border MTA compares IP address of connection to DNS published list of approved senders for that domain.

DNS published list:

Approved senders for sec450.com:
1.1.1.1 = Allowed to send
Everything else = Not allowed



SPF in a Nutshell

Here's a pictorial view of how SPF works. Every incoming message to the receiving border MTA has a claimed domain it came from (from the EHLO message and FROM address), and the actual IP address that made the connection to the MTA. The receiving MTA then takes that domain name and does a DNS lookup for a special record that contains a list of sources that can send email for that domain. In this case, if the email was from user@sec450.com and came from IP address 1.1.1.1, the receiving MTA would find in the DNS record that 1.1.1.1 is an IP allowed to send email, and mark the message as coming from a validated source. If an attacker tried to falsely label an email as from sec450.com and send it from their own IP address, that IP would not be on the DNS published whitelist, and since they can't spoof what IP they are communicating from (because that's impossible with TCP), there is no way to get the email marked as from a validated source.

SPF Results

SPF check result options:

- **Pass:** The client is listed as authorized for sending email
- **None/Neutral:** Either there is no policy listed, or the policy does not address the source specifically
- **Soft fail:** Between neutral and fail, client is allowed, but should be treated as suspicious (when "~all" is used)
- **Fail:** The client is listed as *unauthorized* for sending email, usually falls under "-all" rule

SPF Results

When the border MTA receives a message and does the DNS SPF record lookup, it must tag the message with the conclusion. According to RFC 7208, SPF checks must return in one of the above results and should be interpreted as follows¹:

- **Pass:** A "pass" result means the client is authorized to inject mail with the given identity. The domain can now, in the sense of reputation, be considered responsible for sending the message. Further policy checks can now proceed with confidence in the legitimate use of the identity.
- **None:** With a "none" result, the SPF verifier has no information at all about the authorization or lack thereof of the client to use the checked identity or identities. The [SPF check] function completed without errors but was not able to reach any conclusion.
- **Neutral:** A "neutral" result indicates that although a policy for the identity was discovered, there is no definite assertion (positive or negative) about the client. A "neutral" result **MUST** be treated exactly like the "none" result; the distinction exists only for informational purposes.
- **Softfail:** A "softfail" result ought to be treated as somewhere between "fail" and "neutral"/"none". The [SPF record indicates] the host is not authorized but is not willing to make a strong policy statement. Receiving software **SHOULD NOT** reject the message based solely on this result, but **MAY** subject the message to closer scrutiny than normal.
- **Fail:** A "fail" result is an explicit statement that the client is not authorized to use the domain in the given identity. Disposition of SPF fail messages is a matter of local policy.

[1] <https://tools.ietf.org/html/rfc7208#section-5.1>

Checking SPF Results in Trace Headers

SPF Results are shown in two locations:

1. Authentication-Results header:

```
Authentication-Results: spf=pass (sender IP is 1.2.3.4)
smtp.mailfrom=sec450.com; outlook.com; dkim=none (message
not signed) header.d=none;outlook.com; dmarc=bestguesspass
action=none header.from=sec450.com;compauth=pass reason=109
```

2. Received-SPF header:

```
Received-SPF: Pass (protection.outlook.com: domain of
sec450.com designates 1.2.3.4 as permitted sender)
receiver=protection.outlook.com; client-ip=1.2.3.4;
helo=sec450.com;
```

Checking SPF Results in Trace Headers

You can check the status of the SPF validation in the message trace headers in 2 locations. One is a header called "Authentication-Results." The contents of this header are governed by RFC 7001¹ which specifies not only SPF, but DKIM and DMARC checks as well (additional checks covered shortly). This header is not meant to give the full detail on these checks. It only requires that the answers be specified.

If you want all the info from the SPF check, the Received-SPF record is the place to find it. This header contains all the detail from the validation, including the reason for the conclusion as well as the client-ip, EHLO string hostname, receiver and other items. The Received-SPF header format is fully specified in RFC 7208.²

[1] <https://tools.ietf.org/html/rfc7001>

[2] <https://tools.ietf.org/html/rfc7208#section-9.1>

Domain Keys Identified Mail (DKIM)

DKIM verifies email source via digital signatures

1. Sender picks header/body/both parts of email to sign (selector)
2. Upon sending a new message:
 - Selector is hashed
 - The hash is encrypted with private key at the email gateway (signed)
3. The receiver uses the domain/selector combo to:
 - Pull the public key for the domain from DNS
 - Hash the same sections from selector
 - Validate decryption under public key creates the same hash

Takeaway: Look for dkim=pass in authentication-results header!

Domain Keys Identified Mail (DKIM)

Domain Keys Identified Mail (DKIM) is another mail source identification scheme but instead of being based on the IP of the source of the email, it is based on digital signatures. To apply a DKIM signature, a sender of signed email must first select which portion of the email they will sign. This is called the *selector*, which can be the body of the email, the headers, or both, or there can be multiple selector options. Once the selector is chosen, the domain owner must generate a public and private key the emails will be signed with and publish that key in a DNS TXT record so that all receivers of that organization's email can find a copy of it.

Once the setup is in place, every time an email is sent, the selector portion of the email is turned into a digital hash and that hash is then encrypted with the private key that matches the public key placed in the DNS entry. To be clear, this step is performed by the MTA sending the email, *not* the client sending the email. It is one key for the entirety of all email leaving the organization, applied centrally. Once the email is received by another organization's email server, that server can then see the domain and selector combination for the message and query DNS for the respective public key. The server then hashes the same section chosen in the selector for themselves and compares the results to the version transmitted in the email encrypted under the private key. If the two match, then the receiver can be sure the email was sent from the organization's infrastructure. Although there are multiple options for what the "Authentication-Results" header may say for the DKIM result, the one you care most about seeing is "dkim=pass", instead of "fail" or "none". A pass result means the message was signed and it passed all verification tests, while none means the message was not signed (indeterminate whether spoofed or not). Fail means the message was signed and did *not* pass verification (a potentially spoofed email).

One Final Problem: How Many "From" Addresses?

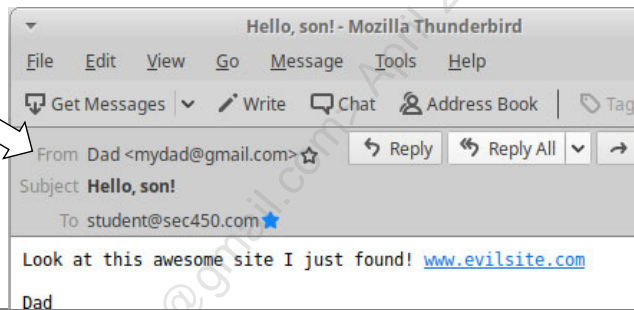
When crafting an email:

```
MAIL FROM:<user@hacked.com>
RCPT TO:<student@sec450.com>
DATA
...
To: student@sec450.com
From: "Dad" <mydad@gmail.com>
...
```

Headers:

```
Return-Path: <user@hacked.com>
To: student@sec450.com
From: "Dad" <mydad@gmail.com>
Subject: Hello, son!
```

- Envelope from used in return path
- **Return-Path** used by SPF for verification!



One Final Problem: How Many "From" Addresses?

One final issue that must be dealt with is what email address to display as the "From" address. Why is that difficult? Because, unfortunately, the multiple standards written that covered email transmission format vs. message format meant that there are numerous from fields that can be used (specifically the RFC5321.MailFrom field vs. the RFC5322.From field).

Think of this difference as the To address and return address on an envelope of a piece of physical mail vs. what is truly written on the letter inside—they may be different and are used for different reasons. The name you type in your SMTP transaction after "MAIL FROM:" is the RFC5321.MailFrom address and will become the "Return-Path" field in the email headers seen by the email received. In the message content of the email, however, you can specify a second and different "From:" line, which is technically the RFC5322.From field. This is important because SPF checks performed by the received rely on the RFC5321 "envelope-from" Return-Path address while email clients will display the RFC5322.From field. This issue is described very well on a blog post on dmarc.org titled, "How Many From Addresses Are There?" You should highly consider reading this post to complement the information in this module.¹

To weaponize this, attackers use the second From address to carefully craft an email that will come from one source but display as if it came from another as shown on the slide above. This gives attackers a method to intentionally confuse the user about who sent them email and if the receiving email server does nothing to solve the problem, it is allowed to happen by design. This trick is possible even while the SPF and DKIM checks pass because the correct information to make these checks pass was in the RFC5321.MailFrom header while the message displayed was in the RFC5322.From field. How do we tackle the issue of mismatched From addresses? Another standard called DMARC.

[1] <https://dmarc.org/2016/07/how-many-from-addresses-are-there/>

DMARC

SMTP has other issues:

- From fields may differ from "**Envelope**" vs. "**Content**"

DMARC builds on SPF and DKIM to fix this

- Allows domain owners to specify if SPF/DKIM is set up
- Enables reporting on authentication failures!
- **To pass DMARC** a message must
 - **Pass SPF and/or DKIM** and be "**aligned**"
 - SPF aligned: **RFC5322.From** matches **RFC5332.MailFrom** field
 - DKIM aligned: **RFC5322.From** matches **DKIM "d="** field

DMARC

To weaponize email by abusing the multiple from address issue, attackers would send an email from one address, while crafting the RFC5322 From address to say another, friendlier looking address. This was possible even while the SPF and DKIM checks pass because the correct information to make these checks pass was in the RFC5321.MailFrom header while the message displayed was in the RFC5322.From field. Therefore, we needed another standard to govern a check that all the From addresses lined up, one that ideally also used SPF and DKIM to also ensure the email was from a verified source with signed contents as well. This is where DMARC steps in.

DMARC works by allowing a domain owner to publish a policy about email from their domain that:

- States whether SPF or DKIM is set up and should be used for messages from their domain
- What to do when authentication fails (nothing, quarantine, or reject the message)
- Tells receivers of fraudulent email how to send reports back to the organization, so that they can keep tabs on if things break or they are the target of spoofing attacks—something that was never possible with email before.

To pass DMARC checks, 2 things must happen, and email must pass the domain owner-specified SPF or DKIM checks and be "aligned." The meaning of being aligned to DMARC depends on if SPF or DKIM or both are used.¹

1. For SPF, being aligned means RFC5322.From header (the one inside the mail content) must match the RFC5321.MailFrom header (the MAIL FROM line in SMTP). The RFC5321.MailFrom line is the one also used for SPF checks.
2. For DKIM, the RFC5322.From header must match the domain in the "d=" field passed in the message's DKIM header signature.

If a domain has SPF, DKIM, and DMARC set up, and all checks pass, receivers of email from that domain can be very certain that organization was the true originator of that email. Be careful, though—this does *not* mean the email is safe. It just means that it's not spoofed. If an employee's machine has been compromised and an attacker can send "real" email from their account, it would still pass all of these checks!

[1] <https://blog.returnpath.com/how-to-explain-dmarc-in-plain-english/>

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

SMTP Summary

Key Points:

- A message *may* be able to be traced to its source through headers
 - SMTP is unauthenticated by nature
 - Without taking extra steps, email can be easily spoofed!
 - In general, you **should not trust** headers beyond your border MTA
- Setting up SPF, DKIM, and DMARC policies prevents spoofing of emails "from" your domain to *yourself and others*
- *Checking* SPF, DKIM, and DMARC on *incoming* mail prevents people from spoofing email to *you*
- ***None guarantee the content of an email is good!***

SMTP Summary

Some key points about the SMTP protocol that we have highlighted throughout this module:

- Although an email *may* be able to be traced back to its source IP address and hostname that created it, there's no guarantee. Clients that use webmail and other solutions that are not programs running on their own machines are unlikely to have the data included. Even when this data does appear in the email, you should question its authenticity. Headers that were created by MTAs before your border MTA are out of your control and should not be trusted. They may be acting maliciously by adding false data. Read the headers carefully as SMTP was designed very naively and has only had security features bolted-on afterward. Many of those security features only work when companies opt themselves in. Realize that without SPF, DKIM, and DMARC set up, it is very easy to send falsely labeled email in several ways. Adjust your level of trust to the amount of verification that has been performed.
- Setting up DNS records for SPF, DKIM, and DMARC is an outstanding way of preventing attackers sending email labeled as if it were coming from *your* organization but will not do much to directly prevent spam coming in to your own organization (minus the important case of attackers trying to spoof internal email). These are technologies that every domain owner must set up on their own domain, largely to protect others. If *every* domain on the internet had SPF, DKIM and DMARC set up, email would be much safer, but unfortunately, according to dmarc.org, only 90% of the Alexa top 100 sites even have SPF records and only 69% utilize DMARC.¹

[1] <https://dmarc.org/stats/alex-top-sites/>

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
- 10. Additional Network Protocols**
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

Server Message Block: SMB

One of the **most commonly attacked services!**

- Windows native, runs with Samba on Linux
- Used for connecting to Windows file shares
- **Port 445**, exposed by default in a domain environment
 - **Attackers use to pivot**, should never be available to/from internet
- For attackers: **Program execution** and **remote login!**
 - **Compare to SSH** capabilities, but must be admin to use
- Used by the most dangerous exploits of the past
 - 9 Equation Group tools

Server Message Block: SMB

You may not realize it, but the service that enables Windows file sharing is one of the ports that are most commonly and successfully used by attackers. Although port 445—SMB—should never be, or rarely is ever exposed to the internet, once attackers get a foothold in the network, the service is usually available from many machines and is commonly used for internal recon and pivoting.

Why is SMB interesting to an attacker? Because the Server Message Block protocol enables an attacker on a remote machine with an administrative level username and password of a victim machine to connect to it, execute any arbitrary command, and interact on a command line. This is a key point you **MUST** understand as a defender: **the capabilities of SMB make password sharing one of the deadliest mistakes in enterprise networks!** How so? **When organizations use the same administrative name and password on multiple machines, such as for the built-in administrator account, a compromise of a single machine (which is easy via phishing) leads to the capability to compromise all other devices with that same username and password.** If your organization uses shared administrative passwords across devices, the network is not segmented with firewalls, and SMB ports are exposed between all devices, you may be a single infected machine away from a full-scale compromise!!

SMB Versions

- CIFS – Windows NT 4.0
- SMB1 – Windows XP, Server 2000, 2003, and 2003 R2
- SMB2 – Windows Vista, Server 2008
- SMB2.1 – Windows 7, Server 2008 R2
- SMB3.0/3.02 – Windows 8/8.1, Server 2012/2012 R2
- SMB3.1 – Windows 10, Server 2016

Clients will use highest version both sides support

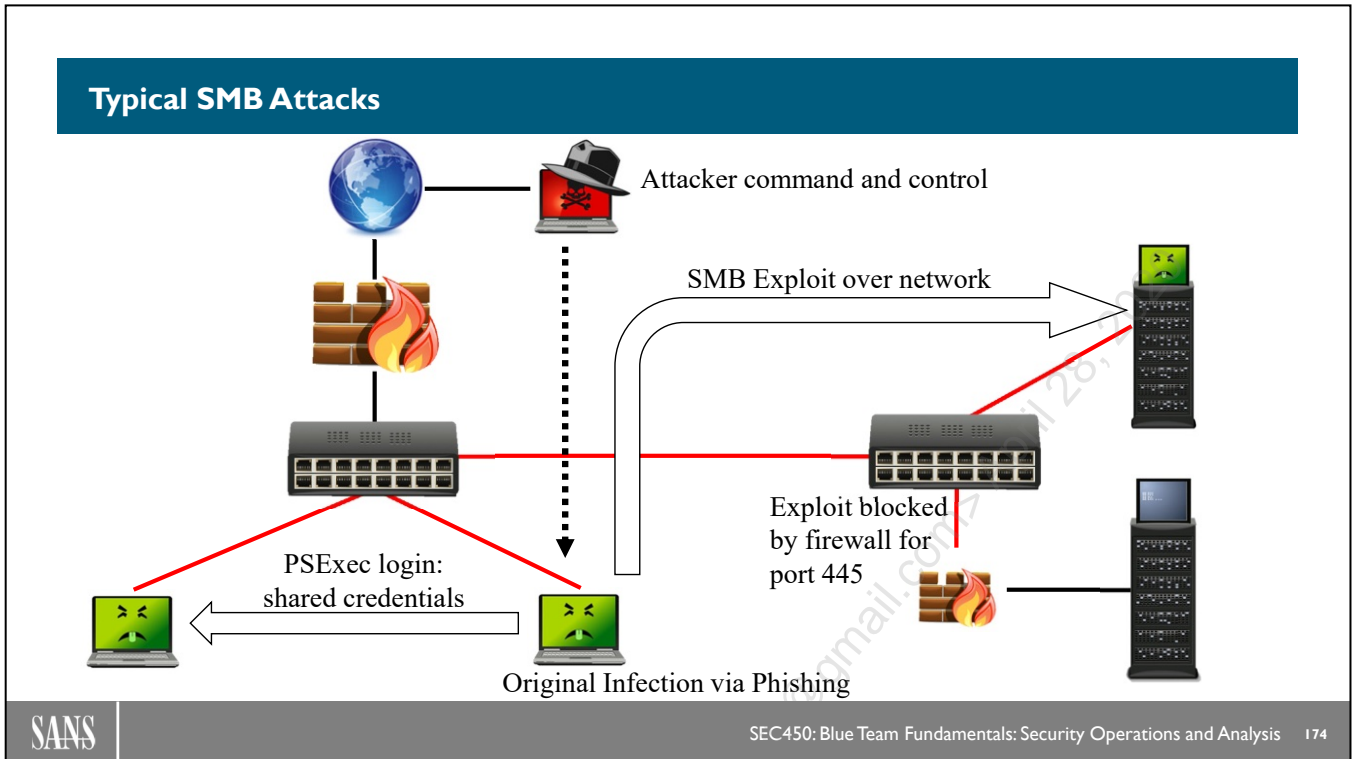
SMB1 MUST be turned off – extremely dangerous

SMB Versions

There are multiple versions of the SMB protocol that have been used throughout the years. The most important thing to know about SMB protocol versions is that you should make every attempt possible to lock down usage to only the newest versions. When establishing an SMB connection, both client and server will send the list of versions they speak and use the highest one supported by both sides.

SMB version 1 is especially bad and should be disabled. Don't take it from me—take it from the person who is in charge of SMB at Microsoft!¹ So many of the SMB-based exploits rely on the non-existent security in SMB1 that removing it should be considered a requirement for a modern defensive strategy. Ideally, SMB2 should be turned off as well. Without SMB2, most of the tools from the Equation Group released by the Shadow Brokers would not have worked.

[1] <https://blogs.technet.microsoft.com/filecab/2016/09/16/stop-using-smb1/>



Typical SMB Attacks

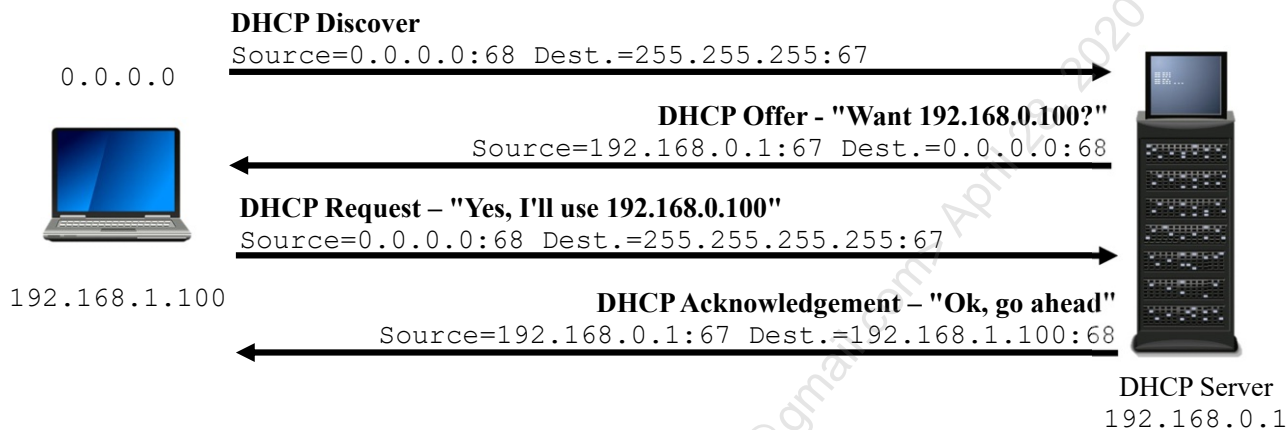
SMB attacks tend to fall into 2 camps—legitimate login credentials used with a tool like PSEXec, or an SMB protocol service-side exploit. This slide demonstrates both.

The original infected laptop in the middle of the slide gives the attacker a foothold—perhaps the victim fell for a phishing email and the attacker was able to stage the SMB attack tools on the device. With remote command and control of the device, the unfortunate choice of the company to use shared credentials across laptops allows the attacker to move from one laptop to the next without issue. If worm-style malware is involved, the infection could also be automatically spread to all devices via this method, this is part of the reason WannaCry and NotPetya were so effective!

SMB also allows attackers to stage exploits such as ETERNALBLUE or one of the other many options and attempt to launch it at any device it can see an open port 445 for on the network as well. If a server with an open SMB port is found that has not been patched for a particular exploit, the attacker can take control of that machine as well, such as the server shown in the upper right. Remember, in the case of exploits as well as "legitimate" tools such as PSEXec, the attacker generally becomes the Administrator of the machine they connect to, not just a user. Combine this capability with the fact that once you become an Administrator of a Windows machine, tools like Mimikatz often allow attackers to steal the passwords of *all* other people currently logged in on that machine, and you can see how open SMB ports can cause a situation to go from bad to worse in a hurry.

Dynamic Host Configuration Protocol (DHCP)

DHCP assigns a host a dynamic IP address using UDP:



Dynamic Host Configuration Protocol (DHCP)

A quick refresher on DHCP—DHCP exists for devices that don't have a statically assigned IP so they can get a dynamic IP assigned to them. The basic process is shown in the slide above. A UDP packet addressed to the broadcast address is sent out so that all DHCP servers can see it and respond with offered addresses. After the first response, the client will then respond to the offer of choice with a request packet and the server will acknowledge the lease. The user then has a dynamically assigned IP address it can use for the duration of the lease time indicated in the offer.

Steps:

1. DHCP Discover: A UDP packet from source IP 0.0.0.0 source port 68 is sent to the broadcast address 255.255.255.255 destination port 67.
2. DHCP Offer: An IP address offer is returned from DHCP server to the user asking for an IP.
3. DHCP Request: Since the client may receive multiple offers if multiple DHCP servers are reachable, the client sends back a DHCP request to the DHCP server that offered the IP it will choose to use. This is also sent to the broadcast address so that all servers that originally responded can read the packet and withdraw their untaken offers.
4. DHCP Acknowledgement: This message is sent back to the client once the offer request has been accepted by the receiving DHCP server.

DHCP for Defenders

Why is DHCP interesting?

- Most incidents require mapping IP address to computer
 - DHCP provides the link from IP address to computer name
 - Computer name can be used to find the device's owner
- Can be used to detect rogue devices!
 - **Hostname:** Matching against naming pattern, black/whitelist
 - **MAC Address OUI**
 - **Blacklist:** Wi-Fi routers, IoT devices,
 - **Whitelist:** Company authorized laptop vendors, etc.

DHCP for Defenders

This protocol is interesting to analysts because so many devices use it to receive an IP address. Once a user has an IP address, they are connected to the network and can start generating normal traffic. If an IDS alert goes off, for example, you will often then need to link the source IP of a device back to that device's name, and ultimately that device's user/owner. DHCP provides the first step in that chain, connecting an IP address someone is using to the name of the machine.

Conversely, DHCP logs also contain a wealth of info that can be directly used for threat detection. One of the threats a network often faces is rogue devices. An attacker may try to sneak into a building and plug a device into the company network, and when this happens, if there is no Network Access Control to stop the device from connecting, it must be detected in some way. Since DHCP servers often will log the hostname, IP address assigned, and MAC address of the device that received a lease, its logs act as a great record of the devices on your network. Consider the possibilities: Hostname is one field that can be matched either against the known naming pattern of authorized devices, or at least against a blacklist of typical Wi-Fi router hostnames like "Linksys" and "D-Link."

The MAC address OUI¹ is another field that can be interpreted for rogue device detection. The first 3 octets of a MAC address make up what is called the Organizationally Unique Identifier, and tells you, if the MAC address is not intentionally spoofed, who the manufacturer of a device is. For example, the OUI of the MAC address 11:22:33:44:55:66:77:88 is 11:22:33. This is useful because if you know all laptops should be Dell devices, the presence of a Lenovo or HP OUI in a MAC address could set off alarm bells. Wireshark's website has a great OUI lookup tool and makes the database of all known OUIs available for download as well.

Remember, none of these methods are foolproof. The hostname and MAC address of a device is a self-reported field that the device itself can set. A clever attacker that knows your PC naming scheme and typical MAC

addresses may disguise their device by cloning something that already exists or making a similar name / MAC address. Nonetheless, it's much better to at least be able to catch some rogue devices, and in the case that the device is something accidentally plugged in by an employee, the name and MAC wouldn't be changed, so this can catch mistakes, too!

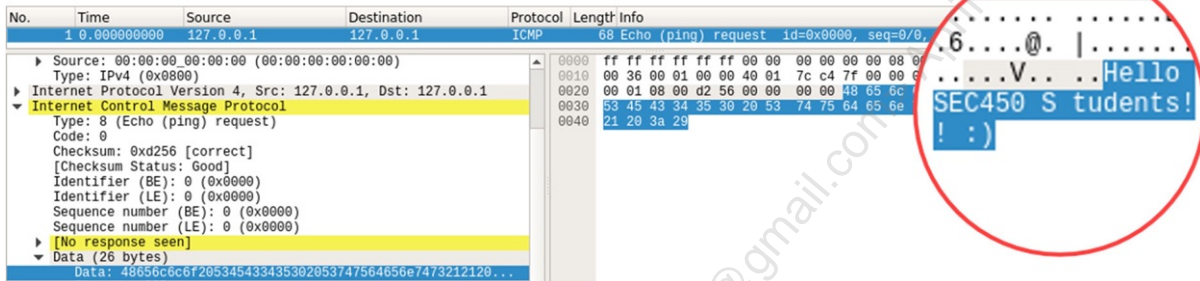
[1] <https://www.wireshark.org/tools/oui-lookup.html>

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

Evil ICMP

You can use ICMP for evil, too!

- Packet contains payload section with arbitrary contents
- Instead of standard payload, smuggle data inside it!
 - 65507 bytes max payload size



Evil ICMP

Can pings be used for evil? You bet! Like most protocols, there is a space within a ping packet to place arbitrary data. So, as with using any protocol for exfiltration, instead of the standard payload, an attacker can choose to pack the ping payload with stolen files, passwords, credit card numbers, or anything else. Each ping has 65507 bytes of space that can be used for anything, if the attacker has the tools required to craft the custom packets. This slide shows a custom ICMP payload crafted with the tool scapy.¹ Although it looks normal in Wireshark, the payload itself was stashed with a special message.

[1] <https://scapy.net/>

FTP

- Old-school protocol for transferring files
- Still used in some networks, but should be phased out
- **Action: Monitor for unexpected use, why?**
 - It's used for exfiltration: Target credit card data was exfil'd via FTP
 - Passwords can be sniffed
 - Data can be tampered with
 - Many old servers are vulnerable to exploitation
- Especially if you do not use it, alert when seen!
- Not just the on perimeter, pay attention to internal, too!

FTP

File Transfer Protocol (FTP) is a legacy protocol that was extremely prevalent on the internet but has now started to fade away. As more and more services shift toward using HTTP, even browsers that used to natively support FTP are moving toward removing it due to lack of use (~0.1% of users per week).¹

As an analyst, why are we interested in FTP? Primarily due to its usefulness for multiple stages of an attack. FTP servers can be exploited to gain access to hosts (much of the software is very old), and they can be used for internal staging of stolen data. To make this worse, passwords can be stolen, and data can be altered since FTP is not encrypted. Additionally, the protocol itself can be used for exfiltration. Many companies may not be watching closely but a well-known protocol that many firewalls will allow make an ideal path for sending out large quantities of data. Don't believe it? Ask Target. The credit cards that were stolen during the Target data breach were purportedly staged on an internal server that had internet access and sent out via FTP.

What do we do about this? If we have a network that doesn't use FTP to transfer any files internally or outbound, it's very easy to keep tabs on anyone's attempt to do so. If there are legitimate uses of FTP within the network, it's easy enough to write an analytic to exclude those uses and alert on anything else. This can begin to highlight any illicit use of the protocol and potentially stop a highly expensive data breach!

[1] <https://www.bleepingcomputer.com/news/google/chrome-and-firefox-developers-aim-to-remove-support-for-ftp/>

SSH

Your **best friend**, and **worst enemy**:

- Good: SSH allows dependable, secure remote connectivity
- Bad: You can use that capability to send *anything* over the tunnel

Need to...

- Tunnel out of your organization, skipping all filters? SSH!
- Route an external device's traffic to inside your network? SSH!
- Forward traffic through a dual-homed network machine? SSH!
- Use X11 forwarding for GUI access from a remote machine? SSH!

Conclusion: **We need to lock down and monitor SSH!**

SSH

Although you may love SSH and use it every day for wonderful and benign server administration, SSH can also easily be used for evil. SSH was designed to allow users to remotely access systems using secure public key authentication with encryption. The thing is, it can do so much more. SSH isn't just for system administrators. Penetration testers and hackers love it, too, because of its capability to tunnel traffic from inside a network to the outside, or the other way around. Here are some of the evil uses of SSH that are possible if anyone is allowed to make an outbound connection:

- Data exfiltration via a standard SSH/SFTP connection. Just connect to the remote endpoint and upload any data you please. It's encrypted, so network security monitoring tools won't see it.
- Want to make a poor-man's VPN to connect an external machine as if it were plugged in on the local network inside an organization? SSH can do that, too. All the attacker needs to do is establish an outbound tunnel and have the remote machine tunnel the traffic in the reverse direction of the previous scenario.
- What if there is a dual-home machine acting as a jump box onto a protected network and attackers need to get into the protected network? If the jump box can be connected to via SSH, they can use local port forwarding to extend the jump box connectivity to any other machine it can SSH to, exposing the protected network much further than intended.
- Linux allows the sending of GUI applications over the network using X11 forwarding. Do you need to open a browser on a victim machine from afar? This can be arranged with SSH as well!

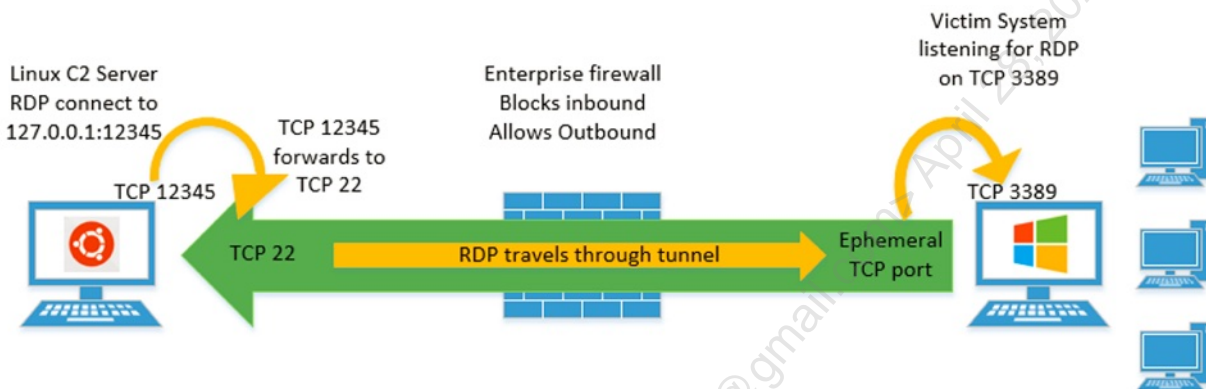
Hopefully, we've made the point that although SSH is an amazing capability, it is a tool that can be used for good or evil. The best way to manage these risks is to carefully restrict where SSH can be used. Whitelisting works best for these types of situations, especially for restricting SSH outbound to the internet. Where it must be allowed for remote system administration, limit who can use it, what network segment they can use it from, and where they can use it to connect to. Heavily monitor the remaining usage. With such a well-defined setup, it will be easy to find when an attacker tries to use it for anything else!

Outbound SSH = Inbound Tunneling

From a FireEye blog post:

*Bypassing Network Restrictions Through RDP Tunneling*¹

January 24, 2019 | by David Pany, Steve Miller, Danielle Desfosses



Outbound SSH = Inbound Tunneling

Here's a real-life example of just how destructive RDP can be. FireEye posted a blog article in January 2019¹ that referenced the above method found used by an APT to gain RDP access to an internal computer. Since the company allowed SSH outbound, the adversary was able to set up a PuTTY link or "plink" to SSH with port forwarding out to an attacker-controlled server. The connection was set up to have any packets that traveled into the established connection on the attacker side get re-routed through the tunnel on the internal victim network side to 127.0.0.1 port 3389 or, Remote Desktop Protocol on the Windows device. In effect, this allowed the attacker to use RDP across the internet to an internal system, merely because the victim organization allowed SSH outbound to arbitrary SSH servers. This is not an attack many people would expect, but it is devastatingly effective as it works not just for RDP, but any protocol and destination IP within the victim network, allowing attackers to freely move inside utilizing the SSH tunnel. Not only that, but all tunneled traffic is encrypted from the defensive team's view, making it impossible to tell whether it's RDP or any other protocol, unless its use is separately detected at the endpoint where the tunnel terminates.

[1] <https://www.fireeye.com/blog/threat-research/2019/01/bypassing-network-restrictions-through-rdp-tunneling.html>

PowerShell Remoting

- **Like SSH for Windows**, but implemented on top of other protocols
 - PSRP = PowerShell Remoting Protocol
 - WSMV = Web Services Management Extensions for Windows Vista
 - SOAP = Simple Object Access Protocol
 - **PowerShell 6.0+ now supports SSH** connections as well
- Also known as **WinRM** (Windows Remote Management)
- Connects using **port 5985 (HTTP) / 5986 (HTTPS)**
- Workgroup or **domain membership required**
- Off by default for desktops
- On by default for servers

PSRP	
WSMV (WinRM)	
SOAP	
HTTP	HTTPS
TCP	
IP	

PowerShell Remoting

Since SSH has not traditionally been available for Windows (although that has recently changed), administrators needed a way to connect to remote machines to run commands and perform other administrative tasks. To meet that need, Microsoft introduced a version of the standardized WS-Management protocol named Windows Remote Management, which is referred to as "WinRM" and is implemented by the winrm.exe program in Windows. WinRM allows administrators to perform what is called PowerShell remoting—starting up an interactive PowerShell session on a remote machine. This uses the "Enter-PSSession" PowerShell cmdlet and looks similar to connecting to a remote Linux machine via SSH.

Although the result is similar, the implementation of the capability is extremely different than SSH. Windows PowerShell Remoting Protocol (PSRP) is implemented using WinRM which is the customized WS-Management implementation Microsoft calls Web Services Management Extensions for Windows Vista (WSMV). These protocols are all transferred over the wire using SOAP formatted (XML style) data inside the body of an HTTP(S) transaction. That means, in effect, remote PowerShell sessions are ultimately implemented by a webserver on the remote side listening for an incoming connection and transferring commands over the network using HTTP with XML in the body—definitely *not* SSH like. One of the main differences between this and normal web traffic is that, by default, WinRM traffic will be used over port 5985 or 5986. Looking for activity on these ports can easily highlight when one machine is connecting to another.

For our purposes, we do not need to go deeper. We merely need to understand that this is a common protocol in use on many networks and be able to recognize it when we see it, as well as understand what it is capable of. Since PowerShell remoting is a remote management framework, it is usable for internal pivoting, backdoor installation, and other attack tactics beyond its intended administrative use. The things to look out for with PowerShell remoting are the activation of its related service (covered later) and unexpected usage, which you can detect by seeing traffic to port 5985 or 5986. Usage of Windows remoting should be defined and any attempts outside of the pre-determined source and destination subnets should be flagged as an alert. The same is true of SSH.

Is Any Protocol Safe!?!?

Not really...if it can deliver a payload, it can be used for evil

- What good is a protocol that can't deliver data?
- Therefore, you must be able to watch for odd behavior
 - Deep packet inspection to log metadata for transactions
 - Protocol compliance on firewalls
 - Network security monitoring
 - Layer 3- and 4-based detection layered on top
 - Outbound default deny
- Well-defined subnets and traffic flow are key to attack detection!

Is Any Protocol Safe?!?

As we've seen throughout the day, almost every protocol is a potential security concern. Protocols are not inherently evil or good, they're just a tool to move data. But if something can move data, it can be used by attackers to move evil data, perform internal pivoting, or facilitate command and control in clever and unexpected ways. The best defense we have as the blue team is a strong monitoring policy and the ability to detect anomalies. Anomaly detection can be complicated, which is why having a well-defined network with a traffic flow that is known can help bring order to the chaos. If you can say which protocols should be used in which areas, coding analytics to find out-of-place connections becomes much easier, and inherently whitelists the ways the network can be used.

Whitelisting is one of the strongest detection tactics because it doesn't rely on signatures, only deviations from known good, which means it can pick up both known and unknown attack styles. Since attackers will eternally be playing cat and mouse with defenders, it's imperative for the blue team to do their best to understand their own network and strive for a monitoring solution that will help them detect intrusions and post-exploitation activity, whether the attackers are using a known method or not.

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. **Day 2 Summary**
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Day 2 Summary

- Understand the pieces of your network and traffic flow
- Know your network security monitoring **collection** types
- Know your points of **visibility** on the network
- **DNS, HTTP(S), and SMTP** are of primary importance
 - DNS and HTTP will help you find and follow attack traffic
 - SMTP is complicated, and phishing is a primary attack vector
 - Understanding auth. technology and headers can identify **spoofing**
 - Many other **common network protocols are all used for evil**
- Tomorrow: Understanding your endpoints!

Day 2 Summary

We've covered a lot of ground in Day 2. The major takeaways from this day are:

- Do your best to understand the major architectural pieces of your network. How does traffic flow? What are the segments? What security controls are keeping them separate?
- Where is your collection monitoring information, how does that drive what stages of an attack you will have evidence of? If someone phishes an employee, moves to a server, stages data on a DMZ server, and exfiltrates it, can you see all of those steps?
- Become familiar with DNS, HTTP, HTTPS and SMTP protocol. Since so many attacks involve these protocols at some point, they are the heavy hitters of the security world. There are additional protocols you should also be familiar with and know what is possible when attackers use them—SMB, ICMP, FTP, SSH, PowerShell Remoting. All can be used for multiple stages of an attack. So it is worth understanding how attackers may do so.

Tomorrow, we will move on from the network and cover the most important concepts to understand about endpoints!

Course Roadmap

- Day 1: Blue Team Tools and Operations
- **Day 2: Understanding Your Network**
- Day 3: Understanding Hosts, Logs, and Files
- Day 4: Triage and Analysis
- Day 5: Continuous Improvement, Analytics, and Automation

Understanding Your Network

1. Network Architecture
2. Traffic Capture and Analysis
3. Understanding DNS
4. DNS Analysis and Attacks
5. Exercise 2.1: Exploring DNS
6. Understanding HTTP(S)
7. HTTP Analysis and Attacks
8. Exercise 2.2: HTTP and HTTPS Analysis
9. Understanding SMTP and Email
10. Additional Network Protocols
11. Day 2 Summary
12. Exercise 2.3: SMTP and Email Analysis

This page intentionally left blank.

Exercise 2.3: SMTP and Email Analysis



Exercise 2.3: SMTP and Email Analysis

Exercise 2.3: SMTP and Email Analysis

Please go to Exercise 2.3 in the SEC450 Workbook or virtual wiki.