599.4

Avoiding Installation, Foiling Command & Control, & Thwarting Lateral Movement

SANS

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

SANS

Avoiding Installation, Foiling Command & Control, & Thwarting Lateral Movement

This page intentionally left blank.

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

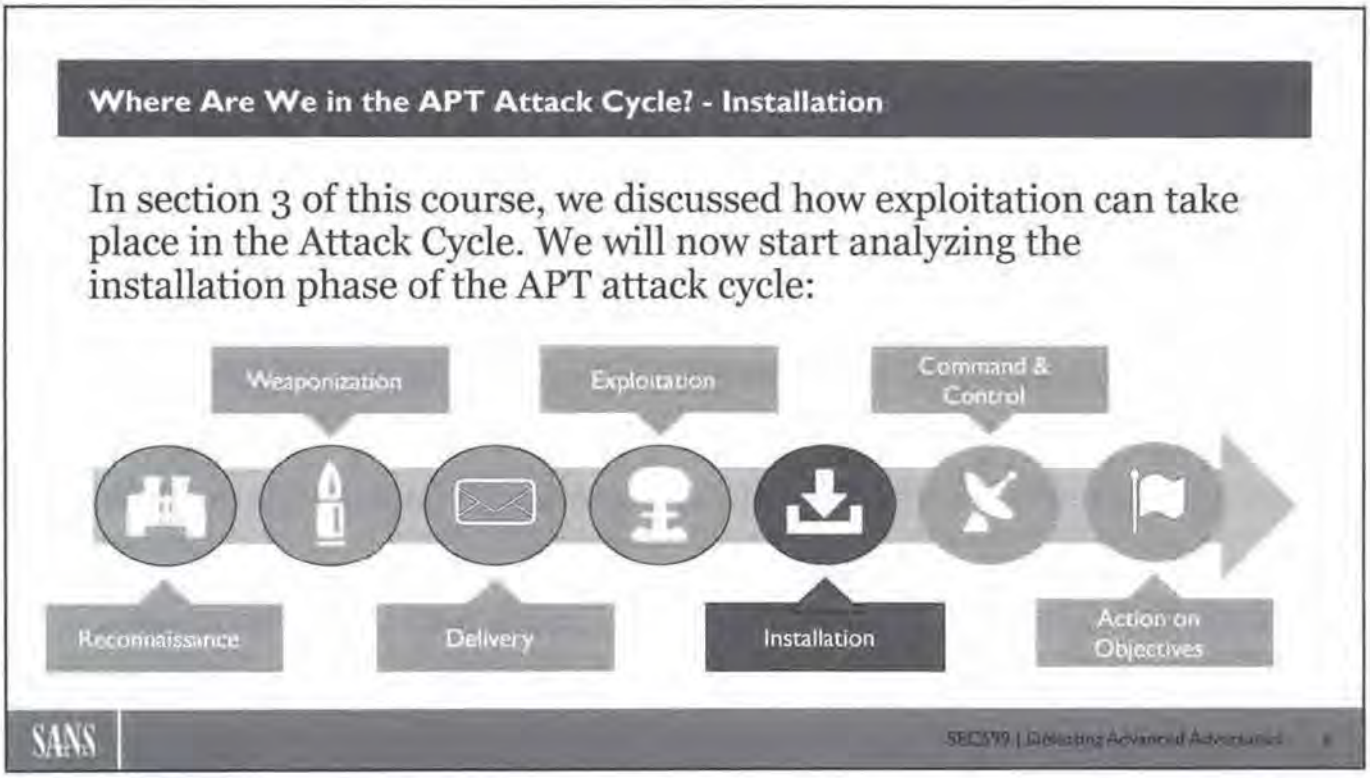
Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

TABLE OF CONTENTS

This page intentionally left blank.

This page intentionally left blank.



Where Are We in the APT Attack Cycle?

Welcome to Day 4 of SEC599! Yesterday, we saw how an adversary can perform initial exploitation of a target environment. Today, we will zoom in on two further steps of the attack: Installation & Command & Control. We will also address common lateral movement strategies, which are typically part of the "Action on Objectives" phases. But let's not get ahead of ourselves and start zooming in on Installation:

Adversary perspective

During the installation step of the attack cycle, adversaries try to achieve persistence on the target machines (if it is part of the goal). When persistence is required, changes must be made to the configuration or software of the machine to achieve persistence. This is necessary because typical computer operating systems like Windows only run started programs as long as the user is logged on. If the user logs off (or the machine is restarted), running programs are terminated. To restart the malicious programs automatically when a user logs on again, persistence mechanisms must be used.

Windows has a large and diverse set of "autorun" options that can be used for persistence. This can be done in the context of a user, so that persistence is achieved only when the same user logs in again, or in the context of a machine, so that persistence is achieved when the machine is started. Persistence can be as simple as a Start entry in the user's Windows menu configured to run the malicious payload again, or as complex as a dedicated backdoor running as a service or even installed in the firmware of the computer. Webshells are typical backdoors left behind on compromised web servers.

Defender perspective

To achieve persistence on a target system, adversaries must make changes to the configuration of said systems. Not only can many of these changes be prevented by hardening, but they can also be detected by monitoring applications like Microsoft's Sysmon. In homogenous environments, configuration base lining can help detect this type of changes. Some host-based Intrusion Prevention Systems monitor autorun configurations too and alert on any changes made to them.

The Windows operating system provides an abundant set of configuration options that can be used to achieve persistence. It should be noted that achieving persistence does not necessarily require the malware to be stored in files on the file system. So, called, "fileless" malware can achieve persistence by storing commands inside autorun entries in the registry. When executed at startup or login, these commands will inject malicious code inside an existing process. The malicious code is often stored in an alphanumeric representation in the registry, like BASE64.

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

Persistence?

Upon successful exploitation, adversaries typically want to **persist** their access on the target environment (e.g. to survive reboots, user logoff, ...) There's two main categories of persistence strategies:

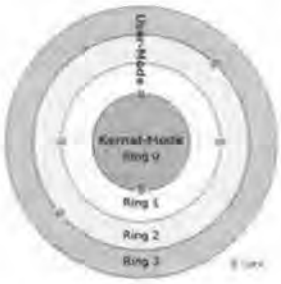


Image Source: Wikipedia

User space

- "Hiding in plain sight"
- Do not always require administrative access to system
- Examples: Web shells, scheduled tasks, user profile, ...

Kernel

- Has additional capabilities to hide itself from investigations
- Require administrative access to system
- Examples: Device drivers, Loadable kernel modules, ...

Both Userspace & Kernel persistence strategies are used by APTs!

Persistence?
Upon successful exploitation, adversaries typically want to persist their access on the target environment (e.g. to survive reboots, user logoff, ...). Depending on the privileges available to the adversary, they could choose to hide in two main parts of the victim system:

- User space: User space is the memory area where application software (and a limited number of drivers) execute. All "user" interactions typically occur in user space.
- Kernel: Kernel space is strictly reserved for running a privileged operating system kernel, kernel extensions, and most device drivers. Code used for persistence in kernel-mode is typically referred to as "rootkits", as they interact with low-level parts of the OS and can thus hide themselves better from investigations.

Advanced adversaries have been known to use both locations for persistence. You might think they would always prefer kernel-mode persistence. This is however not true: In some cases, the very presence of a "rootkit" could signal something is suspicious... They might thus prefer to compromise the user environment and thus limit themselves to "hiding in plain sight". Adding a scheduled task, setting a "Run" registry key, adapting the user profile, ...

What Persistence Strategy Is Used? (1)

Next to the "user space" vs "kernel" question, there are other items that will determine the type of persistence strategy used:

- What is the function of the target system?
- Is it available from the Internet?
- Does the adversary have administrative privileges to the system?
- ...

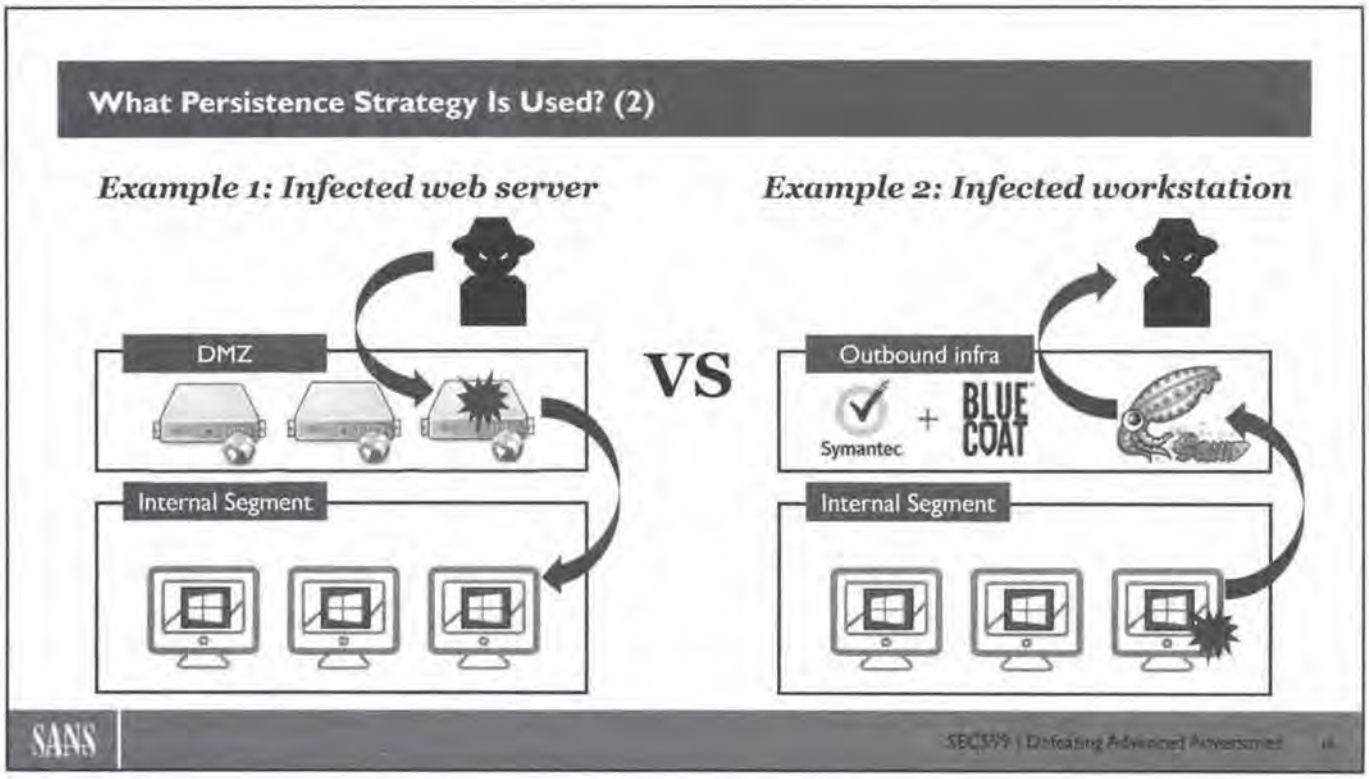
! Successful persistence does not always require administrative privileges !

SANS

What Persistence Strategy Is Used? (1)
We already discussed the "user space" vs "kernel" question in the previous slide. There are however a number of other questions that will determine what type of persistence strategy is opted for by the adversary:

- What is the function of the target environment?
  If it's a workstation, we can assume that reboots & user logons will occur frequently and thus a persistence strategy related to the user profile could be a viable option.
- Is it available from the Internet?
  If it is, the adversary could opt to place a backdoor in an existing service (e.g. web shell) and use that to directly reconnect to the system.
- Does the adversary have administrative privileges to the system?
  If administrative privileges are available, the adversary could opt to attempt installing a rootkit that operates in kernel-mode.

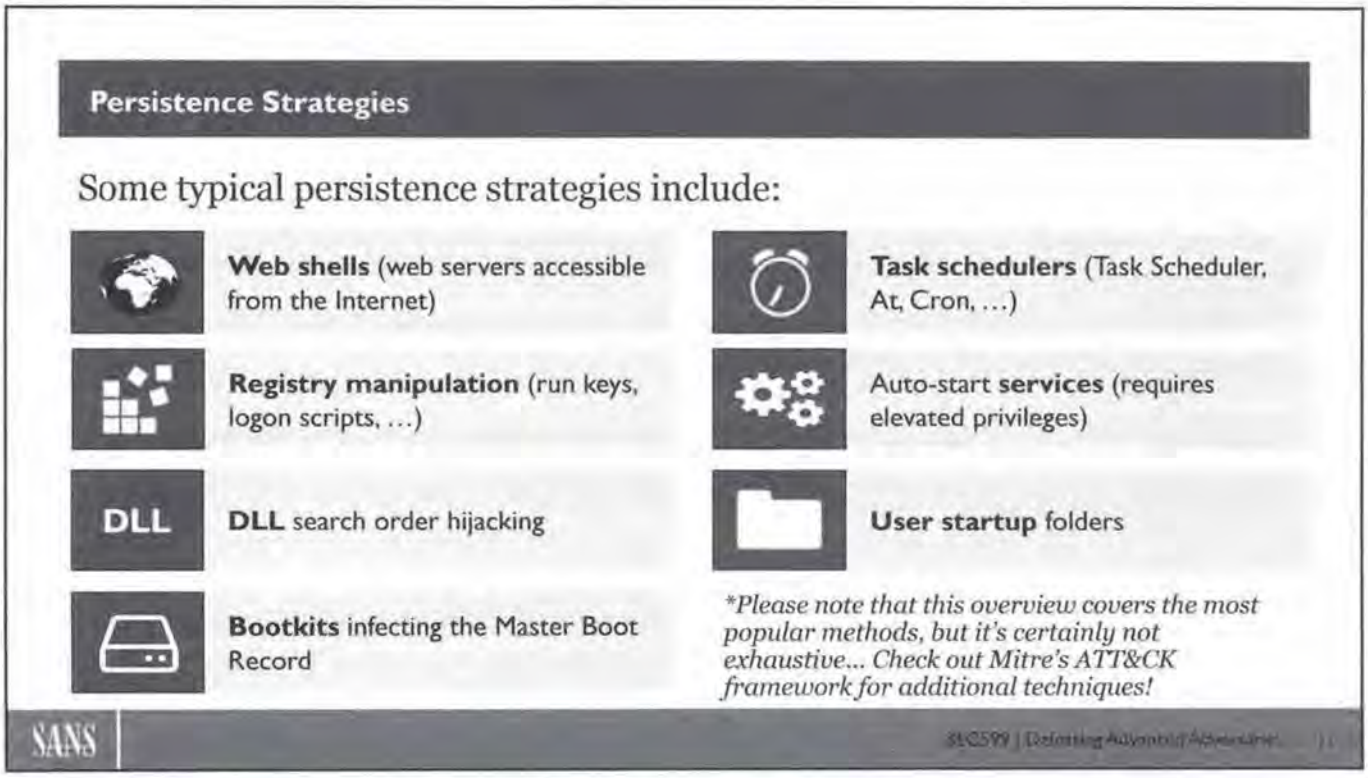
Even if administrative privileges are helpful to the adversary, persistence can be achieved without having them as well!



What Persistence Strategy Is Used? (2)

The diagram in the slide provides an insight in two different types of persistence:

- On the left, we can see an infected web server, where the adversary could now deploy a backdoor as part of the web application. This backdoor can be used at will by the adversary as a pivot point to further compromise other parts of the internal network.
- On the right, we have a situation where the adversary has compromised a workstation already inside the environment. As the adversary would like to retain control over the infected workstation, he will opt for a persistence strategy that will automatically launch a backdoor that will attempt to connect towards the adversary again.

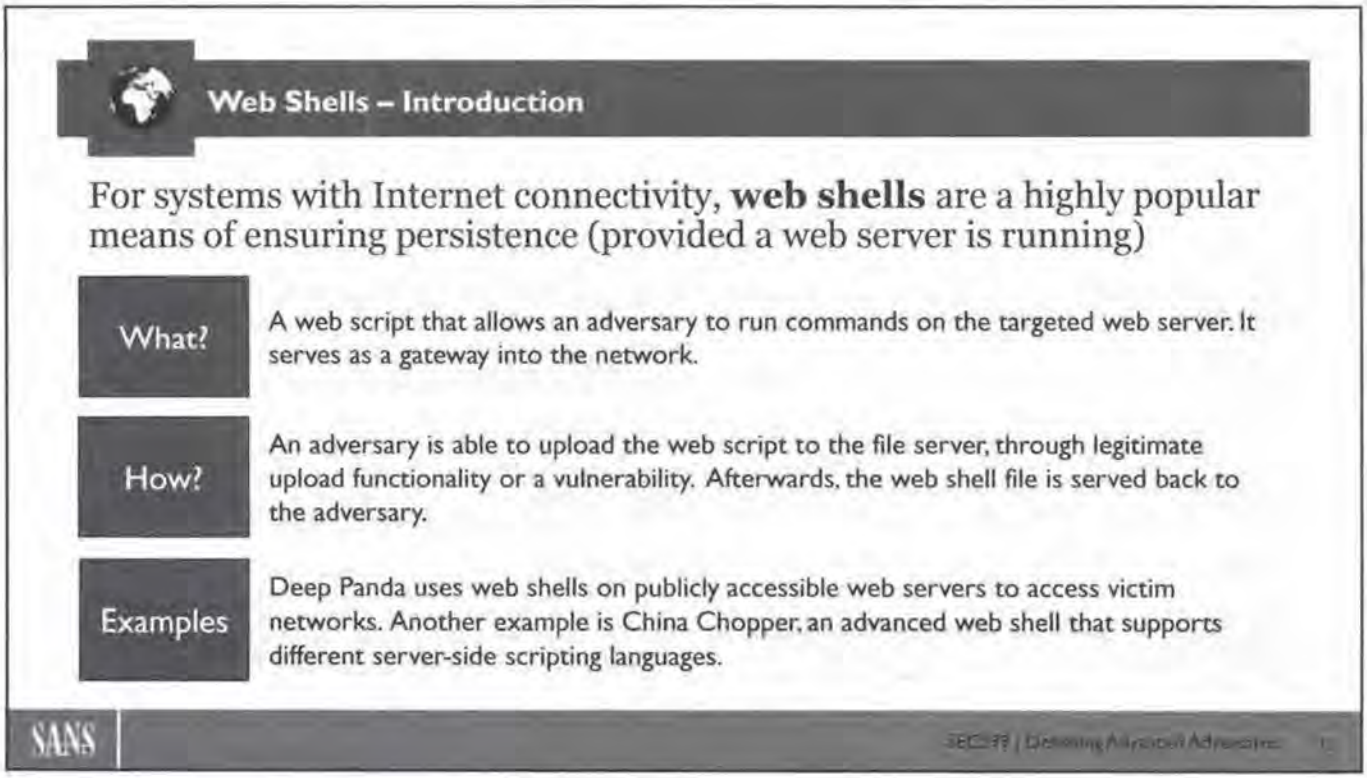


Persistence Strategies

A number of common persistence strategies are listed below. These will be explained in more detail in the following slides.

- Web shells: Only useful for Internet-accessible systems. Often used as an initial entry point in an environment.
- Task schedules: All major operating systems have task schedulers available (Task Scheduler, At, Cron, ...). A highly popular means of persistence.
- Registry manipulation: A number of registry keys are used during the start-up / user logon process. These can be abused by adversaries to add malicious code to be executed upon start-up / logon.
- Auto-start services: Many systems use a number of services that will automatically launch at start-up (often even with elevated privileges). Adversaries could add additional services that appear to have normal names or functions (will require administrative privileges);
- DLL search order hijacking: Highly interesting technique abusing the way Windows prioritizes the loading of DLL's;
- User startup folders: much like registry manipulation, adversaries could implement shortcuts / scripts in a user's startup folder, which is executed upon logon;
- Bootkits are used to affect the system upon startup. Malicious code that infects the Master Boot Record (MBR) will run even before the Operating System is launched.

We will analyze all of these techniques in the upcoming section!



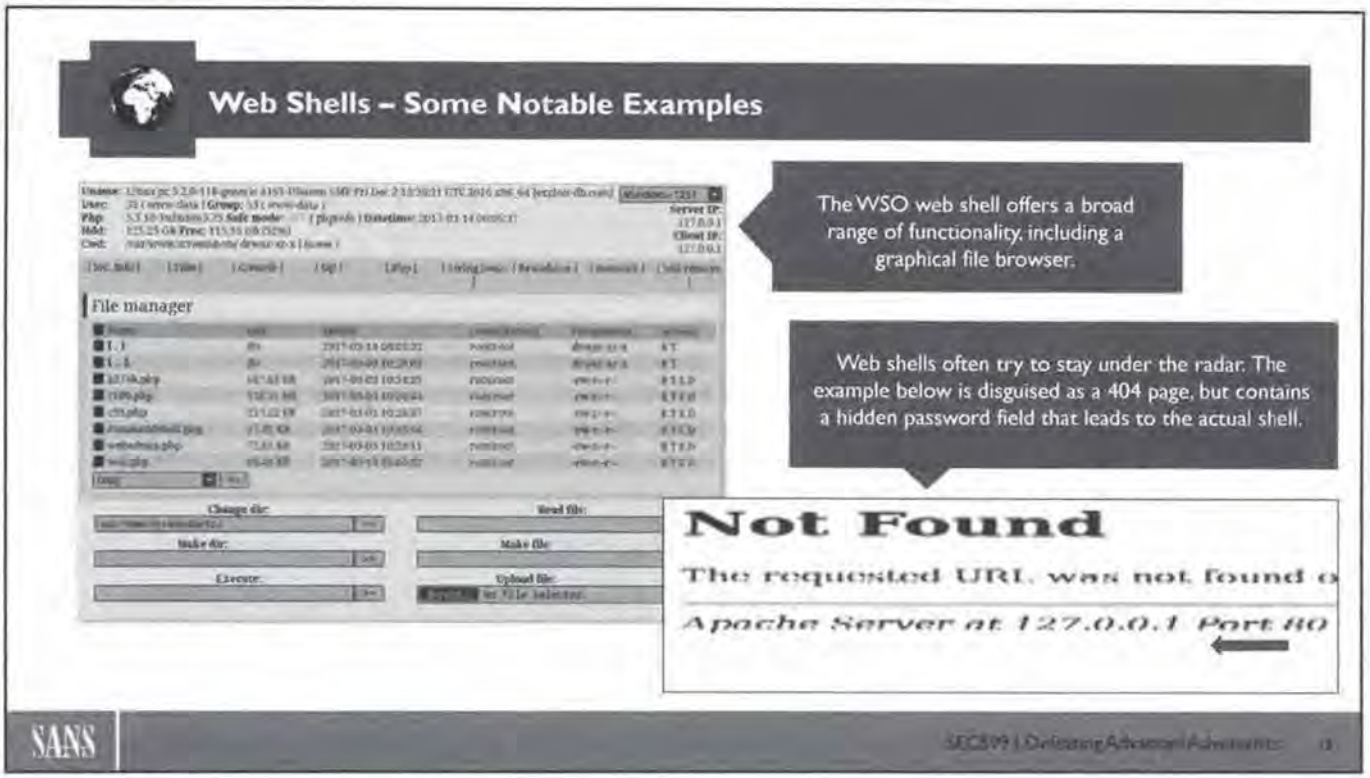
Web Shells – Introduction

A web script allows an adversary to run commands on the targeted web server. It can serve as a gateway into the network. Web shells can provide a simple interface that allows to run single commands, or consist of an advanced GUI with multiple types of functionality, such as direct file access, database connections, or network reconnaissance to explore the internal network.

For an adversary to be able to abuse a web shell on a web server, the web shell first has to be uploaded. This could either be done through a legitimate upload function provided by the web server, or might be possible due to a vulnerability present in the web application or web server software. Once the adversary has been able to upload the web shell, it has to be served back. If the file is accessible, but not interpreted as web script, and thus shown back as simple text, the adversary will not be able to execute commands. The web server has to interpret the web shell's script and serve that back to the adversary.

Some famous examples are web shells used by the Deep Panda threat group, who used them as primary access back into victim organizations in the defense, legal, telecommunication and financial industries. This was an interesting approach as web shells were mostly only seen as a first stage into obtaining a foothold in the target network, after which they would be abandoned as soon as a second stage malware was communicating back to the adversaries. The usage of web shells, gave Deep Panda some advantages, such as the absence of C&C beacon traffic, a low detection rate by AV products, and the ease to switch source IP addresses, making it difficult for the defenders to block known C&Cs. More information on Deep Panda web shells can be found here: https://www.crowdstrike.com/blog/mo-shells-mo-problems-deep-panda-web-shells/

An advanced example of a web shell is the China Chopper shell, which supports server payloads for many different kinds of server-side scripting languages and contains functionality to access files, connect to a database, and open a virtual command prompt. An analysis of China Chopper can be found here: https://www.fireeye.com/blog/threat-research/2013/08/breaking-down-the-china-chopper-web-shell-part-i.html.



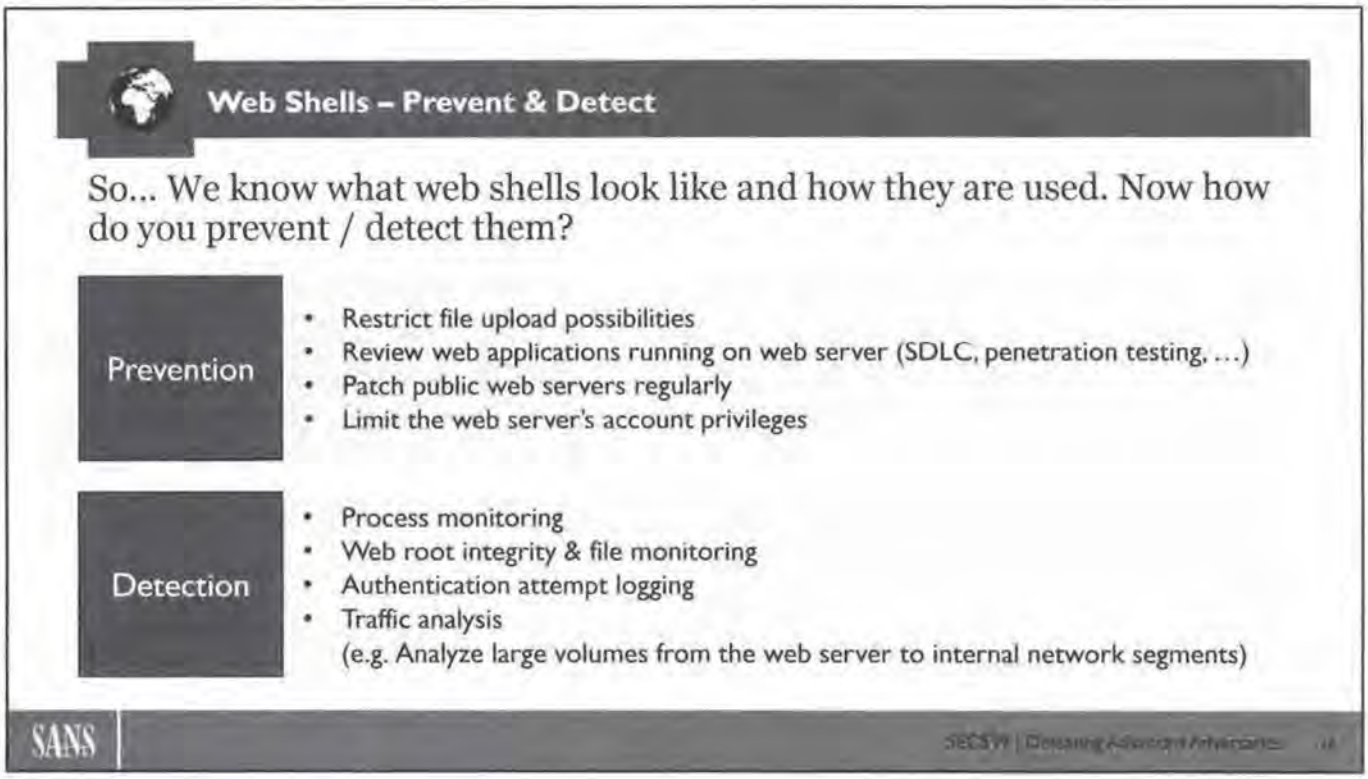
Web Shells - Some Notable Examples

A number of shells offer the creation of a botnet in as little as a click, launching standalone processes that either connect to a command and control server or listen for commands over an insecure TCP connection. Some allow performing port scans to find potentially exploitable services. Others enable fraudsters to schedule denial of service attacks. There are shells dedicated to sending bulk spam emails, testing stolen credentials against popular websites (such as PayPal or Amazon), cracking passwords, and automatically defacing websites. With such a wide array of powerful features, it is unsurprising how popular web shells are with cyber criminals.

A popular and feature-rich web shell is WSO. WSO offers, among others file management, such as browsing directory contents in a GUI, but also both bind shell and back connect options. Selecting one of these options will launch a standalone process that will connect to or listen for a connection from a remote command and control server - an easy method for the creation of a botnet.

Web shells will often try to stay under the radar to avoid detection by server admins or other hackers. A particularly common ploy is that of fake error pages, used by some variants of the C99 web shell. These shells attempt to recreate the default Apache error pages, usually 404 Not Found or 403 Forbidden. When viewed in a web browser, these fake pages can easily be mistaken for legitimate error messages. However, when compared side-by-side, discrepancies can be found by looking for incorrect or omitted version numbers, hostnames, URLs, and HTML titles. These fake error pages also contain hidden password fields, which provide access to the web shell: some variants simply set the background and border colors to match the page background, while others add JavaScript that reveals the password form when the port number is clicked.

Another notable method for avoiding detection is prefixing the web shell scripts with small excerpts of image file headers – most commonly those from the GIF89a specification. When processed by the PHP interpreter, these bytes are ignored and passed through to the web browser, displaying the text "GIF89a". Automated tools such as the open-source utility file use these magic bytes as a fingerprint to identify the file type, mistaking the malicious PHP script for an image. Reference: https://news.netcraft.com/archives/2017/05/18/web-shells-the-criminals-control-panel.html



Web Shells – Prevent & Detect

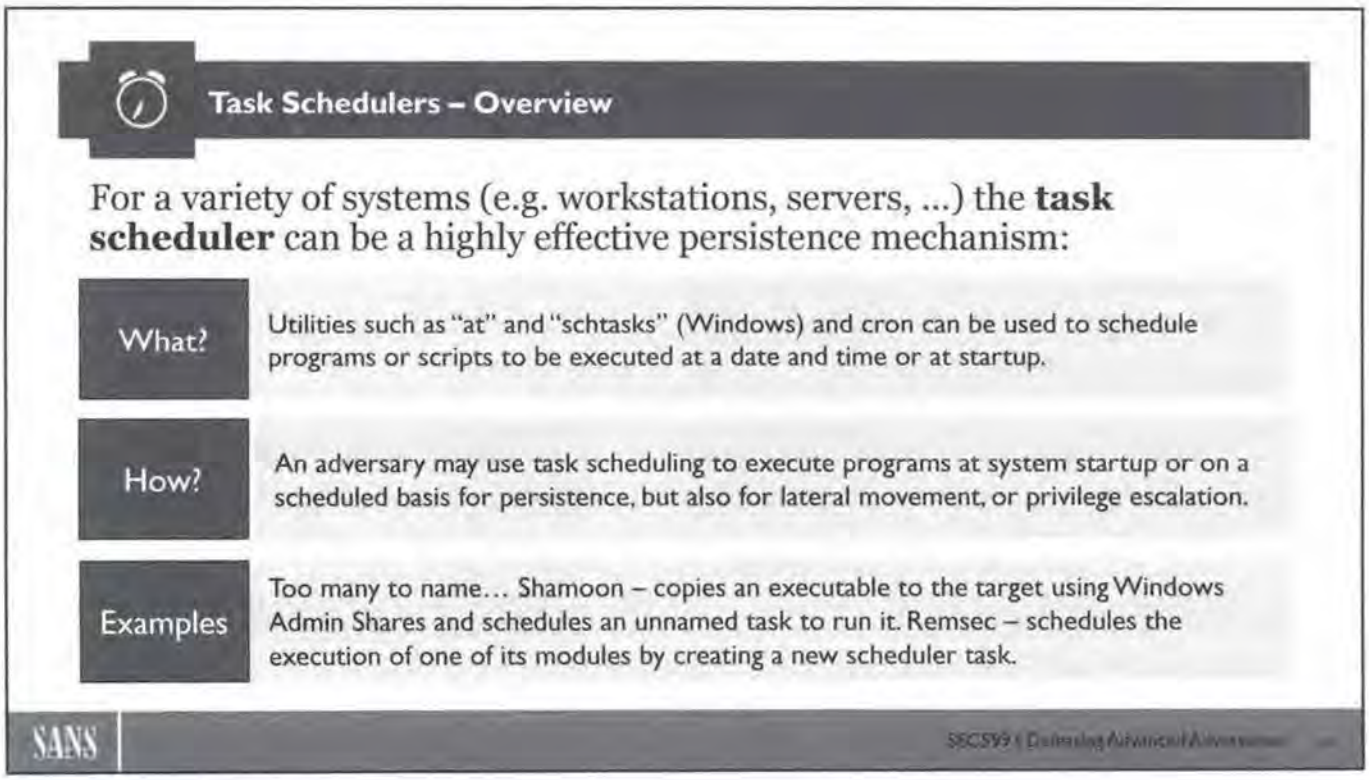
It is easier to prevent web shells from being abused as a persistence mechanism in your server or network than to detect them.

If your web application has a file upload function, it should have some restrictions. First of all, it's possible to limit the types of files that can be uploaded (based file header for example). It could be wise to disallow users to upload web scripts of various types. In case all kinds of files are allowed to be uploaded, it's possible to restrict how users access these files. Avoid having the web server interpret uploaded scripts, for example by changing the file names for all uploaded files or by only allowing users to download them.

In case file upload is not allowed, measures should be taken to avoid unforeseen file upload. Make sure adversaries cannot abuse known vulnerabilities that could allow remote code execution or file inclusion by regularly patching your externally reachable web servers.

In case adversaries do succeed in uploading and abusing a web shell, it is still possible to limit the possible damage they can do. Audit account and group permissions of the web server's account and make sure it does not have local root privileges or access to unnecessary files and folders. If the web server is part of an Active Directory, make sure accounts that are used to manage it do not overlap with permissions for the internal network.

In case all prevention measures failed, the web shell might still be detected. However, they can be difficult to detect, since they do not initiate connections. The portion of the shell that resides on the web server might also be small and innocent looking. Process monitoring could be used to detect suspicious actions such as command execution or file access outside the web directory. Often, an adversary will try to go for the /etc/passwd file. File monitoring could be used to detect web shell code being injected into other web application files. In case files are changed that do not match updates to the web content, this might be an indication of unauthorized changes by an adversary. Additionally, log authentication attempts to the server to avoid potential brute forcing and monitor network traffic for suspicious activity to and from the web server and the internal network.

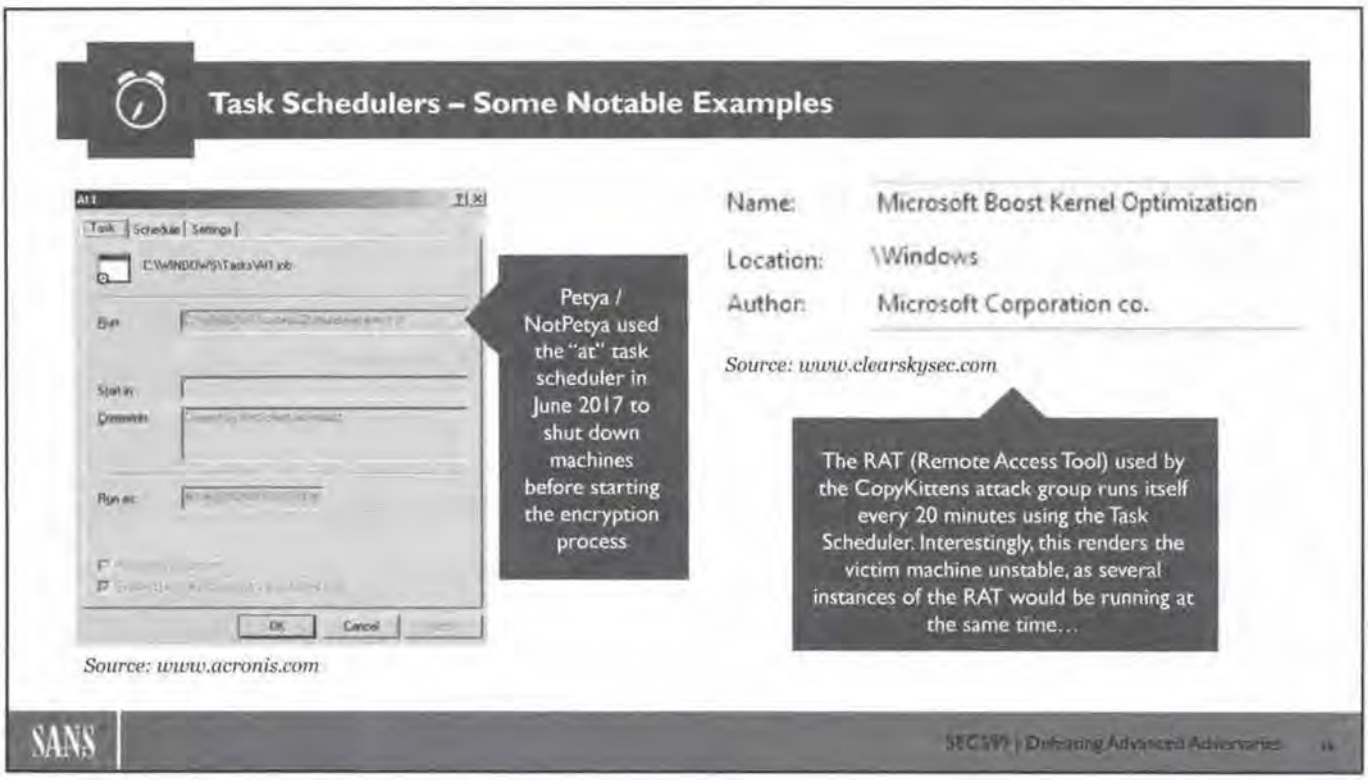


Task Schedulers - Overview
Utilities such as "at" and "schtasks", along with the Windows Task Scheduler, can be used to schedule programs or scripts to be executed at a date and time or at startup. The account used to create the task must be in the Administrators group on the local system. A task can also be scheduled on a remote system, provided the proper authentication is met to use RPC and file and printer sharing is turned on.

An adversary may use task scheduling to execute programs at system startup or on a scheduled basis for persistence, but also for lateral movement, or privilege escalation to SYSTEM, or running a process under the context of a specified account.

A famous case that makes use of scheduled tasks is Shamoon. It copies an executable payload to the target using Windows Admin Shares and schedules an unnamed task to run the malware.
Remsec – schedules the execution of one of its modules by creating a new scheduler task.
https://www.fireeye.com/blog/threat-research/2016/11/fireeye_respondsto.html

Another well-known example is Remsec, a modular backdoor aimed at espionage, which was used by Strider, a.k.a. ProjectSauron. The execution of one of its modules was scheduled using a new scheduler task. An analysis by Kaspersky can be found here: https://securelist.com/files/2016/07/The-ProjectSauron-APT_Technical_Analysis_KL.pdf.

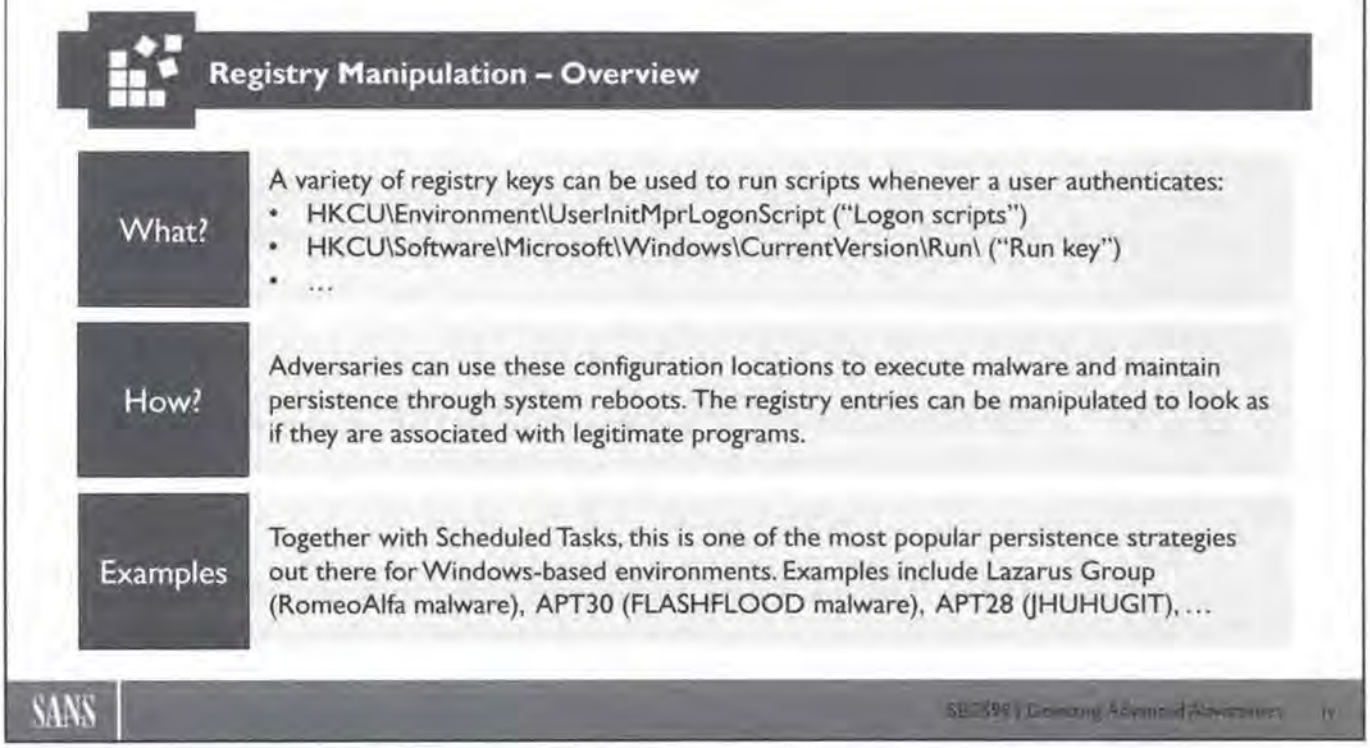


Task Schedulers – Some Notable Examples

So, let's have a look at some notable examples of persistence through task schedulers. Due to the simplicity of this attack, it's one of the most popular persistence strategies. Some examples include:

- The Metasploit Meterpreter has a built-in "automated persistence" method that will rely on a scheduled task being added;
- Petya / NotPetya used the "at" task scheduler in June 2017 to shut down machines before starting the encryption process;
- The RAT (Remote Access Tool) used by the CopyKittens attack group runs itself every 20 minutes using the Task Scheduler. Interestingly, this renders the victim machine unstable, as several instances of the RAT would be running at the same time…

From a forensic perspective, scheduled tasks (both using the Task Scheduler and "at") do leave some artefacts we can use to further analyze & investigate.



Registry Manipulation - Overview

Adding an entry to the "Run keys" in the Registry or startup folder will cause the program referenced to be executed when a user logs in. The program will be executed under the context of the user and will have the account's associated permissions level. Some examples include:

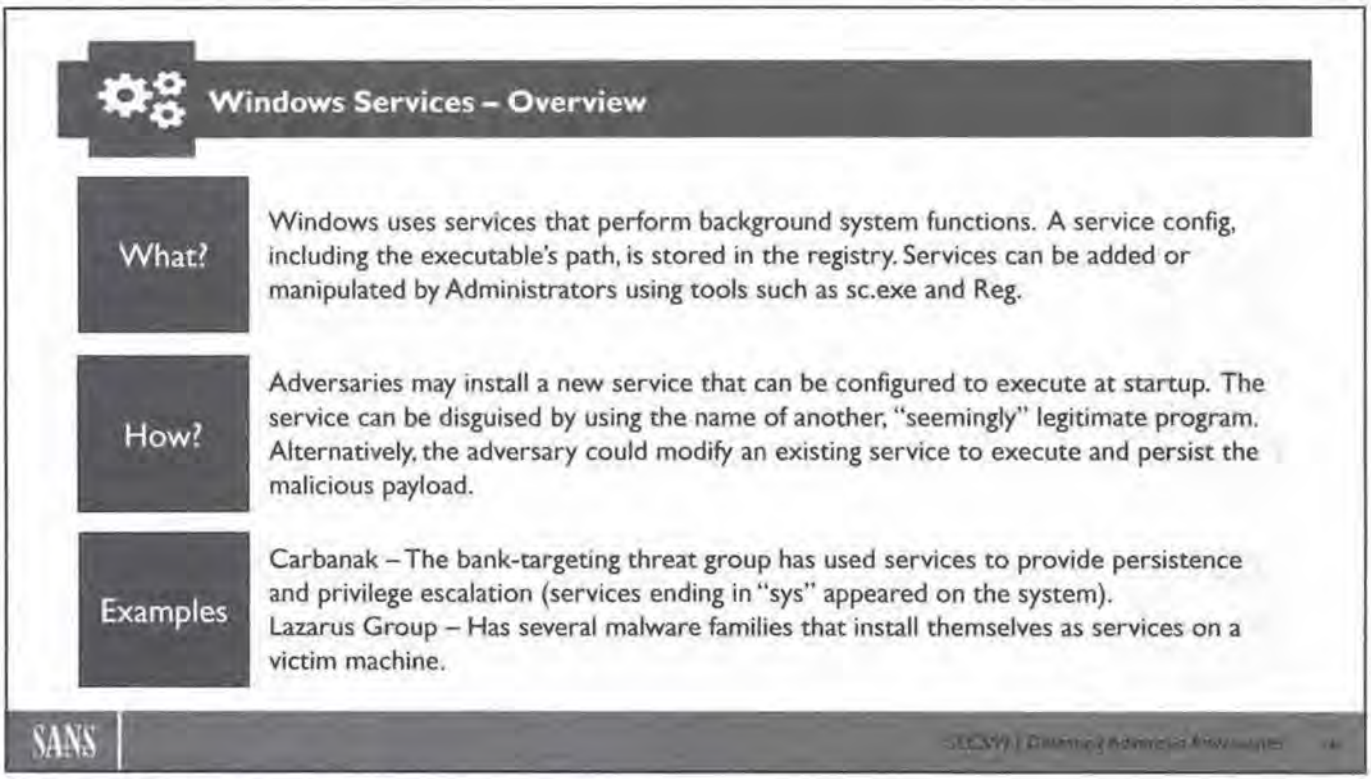
- Login scripts under HKCU\Environment\UserInitMprLogonScript
- "Run" registry keys under a variety of possible locations, including HKCU\Software\Microsoft\Windows\CurrentVersion\Run\

Adversaries can use these configuration locations to execute malware and maintain persistence through system reboots. The Registry entries can be manipulated to look as if they are associated with legitimate programs.

Lazarus Group made use of RomeoAlfa malware, which has persistence by saving itself in the Start menu folder.

FLASHFLOOD – Malware developed by APT30 that allows to exfiltrate data across air-gaps. It creates an entry in the Run key. The following reference contains a section on the FLASHFLOOD malware: https://www2.fireeye.com/rs/fireye/images/rpt-apt30.pdf.

JHUHUGIT is a piece of malware used by APT28. It registers a Windows shell script under the Registry key HKCU\Environment\UserInitMprLogonScript to establish persistence. A good analysis of the APT28 group, including the use of Logon scripts is located here: https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part1.pdf



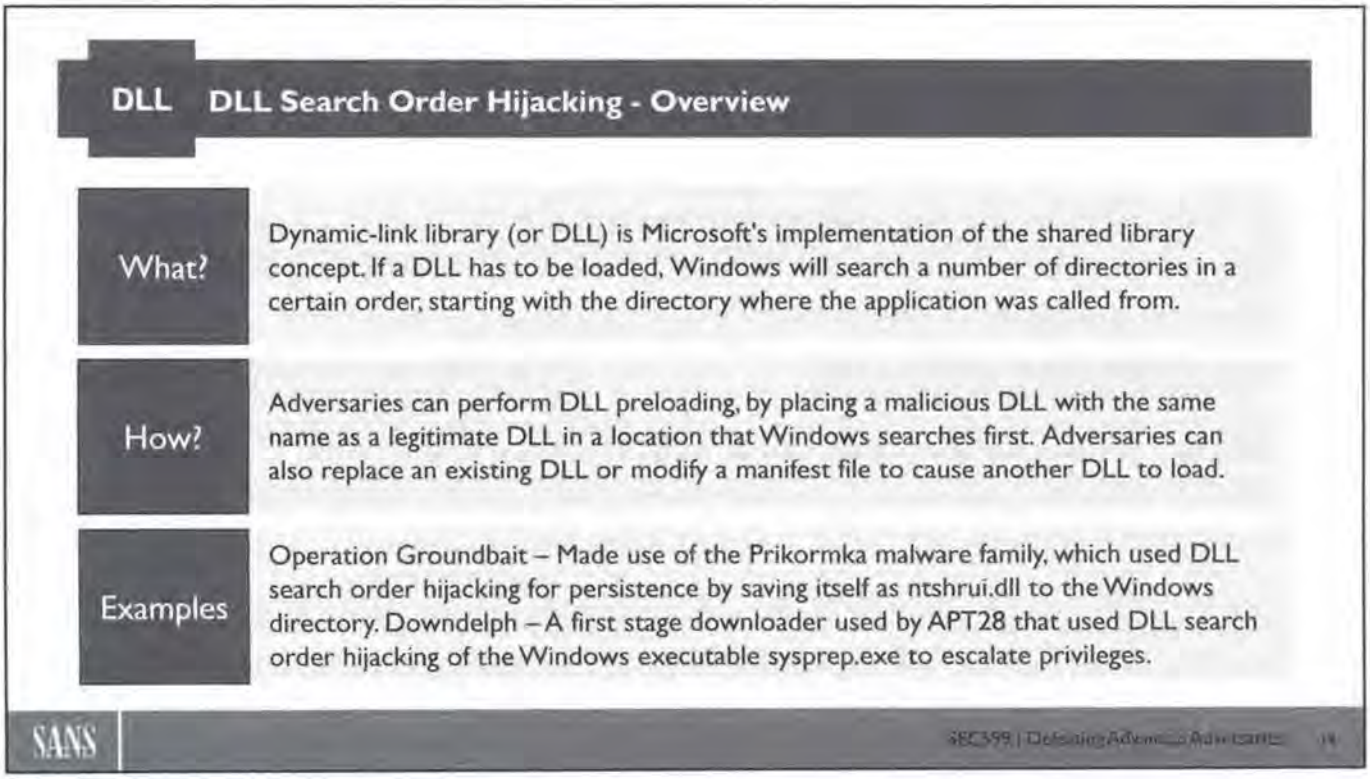
Windows Services – Overview
When booting, Windows can start programs or applications called services that perform background system functions. A service's configuration information, including the file path to the service's executable, is stored in the Windows Registry. Service configurations can be modified using utilities such as sc.exe and Reg.

Adversaries may install a new service that can be configured to execute at startup by using utilities to interact with services or by directly modifying the Registry. The service can be disguised by using the name of another, legitimate program. Services may be created with administrator privileges but are executed under SYSTEM privileges, so an adversary can use a service to escalate privileges from administrator to SYSTEM. Services can also be executed directly.

Instead of creating a new service, an adversary can also modify an existing service to execute and persist the malicious payload. The usage of existing services is a type of masquerading that may make detection analysis more challenging. Modifying existing services could interrupt their functionality or enable services that are disabled or otherwise not commonly used.

Carbanak is a threat group that mainly targets banks. Their malware makes use of services to provide persistence and privilege escalation. An analysis on the Carbanak APT can be found here: https://securelist.com/files/2015/02/Carbanak_APT_eng.pdf.

Lazarus Group is a threat group that was responsible for the attack against Sony Pictures Entertainment. Several of their malware families install themselves as new services on victim machines.
A very detailed report on Operation Blockbuster, carried out by the Lazarus Group, can be found here: https://www.operationblockbuster.com/wp-content/uploads/2016/02/Operation-Blockbuster-Report.pdf

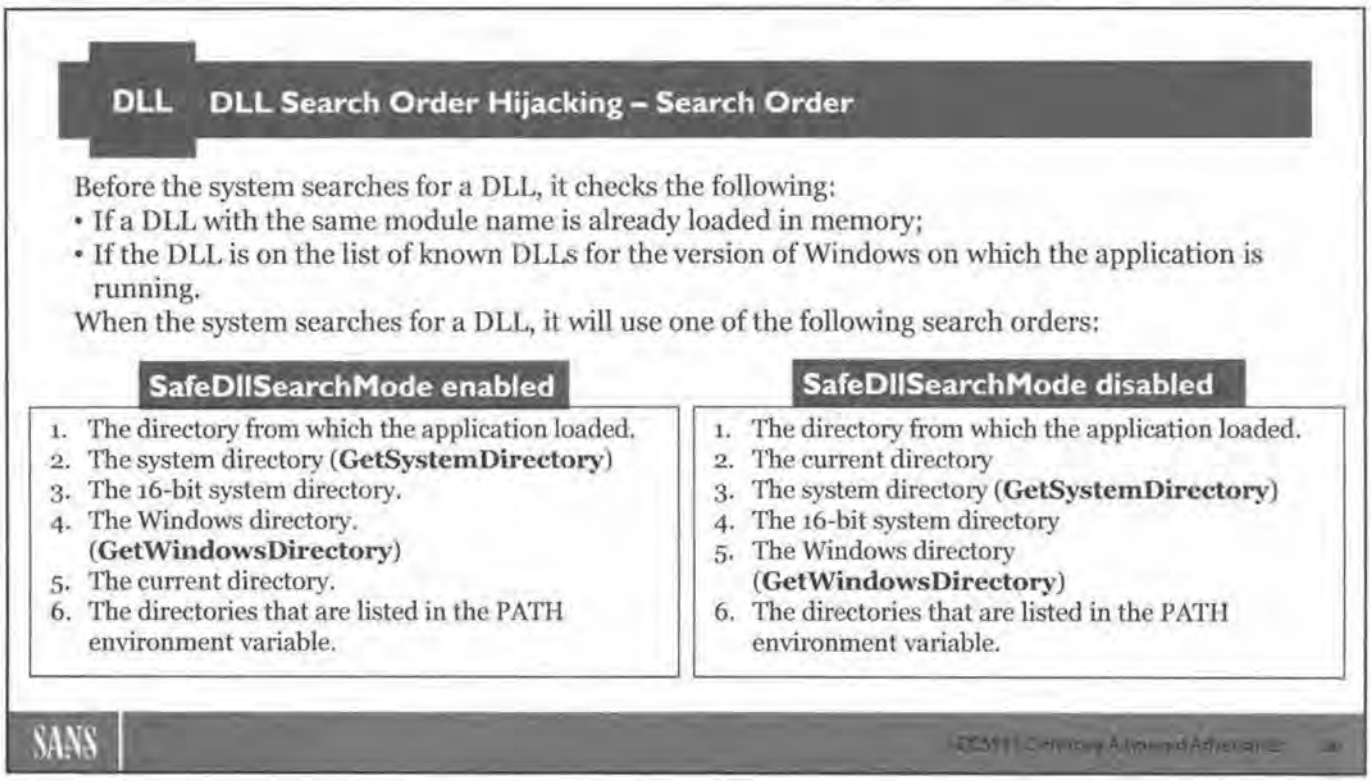


DLL Search Order Hijacking - Overview

Dynamic-link library (or DLL) is Microsoft's implementation of the shared library concept. Windows will follow a specific search order when a DLL has to be loaded. Adversaries can perform DLL preloading, by placing a malicious DLL with the same name as a legitimate DLL in a location that Windows searches first. That way, when Windows encounters the malicious DLL, it will be loaded instead of the legitimate one. Adversaries can also replace an existing DLL or modify a manifest file to cause another DLL to load. The malicious DLL can also be configured to load the legitimate DLLs they are meant to replace, which means the application keeps functioning as it normally would.

Operation Groundbait was mostly observed in Ukraine and used for surveillance. It made use of a malware family called Prikormka (which roughly translates to groundbait from Russian), which employed DLL search order hijacking for persistence by saving itself as ntshrui.dll to the Windows directory so it will load before the legitimate ntshrui.dll saved in the System32 subdirectory. Info on Operation Groundbait can be found here: https://www.welivesecurity.com/wp-content/uploads/2016/05/Operation-Groundbait.pdf

Another example is Downdelph, a first stage downloader that was (although rarely) used by APT28. It also made use of DLL search order hijacking, but targeted the Windows executable sysprep.exe, with the goal of escalating privileges. An analysis can be found here: https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part3.pdf



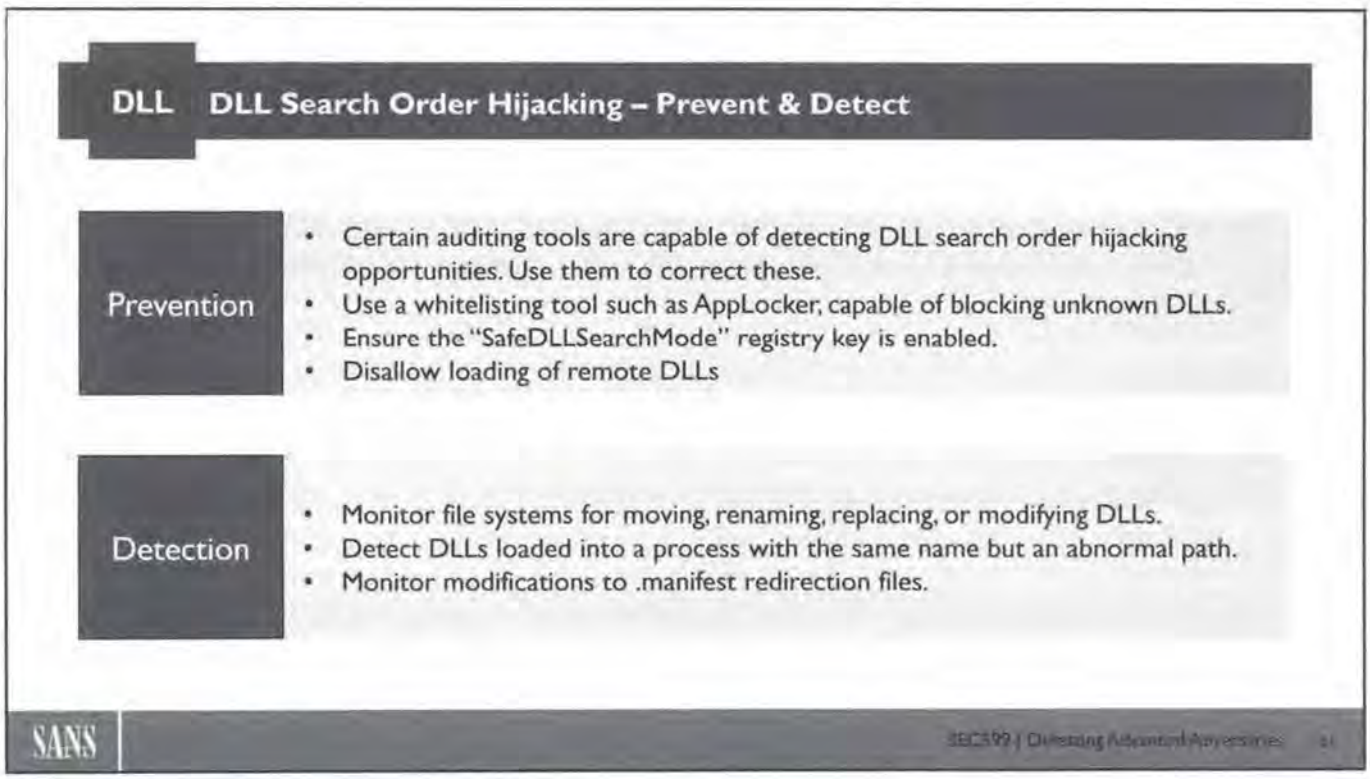
DLL Search Order Hijacking – Search Order
Before the system searches for a DLL, it checks the following:

- If a DLL with the same module name is already loaded in memory, the system uses the loaded DLL, no matter which directory it is in. The system does not search for the DLL.
- If the DLL is on the list of known DLLs for the version of Windows on which the application is running, the system uses its copy of the known DLL (and the known DLL's dependent DLLs, if any). The system does not search for the DLL. For a list of known DLLs on the current system, see the following registry key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs.

The standard DLL search order used by the system depends on whether safe DLL search mode is enabled or disabled. Safe DLL search mode places the user's current directory later in the search order.

If **SafeDllSearchMode** is enabled, the search order is as follows:
- The directory from which the application loaded.
- The system directory. Use the **GetSystemDirectory** function to get the path of this directory.
- The 16-bit system directory. There is no function that obtains the path of this directory, but it is searched.
- The Windows directory. Use the **GetWindowsDirectory** function to get the path of this directory.
- The current directory.
- The directories that are listed in the PATH environment variable. Note that this does not include the per-application path specified by the **App Paths** registry key. The **App Paths** key is not used when computing the DLL search path.

If **SafeDllSearchMode** is disabled, the search order is as follows:
- The directory from which the application loaded.
- The current directory.
- The system directory. Use the **GetSystemDirectory** function to get the path of this directory.
- The 16-bit system directory. There is no function that obtains the path of this directory, but it is searched.
- The Windows directory. Use **the GetWindowsDirectory** function to get the path of this directory.
- The directories that are listed in the PATH environment variable. Note that this does not include the per-application path specified by the **App Paths** registry key. The **App Paths** key is not used when computing the DLL search path.



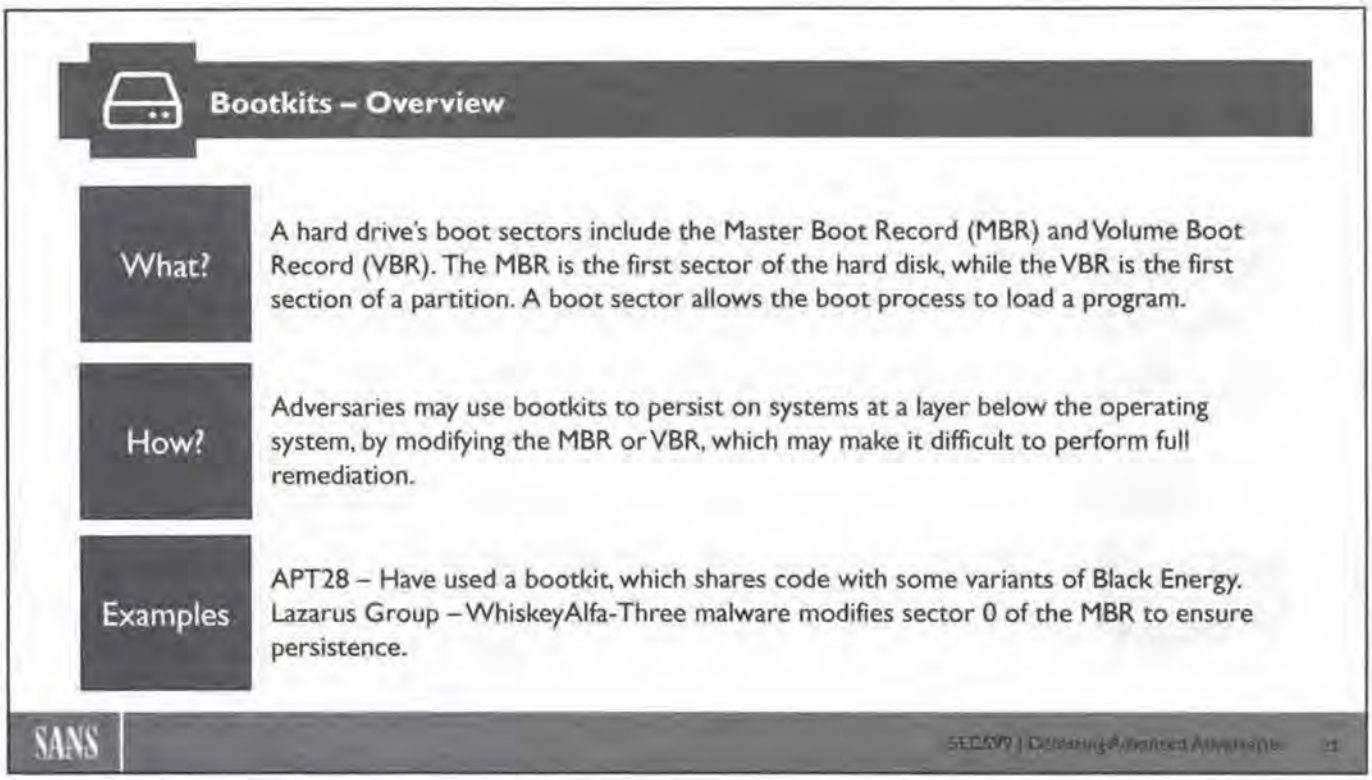
DLL Search Order Hijacking – Prevent & Detect

In order to prevent & detect DLL search order hijacking attacks there are a few controls we can consider:

- Certain auditing tools are capable of detecting DLL search order hijacking opportunities. You can use these tools to identify potential DLL search order issues and correct them. An example tool is Powerup (http://www.verisgroup.com/2014/06/17/powerup-usage/).
- Additionally, a whitelisting tool such as AppLocker, capable of blocking unknown DLLs, can be used as well.
- Finally, in order to reduce the opportunity of DLL search order hijacking vulnerabilities, ensure the "SafeDLLSearchMode" registry key is enabled.
- Disallow loading of remote DLLs, which could take place if an application sets its current directory to a folder on a share.

In order to detect DLL search order hijacking attacks, the following controls can be considered:

- Monitor file systems for moving, renaming, replacing, or modifying DLLs. Changes in the set of DLLs that are loaded by a process (compared with past behavior) that do not correlate with known software, patches, etc., are suspicious.
- Monitor DLLs loaded into a process and detect DLLs that have the same file name but abnormal paths.
- Modifications to or creation of .manifest and .local redirection files that do not correlate with software updates are suspicious.



Bootkits - Overview

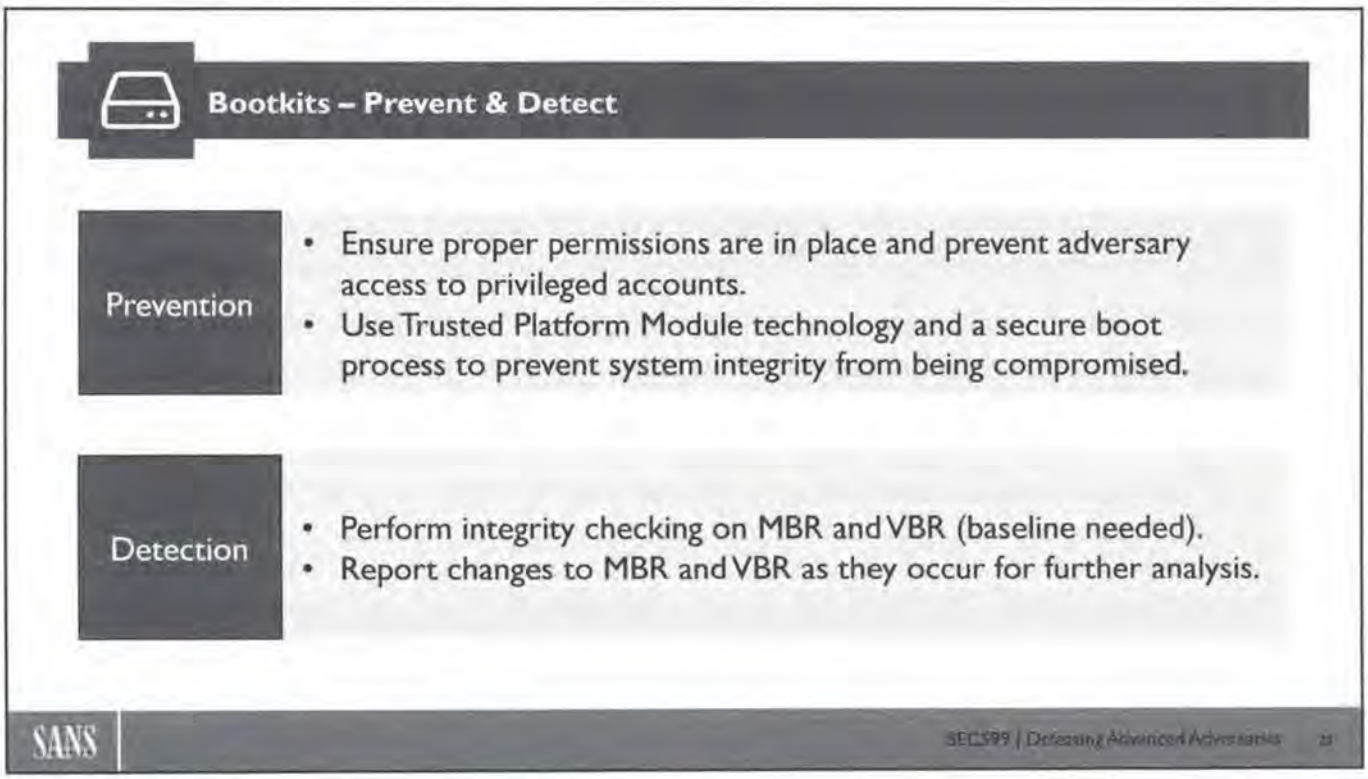
A boot sector is a region of a hard drive that contains machine code to be loaded into random-access memory (RAM) by a computer's built-in firmware. The purpose of a boot sector is to allow the boot process of a computer to load a program (usually, but not necessarily, an operating system) stored on the same storage device. A hard drive's boot sectors include the Master Boot Record (MBR) and/or Volume Boot Record (VBR):

- A Master Boot Record (MBR) is the first sector of a data storage device that has been partitioned. The MBR sector may contain code to locate the active partition and invoke its Volume Boot Record.
- A Volume Boot Record (VBR) is the first sector of a data storage device that has not been partitioned or the first sector of an individual partition on a data storage device that has been partitioned. It may contain code to load an operating system (or other standalone program) installed on that device or within that partition.

Adversaries may use bootkits to persist on systems at a layer below the operating system, by modifying the MBR or VBR, which may make it difficult to perform full remediation. The MBR is the section of disk that is first loaded after completing hardware initialization by the BIOS. An adversary who has raw access to the boot drive may overwrite this area, diverting execution during startup from the normal boot loader to adversary code. The MBR passes control of the boot process to the VBR. Similar to the case of MBR, an adversary who has raw access to the boot drive may overwrite the VBR to divert execution during startup to adversary code.

Once again, APT28 comes into play. They have used a bootkit that shares some of its code with variant of the Black Energy malware. More information here: https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part3.pdf

The Lazarus Group is no stranger to us either. Their WhiskeyAlfa-Three malware modifies sector 0 of the MBR to ensure persistence of their malicious programs. Detailed info on this (and other) malware can be found here: https://operationblockbuster.com/wp-content/uploads/2016/02/Operation-Blockbuster-Destructive-Malware-Report.pdf.



Bootkits – Prevent & Detect

Ensure proper permissions are in place to help prevent adversary access to privileged accounts necessary to perform modifications to the boot sectors. Use Trusted Platform Module (TPM) technology and a secure or trusted boot process to prevent system integrity from being compromised.

A TPM is a microcontroller that can be used to store platform measurements that help ensure that the platform remains trustworthy. The primary scope of a TPM is to assure the integrity. In this context "integrity" means "behave as intended", and a "platform" is generically any computer platform – not limited to PCs or a particular operating system: start the power-on boot process from a trusted condition and extend this trust until the operating system has fully booted and applications are running. A good summary on TPM is available here: https://www.trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-Summary_04292008.pdf

Perform integrity checking on MBR and VBR. To do this, take snapshots of MBR and VBR and compare against known good samples (i.e., a baseline). In case changes to the MBR or VBR have taken place, further analysis should be performed to determine malicious activity.

So... How Do We Prevent Persistence?

Although several different persistence techniques exist, the following recommendations will go a long way:

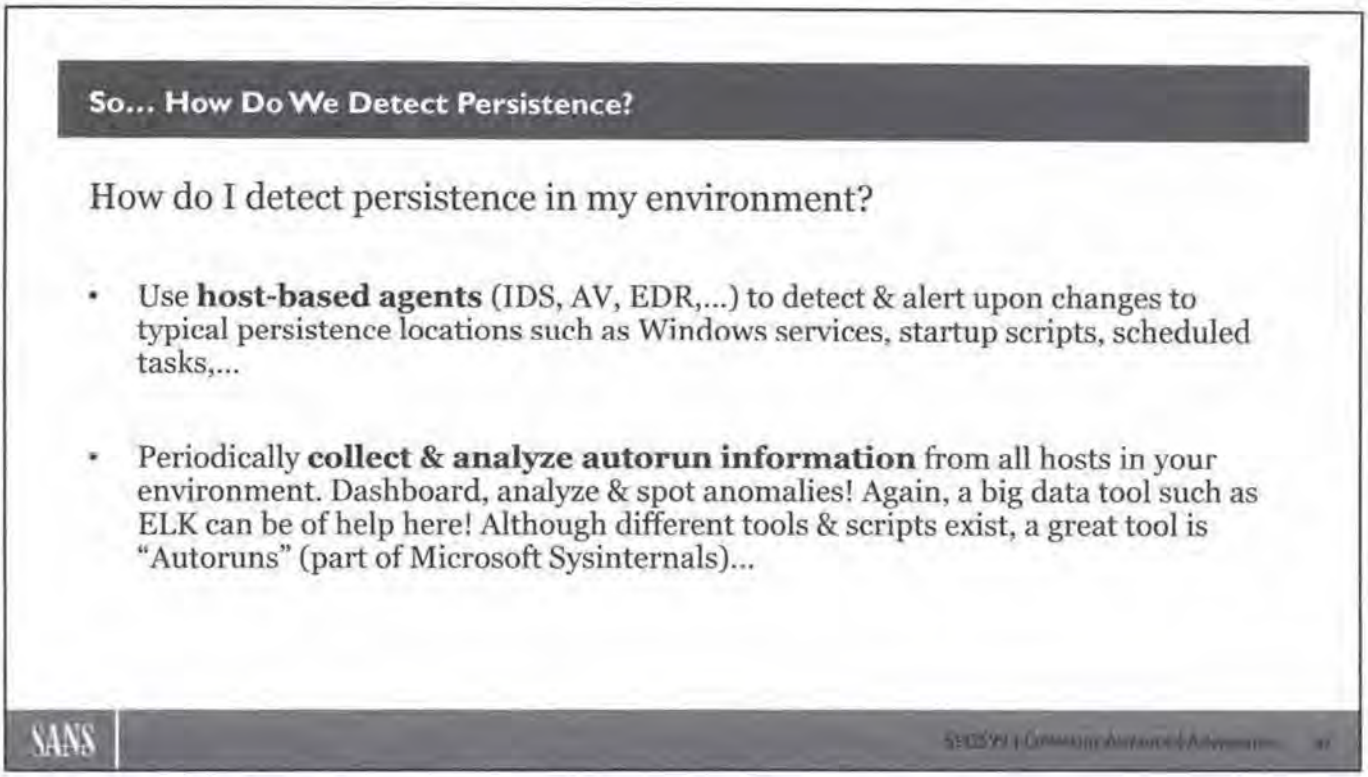
- **Limit user privileges** to prevent the installation of bootkits or Windows services;
- **Remediate vulnerabilities** that could allow persistence (e.g. DLL search order hijacking);
- **Block unneeded utilities or software** that could be used to schedule tasks (for example with AppLocker or software restriction policies).

Given the large number of techniques available to the adversary, we should not only focus on prevention, but should also assess how we can detect persistence in our environment...

So... How Do We Prevent Persistence?
Several different persistence techniques exist, some of which require a specific approach, which we have highlighted in the techniques above.
The following recommendations will, however, go a long way to prevent persistence from succeeding in your environment:

- Limit user privileges to prevent the installation of bootkits or Windows services. We will discuss this in more depth during the Privilege Escalation & Active Directory sections of this course;
- Remediate vulnerabilities that could allow persistence (e.g. DLL search order hijacking);
- Block unneeded utilities or software that could be used to schedule tasks (for example with AppLocker or software restriction policies).

As a number of different persistence opportunities exist and it will be difficult to deny an adversary to persist once he / she actually obtains access to an environment, we strongly advising taking efforts to detect persistence as well!



So... How Do We Detect Persistence?

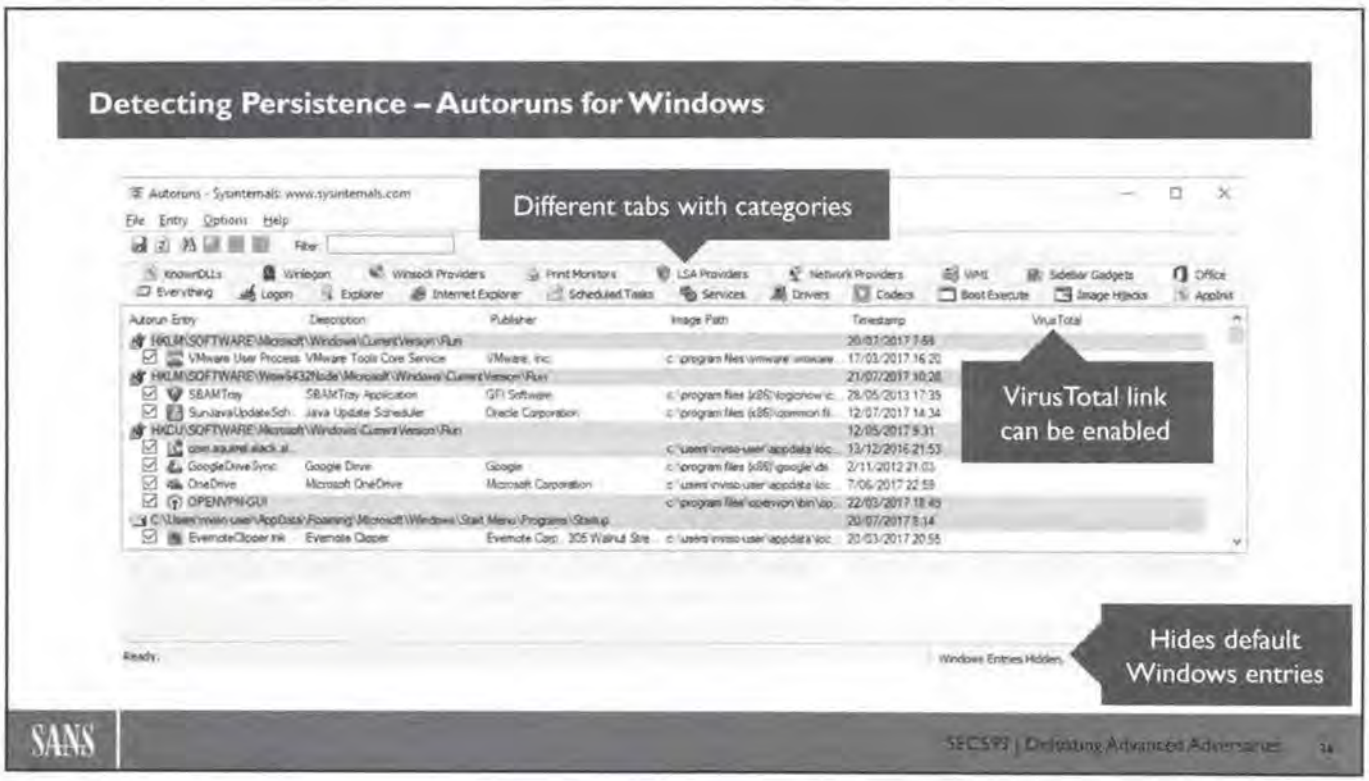
Although adversaries are continuously changing & adapting the way they penetrate our networks, they only have a number of options available for persistence. Two main strategies exist to detect persistence strategies in your environment:

Host-based agents (IDS, AV, EDR, ...) can help you detect & alert upon changes to typical persistence locations such as Windows services, startup scripts, scheduled tasks,...

Detecting persistence often relies on finding anomalies in your environment. Monitor scheduled task creation from common utilities using command-line invocation. Legitimate scheduled tasks could be created during installation of new software or through system administration functions, so beware of false positives. Monitor process execution from the Windows Task Scheduler, taskeng.exe, and changes to the Windows Task Scheduler stores (%systemroot%\System32\Tasks) for change entries related to scheduled tasks that do not correlate with known software, patch cycles, etc. Data and events should not be viewed in isolation but as part of a chain of behavior that could lead to other activities, such as network connections made for C&C, learning details about the environment through discovery, and lateral movement.

Tools such as Sysinternals Autoruns may also be used to detect system changes that could be attempts at persistence, including listing currently scheduled tasks. Look for changes to tasks that do not correlate with known software, patch cycles, etc. Suspicious program execution through scheduled tasks may show up as outlier processes that have not been seen before when compared against historical data.

Monitor processes and command-line arguments for actions that could be taken to create tasks. Remote access tools with built-in features may interact directly with the Windows API to perform these functions outside of typical system utilities. Tasks may also be created through Windows system management tools such as Windows Management Instrumentation (WMI) and PowerShell, so additional logging may need to be configured to gather the appropriate data.



Detecting Persistence – Autoruns for Windows

Autoruns has "the most comprehensive knowledge of auto-starting locations of any startup monitor". It attempts to show you a full overview of what programs or scripts are configured to run during system boot or user login or when built-in Windows applications such as Explorer or Internet Explorer starts. Some of the example locations it monitors include:

- Run, RunOnce and other Registry keys
- Explorer shell extensions
- Scheduled tasks
- Auto-start services
- ...

Autoruns is by default configured to hide default, known, Windows entries, providing you with an in-depth view of what is executed once the computer boots.

Persistence – Additional References

Some additional resources concerning persistence mechanisms:

- An extensive list of persistence mechanisms can be found here:
  https://attack.mitre.org/wiki/Persistence
- SANS DFIR blog post on DLL hijacking
  https://digital-forensics.sans.org/blog/2015/03/25/detecting-dll-hijacking-on-windows/
- Autoruns for Windows
  https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

Persistence – Additional References
Some additional resources concerning persistence mechanisms:

- An extensive list of persistence mechanisms can be found here:
  https://attack.mitre.org/wiki/Persistence
- SANS DFIR blog post on DLL hijacking
  https://digital-forensics.sans.org/blog/2015/03/25/detecting-dll-hijacking-on-windows/
- Autoruns for Windows
  https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

Exercise – Catching Persistence

The objective of the lab is to detect persistence strategies implemented on one of our Windows machines! The high-level steps of the exercise are listed below, but feel free to refer to the LODS workbook for additional information.

High-level exercise steps:

1. Run autoruns & Malwarebytes Anti-Rootkit on our Windows system
2. Analyze the output & identify the malicious persistence mechanism
3. Create a GPO to run autoruns periodically on all Windows systems
4. *Optional: Dashboard the autoruns output in ELK stack for baselining*

Exercise – Catching Persistence

The objective of the lab is to detect a number of persistence strategies implemented on one of our Windows machines! The high-level steps of the exercise are listed below, but feel free to refer to the LODS workbook for additional information. Throughout the exercise, you will complete the following high-level steps:

1. Run autoruns & Malwarebytes Anti-Rootkit on our Windows system
2. Analyze the output & identify the malicious persistence mechanism
3. Create a GPO to run autoruns periodically on all Windows systems
4. Optional: Dashboard the autoruns output in ELK stack for baselining

For additional guidance & details on the lab, please refer to the LODS workbook.

Exercise – Detecting Persistence – Conclusions

During this exercise, we used two different tools to try catching persistence in our environment. Some conclusions from this exercise:

Autoruns and Malwarebytes Anti-Rootkit are both useful tools that can help us detect typical persistence strategies implemented by malware

elastic

An interesting, enterprise-compatible, approach would be to script these tools and have them run on a periodic basis (e.g. once every week). Its results could then be dash-boarded, after which anomalies / outliers can be detected.

Exercise – Detecting Persistence – Conclusions

During this exercise, we used two different tools to try catching persistence in our environment. Some conclusions from this exercise:

- Autoruns and Malwarebytes Anti-Rootkit are both useful tools that can help us detect typical persistence strategies implemented by malware;
- An interesting, enterprise-compatible, approach would be to script these executables and have them run on a periodic basis (e.g. once every week). Its results could then be dash-boarded, after which anomalies / outliers can be detected;
- Once anomalies are detected, they can be further investigated (e.g. suspicious binaries can be analyzed to assess how they behave).

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

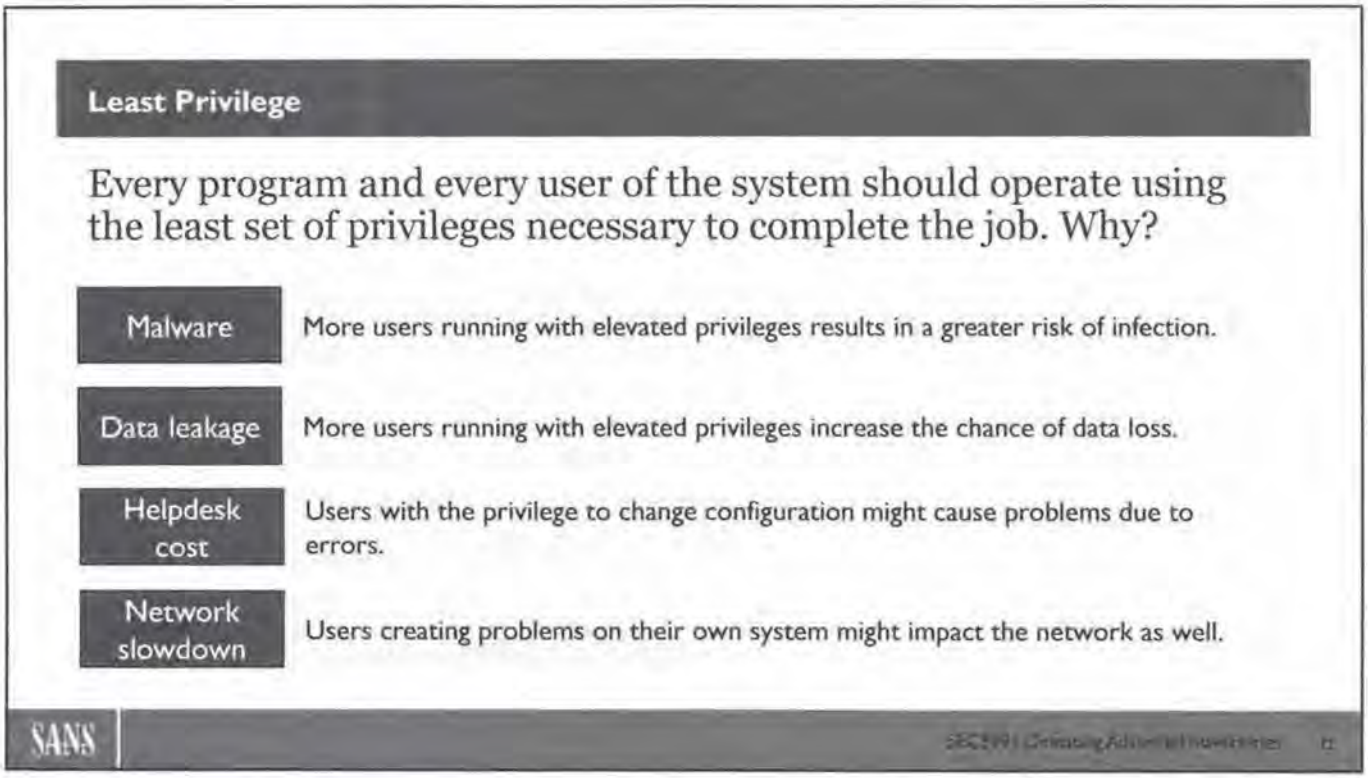
SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.



Least Privilege

The principle of least privilege could be defined as follows: "Every program and every user of the system should operate using the least set of privileges necessary to complete the job." - Saltzer and Schroeder. Granting permissions to a user beyond the scope of the necessary rights of an action can allow that user to obtain or change information in unwanted ways. Some specific examples of why implementing least privilege in your organization are listed below.

First of all, there's a greater risk of malware infection. Allowing users to download and/or install any application they want means they are allowed to install potential malware. Not only malware downloaded from the internet, but also malware making use of a USB key as an infection vector could be installed.

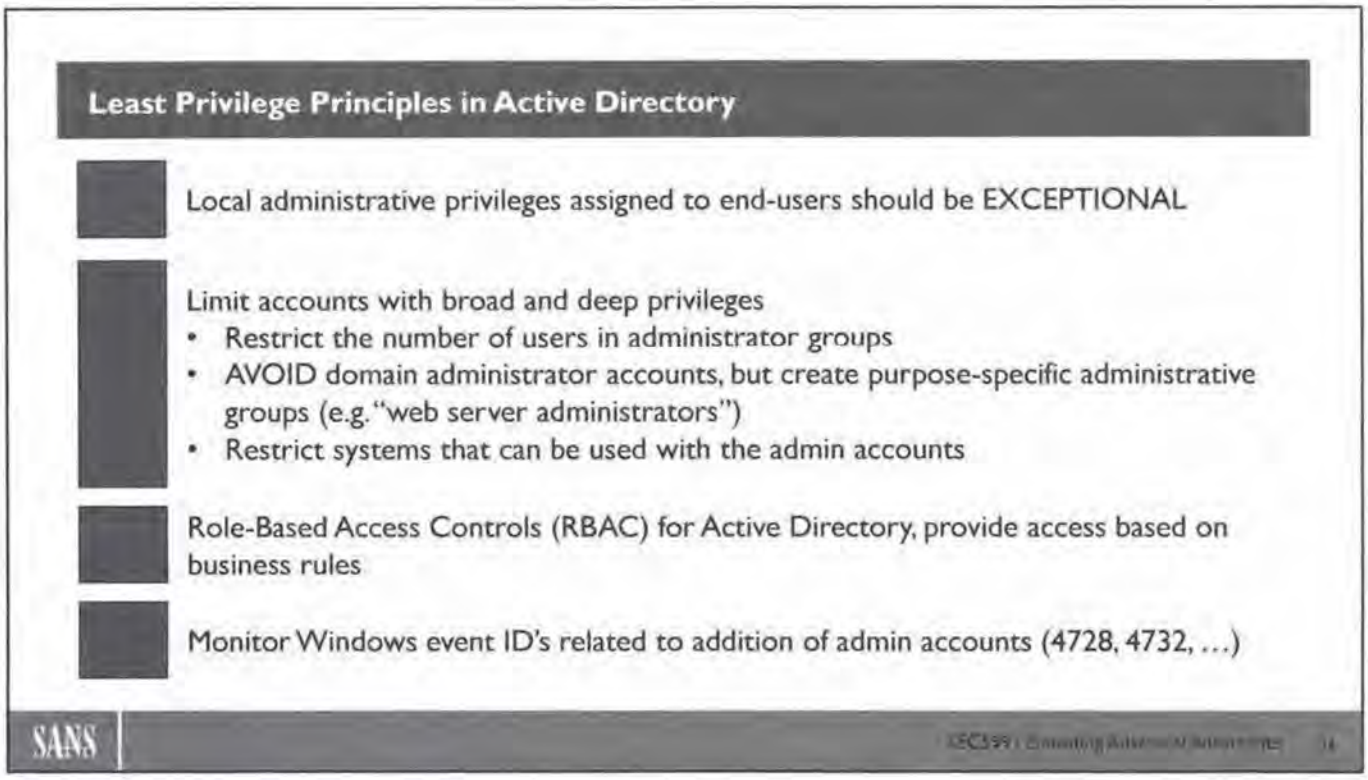
In attacks in which the target is an organization's intellectual property, accounts that have been granted powerful privileges within applications can be targeted to allow exfiltration of data. Although the accounts that have access to sensitive data may have been granted no elevated privileges in the domain or the operating system, accounts that can manipulate the configuration of an application or access the information the application provides present an increased risk. Additionally, adversaries could target the accounts that control access to the file stores, accounts that have direct access to the files, or even groups or roles that have access to the files. For example, if a file server is used to store contract documents and access is granted to the documents by the use of an Active Directory group, an adversary who can modify the membership of the group can add compromised accounts to the group and access the contract documents. In cases in which access to documents is provided by applications such as SharePoint, adversaries can target the applications as described earlier.

Another potential issue is the increased cost of helpdesk and IT support. If users have the privileges to change system configurations, they might make changes that result in malfunctions. Be it accidentally, or on purpose (i.e., they think they know what they are doing), various things could go wrong, such as files or shares that are suddenly no longer accessible, network connection issues, etc. This could result in an increased number of calls to the IT helpdesk.

The issue might not just be limited to their own system, but could result in network issues for other users as well. For example, when changing properties of a networked file share, other users might lose their ability to view or modify files and thus have their ability to perform their work reduced.

References:

- http://windowsitpro.com/blog/least-privilege-security-windows-7-vista-and-xp
- https://www.us-cert.gov/bsi/articles/knowledge/principles/least-privilege



Least Privilege Principles in Active Directory

Unfortunately, the path of least resistance in many environments has proven to be the overuse of accounts with broad and deep privilege. Broad privileges are rights and permissions that allow an account to perform specific activities across a large cross-section of the environment, for example, an account that is administrator on all servers.

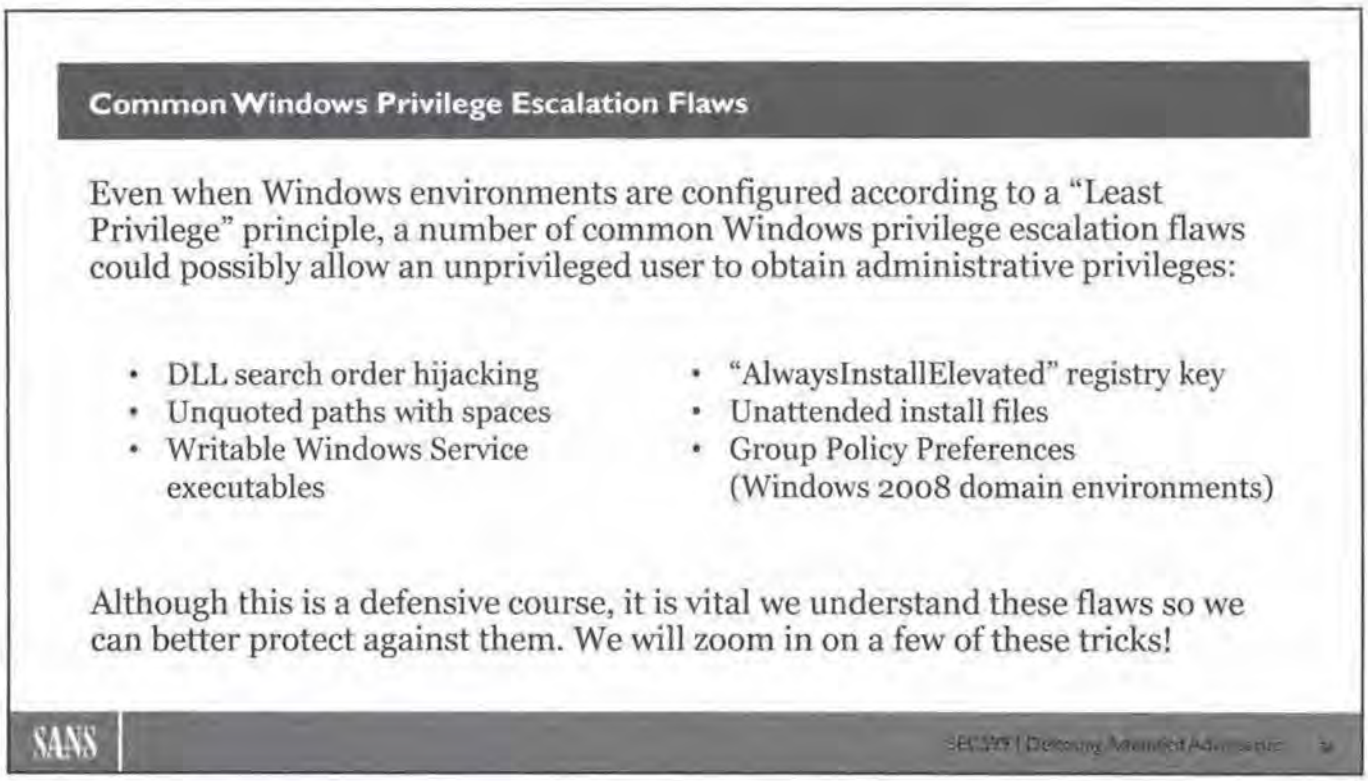
You should consider carefully whether users require administrative rights on their workstations, and if they do, a better approach may be to create a separate local account on the computer that is a member of the Administrators group. When users require elevation, they can present the credentials of that local account for elevation, but because the account is local, it cannot be used to compromise other computers or access domain resources. As with any local accounts, however, the credentials for the local privileged account should be unique; if you create a local account with the same credentials on multiple workstations, you expose the computers to pass-the-hash attacks. Securing local admin accounts: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/appendix-h--securing-local-administrator-accounts-and-groups

Deep privileges are powerful privileges that are applied to a small number of users, such as giving an engineer full administrator rights on a certain server. Neither broad privilege nor deep privilege is necessarily dangerous, but when many accounts in the domain are permanently granted broad and deep privilege, an adversary can easily compromise one account and use it to move laterally through the network and potentially compromise additional accounts. We will see some examples later this day, during the "attacks on AD" section. Unsurprisingly, the number of administrators should be limited, along with the number of systems where on which admins can use their accounts.

Generally speaking, role-based access controls (RBAC) are a mechanism for grouping users and providing access to resources based on business rules. In the case of Active Directory, implementing RBAC for AD is the process of creating roles to which rights and permissions are delegated to allow members of the role to perform day-to-day administrative tasks without granting them excessive privilege. RBAC for Active Directory can be designed and implemented via native tooling and interfaces, by leveraging software you may already own, by purchasing third-party products, or any combination of these approaches.

Finally, it's a good idea to monitor your Windows environment for the following interesting event IDs:

- 4728: A member was added to a security-enabled global group (for administrative groups in the domain)
- 4732: A member was added to a security-enabled local group (for local administrative groups)

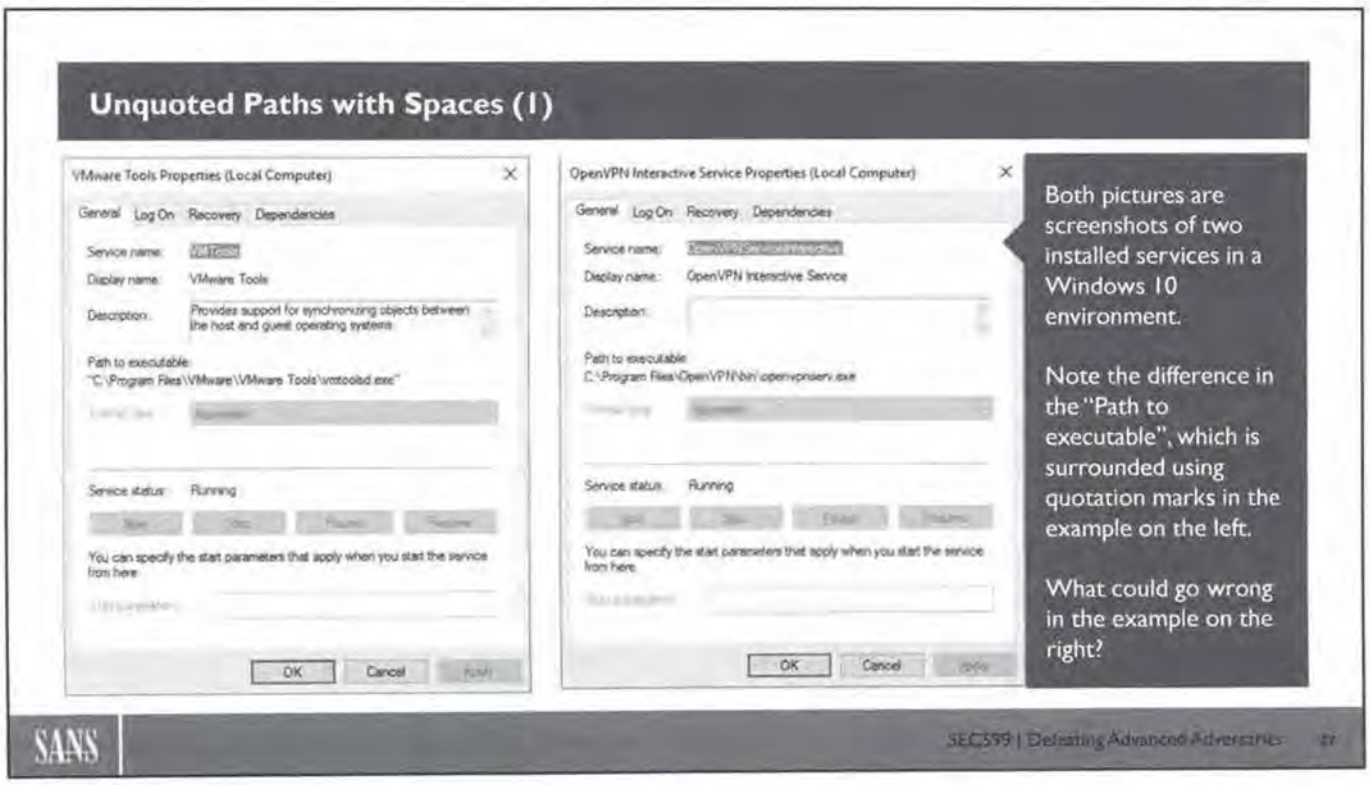


Common Windows Privilege Escalation Flaws

Even when Windows environments are configured according to a "Least Privilege" principle and the number of administrative accounts is restricted, a number of common Windows privilege escalation flaws could possibly allow an unprivileged user to obtain administrative privileges. Local privilege escalation issues could be caused by outdated / unpatched software that exposes the core OS to vulnerabilities. More often, however, they are the result of a number of misconfigurations or mistakes that adversaries can abuse. The below is a non-exhaustive list of some commonly used attack strategies:

- DLL search order hijacking (which we discussed already in the persistence section of this material);
- Unquoted paths with spaces
- Writable Windows Service executables
- "AlwaysInstallElevated" registry key
- Unattended install files
- Group Policy Preferences (Windows 2008 domain environments)

Although this is a defensive course, it is vital we understand these flaws so we can better protect against them. We will zoom in on a few of these tricks!



Unquoted Paths with Spaces (1)

Unquoted paths with spaces can be a serious issue in Windows Services configurations. Let's analyze an example:

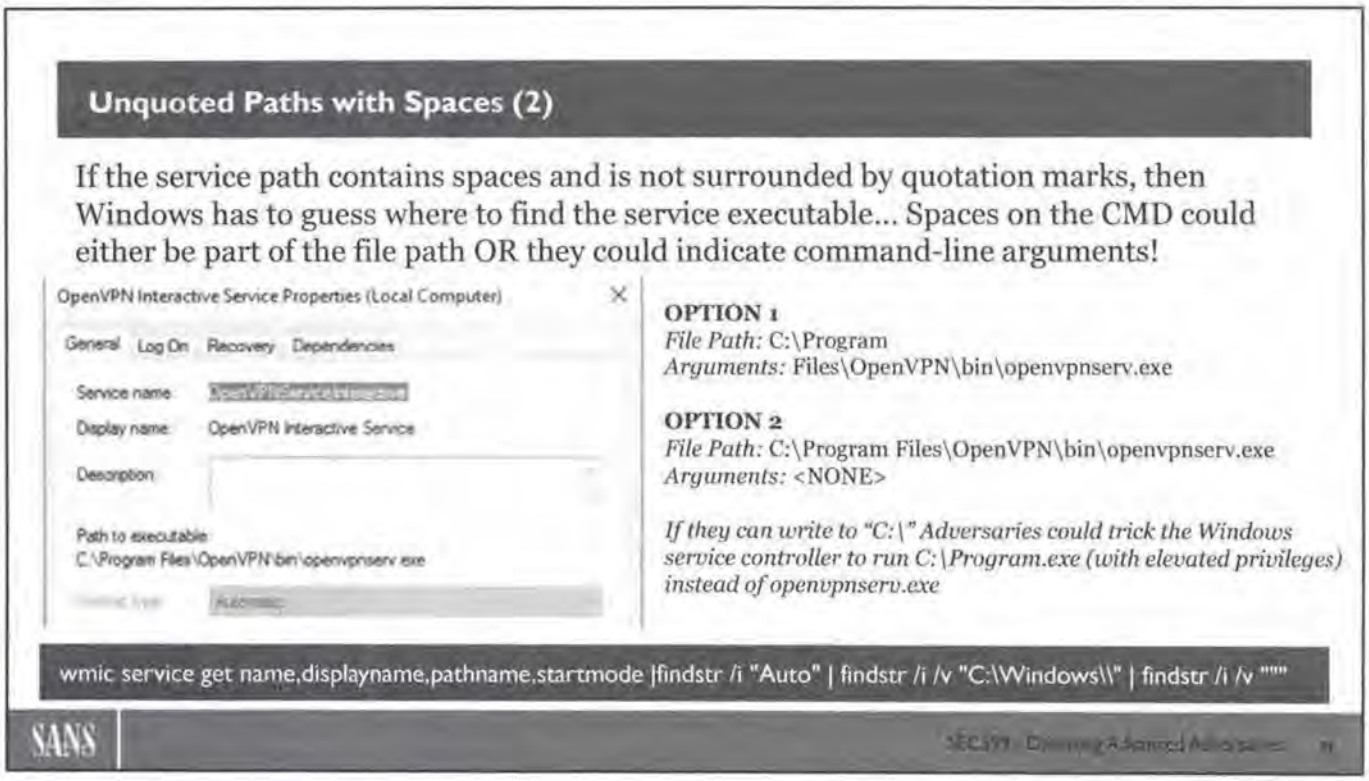
- The picture on the left is a screenshot of a recent VMWare Tools service installed on a Windows 10 host. Note the **Path to executable**, which is set to:

"C:\Program Files\Vmware\Vmware Tools\vmtoolsd.exe"

- The picture on the right is a screenshot of a recent OpenVPN installed on a Windows 10 host. Note the **Path to executable**, which is set to:

C:\Program Files\OpenVPN\bin\openvpnserv.exe

The Vmware Tools service is clearly **NOT VULNERABLE** to the issue, as the executable path is properly delimited using quotes. The OpenVPN service, however, raises some questions… Can you think of what could go wrong?



Unquoted Paths with Spaces (2)
If the service path contains spaces and is not surrounded by quotation marks, then Windows has to guess where to find the service executable… Spaces on the CMD could either be part of the file path OR they could indicate command-line arguments!

So, let's have a look at that OpenVPN service… Windows tries to start this service "automatically" (so on boot). But how does Windows try to start the service? It, of course, tries to load the "executable path". But, how is it interpreted? Given the space in the path name, there's an ambiguity here:

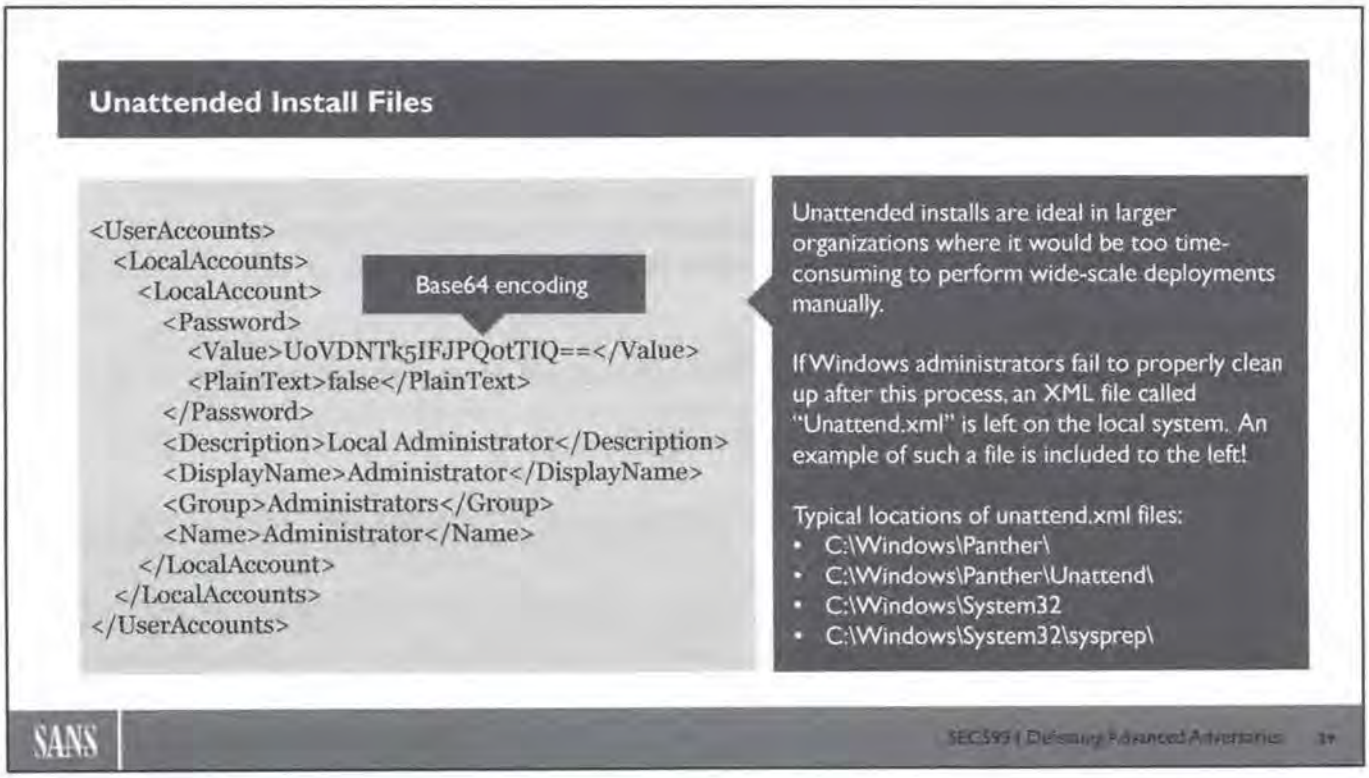
OPTION 1
Logic: Windows considers the space at the end of the filename and assumes everything that follows are arguments passed to that executable.
File Path: C:\Program
Arguments: Files\OpenVPN\bin\openvpnserv.exe

OPTION 2
Logic: The excepted logic, Windows considers the space as part of the file path and finds the correct executable.
File Path: C:\Program Files\OpenVPN\bin\openvpnserv.exe
Arguments: <NONE>

Windows fill first attempt option 1. If the adversary could thus write to "C:\", he could create a program called "Program.exe" that would subsequently be executed with the privileges of the OpenVPN Windows service. In this specific case, there is no privilege escalation as an unprivileged user cannot write to "C:\" either.

An excellent command line to find these types of services on our own machine is below (originally posted by Danial Compton):

wmic service get name,displayname,pathname,startmode |findstr /i "Auto" | findstr /i /v "C:\Windows\\" | findstr /i /v """



Unattended Install Files

Unattended installs are often used in enterprise organizations where it would be too time-consuming to perform wide-scale deployments manually. If Windows administrators fail to properly clean up after this process, an XML file called "Unattend.xml" is left on the local system. An example of such a file is included to the left!

As you can see, it includes the password in a base64 encoded format, which means it can be very easily decoded. Now, where can you find these xml files? It depends… Some good candidates to check include:

- C:\Windows\Panther\
- C:\Windows\Panther\Unattend\
- C:\Windows\System32
- C:\Windows\System32\sysprep\

Keep an eye out on your environment and search for these files periodically… Upon discovery, they should be immediately deleted!

Group Policy Preferences (GPP) (1)

GPP's were used to allow administrators to create domain policies with embedded credentials. These policies allowed administrators to for example change local accounts or embed credentials for the purposes of mapping drives.

While highly useful, the storage mechanism used for such credentials is insecure: The GPP's are stored in XML files on the SYSVOL (Windows domain share accessible to all domain users) and the password is stored encrypted with a known 32-byte AES key (it was published by Microsoft in an MSDN article)

Although this vulnerability was addressed by Microsoft in MS14-025, existing GPP's with passwords were not removed (this has to be performed manually....)

Group Policy Preferences (GPP) (1)
GPP's were used to allow administrators to create domain policies with embedded credentials. These policies allowed administrators to for example change local accounts or embed credentials for the purposes of mapping drives.

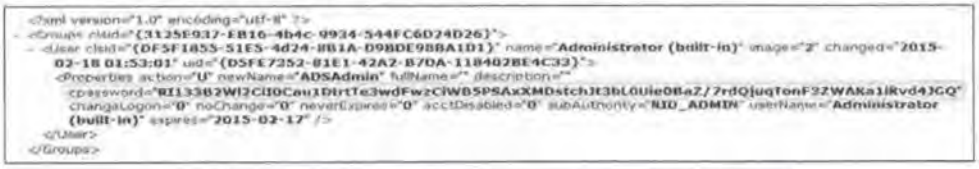
While highly useful, the storage mechanism used for such credentials is insecure: The GPP's are stored in XML files on the SYSVOL share (Windows domain share accessible to all domain users) and the password is stored encrypted with a known 32-byte AES key (it was published by Microsoft in an MSDN article - https://msdn.microsoft.com/en-us/library/cc422924.aspx).

Although this vulnerability was addressed by Microsoft in MS14-025, existing GPP's with passwords were not removed (this has to be performed manually...)

Group Policy Preferences (GPP) (2)

In order to find these passwords in your own environment, you could run the following from any domain-authenticated user session:

findstr /S cpassword %LOGONSERVER%\sysvol*.xml



The screenshot above shows the XML file when it would be opened using a normal text editor. Note the value for the "cpassword". On the left we can see the encryption key used by Microsoft, which is published in an MSDN article. Finally, on the right we see the GPP Metasploit module automatically extracting & decrypting the password…

SANS | SEC599 | Defeating Advanced Adversaries 41

Group Policy Preferences (GPP) (2)
In order to find these passwords in your own environment, you could run the following from any domain-authenticated user session:

findstr /S cpassword %LOGONSERVER%\sysvol*.xml

This one-line command will search for the string "cpassword" in any .xml file under the domain controller's publicly accessible sysvol network share.

The screenshots on the slide provide some additional context:

- The first screenshot shows an identified XML file with a cpassword value when it would be opened using a normal text editor. Note the value for the "cpassword";
- On the bottom left we can see the encryption key used by Microsoft, which is published in an MSDN article;
- Finally, on the bottom right we see the GPP Metasploit module automatically extracting & decrypting the password…

Next to Metasploit modules, several PowerShell scripts exist that will do the same thing, which of course reduces the detection rate of such tools.

Introducing BeRoot.exe & PowerUp

So... How can we check our own environments for such vulnerabilities? One of the most effective ways of doing so is to leverage pentester tools for that! Some examples:

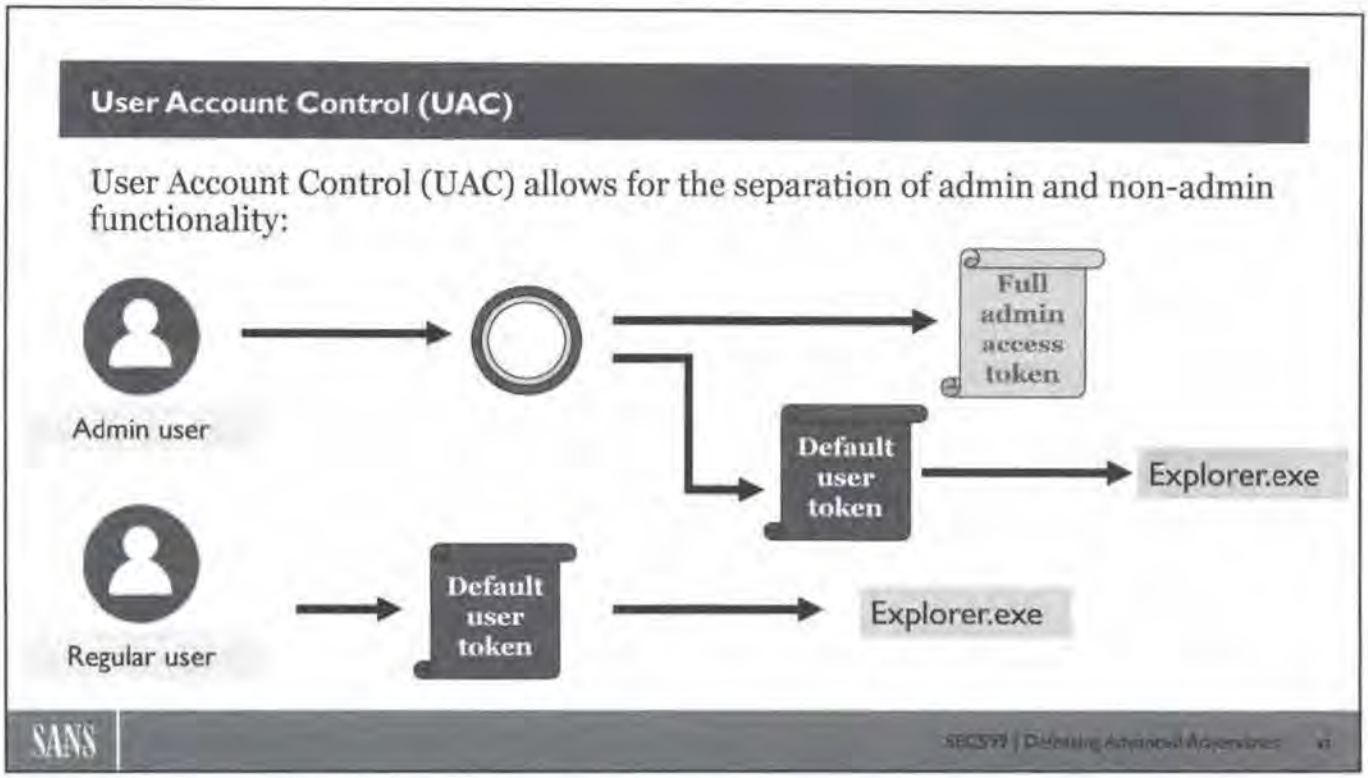
BeRoot is a post exploitation tool to check common Windows misconfigurations to find a way to escalate our privilege;

PowerUp (now part of Empire) is a pure-PowerShell script that will use the techniques mentioned above (and much more) to try to escalate privileges!

Introducing BeRoot.exe & PowerUp

In order to better protect ourselves against these typical privilege escalation attacks, it's a good idea to use tooling that allows us to easily assess how vulnerable we are, so we can fix any identified vulnerabilities. Two tools are very easy to run and use:

- BeRoot is a post-exploitation tool to check common Windows misconfigurations. It can be downloaded as a stand-alone binary. Upon running, it will attempt any privilege escalation techniques and will return a result.
- PowerUp is a pure-PowerShell script that will use the techniques mentioned above (and much more) to try to escalate privileges. PowerUp used to be a stand-alone PowerShell script, but it's now been included in the Empire PowerShell post-exploitation framework.

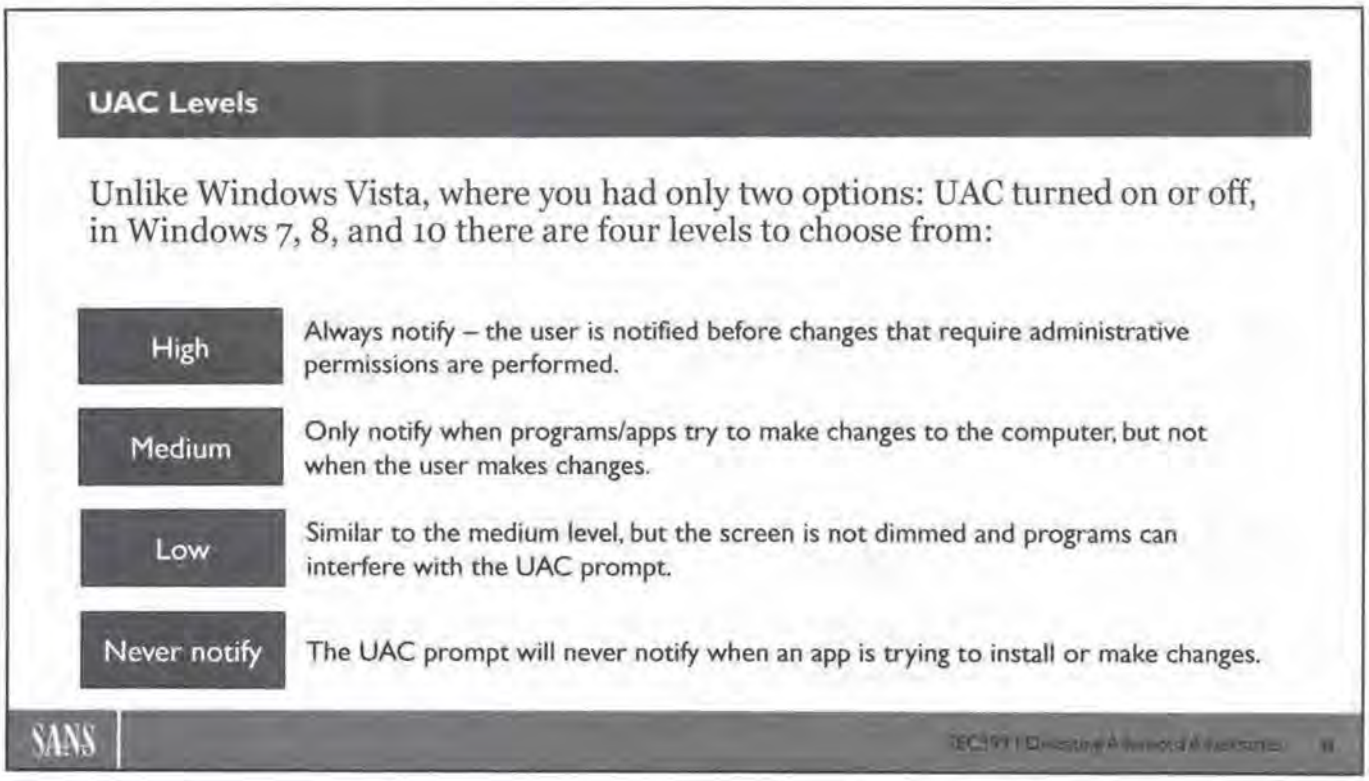


User Account Control (UAC)

Microsoft has continued to improve the security in each version of Windows and as a result, have reduced the operating system's attack surface. One of the most common and widespread security holes has been the granting of administrator privileges, to otherwise standard users, often to resolve application and configuration issues. User Account Control (UAC) is a security component that enables users to perform common tasks as non-administrators or as administrators without having to switch users, log off, or use Run As. User accounts that are members of the local Administrators group run most applications as a standard user. By separating user and administrator functions, UAC helps users move toward using standard user rights by default. So, if a user is logged on as a local administrator, then UAC disables a user's administrator rights and prompts the user when their administrator rights are required. If a user logs on as a standard user they are asked to provide the credentials of an administrator account if they attempt to perform any task that requires administrator rights. Ultimately the user has control over when to use administrator privileges as UAC does not have the ability to be controlled via centralized policy.

By default, standard users and administrators access resources and run apps in the security context of standard users. When a user logs on to a computer, the system creates an access token for that user. The access token contains information about the level of access that the user is granted, including specific security identifiers (SIDs) and Windows privileges. With the built-in UAC elevation component, standard users can easily perform an administrative task by entering valid credentials for a local administrator account. The default, built-in UAC elevation component for standard users is the credential prompt.

When an administrator logs on, two separate access tokens are created for the user: a standard user access token and an administrator access token. The standard user access token contains the same user-specific information as the administrator access token, but the administrative Windows privileges and SIDs are removed. The standard user access token is used to start apps that do not perform administrative tasks (standard user apps). The standard user access token is then used to display the desktop (explorer.exe). Explorer.exe is the parent process from which all other user-initiated processes inherit their access token. As a result, all apps run as a standard user unless a user provides consent or credentials to approve an app to use a full administrative access token.



UAC Levels

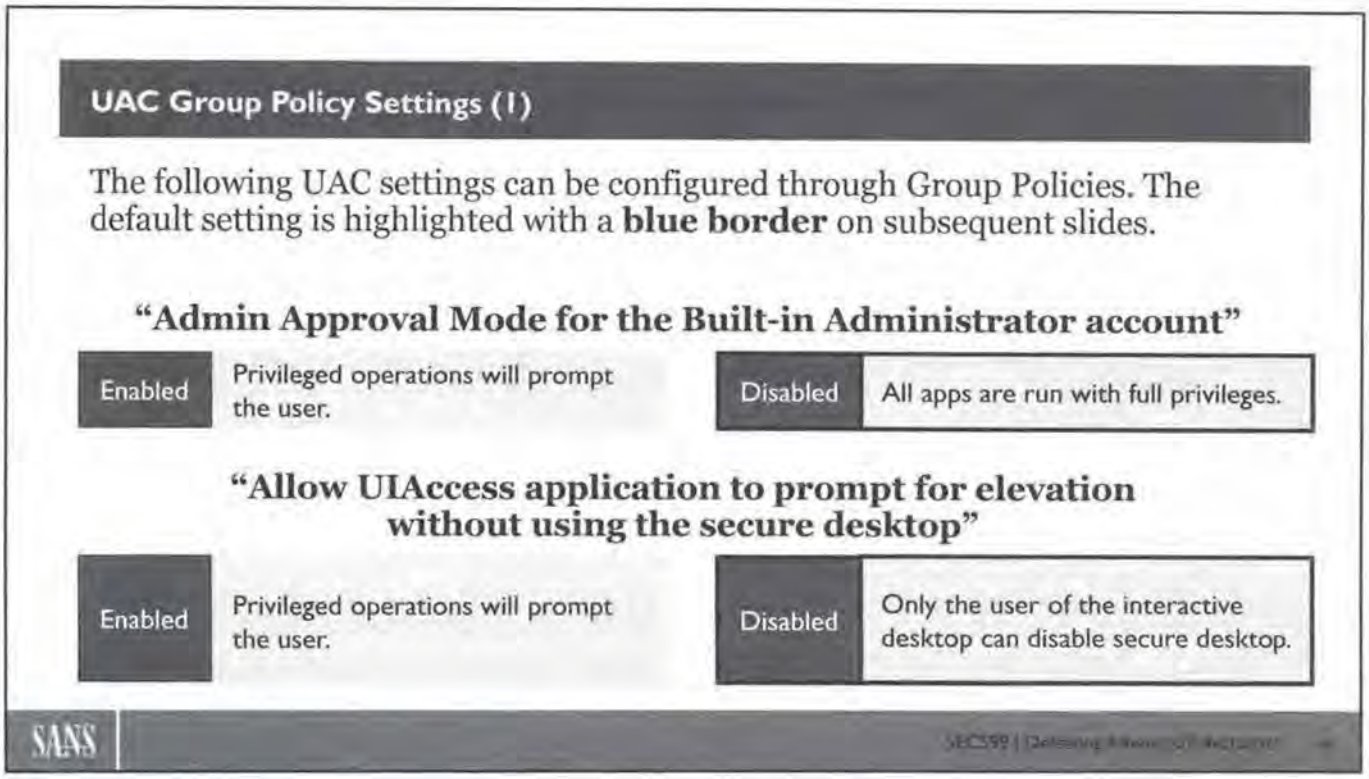
Unlike Windows Vista, where you had only two options: UAC turned On or Off, in Windows 7, 8, and 10 there are four levels to choose from. The differences between them are the following:

- High: Always notify - at this level you are notified before applications and users make changes that require administrative permissions. When a UAC prompt shows up, the desktop is dimmed as shown in the screenshot below. You must choose Yes or No before you can do anything else on the computer. Security Impact: this is the most secure setting and the most intruding as well.

- Medium: Notify me only when programs/apps try to make changes to my computer - this is the default level and UAC notifies you only before programs make changes that require administrative permissions. If you manually make changes to Windows, then a UAC prompt is not shown. This level is less intruding as it doesn't stop the user from making changes to the system, it only shows prompts if an application wants to make changes. When a UAC prompt is shown, the desktop is dimmed and you must choose Yes or No before you can do anything else on your computer. Security Impact: this is less secure than the first setting because malicious programs can be created to simulate the keystrokes or mouse movements made by a user and change Windows settings.

- Low: Notify me only when programs/apps try to make changes to my computer (do not dim my desktop) - this level is identical to the one above except the fact that, when a UAC prompt is shown, the desktop is not dimmed and other programs are able to interfere with it. Security Impact: this level is even less secure as it makes it easy for malicious programs to simulate keystrokes or mouse moves that interfere with the UAC prompt.

- Never notify: At this level, UAC is turned off and it doesn't offer any protection against unauthorized system changes. Security Impact: if you don't have a good security solution you are very likely to encounter security issues with your PC. With UAC turned off it is much easier for malicious programs to infect your computer and take control.

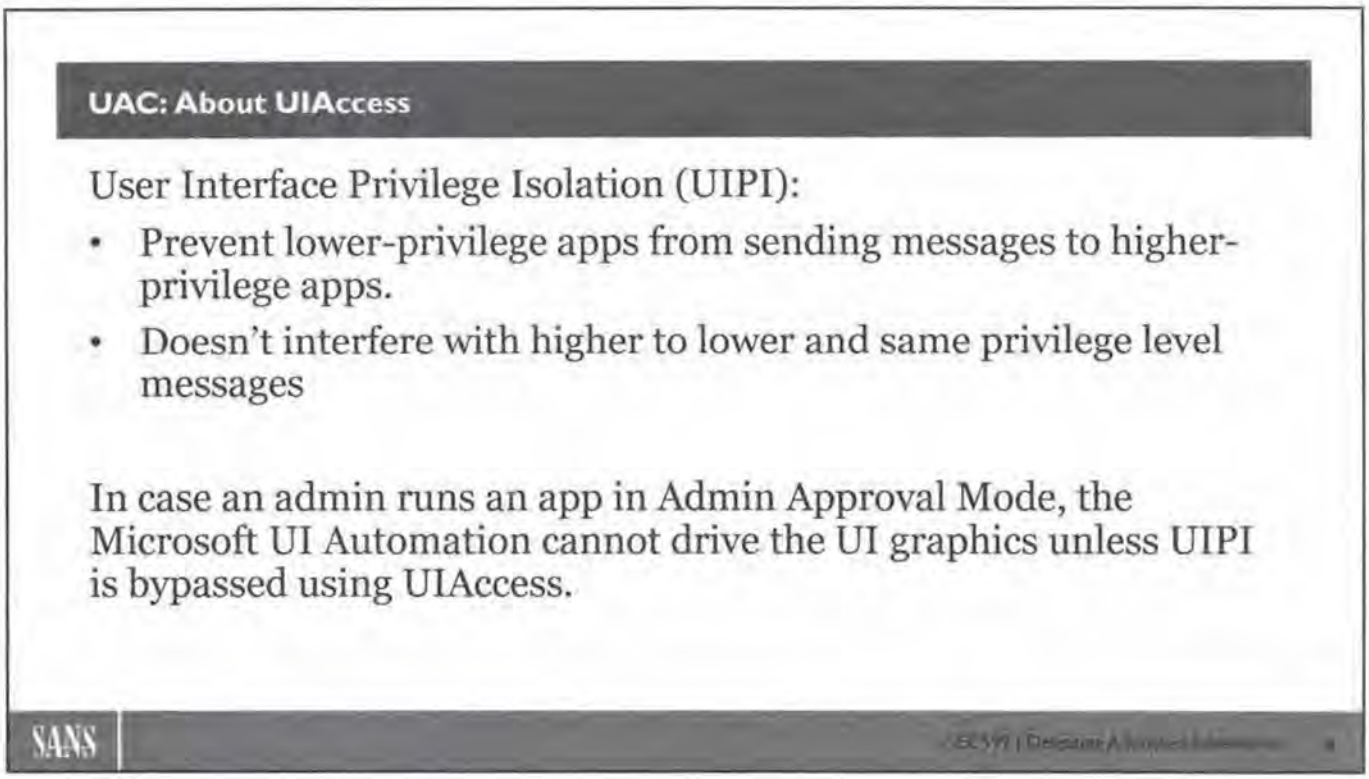


UAC Group Policy Settings (1)
You can use security policies to configure how User Account Control works in your organization. They can be configured locally by using the Local Security Policy snap-in (secpol.msc) or configured for the domain, OU, or specific groups by Group Policy. In every slide, the default setting is marked with a blue border.

User Account Control: Admin Approval Mode for the Built-in Administrator account
This policy setting controls the behavior of Admin Approval Mode for the built-in Administrator account.
Enabled - The built-in Administrator account uses Admin Approval Mode. By default, any operation that requires elevation of privilege will prompt the user to approve the operation.
Disabled (Default) - The built-in Administrator account runs all applications with full administrative privilege.

Admin Approval Mode (AAM) is a UAC configuration in which a split user access token is created for an administrator. When an administrator logs in, the user is assigned two separate access tokens. Without AAM, an administrator account receives only one access token, which grants that administrator access to all Windows resources. As it is disabled by default, it is recommended to enable this option.

User Account Control: Allow UIAccess application to prompt for elevation without using the secure desktop
This policy setting controls whether User Interface Accessibility (UIAccess or UIA) programs can automatically disable the secure desktop for elevation prompts used by a standard user.
Enabled - UIA programs, including Windows Remote Assistance, automatically disable the secure desktop for elevation prompts. If you do not disable the "User Account Control: Switch to the secure desktop when prompting for elevation" policy setting, the prompts appear on the interactive user's desktop instead of the secure desktop.
Disabled (Default) - The secure desktop can be disabled only by the user of the interactive desktop or by disabling the "User Account Control: Switch to the secure desktop when prompting for elevation" policy setting.

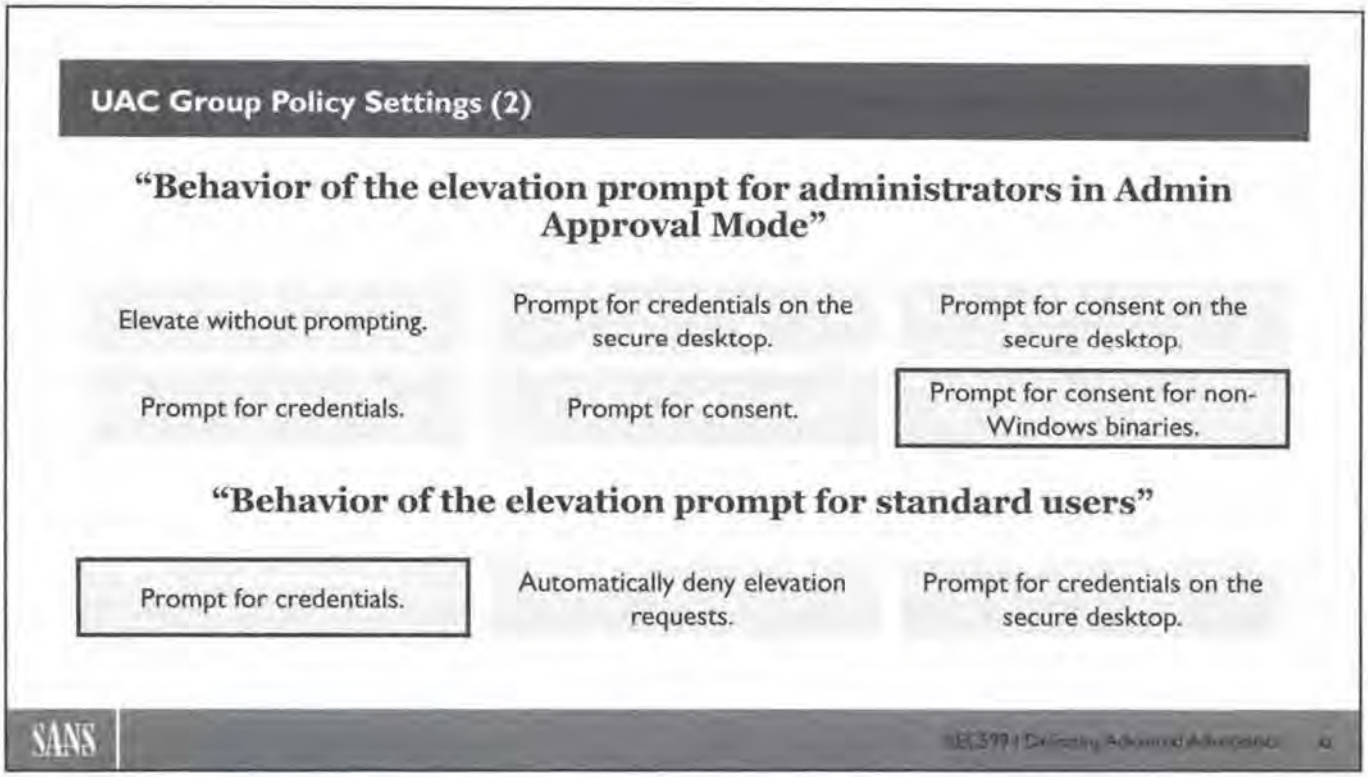


UAC: About UIAccess

User Interface Privilege Isolation (UIPI) implements restrictions in the Windows subsystem that prevent lower-privilege applications from sending messages or installing hooks in higher-privilege processes. Higher-privilege applications are permitted to send messages to lower-privilege processes. UIPI does not interfere with or change the behavior of messages between applications at the same privilege (or integrity) level.

Microsoft UI Automation is the current model to support accessibility requirements in the Windows operating systems. Applications that are designed to support an accessible user experience control the behavior of other Windows applications on behalf of the user. When all applications on the automation client computer and server are running as a standard user (that is, at a medium integrity level), the UIPI restrictions do not interfere with the Microsoft UI automation model.

However, there might be times when an administrative user runs an application with elevated privilege based on UAC in Admin Approval Mode. Microsoft UI Automation cannot drive the UI graphics of elevated applications on the desktop without the ability to bypass the restrictions that UIPI implements. The ability to bypass UIPI restrictions across privilege levels is available for UI automation programs by using UIAccess. If an application presents a UIAccess attribute when it requests privileges, the application is stating a requirement to bypass UIPI restrictions for sending messages across privilege levels.



UAC Group Policy Settings (2)

User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode

This policy setting controls the behavior of the elevation prompt for administrators.

Elevate without prompting - Allows privileged accounts to perform an operation that requires elevation without requiring consent or credentials.

Note: Use this option only in the most constrained environments.

Prompt for credentials on the secure desktop - When an operation requires elevation of privilege, the user is prompted on the secure desktop to enter a privileged user name and password. If the user enters valid credentials, the operation continues with the user's highest available privilege.

Prompt for consent on the secure desktop - When an operation requires elevation of privilege, the user is prompted on the secure desktop to select either Permit or Deny. If the user selects Permit, the operation continues with the user's highest available privilege.

Prompt for credentials - When an operation requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

Prompt for consent - When an operation requires elevation of privilege, the user is prompted to select either Permit or Deny. If the user selects Permit, the operation continues with the user's highest available privilege.

Prompt for consent for non-Windows binaries (Default) - When an operation for a non-Microsoft application requires elevation of privilege, the user is prompted on the secure desktop to select either Permit or Deny. If the user selects Permit, the operation continues with the user's highest available privilege.

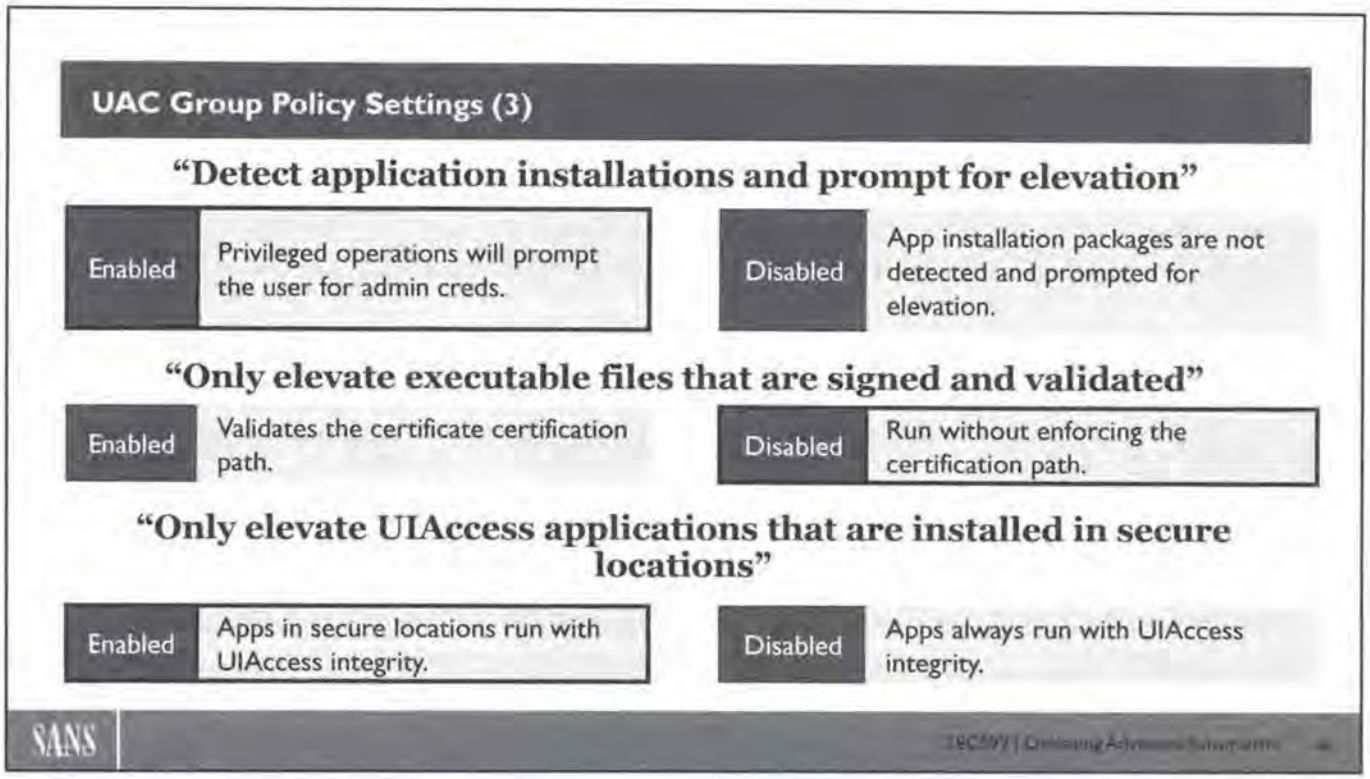
User Account Control: Behavior of the elevation prompt for standard users

This policy setting controls the behavior of the elevation prompt for standard users.

Prompt for credentials (Default) - When an operation requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

Automatically deny elevation requests When an operation requires elevation of privilege, a configurable access denied error message is displayed. An enterprise that is running desktops as standard user may choose this setting to reduce help desk calls.

Prompt for credentials on the secure desktop When an operation requires elevation of privilege, the user is prompted on the secure desktop to enter a different user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.



UAC Group Policy Settings (3)

User Account Control: Detect application installations and prompt for elevation

This policy setting controls the behavior of application installation detection for the computer.

Enabled (Default) - When an app installation package is detected that requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

Disabled - App installation packages are not detected and prompted for elevation. Enterprises that are running standard user desktops and use delegated installation technologies, such as Group Policy or System Center Configuration Manager should disable this policy setting. In this case, installer detection is unnecessary.

User Account Control: Only elevate executable files that are signed and validated

This policy setting enforces public key infrastructure (PKI) signature checks for any interactive applications that request elevation of privilege. Enterprise administrators can control which applications are allowed to run by adding certificates to the Trusted Publishers certificate store on local computers.

Enabled - Enforces the certificate certification path validation for a given executable file before it is permitted to run.

Disabled (Default) - Does not enforce the certificate certification path validation before a given executable file is permitted to run.

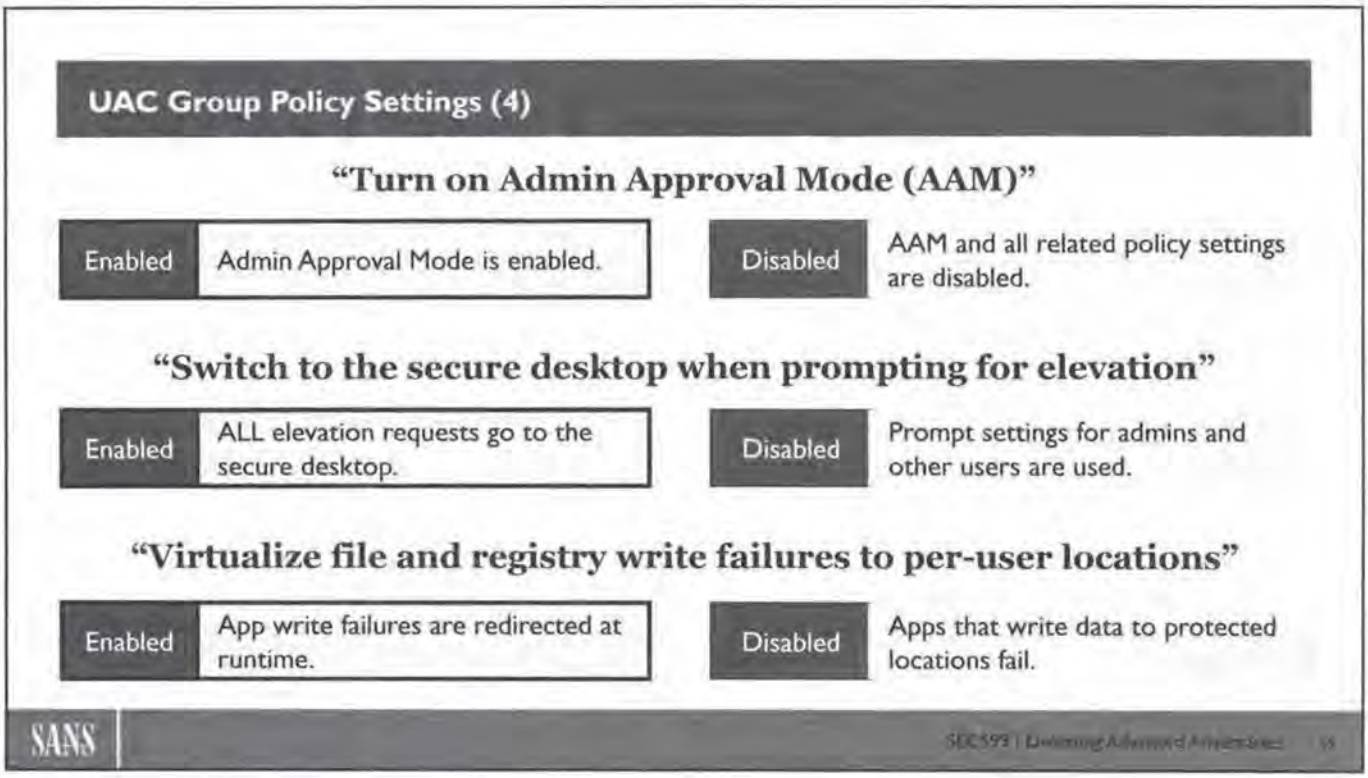
User Account Control: Only elevate UIAccess applications that are installed in secure locations

This policy setting controls whether applications that request to run with a User Interface Accessibility (UIAccess) integrity level must reside in a secure location in the file system. Secure locations are limited to the following: - ...\Program Files\, including subfolders - ...\Windows\system32\ - ...\Program Files (x86)\, including subfolders for 64-bit versions of Windows+

Note: Windows enforces a digital signature check on any interactive app that requests to run with a UIAccess integrity level regardless of the state of this security setting.

Enabled (Default) - If an app resides in a secure location in the file system, it runs only with UIAccess integrity.

Disabled - An app runs with UIAccess integrity even if it does not reside in a secure location in the file system.



UAC Group Policy Settings (4)

User Account Control: Turn on Admin Approval Mode

This policy setting controls the behavior of all User Account Control (UAC) policy settings for the computer. If you change this policy setting, you must restart your computer.

Enabled (Default) - Admin Approval Mode is enabled. This policy must be enabled and related UAC policy settings must also be set appropriately to allow the built-in Administrator account and all other users who are members of the Administrators group to run in Admin Approval Mode.

Disabled - Admin Approval Mode and all related UAC policy settings are disabled. Note: If this policy setting is disabled, the Security Center notifies you that the overall security of the operating system has been reduced.

User Account Control: Switch to the secure desktop when prompting for elevation

This policy setting controls whether the elevation request prompt is displayed on the interactive user's desktop or the secure desktop.

Enabled (Default) - All elevation requests go to the secure desktop regardless of prompt behavior policy settings for administrators and standard users.

Disabled - All elevation requests go to the interactive user's desktop. Prompt behavior policy settings for administrators and standard users are used.

Virtualize file and registry write failures to per-user locations

This policy setting controls whether application write failures are redirected to defined registry and file system locations. This policy setting mitigates applications that run as administrator and write run-time application data to %ProgramFiles%, %Windir%, %Windir%\system32, or HKLM\Software.+

Enabled (Default) App write failures are redirected at run time to defined user locations for both the file system and registry.

Disabled Apps that write data to protected locations fail.

UAC Bypass Techniques

Given the many different settings available to fine-tune UAC settings in a Windows environment, vulnerabilities are bound to arise! Several tools have implemented **UAC bypass techniques**:

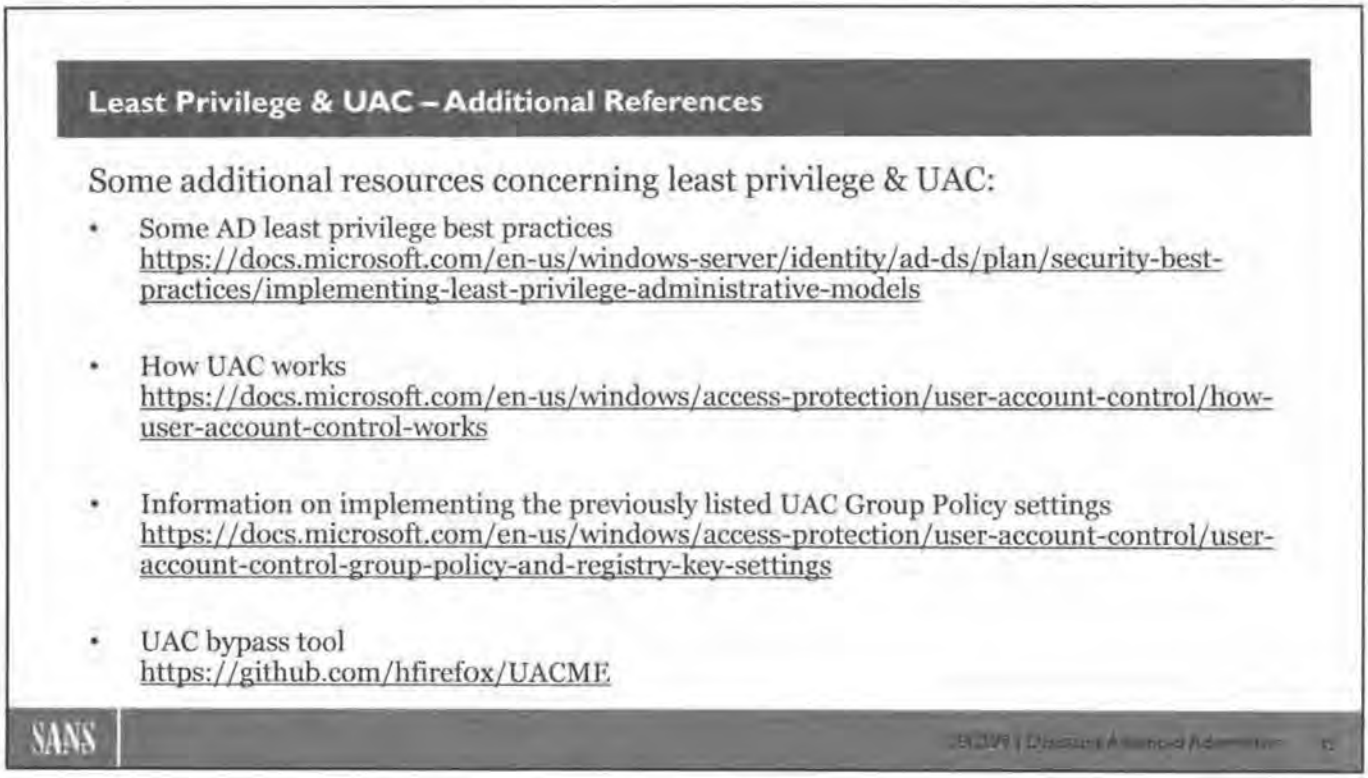
- In Windows 7, sysprep.exe is vulnerable to a DLL search order hijacking vulnerability, which is exploited by the Metasploit framework "bypassuac" module (for Windows 7)
- PowerShell Empire uses several bypass UAC techniques. In hardened Windows 10 environments (without vulnerabilities), it will spawn a UAC window asking the user to "Permit" a legitimately looking system component from obtaining admin credentials (success rate depends on the UAC settings)
- An interesting tool that tries 30+ UAC bypass techniques can be found at https://github.com/hfirefox/UACME

UAC Bypass Techniques

Given the many different settings available to fine-tune UAC settings in a Windows environment, vulnerabilities due to misconfigurations are bound to arise! Several tools have implemented UAC bypass techniques:

- In Windows 7, sysprep.exe (used to configure a Windows installation upon install) is vulnerable to a DLL search order hijacking vulnerability, which is exploited by the Metasploit framework "bypassuac" module (for Windows 7)
- PowerShell Empire uses several bypass UAC techniques. In hardened Windows 10 environments (without vulnerabilities), it will spawn a UAC window asking the user to "Permit" a legitimately looking system component from obtaining admin credentials (success rate depends on the UAC settings)
- An interesting tool that tries 30+ UAC bypass techniques can be found at https://github.com/hfiref0x/UACME

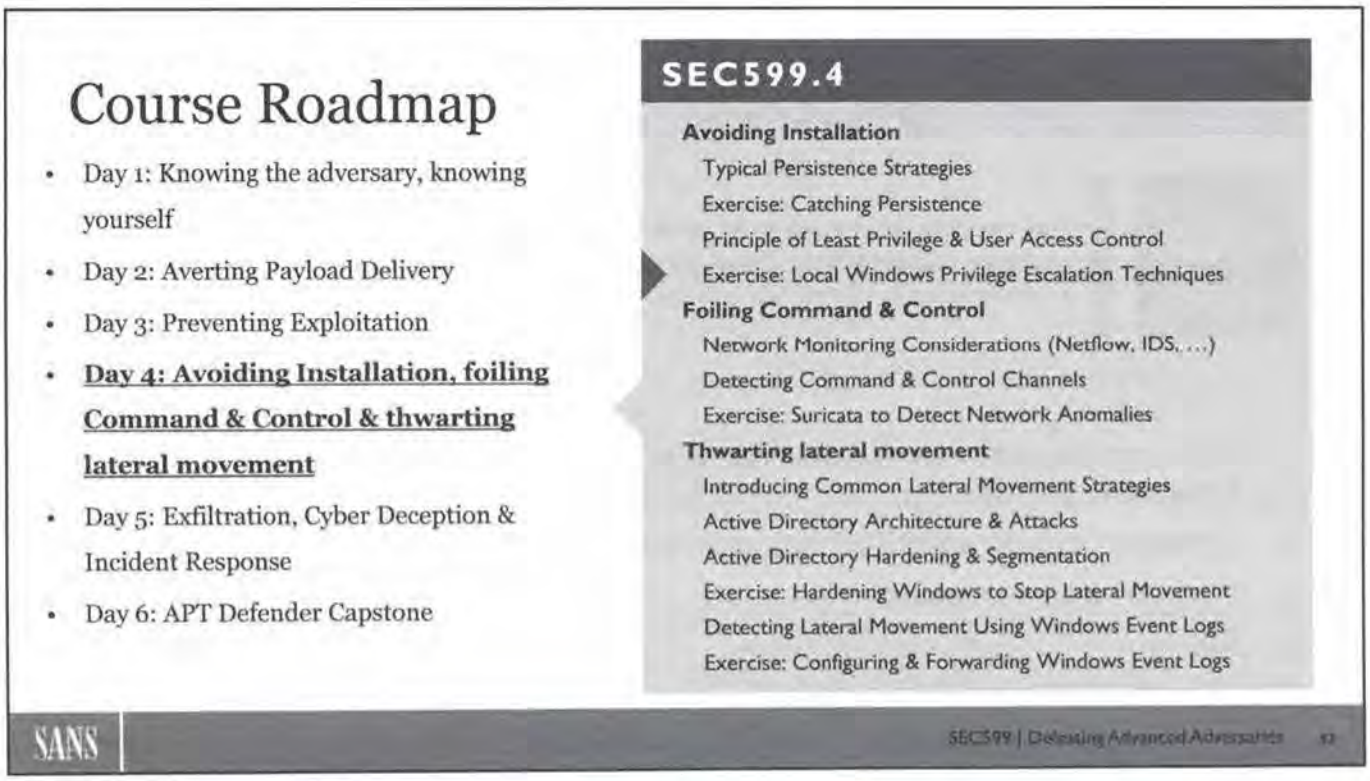
We will attempt to use these tools in an upcoming exercise!



Least Privilege & UAC – Additional References

Some additional resources concerning least privilege & UAC:

- Some AD least privilege best practices
  https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/implementing-least-privilege-administrative-models
- How UAC works
  https://docs.microsoft.com/en-us/windows/access-protection/user-account-control/how-user-account-control-works
- Information on implementing the previously listed UAC Group Policy settings
  https://docs.microsoft.com/en-us/windows/access-protection/user-account-control/user-account-control-group-policy-and-registry-key-settings



This page intentionally left blank.

Exercise – Local Windows Privilege Escalation Techniques

The objective of the lab is to audit our own Windows environment for privilege escalation vulnerabilities using different offensive tools. We will then harden our environment, after which we will again test the effectiveness of our tools.

High-level exercise steps:

1. Test our Windows environment for local privilege escalation flaws using beroot.exe & PowerUp.
2. *Optional: exploit identified issues to become local administrator*
3. Fix identified vulnerabilities
4. Retest our Windows environment

Exercise – Local Windows Privilege Escalation Techniques
The objective of the lab is to audit our own Windows environment for privilege escalation vulnerabilities using different offensive tools. We will then harden our environment, after which we will again test the effectiveness of our tools.

The high-level steps of this exercise include:
1. Test our Windows environment for local privilege escalation flaws using beroot.exe & PowerUp.
2. Optional: exploit identified issues to become local administrator
3. Fix identified vulnerabilities
4. Retest our Windows environment

For additional guidance & details on the lab, please refer to the LODS workbook.

Exercise – Local Windows Privilege Escalation Techniques - Conclusions

During this exercise, we used two tools to "audit" our Windows environment for privilege escalation vulnerabilities.

Note that the vulnerabilities identified are often "configuration" mistakes that cannot just be fixed by applying a patch. Following this idea, we manually applied fixes to secure our environment.

It's a good idea to periodically analyze your environment for such techniques. These tools could perfectly be deployed in a similar fashion to Autoruns, where you could make them part of a scheduled task that is executed upon system start (using, for example, GPO's).

SANS

Exercise – Local Windows Privilege Escalation Techniques – Conclusions
During this exercise, we used two tools to "audit" our Windows environment for privilege escalation vulnerabilities. Note that the vulnerabilities identified are often "configuration" mistakes that cannot just be fixed by applying a patch. Following this idea, we manually applied fixes to secure our environment.

It's a good idea to periodically analyze your environment for such techniques. These tools could perfectly be deployed in a similar fashion to Autoruns, where you could make them part of a scheduled task that is executed upon system start (using, for example, GPO's).

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

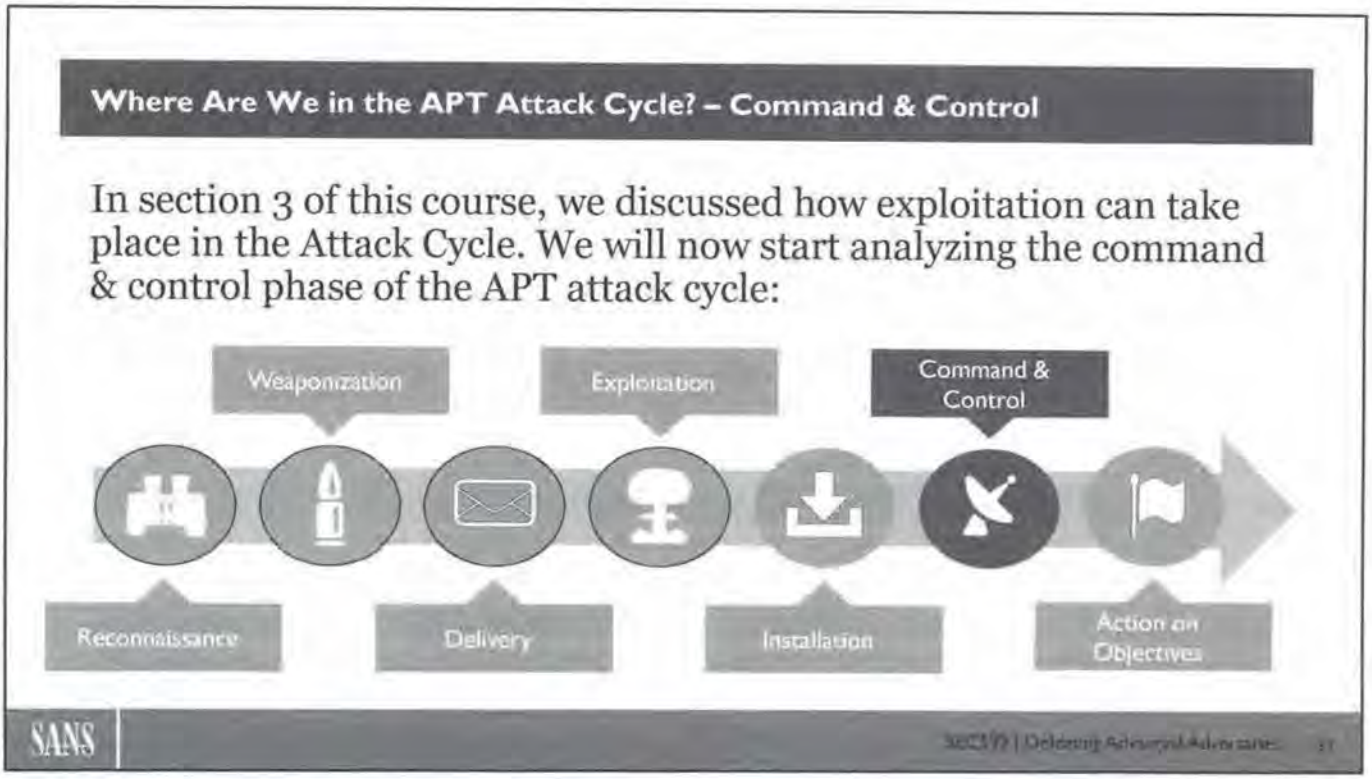
SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.



Where Are We in the APT Attack Cycle? - Command & Control

Programming malware to perform all malicious actions automatically and autonomously can be quite challenging for adversaries, especially when they have incomplete information about their targets. Often the adversaries will want the malware to have capabilities to be controlled remotely so that they can instruct the malware with the appropriate actions to take. Adversaries attempt to keep this control by the use of a Command & Control infrastructure and channel.

Adversary perspective

During weaponization, adversaries will already decide on the communication channel to be used, as it will be built into the malicious payload. In order to avoid detection and to increase the chances of the outbound connectivity being allowed, adversaries will select a commonly used protocol such as HTTP(S), DNS, email or even social media. Cases have also been identified where custom TCP protocols were developed. The endpoint of communication channels are called Command & Control servers; these are servers under the control of the adversaries, but not necessarily owned by the adversaries. In targeted attacks, adversaries might first compromise other systems and use these as Command & Control servers.

A varying degree of stealth can be built into these C&C channels, for example masquerading the communication as an HTTP connection with a music streaming service. Another example we've seen is the use of steganography in pictures. This doesn't necessarily imply that the communication is complex. For ransomware, for example, the C&C infrastructure is often only used to report back the encryption keys.

Defender perspective

Communication between the malware on target systems and adversaries is a good opportunity for us to stop the attack, provided we detect and block it almost instantaneously. We can limit network communications from our network to the Internet via control points like proxies and firewall. In an open network where internal clients have full access to the Internet via a NAT gateway, controlling, filtering and monitoring is exponentially harder than in a network where all communications go through filtering devices. Of course, an open network can be a business requirement and something you have to live with.

Proxies not only allow us to block and filter traffic, but it also gives us the opportunity to log and inspect the traffic for patterns or anomalies (e.g. beaconing behavior). Beaconing behavior is when malware periodically attempts to connect back to its Command & Control server. If this is done using a fixed time interval, it could form a pattern we can attempt to detect.

Network Monitoring Considerations

Before we start zooming in on the specifics of Command & Control channels, we want to spend a bit of time to discuss network monitoring!

- Should we monitor only NetFlow or full packet capture?
- At what point of the network should we perform monitoring?
- How do we handle SSL/TLS?
- How long should we retain NetFlow / full packet captures?

Even if we discuss this topic while addressing Command & Control, network monitoring can help you detect several stages of the APT Attack Cycle!

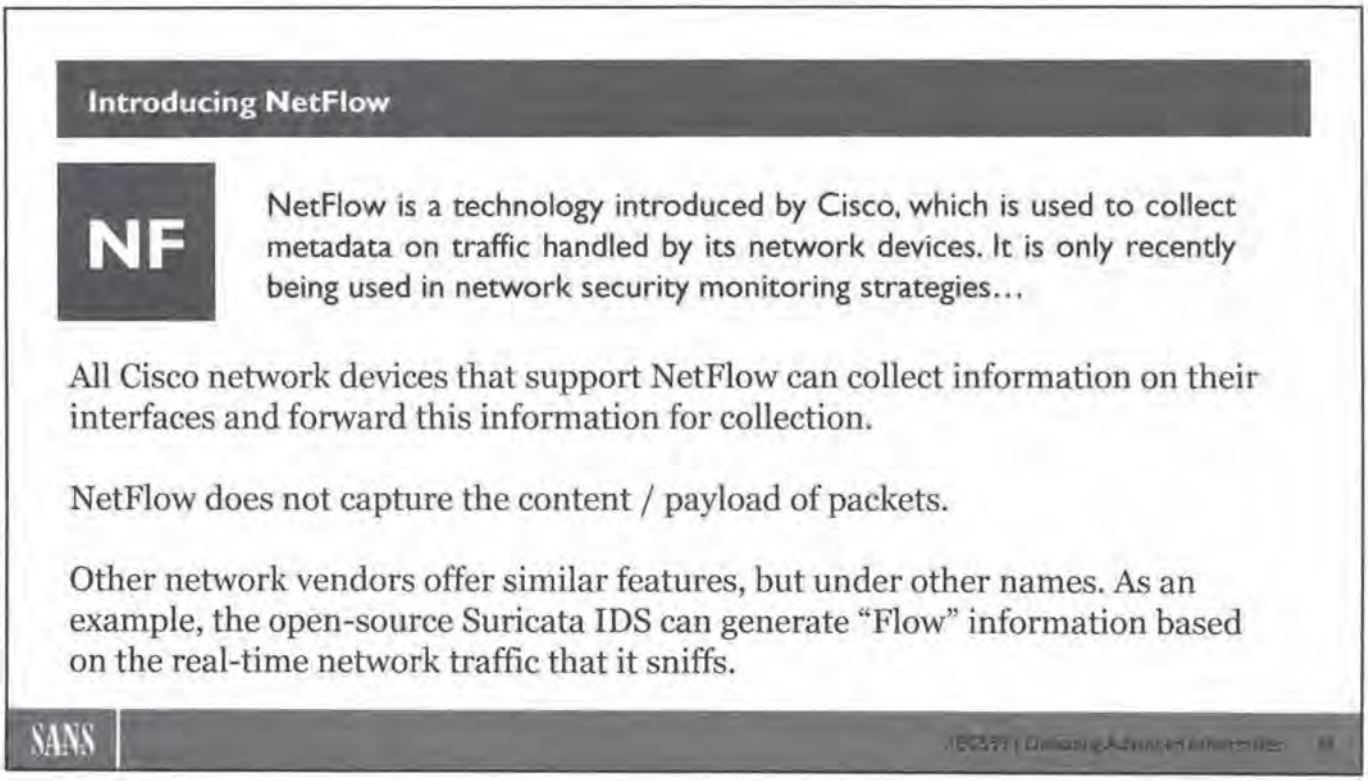
Network Monitoring Considerations

Before we start zooming in on the specifics of Command & Control channels, we want to spend a bit of time to discuss network monitoring… When advanced adversaries attack our infrastructure, they will be forced to use our network. Even if they have physical access to one machine, they will need to use the network to move laterally. That is why in our network architecture design, we apply segmentation so that we have chokepoints where we can monitor network traffic.

So, let's agree we need to monitor network traffic. But, how? Should we do full packet capture or should we just focus on NetFlow? Furthermore, how do we handle SSL / TLS encrypted protocols? Because we are bound to miss some attacks, it is a good idea to keep some kind of "history" of network traffic, so that this history can be consulted post-facto.

But even if we would monitor all network traffic, we would still miss detecting some attacks. For example, because these attacks "hide in plain sight", by using existing network traffic and injecting their own messages. A very difficult technique to detect for example is steganography: steganography conceals messages inside other messages. The typical example of steganography is an existing picture where some bits are altered to encode the hidden message. These altered bits do not change the overall aspect of the picture. The same principle can be applied to network traffic, for example by adding whitespace characters to the headers of an HTTP request.

Even if we discuss this topic while addressing Command & Control, it's important to note that network monitoring can be highly useful to detect several other stages of the APT Attack Cycle as well!



Introducing NetFlow

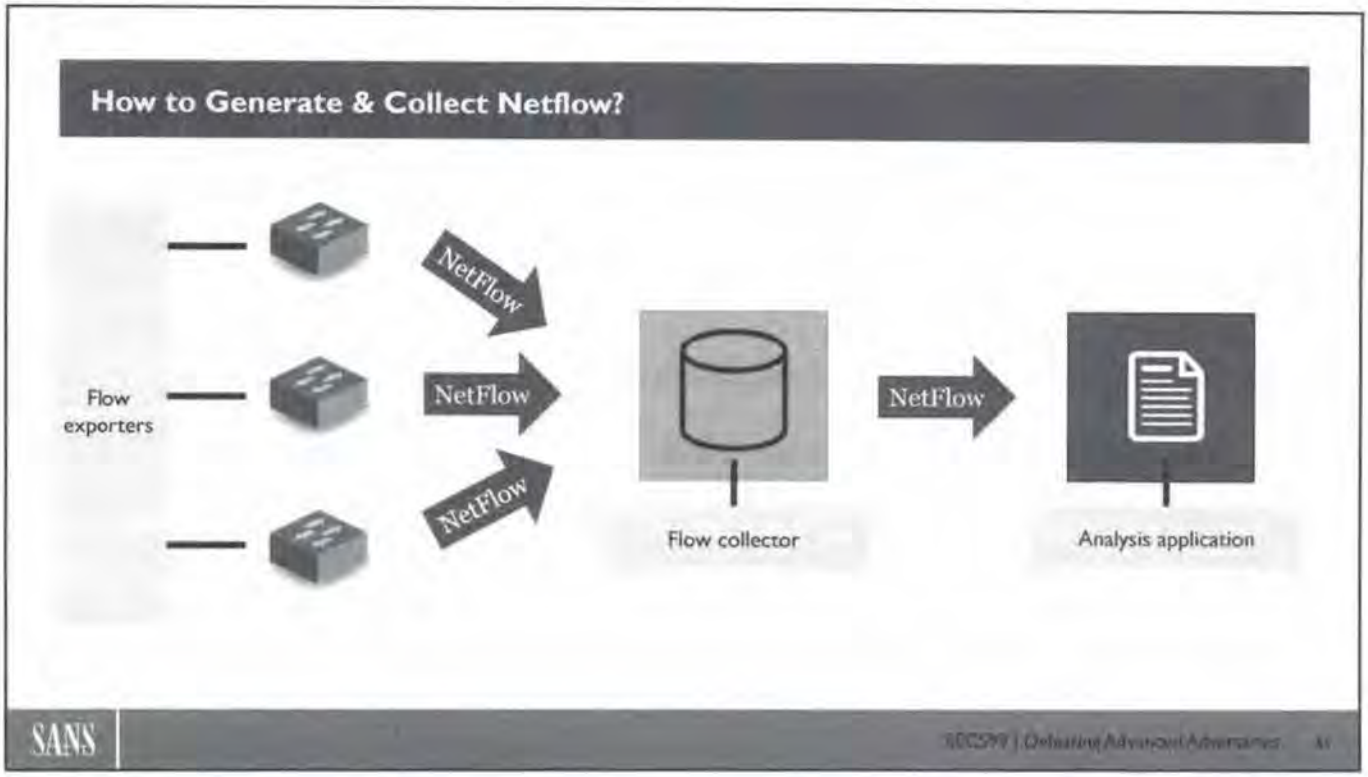
NetFlow is a technology by Cisco introduced on their network devices to collect metadata from network traffic that flows through these devices. The keyword here is metadata: this means that the traffic itself is not captured, but properties of the traffic.

Take for example a TCP connection. NetFlow will not capture the content of the TCP connection, but it will log properties of the TCP connection (metadata) like these:

- Date and time of the connection
- Source IP address
- Source port
- Destination IP address
- Destination port
- Number of packets transmitted and received
- Number of bytes transmitted and received

This aggregated information is collected by network devices on the interfaces that are configured to generate NetFlow data and then forwarded.

Other network vendors offer similar features but under other names. As an example, the open source Suricata IDS can generate "Flow" information based on the real-time network traffic that it sniffs.



How to Generate & Collect NetFlow?
In this network diagram, we illustrate how NetFlow data is generated and collected.

Different network devices in our corporate network (switches, routers, ...) are configured to generate NetFlow data. These Cisco devices need to support the generation of NetFlow data (this depends on the IOS version and features), and the interfaces from which we want NetFlow data need to be configured to generate this data. These devices are called flow exporters.

This data, generated by many devices in our corporate network, is then forwarded to one or more flow collectors. A flow collector is a network device or computer with enough persistent storage to keep the NetFlow data of many exporters for long periods of time (months).

When necessary, analysts can use an analysis application to retrieve and analyze the NetFlow data. Originally NetFlow was implemented to provide statistics to network administrators to help them with managing and adapting their network infrastructure with the growing needs of their enterprise.

But this data can also be used after an attack was detected. For example, when a compromised computer is detected several months after the attack took place, NetFlow data can be consulted to have a list of all connections that this computer initiated to other computers. Depending on the configuration of your NetFlow infrastructure, NetFlow data will be available for all computers, or only for computers in another segment of the network (depending on the place where NetFlow data is exported).

Introducing Full Packet Capture (FPC)

Full Packet Capture is the concept of sniffing and storing all network traffic using for example a network tap. As opposed to NetFlow, Full Packet Capture will capture the full packet payload / content.

- Large enterprise networks generate A LOT of traffic, so storage could be an issue!
  - *But really, how expensive are disks these days?*
- For proper packet inspection, consider using an SSL interception solution!
- The advantages of FPC over NetFlow could be limited if there's a lot of encrypted protocols...
- Some critical government institutions will collect & store full packet captures for a large period of time (e.g. 1 to 2 years). This will allow them to retroactively hunt their logs / packet captures for attack campaigns that are identified in the future!

SANS

Introducing Full Packet Capture (FPC)

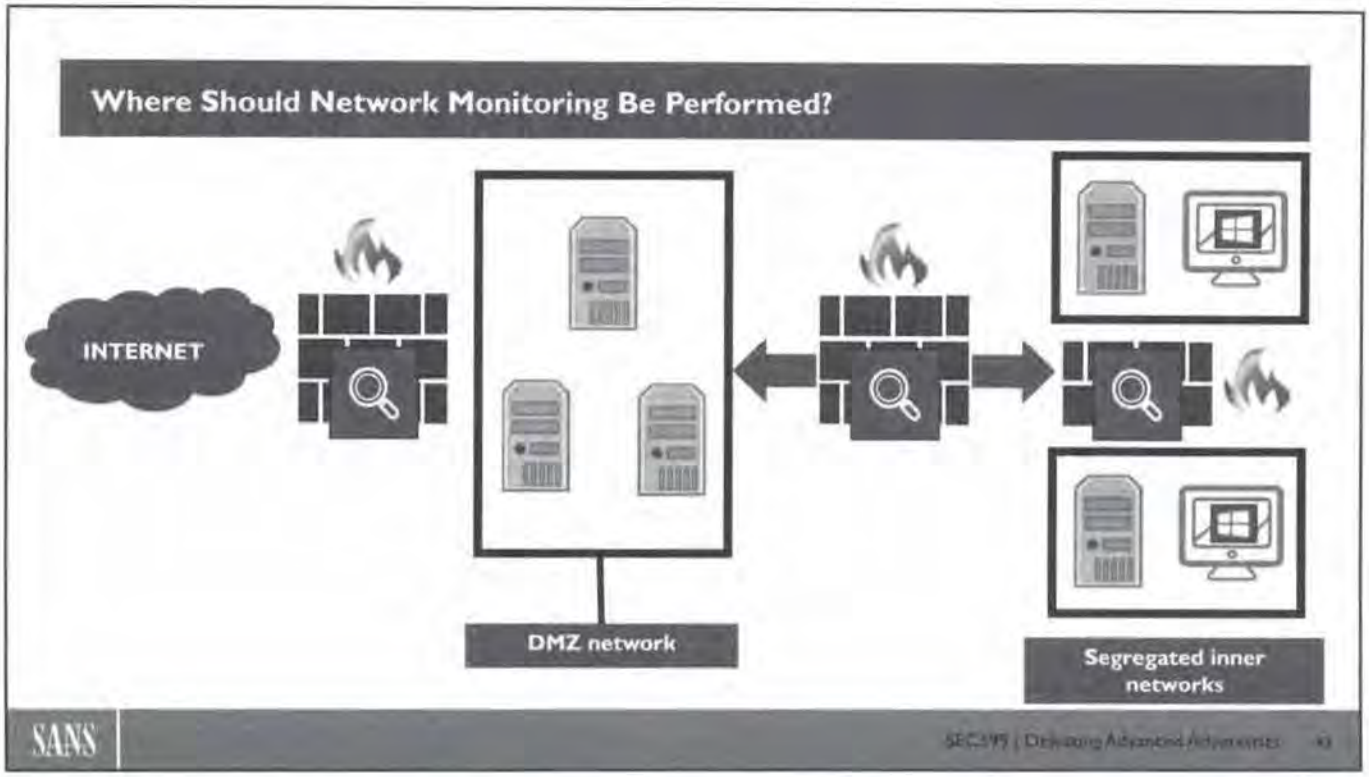
Where NetFlow (and similar technologies) will capture metadata, full packet capture will capture and store the complete content of network packets. This means that for a TCP connection, we not only have all the metadata like NetFlow would provide, but also the content of the TCP transmission itself: all the data that was transmitted and received by the computer.

Full packet capture will generate a lot of data, Gigabytes to Terabytes of data per day, depending on the network. This is why it is not possible to perform full packet capture on all computers. When full packet capture is implemented, it is typically done by tapping the network connections at key points in the corporate network infrastructure. These key points are for example the chokepoints where traffic flows between network segments. A good starting point for full packet capture is at the perimeter: capture all traffic between the corporate network and the Internet.

Not only does full packet capture help analysts to investigate attacks post-facto, but it also allows for the detection of attacks post-facto. This is done by processing all captured data of a given period (for example the last month) with an IDS like Snort or Suricata. The IDS needs to be provided with the detection rules and IOCs of the latest attack methods so that it can detect attacks that happened in the past and were not detected.

A good example is the Heartbleed vulnerability: this is a vulnerability in OpenSSL that allowed an adversary to obtain private data from SSL/TLS webservers. This private data could contain the encryption keys, for example. After Heartbleed was discovered and disclosed, several IDS rules were created to detect Heartbleed attacks. By searching all captured network traffic for evidence of an Heartbleed attack, a corporation could determine (post-facto) if it had been attacked and if data was leaked.

Although full packet capture produces a huge amount of data, it must be taken into account that large capacity magnetic harddisks have become cheap.



Where Should Network Monitoring Be Performed?
To perform full packet capture, one has to decide where to capture packets inside the network.

This will depend upon the type of attacks to be detected, and be influenced by the network architecture.

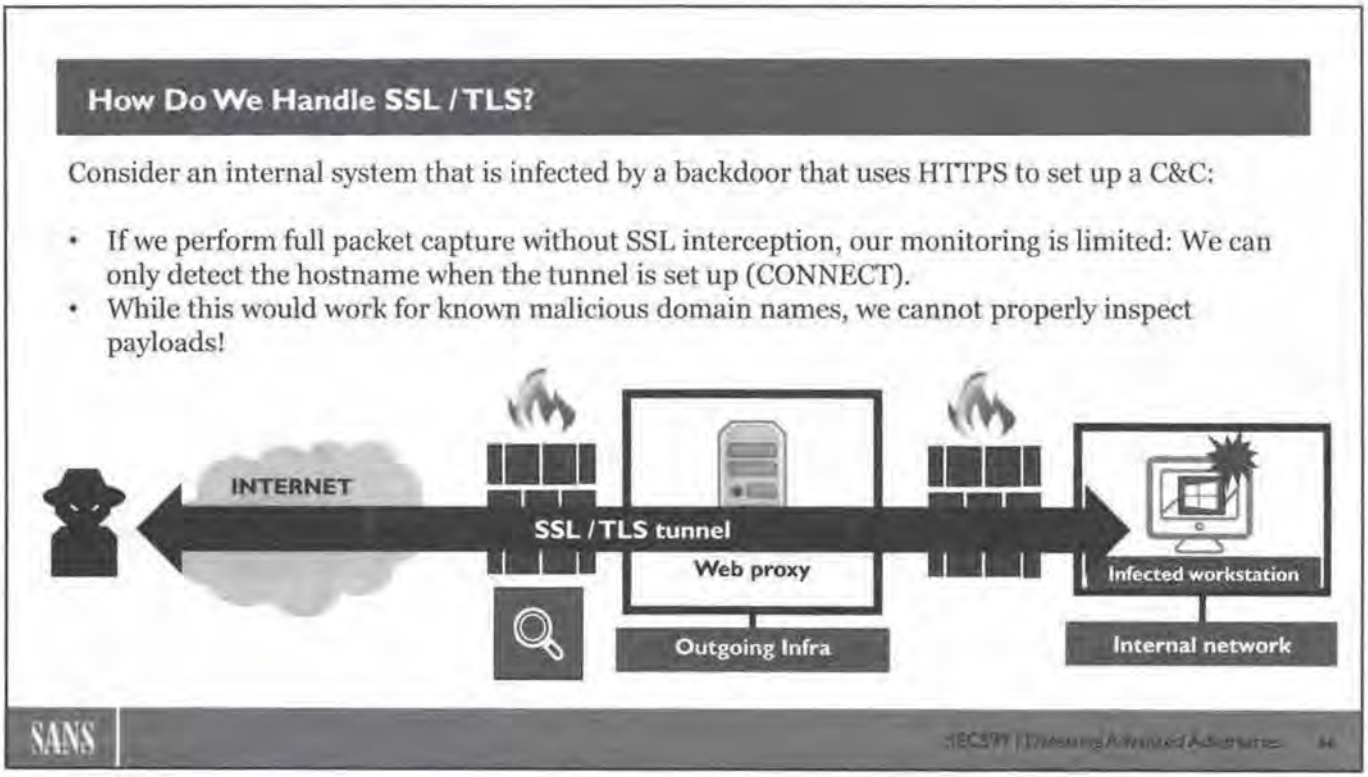
- We recommend starting at the perimeter as a priority: the border between the corporate network and the Internet. Adversaries have many different tricks up their sleeves, but in a lot of cases they will need to enter and leave the network (e.g. set up C&C channels or exfiltrate data).

- In later stages, you can expand your scope to also include the borders of internal network zones. This could be useful to detect lateral movement inside the environment!

A router is a good point to start capturing traffic. Although routers like Cisco routers have IOS commands to perform packet capture on the router itself, routers don't have enough persistent storage to keep traffic for more than a couple of minutes (this feature is often used for troubleshooting, not for long-term packet capture). But routers with decent management features can be configured to duplicate network traffic that flows through one of its interfaces to another interface: this is called a network tap. This network tap interface can then be connected to a device that will perform the full packet capture. This can be a computer with enough persistent storage (mechanical harddisks) to store the captured traffic as pcap files for several months.

It is possible to aggregate the network traffic of several interfaces to one tap interface, but attention must be paid not to exceed the bandwidth of the network tap interface. For example, if all interfaces on a router can manage 10 Gigabit and 2 interfaces that we want to aggregate process 6 Gigabit of traffic, then the resulting traffic will be 12 Gigabit which exceeds the capacity of the tap interface.

Just capturing and storing network traffic is not a CPU intensive task: the machine that stores the full packet capture does not have to be a high-end machine.



How Do We Handle SSL / TLS?

When a proxy has to establish a TLS connection for a client that starts an HTTPS conversation, a normal proxy will create a tunnel. A TLS connection is encrypted end-to-end, and the encryption keys are only known by the end-points.

The proxy is not capable to decrypt the traffic. To initiate a tunnel for the use of TLS, the client (the workstation for example) makes a connection to the proxy and issues a CONNECT command. Here is an example for the Google website:

CONNECT www.google.com

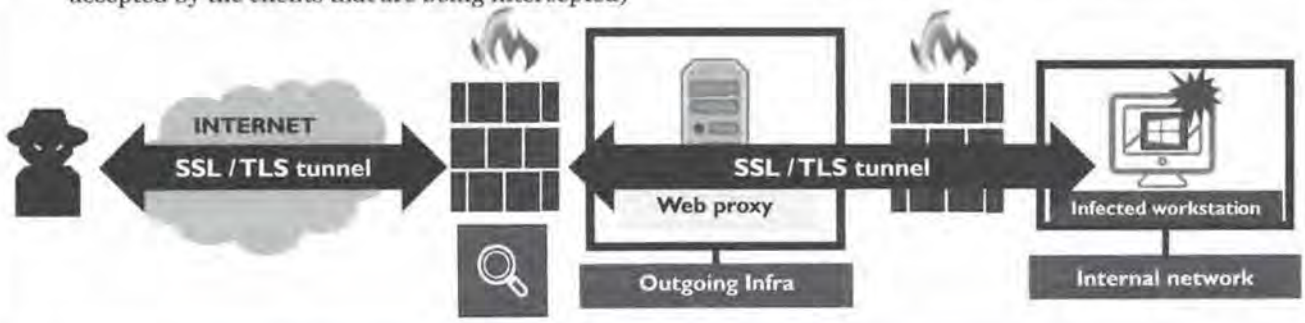
When the proxy receives this command, it creates a tunnel: the TLS traffic it receives from the client is forwarded to the Google website, and the TLS traffic from Google is forwarded to the client. The proxy operates in a transparent mode for this connection: this allows both end-points (the workstation and Google's webserver) to perform the TLS handshake and start an encrypted conversation.

The only action that the proxy can undertake (besides flat-out dropping all TLS connections), is categorizing the website based on its domain, and block it if it falls in a blocked category. The proxy can categorize the domain (but not the URL) because the domain is communicated in cleartext with the CONNECT command. The complete URL, however, is encrypted in the TLS channel.

How Do We Handle SSL / TLS? – Interception

During SSL interception, one of the systems between the Internet and the internal systems will "break" the connection. Instead of allowing the internal system to create a tunnel to the Internet system, the interceptor will set up two tunnels: A tunnel between the Internet and itself and a tunnel between the internal workstation and itself.

- SSL interception is typically performed by web proxies (which will of course require the root CA to be accepted by the clients that are being intercepted)



SANS

How Do We Handle SSL / TLS? – Interception

Because encrypted HTTP traffic becomes the norm on the Internet, network devices that perform deep inspection of network traffic are becoming "blind". A solution to this problem is TLS interception.

A proxy that supports TLS interception works as follows:

- When the client starts a TLS connection, the TLS intercepting proxy does not forward the TLS connection to the web server, but it establishes a TLS connection with the client (TLS 1).
- The client accepts this TLS connection from the proxy because the proxy uses a certificate (with matching private key) that is trusted by the client. This can be done by using a certificate generated by a corporate PKI that has its root certificate installed on all corporate machines.
- Next, the proxy will establish its own TLS connection with the web server (TLS 2).
- One set of keys is used to encrypt TLS connection 1, and another set is used to encrypt TLS connection 2.
- Since the proxy is now an endpoint for both TLS connections, it can decrypt the traffic from both sides.
- The proxy decrypts the traffic from one channel and sends it encrypted in the other channel.

This allows the proxy to inspect the "encrypted" traffic, and it can also create a network tap for use by other network devices.

What Kind of Technology Is Available?

Many commercial & open-source tools exist that that can meet a variety of our needs:

- SSL / TLS interception
- Full packet capture
- IDS alerting
- Protocol anomaly detection

SURICATA

Some commercial examples include the typical firewall vendors: PaloAlto, Juniper, Cisco, Fortinet, Sophos, ... Open-source alternatives for network monitoring include Snort, Suricata, Bro, ... They can typically be configured in IDS (passive) or IPS mode (inline).

For both commercial & open-source tools, the architecture & configuration of the device will be essential. If they are not correctly positioned in the network they will not fulfill their full potential.

SANS

What Kind of Technology Is Available?

Many commercial & open-source tools exist that that can meet a variety of our needs… We are typically looking for the following types of functions:

- SSL / TLS interception
- Full packet capture
- IDS alerting
- Protocol anomaly detection
- …

Some commercial examples include the typical firewall vendors: PaloAlto, Juniper, Cisco, Fortinet, Sophos, … Open-source alternatives for network monitoring include Snort, Suricata, Bro, … They can typically be configured in IDS (passive) or IPS mode (inline).

In inline mode, the IDS/IPS has at least 2 network interfaces: one network interface is connected to the corporate network and the second network interface is connected to the Internet. This way, the network traffic flows through the IDS/IPS device and is available for inspection, while the IPS can block traffic for which alerts are generated. The disadvantage of inline mode is availability; when the IDS/IPS device goes down, network traffic no longer flows through the device and the corporate network is severed from the Internet.

Passive mode does not have this inconvenience. In passive mode, the IDS/IPS device will receive network traffic on one interface via a network tap (just like full packet capture). If the device goes down, the network flow is not impacted. The IPS can no longer block the traffic in passive mode, but there is a solution for this problem: the IPS device can send an instruction to the network device with the network tap to block the traffic detected by the IPS.

For both commercial & open-source tools, the architecture & configuration of the device will be essential. If they are not correctly positioned in the network they will not fulfill their full potential.

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

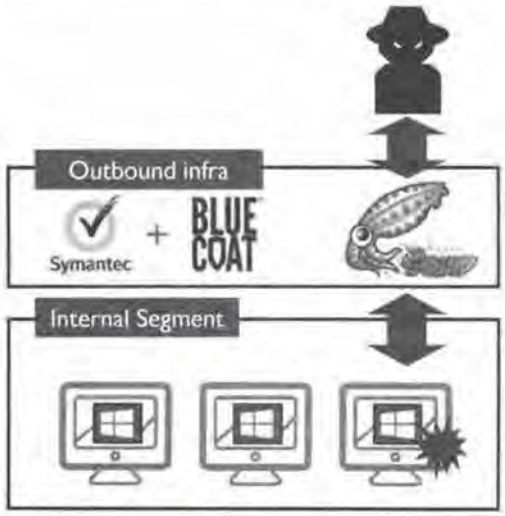
Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

Command & Control Channels - Topology



Typical scenario

- A machine on the corporate network is compromised: malware authored by our adversaries is running on said machine.
- This machine has access to the Internet, but it has to traverse proxies and firewalls (which gives us a control / inspection point).
- The command & control server, operated by the adversaries, is located on the Internet.

SANS

Command & Control Channels - Topology
The typical topology of a command and control channel inside a corporate network is the following:

- A machine on the corporate network is compromised: malware authored by our adversaries is running on said machine.
- This machine has access to the Internet.
- The network connection to the Internet has to traverse proxies and firewalls.
- The command & control server, operated by the adversaries, is located on the Internet.

Without corporate proxy and firewalls, the compromised machine would have unrestricted access to the Internet, and our adversaries would be able to use any network protocol as command & control channel.

With corporate proxy and firewalls, however, Internet access is restricted and our adversaries have to use network protocols that are allowed. Typically, all UDP connections are blocked (blocking DNS) and TCP is only allowed over certain ports, like FTP, HTTP, and HTTPS.

If the proxy/firewalls inspect the type of network protocol that is used through the allowed ports, then our adversaries are further restricted in the protocols they can use. For example, SSH over port 80 would be blocked.

Proxies categorize URLs in categories, and network administrators use these categories to block certain web sites. Typically, categories like adult content are blocked, and categories like banking are allowed. Depending on the network policies, uncategorized websites are blocked or allowed. By blocking uncategorized websites, we further restrict the number of command & control channels that our adversaries can use because they often use machines without domain names and that are thus uncategorized.

Introducing Command & Control Channels

Once adversaries have obtained access to a target environment, they want to USE this access to attain their objectives. This could include compromising additional hosts, stealing sensitive data,... In order to achieve these objectives, adversaries need a Command & Control channel!

Many different C&C channels exist, each with their own detection / prevention challenges:

- Dedicated C&C infrastructure
- Compromised infrastructure
- ...
- Social media services
- Cloud storage providers

Introducing Command & Control Channels

Once adversaries have obtained access to a target environment, they want to USE this access to attain their objectives. This could include compromising additional hosts, stealing sensitive data, ... In order to achieve these objectives, adversaries need a Command & Control channel!

For example, malware can spy on computers by capturing keystrokes, taking screenshots or screen recordings, recording audio from the built-in microphone, ... If this malware would be implanted on all your corporate machines and automatically send all captured data to the adversaries, they would be overwhelmed with data and you would see a spike in outgoing traffic.

What (advanced) adversaries will do in this case is use the implanted malware to locate computers with interesting data and then activate the spying capabilities of said malware. This requires communication between the malware and the adversaries: this is done via so-called command & control channels. On one end, there is the compromised, corporate machine, and on the other end the server under control of the adversary: the command & control server (C2).

Many communications channels can be used as C2 channel. Malware authors started to use custom made, proprietary command and control channels over TCP connections, but later on, this evolved into using existing channels like DNS and HTTP. The reason is that these custom channels would be more easily detected and blocked in corporate networks.

Detecting Command & Control Channels

While in the past, exotic or custom protocols were often used, malware authors now prefer to use C2 channels that "blend into the noise":

- We are referring to the "noise" produced by network traffic of social media like Twitter, Facebook, Instagram, even Dropbox, ...
- To be fair, a lot of modern web traffic genuinely looks malicious ☺
- CDN networks are all generating obfuscated, long, HTTP queries that look strange... This is not helping us!

We will illustrate this issue by zooming in one some of the most interesting C&C channels used by adversaries!

Detecting Command & Control Channels

To remain undetected when malware uses command & control channels, malware authors have started to adopt a strategy to "blend into the noise". By this, we mean that malware authors will use existing channels with a high degree of interaction: messages that flow between clients and servers with high frequency.

These types of channels can be found in social media applications. Social media produces messages at a high frequency, many corporate users use social media, also for business reasons (LinkedIn for example) and most social media applications have a public aspect. These factors explain why malware authors start to use social media channels as command & control channels: they can "blend into the noise".

Take for example malware that uses Twitter as a communication channel. The client (the malware) will have the credentials of a Twitter account and use this to read and write messages. These messages will go to the Twitter servers, like any other Twitter communication. The command & control server has another set of credentials for a Twitter account. Both accounts follow each other, and can thus communicate with each other, publicly or privately (via direct messages). If steganography is used to encode a command in a seemingly innocuous Twitter message, it becomes very hard to uncover these channels.

Twitter is just one example. There are many social media applications that are very popular with your corporate users, like Facebook, Instagram, Pinterest, Google+, Dropbox, … All these social media applications can be used as command and control channel.

In the next slides, we will discuss some famous C&C channels that were used in real attacks!

Famous Command & Control Channels – Turla Likes Britney Spears ☺

A variant of Turla communicated by reading and writing comments on Britney Spears' Instagram account.

Example comment
"asmith2155: #2hot make loved to her, uupss #Hot #X"

- The malware would read comments posted for this picture, and calculate a custom hash. If the custom hash is 183, then the malware knows the comment is a from the C&C server and that it must decode it.
- The comment contains a hidden special UNICODE character: \200d. This is a non-printable character called "Zero Width Joiner". Like its name implies, it does not take up space but it is present.
- This character, together with a couple of other characters like # and @, are used as prefix to the encoded URL:
  <200d>#2hot ma<200d>ke lovei<200d>d to <200d>her, <200d>uupss <200d>#Hot <200d>#X

This gives us the following string: 2kdhuHX, or URL http://bit(.)ly/2kdhuHX!

SANS

Famous Command & Control Channels – Turla Likes Britney Spears ☺
There exists a variant of the Turla malware that uses Instagram as a command & control channel.

Not via Instagram accounts that would communicate directly with each other, but via Instagram accounts that would post comments on pictures posted by a very popular Instagram user: Britney Spears.

To send a message to the malware, the command and control channel would post a link (for example to pastebin) shortened with a URL shortening service like bitly. This URL would not be posted directly to Britney Spears' picture as a comment, but it would be encoded with steganography in a seemingly innocuous comment.

An example of such a comment would be: #2hot make loved to her, uupss #Hot #X

The malware would read comments posted for this picture, and calculate a custom hash. If the custom hash is 183, then the malware knows the comment is a command from the command & control server and that it must decode it.

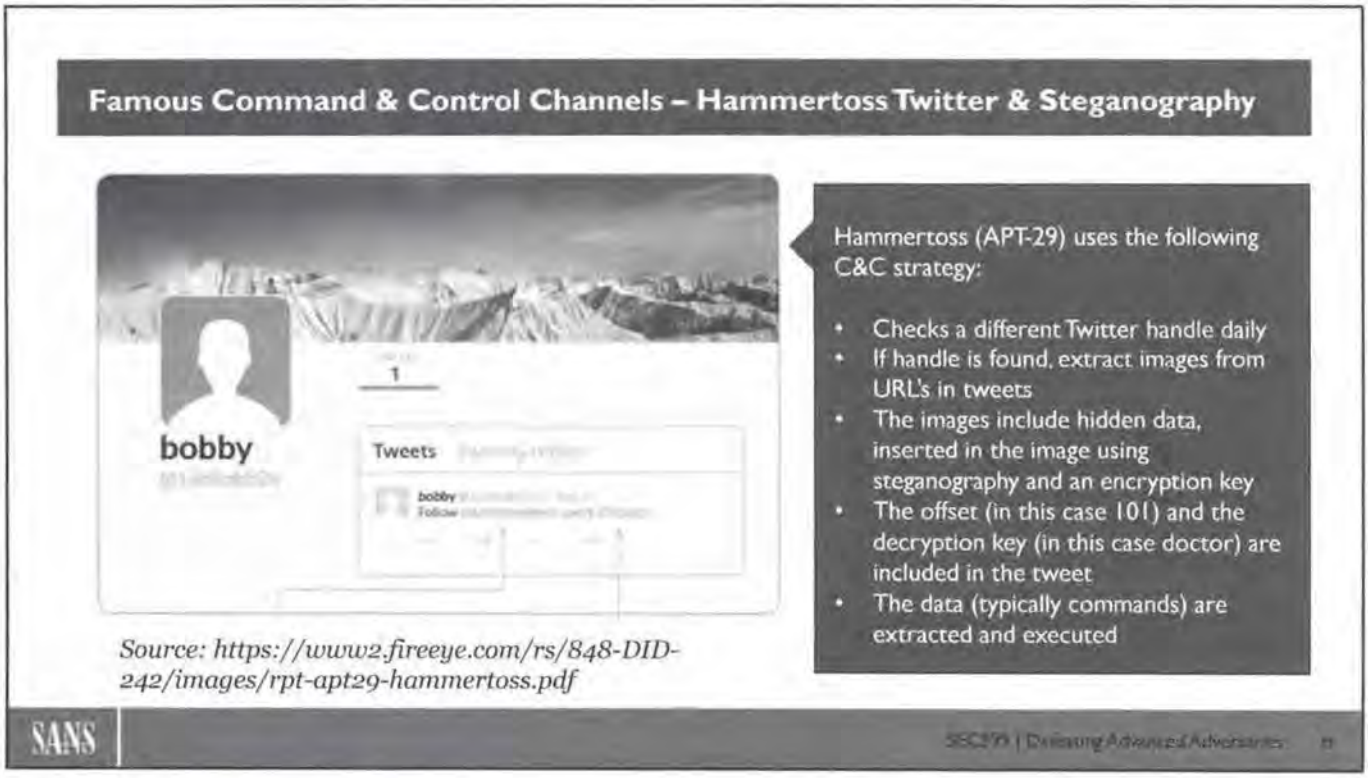
It cannot be seen in the text of the comment, but the comment contains a special UNICODE character: \200d. This is a non-printable character called "Zero Width Joiner". Like its name implies, it does not take up space but it is present.

This character, together with a couple of other characters like # and @, are used as prefix to the encoded URL.

The malware extracts the URL from the message by taking the character that follows the prefix character.
Here is the message with the special UNICODE character made visible (<200d>):
<200d>#2hot ma<200d>ke lovei<200d>d to <200d>her, <200d>uupss <200d>#Hot <200d>#X

This gives us the following string: 2kdhuHX, or URL http://bit(.)ly/2kdhuHX

https://www.welivesecurity.com/2017/06/06/turlas-watering-hole-campaign-updated-firefox-extension-abusing-instagram/



Famous Command & Control Channels – Hammertoss Twitter & Steganography

Another interesting example of a command & control channel is the use of steganography in Twitter images by the HAMMERTOS (by APT-29). In short, the following took place:

- Once installed, the HAMMERTOSS backdoor generates & looks for a specific Twitter handle (a different one every day). A built-in algorithm will generate a daily handle (e.g. 1abBob52b) and will subsequently check this twitter account.
- If the APT group has not registered the handle, the malware will take no action and wait for the next day.
- The APT group will register the protocol name and post tweets using the account. The tweets have the following format:
  - They include a link to an image
  - The image includes a hidden message (typically a command that is to be executed)
  - The message is hidden in the image using steganography (it is encrypted and placed in the image using a symmetric encryption key)
  - The offset (where the message is hidden) and the encryption key are included in the tweet
- Upon finding an image, the backdoor will download the image, extract the message and execute the requested command.

An excellent analysis of the HAMMERTOSS backdoor was done by FireEye analysts and can be found here:
https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf

Famous Command & Control Channels – DropSmack

DropSmack is an offensive toolkit developed by Jake Williams from Rendition Infosec. It was presented at Blackhat in 2013 and essentially uses Dropbox as a C&C channel!

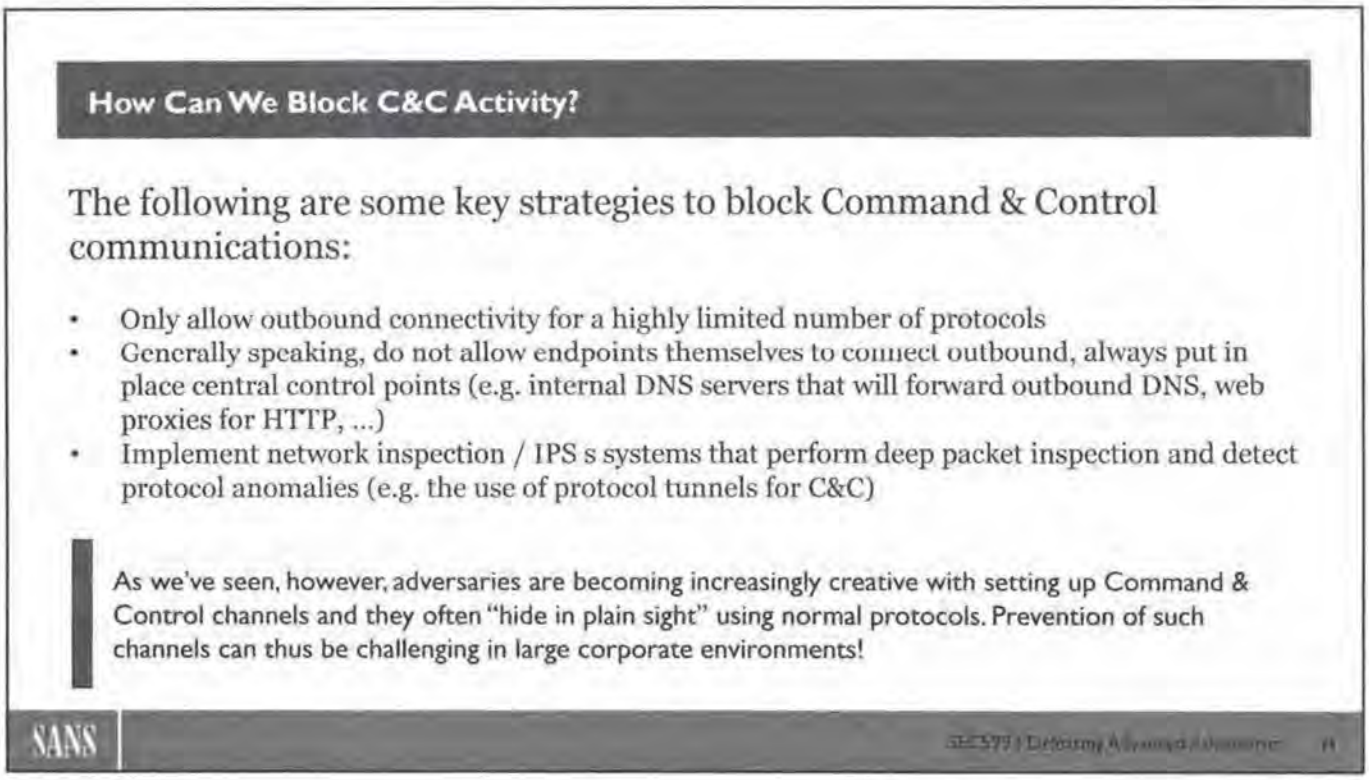
DropSmack leverages standard Dropbox connectivity to set up an "easy" channel for both command & control and data exfiltration. Commands are included as text files which are subsequently "sync'ed" to infected hosts in the internal network.

As DropSmack uses built-in Dropbox functionality, there really isn't a lot we can do except for disabling Dropbox synchronization in our internal network.

Famous Command & Control Channels – DropSmack
DropSmack is an offensive toolkit developed by Jake Williams from Rendition Infosec. It was presented at Blackhat in 2013 and essentially uses Dropbox as a C&C channel!

- DropSmack leverages standard Dropbox connectivity to set up an "easy" channel for both command & control and data exfiltration. Commands are included as text files which are subsequently "sync'ed" to infected hosts in the internal network. It can of course also be used in the other direction to "sync" files out of the network…
- As DropSmack uses built-in Dropbox functionality, there really isn't a lot we can do except for disabling Dropbox synchronization in our internal network.

It's important to note that this is not really "a vulnerability" in Dropbox. This type of technique could be used against the vast majority of cloud-based file sharing services. This opens the discussion whether we should allow cloud-based file sharing within the enterprise. Given the rise of cloud within enterprises, this will be a hot discussion topic ☺



How Can We Block C&C Activity?

The following are some key strategies to block Command & Control communications:

- Only allow outbound connectivity for a highly limited number of protocols
- Generally speaking, do not allow endpoints themselves to connect outbound, always put in place central control points (e.g. internal DNS servers that will forward outbound DNS, web proxies for HTTP, ...)
- Implement network inspection / IPS s systems that perform deep packet inspection and detect protocol anomalies (e.g. the use of protocol tunnels for C&C)
- Configure network devices to check that the type of traffic matches the port. For example, we only allow HTTP on port 80, we block SSH over port 80 but allow it over port 22.

As we've seen, however, adversaries are becoming increasingly creative with setting up Command & Control channels, and they often "hide in plain sight" using normal protocols. Prevention of such channels can thus be challenging in large corporate environments! Let's investigate how we can try detecting the use of Command & Controls!

How Can We Detect C&C Activity?

Next to the "blocking" controls listed on the previous slide, here are some interesting ideas to detect C&C activity in your environment. You will require logs & network traffic of your perimeter devices:

- Review end-point systems for applications that are connecting to external systems (this can, for example, be easily achieved using OSQuery, which we will use tomorrow)
- Look for unknown protocols that are not expected in your environment
- Look for beaconing behavior
  (malware typically checks in with its C&C server on a periodic basis)
- Unusual traffic volumes (could be a sign of data exfiltration)
- Investigate typical C&C protocols
  - HTTP: User-Agent, HTTP Referer, ...
  - DNS: Query length, Query types, Query entropy,
  - Find values that are not normal for your environment and alert upon them
    (e.g. A User-Agent that does not match your organization's default browser type)

SANS

How Can We Detect C&C Activity?
Next to the "blocking" controls listed on the previous slide, here are some interesting ideas to detect C&C activity in your environment. In any case, if we want to start detecting Command & Control activity, we will need to start performing network monitoring & collecting logs from our perimeter devices. Additionally, it might be worth reviewing end-points for applications that are connecting to external systems (this can, for example, be easily achieved using OSQuery, which we will use tomorrow)

Furthermore, look for unknown protocols that are not expected in your environment. This is fairly easy, as exotic protocols will raise alerts and are easy to spot.

Implement protocol specific gateways (such as outbound web proxies) that can analyze the protocol and analyze protocol fields & settings. Highly popular protocols used for C&C connectivity include HTTP & DNS. In HTTP traffic, interesting fields to analyze include, for example, the User-Agent and the "Referer" request headers. For DNS, we can look at the query length, query types & entropy.

Even advanced adversaries will not always invent new protocols, but reuse existing command & control protocols. By configuring IDS & IPS systems with up-to-date rules that detect various known command & control channels, we increase our chances to detect known command and control protocols. Properly configuring proxies and firewalls to only allow known and trusted protocols and websites, we can further reduce the use of command & control channels.

Anomaly-based network detection systems can help us detect new, unknown command & control channels. These detection systems come with advanced network protection devices and are sometimes running on dedicated computers to provide full processing power. Detecting beaconing is one such application. By restricting the initiation of network connections to devices inside our network, the command & control server cannot connect directly to the compromised machine. It is the compromised machine that has to initiate the connection on a regular basis to check if the command & control server has instructions. This is called beaconing, and by analyzing the frequency of connections, regular beaconing can be detected.

One of the problems we face is that more and more our adversaries will use encrypted command & control channels (like SSL/TLS), which drastically reduces our opportunities to inspect traffic. TLS interception can help us here.

How Can We Detect C&C Activity? – Freq.Py

Mark Baggett (a SANS Instructor & ISC Handler) wrote a highly useful python tool called "**freq.py**" that can be used to perform frequency analysis to detect suspicious hostnames!

Some malware leverages domain generation algorithms (DGA), which is the primary target of freq.py:

- In typical English words, there are character pairs that are more frequent than others
  - "TH": There is a 40% chance that a T will be followed by an H
  - "QU": There is a 97% chance that a Q will be followed by a U
- Freq.py will analyze DNS logs and "score" all hostnames for "randomness". DGA algorithms will immediately pop up!

PROBLEM: CDN / ad networks will also be highlighted by "freq.py"
PROBLEM 2: Modern malware often uses random words, not random letters

SANS

How Can We Detect C&C Activity? – Freq.py

Mark Baggett (a SANS Instructor & ISC Handler) wrote a highly useful python tool called "freq.py" that can be used to perform frequency analysis to detect suspicious hostnames! Some malware leverages domain generation algorithms (DGA), an interesting example is the Zeus banking trojan.

Malware families that use DGA are the primary detection target of freq.py… How does it work?

- In typical English words, there are character pairs that are more frequent than others:
  - "TH": There is a 40% chance that a T will be followed by an H
  - "QU": There is a 97% chance that a Q will be followed by a U
- Freq.py will analyze DNS logs and "score" all hostnames for "randomness". DGA algorithms will immediately pop up!

The development and usage of freq.py is described in detail in Mark Baggett's SANS ISC post: https://isc.sans.edu/forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893/

It's important to note that this approach has the following limitations:

1. First of all, CDN / ad networks will typically use randomly generated (sub-)domain names as well, which will also be highlighted by "freq.py". This could be solved by whitelisting certain domains from the list (then again, you might whitelist actual malicious domains)
2. Secondly, modern malware that relies on DGA sometimes uses a new approach: instead of using random letter, they use random words to generate domain names. This will defeat freq.py, as it's looking for uncommon characters pairs, not uncommon combinations of words…

How Can We Detect C&C Activity? – RITA

RITA (Real Intelligence Threat Analytics) is a project first started by Black Hills Information Security which is now being further maintained by Offensive CounterMeasures. You can find its source code on GitHub!

RITA is an open source framework for network traffic analysis. It was designed to ingest Bro (network security monitor) logs and has the following interesting analysis options:

- Search for signs of **beaconing behavior** in and out of your network;
- Search for signs of **DNS based covert channels**;
- Search for **suspicious, long, URL's** that are often indicate of malware;

RITA supports a few other items, but the above are most relevant for detecting C&C traffic!

How Can We Detect C&C Activity? – RITA

RITA (Real Intelligence Threat Analytics) is a project first started by Black Hills Information Security which is now being further maintained by Offensive CounterMeasures. You can find its source code on the following GitHub page: https://github.com/ocmdev/rita

RITA is an open source framework for network traffic analysis. It was designed to ingest Bro (network security monitor) logs and has the following interesting analysis options:

- Search for signs of beaconing behavior in and out of your network;
- Search for signs of DNS based covert channels;
- Search for suspicious, long, URL's that are often indicative of malware;

RITA does support other analysis types as well, but the ones listed above are most relevant to detect Command & Control traffic. Please refer to RITA's github page for additional information, including for example install, configuration & usage guidelines.

Command & Control Channels – Additional References

Some additional resources concerning command & control channels:

- Turla's use of Instagram
  https://www.welivesecurity.com/2017/06/06/turlas-watering-hole-campaign-updated-firefox-extension-abusing-instagram/
- FireEye – Hammertoss report
  https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf
- DropSmack – Blackhat paper
  https://media.blackhat.com/eu-13/briefings/Williams/bh-eu-13-dropsmack-jwilliams-wp.pdf
- RITA:
  https://github.com/ocmdev/rita

Command & Control Channels – Additional References

Turla's use of Instagram
https://www.welivesecurity.com/2017/06/06/turlas-watering-hole-campaign-updated-firefox-extension-abusing-instagram/

FireEye – Hammertoss report
https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf

DropSmack – Blackhat paper
https://media.blackhat.com/eu-13/briefings/Williams/bh-eu-13-dropsmack-jwilliams-wp.pdf

Black Hill's RITA:
https://www.blackhillsinfosec.com/?page_id=4417

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

Exercise – Using Suricata to Detect Network Anomalies

The objective of the lab is to detect a new attack technique that is being used to download executables to compromised systems. We will first configure Suricata on PfSense, after which we will write a custom rule to detect this new technique.

High-level exercise steps:

1. Configure Suricata on PfSense to perform IDS alerting
2. Write new IDS rule to spot new type of attack technique
3. Simulate attack using new attack technique and confirm successful detection

SANS | SEC599 | Defeating Advanced Adversaries

Exercise – Using Suricata to Detect Network Anomalies
The objective of the lab is to detect a new attack technique that is being used to download executables to compromised systems. We will first configure Suricata on PfSense, after which we will write a custom rule to detect this new technique (TTP).

The following are the high-level attack steps:

1. Configure Suricata on PfSense to perform IDS alerting
2. Write new IDS rule to spot new type of attack technique
3. Simulate attack using new attack technique and confirm successful detection

For additional guidance & details on the lab, please refer to the LODS workbook.

Exercise – Using Suricata to Detect Network Anomalies – Conclusions

During this exercise, we relied on Suricata IDS to detect suspicious HTTP user agents in our environment:

Based on intelligence we received, we understood that "certutil.exe" is being used as a novel technique to bypass application whitelisting and download malicious payloads.

We subsequently created an IDS rule to detect this type of outbound network connectivity!

We will focus more on network traffic analysis tomorrow when we discuss data exfiltration!

Exercise – Using Suricata to Detect Network Anomalies – Conclusions
During this exercise, we relied on Suricata IDS to detect suspicious HTTP user agents in our environment:

- Based on intelligence we received, we understood that "certutil.exe" is being used as a novel technique to bypass application whitelisting and download malicious payloads. We subsequently created an IDS rule to detect this type of outbound network connectivity!

We will focus more on network traffic analysis tomorrow when we discuss data exfiltration!

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

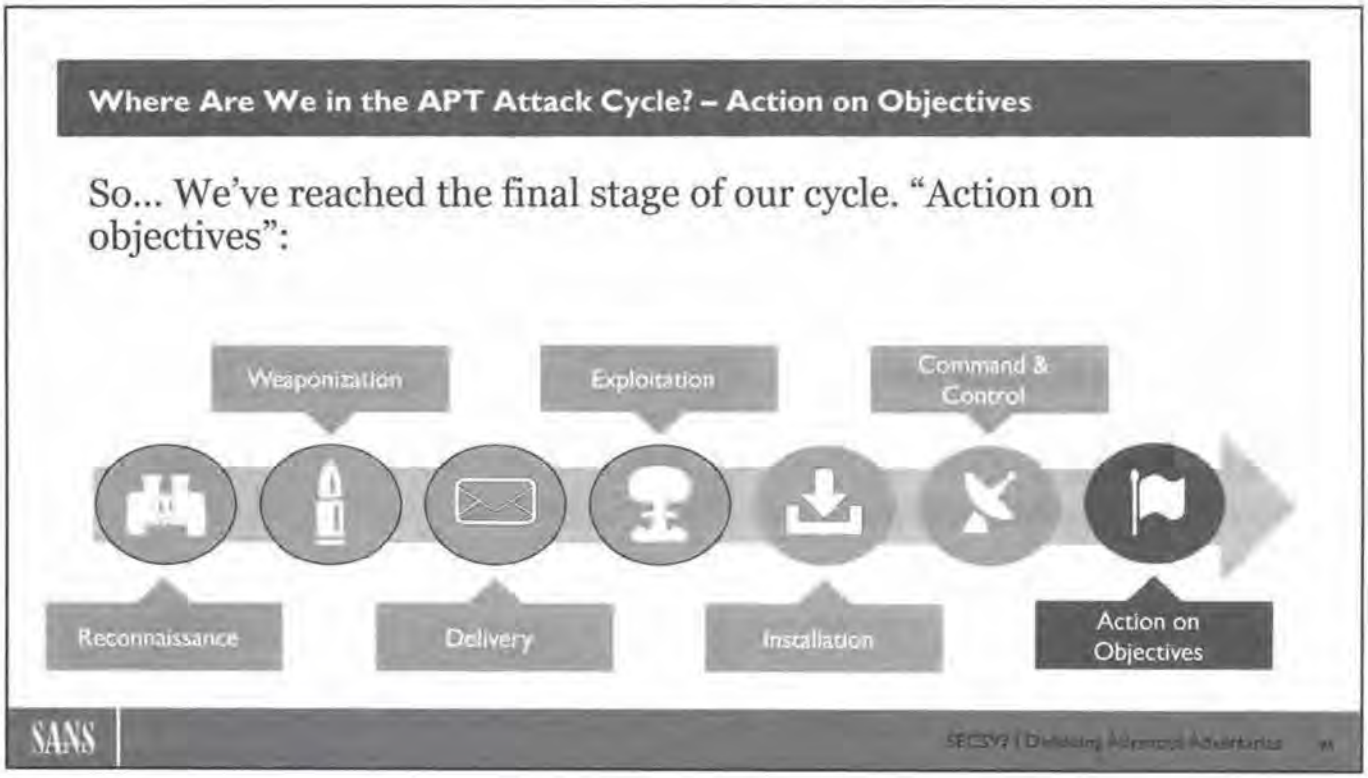
SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.



Where Are We in the APT Attack Cycle? – Action on Objectives
Once the adversary has managed to persist malware on an initial system, he can now gear up to start working on his actual objectives.

Adversary perspective
Once the attack reaches this step of the kill chain, everything has been put into place to enable the adversaries to perform the necessary actions on the targets to reach their objectives. The actions they will take depend on these objectives, and can be all sort.

For example, once they have a foothold in the infrastructure, adversaries can start a new digital kill chain: they start with reconnaissance of the internal network to identify interesting targets to attack. This will typically be followed by lateral movement. Lateral movement is the term used to indicate that adversaries are spreading in the network, moving from computer to computer. Once inside, lateral movement is often facilitated by the "openness" of the internal network (so-called "egg-shell" problem). Old school design of a secure network puts many of the security controls at the perimeter of the network, and not inside the network. Once adversaries penetrated the perimeter without detection, they encounter fewer obstacles to move inside the internal network.

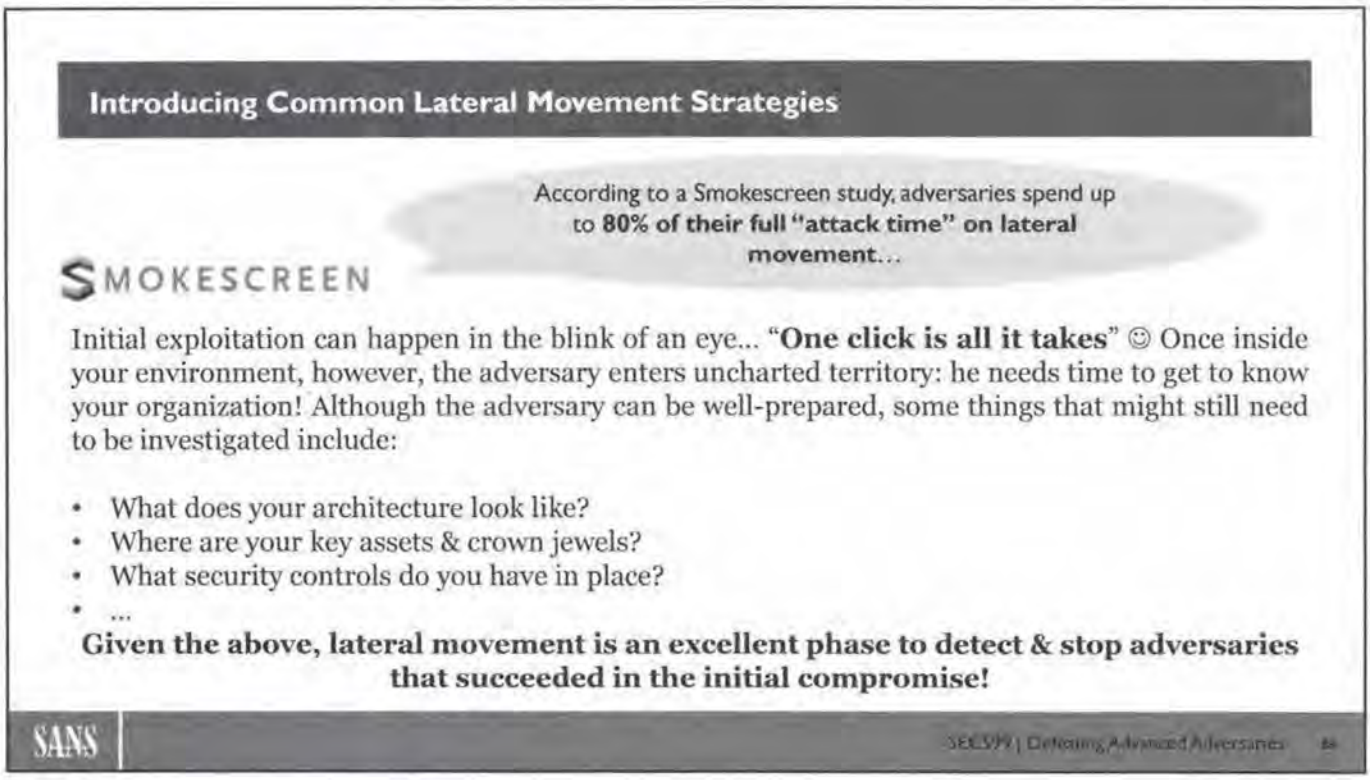
When adversaries reach their targets through lateral movement, they will "finalize the kill". If the objective is espionage, they will collect and exfiltrate data. If the objective is to interfere with the target, they will start making modifications. This can be corrupting, deleting or overwriting of data and systems, or covertly modify data and configurations to change operations within the target. For example, data modifications can be introduced in payment systems to steal money by wire transfer. We have even observed malware samples that modify payroll data on cloud systems to introduce new, fake, employees in the staff database and have their wages paid to bank accounts owned by criminals or their money mules.

Defender perspective
When adversaries progress this far in the kill chain, they have defeated the majority of previous defenses. For the adversary, everything is in place for the final push.

Depending on the objectives and the complexity of the attack, there might be a lot of activity required from the adversary, which could give us more opportunities to detect or block the attack. During this step, which can be the longest in absolute time of all steps in the digital kill chain, adversaries typical perform following actions (not all during the same attack):

Lateral movement: adversaries do reconnaissance of the internal network and move from system to system to reach the systems they target. Lateral movement can generate a lot of evidence, offering opportunities to defenders for detection. To be able to detect lateral movement, appropriate controls inside the internal network need to be put in place, such as firewalls and intrusion detection system between different segments of the internal network. Furthermore, adversaries could attempt to reuse previously compromised credentials, which could raise suspicion. For example, if user A is currently working in the New York office and a login is detected for user A via VPN from a location in Turkey, something unusual is going on and should be further investigated.

Data exfiltration: when the objective is to obtain information, it has to be transferred to the adversaries' systems once it is located and accessed. Exfiltration of data is typically a network activity and as such leaves traces. Large amounts of data exfiltration (gigabytes or terabytes) are detectable by graphing the consumed network bandwidth versus a time axis. Dedicated system can be put in place to monitor for data exfiltration: Data Loss Prevention systems. DLP can be as simple as looking for tags on the network, such as the string "strictly confidential" in uploaded documents. But such simple detections are also simple to bypass. For example, just compressing or encrypting a document before uploading hides all strings inside the document.

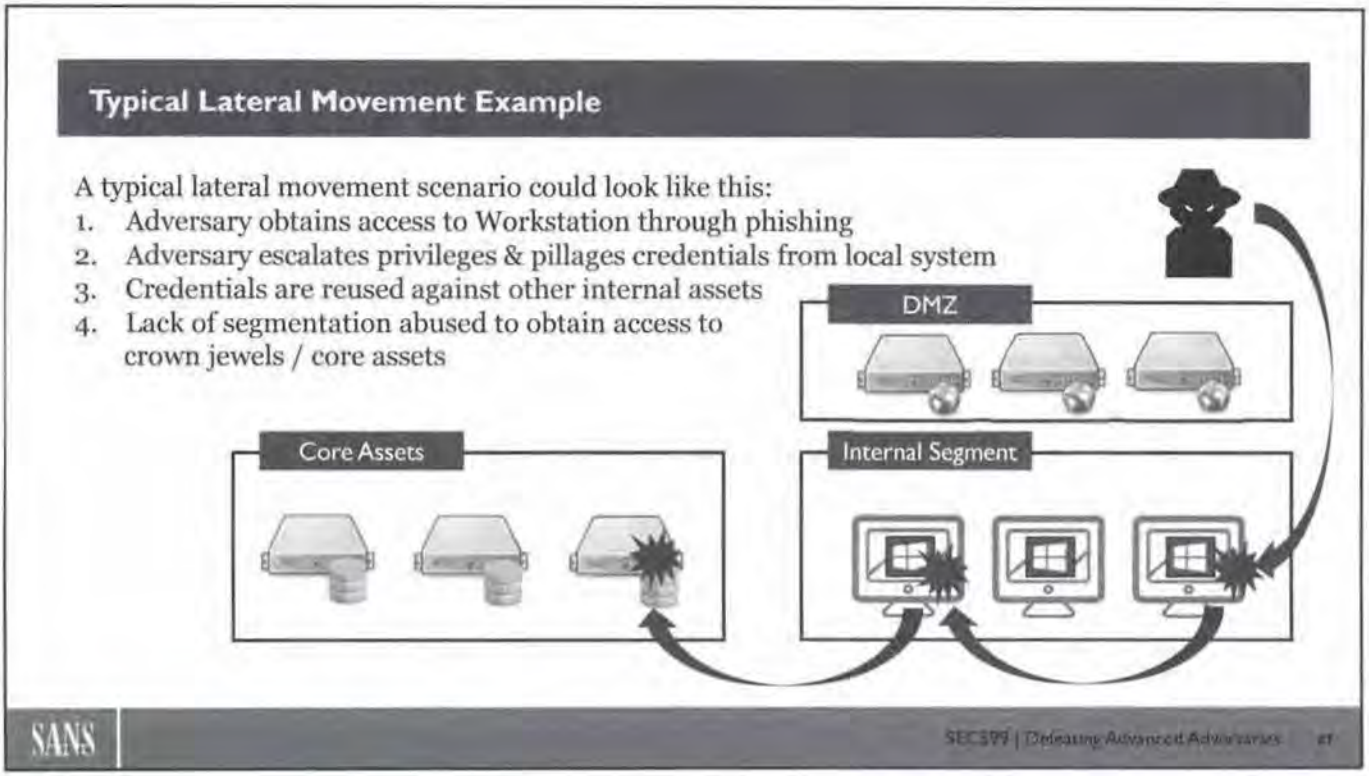


Introducing Common Lateral Movement Strategies

Initial exploitation can happen in the blink of an eye... "**One click is all it takes**". ☺ Once inside your environment, however, the adversary enters uncharted territory; he needs time to get to know your organization! Although the adversary can be well-prepared, some things that might still need to be investigated include:

- What does your architecture look like?
- Where are your key assets & crown jewels?
- What security controls do you have in place?
- ...

According to a Smokescreen study (a vendor of cyber deception technology), adversaries spend up to 80% of their full "attack time" on lateral movement! Given the above, lateral movement is an excellent phase to detect & stop adversaries that succeeded in the initial compromise!

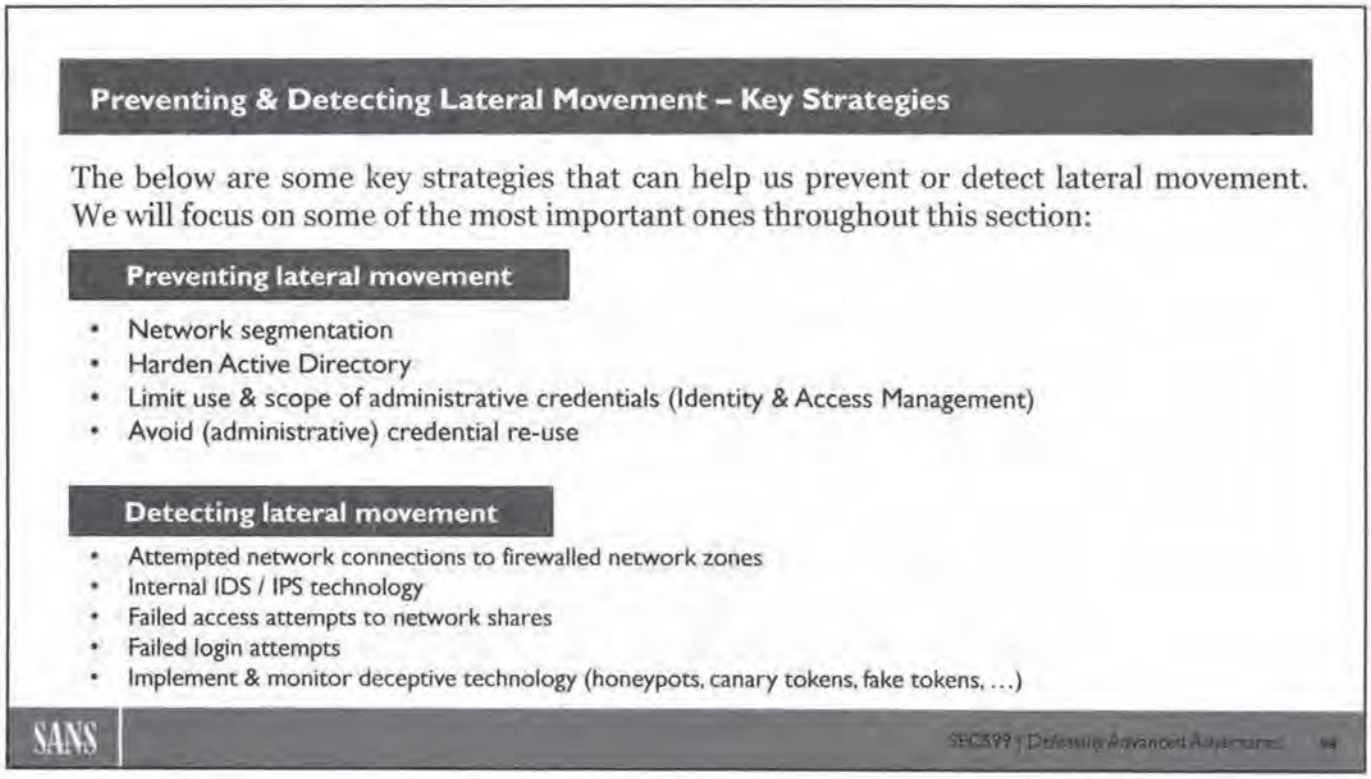


Typical Lateral Movement Example

So, what does a typical lateral movement scenario look like? It looks something like this:

- An adversary obtains initial access to the environment by compromising a workstation through phishing
- The adversary escalates privileges on the machine and steals credentials from local system
- Credentials are re-used against other internal assets
- Lack of segmentation is abused to obtain access to crown jewels / core assets

This is, of course, a bit of a simplistic view, as lateral movement could include a number of different hops before access to the cron jewels is actually obtained. It's important to note here, however, that the adversary is "trying" stuff and thus might make mistakes which could trigger alerts and could allow us to detect their activity!



Preventing & Detecting Lateral Movement – Key Strategies
The below are some key strategies that can help us prevent or detect lateral movement. We will focus on some of the most important ones throughout this section!

How can we prevent lateral movement?
- Network segmentation
- Harden Active Directory
- Limit use & scope of administrative credentials (Identity & Access Management)
- Avoid (administrative) credential re-use

How can we detect lateral movement?
- Attempted network connections to firewalled network zones
- Internal IDS / IPS technology
- Failed access attempts to network shares
- Failed login attempts
- Implement & monitor deceptive technology (honeypots, canary tokens, fake tokens, ...)

Given its primacy, nearly every large breach has leveraged Active Directory for lateral movement. We will thus pay extra attention to lateral movement strategies on Active Directory!

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

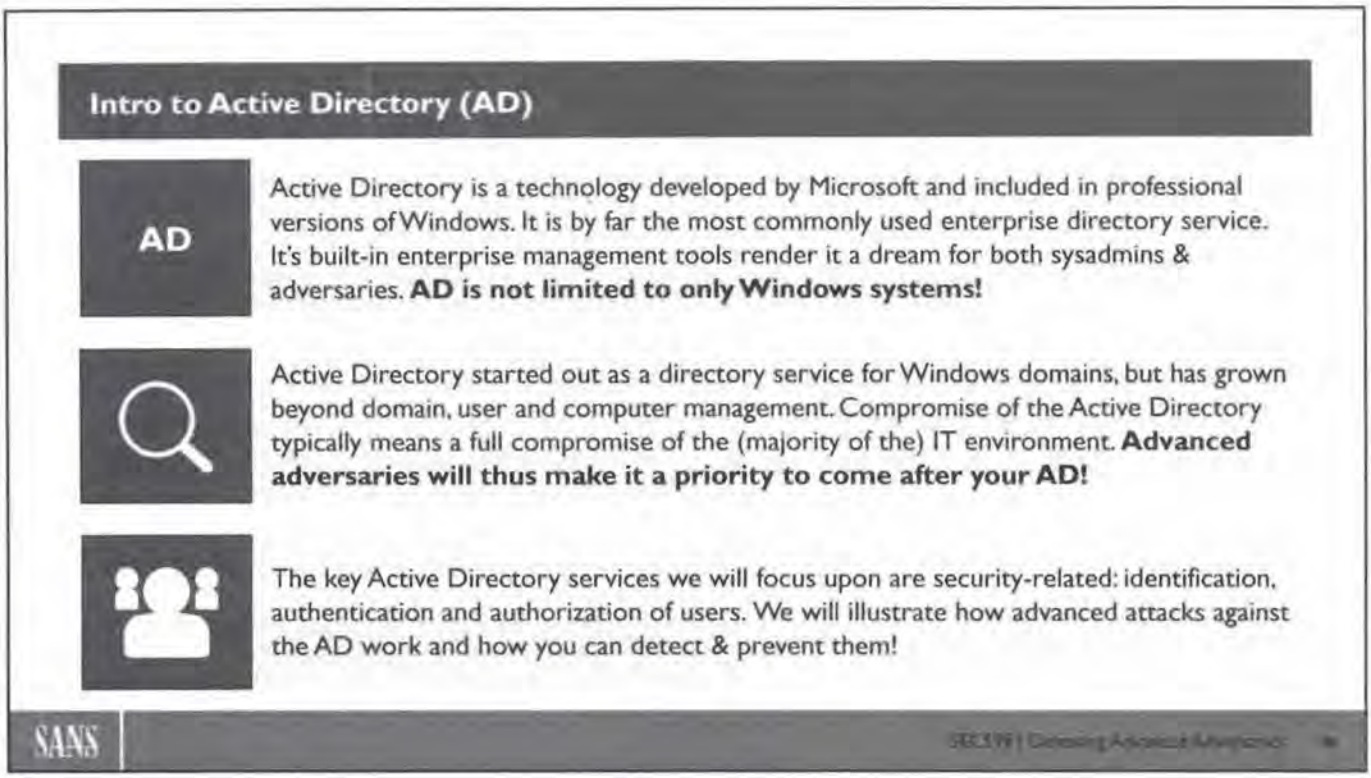
SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.



Intro to Active Directory (AD)

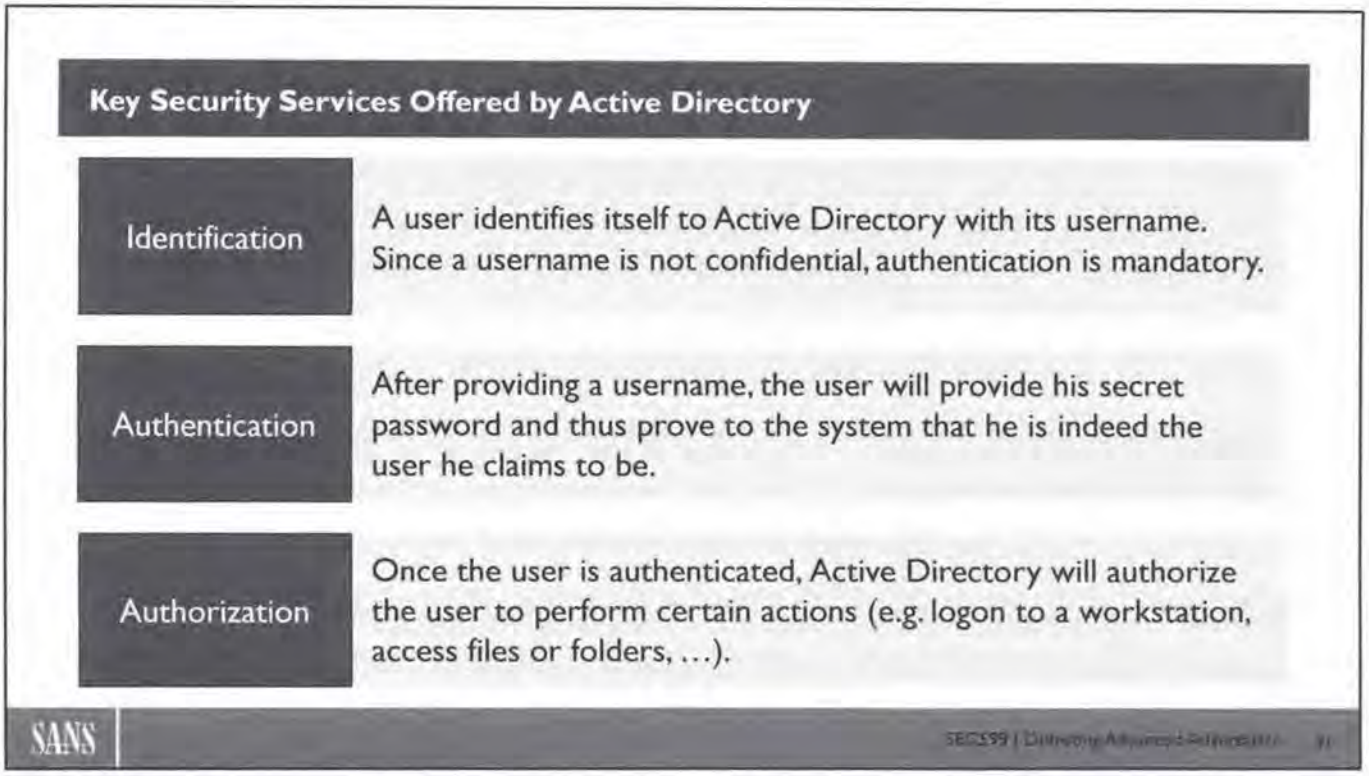
In the nineties, Microsoft introduced a professional version of Windows: Windows NT (New Technology). Windows machines (servers and workstations) were grouped in "domains". Domains are logical groups of machines. They are managed by specialized Windows machines: the domain controller (primary domain controller and backup domain controllers). Domains allowed for centralized management of users and machines.

With Windows 2000 (released in 2000), Microsoft released Active Directory. Active Directory, when it was introduced, was a directory service for Windows domains: like a phone directory, it contains meta-data about all the entities it contains: users, groups, machines, …

With Windows NT, domains were standalone: if an organization needed a domain for its production environment and a domain for its research environment, then 2 domains had to be created (with the corresponding domain controllers) and there were little facilities to establish relationships between the 2 domains, except sharing credentials. With the introduction of Active Directory, a hierarchy of domains could be built: a domain tree where trust relationships could be established between domains.

An important feature of Active Directory is Group Policies: these are various configuration settings (for computers and users) that can be configured and managed centrally and applied throughout the members of the domain.

A key feature of Active Directory is security: identification, authentication, and authorization of users (and computers). It's this feature of Active Directory that is often attacked by advanced adversaries.



Key Security Services Offered by Active Directory
When a user logs on to a machine via Active Directory, many actions are performed. The actions we are interested in, are those that pertain to security, and more specifically: identification, authentication, authorization.

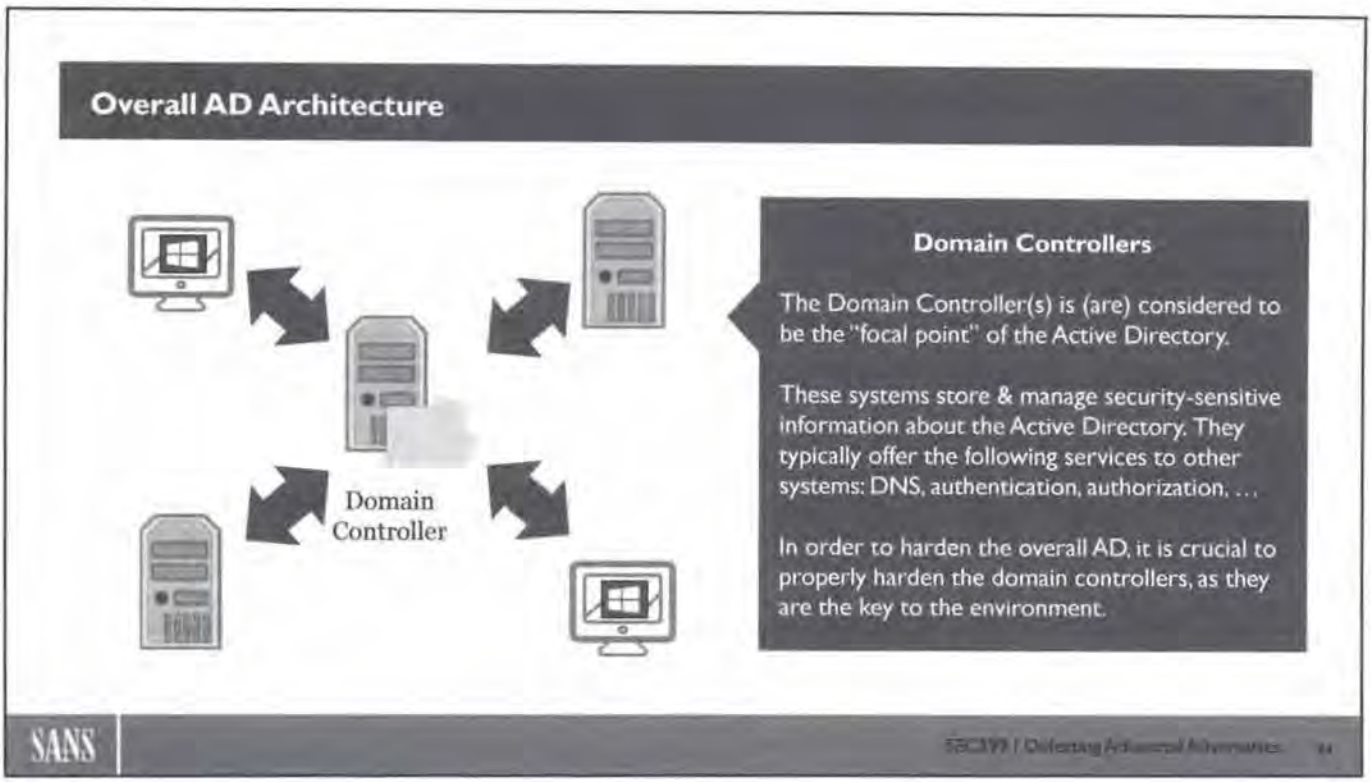
In a normal situation, when logging onto Active Directory, the user is presented with a logon screen where he has to provide his credentials: username and password. This information must be typed into the fields via a keyboard.

The username is what uniquely identifies the user: by typing his username, Active Directory knows which useraccount wants to authenticate. The username is not confidential, and it is displayed when the user types it. Since the username is not confidential, many users know the usernames of other users. In organizations that use the Active Directory username as email address, it's obvious that usernames are not confidential.

Because of the non-confidential nature of usernames, it would be easy for a user to identify as another user to Active Directory, just by typing that username. Because this is what identifies a user to active directory. However, when it comes to security, Active Directory does not blindly identify a user by his username without further checks.

To prove to Active Directory that a user is who he claims to be, the user has to authenticate by providing the second part of the credential: the password. Because a password authenticates a user, it has to be kept secret: only the user must know the password. If an adversary knows the password of a user, the adversary can authenticate to Active Directory as that user, and the adversary will obtain all the rights this user has.
When typing a password into the password field, it is kept confidential: each typed character of the password is hidden by showing an asterisk (*).

When a user provides valid credentials (an existing username with the corresponding, valid password) to Active Directory, Active Directory looks up information for this user account to decide what the account is authorized to do. For example, the account might not be authorized to logon to the computer, and then a message will be displayed and the user will not be given access to the computer.



Overall AD Architecture

When a user logs on to a machine via Active Directory, many actions are performed. The actions we are interested in, are those that pertain to security, and more specifically: identification, authentication, authorization.

In a normal situation, when logging onto Active Directory, the user is presented with a logon screen where he has to provide his credentials: username and password. This information must be typed into the fields via a keyboard.

The username is what uniquely identifies the user: by typing his username, Active Directory knows which useraccount wants to authenticate. The username is not confidential, and it is displayed when the user types it. Since the username is not confidential, many users know the usernames of other users. In organizations that use the Active Directory username as email address, it's obvious that usernames are not confidential.

Because of the non-confidential nature of usernames, it would be easy for a user to identify as another user to Active Directory, just by typing that username. Because this is what identifies a user to active directory. However, when it comes to security, Active Directory does not blindly identify a user by his username without further checks.

To prove to Active Directory that a user is who he claims to be, the user has to authenticate by providing the second part of the credential: the password. Because a password authenticates a user, it has to be kept secret: only the user must know the password. If an adversary knows the password of a user, the adversary can authenticate to Active Directory as that user, and the adversary will obtain all the rights this user has.

When typing a password into the password field, it is kept confidential: each typed character of the password is hidden by showing an asterisk (*).

When a user provides valid credentials (an existing username with the corresponding, valid password) to Active Directory, Active Directory looks up information for this user account to decide what the account is authorized to do. For example, the account might not be authorized to logon to the computer, and then a message will be displayed and the user will not be given access to the computer.

How Does Authentication in AD Work?

In an Active Directory domain, the main authentication mechanism is Kerberos.

Kerberos is a network authentication protocol based on tickets. The protocol allows 2 parties (a client and a server for example) to authenticate to each other over an insecure network channel, provided that both parties trust a third party; the Kerberos server!

The main components of in a Kerberos transaction are:

- The KDC (Kerberos Distribution Center)
- The client requesting access
- The service the client is attempting to obtain access to

When Kerberos authentication is not available, Windows reverts to NTLMv2!

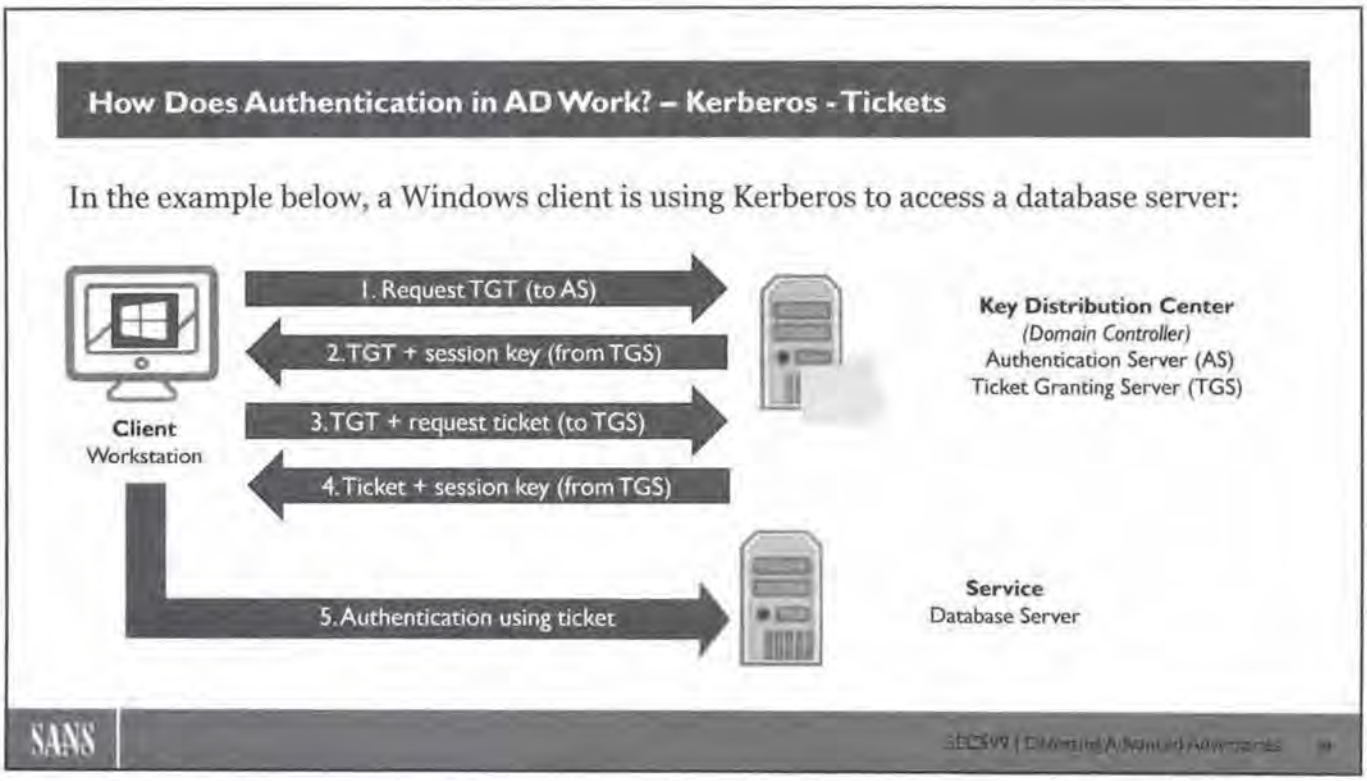
How Does Authentication in AD Work?

With Active Directory, Microsoft introduced a new authentication scheme called Kerberos. Kerberos is a network authentication protocol based on tickets, developed by MIT. The protocol allows 2 parties (a client and a server for example) to authenticate to each other over an insecure network channel, provided that both parties trust the third party: the Kerberos server. Given these three components, Kerberos is named after the three-headed mythological dog.

Each party in a Kerberos environment authenticates to the Kerberos server and receives a ticket. More precisely, this is a ticket-granting-ticket: a ticket that can be used to request tickets. When one party (client) wants to use a service from a second party (server), the first party uses its ticket-granting-ticket to obtain a ticket for the second party from the Kerberos server, and then presents it to the second party, thereby authenticating to the third party. Since both parties trust the Kerberos server, a ticket provided by the Kerberos server is trusted by both parties, leading to authentication and authorization.

Before Kerberos was introduced, pre-Active Directory Windows domains (Windows NT) relied on NTLM challenge / response authentication. NTLM provides authentication between 2 parties, without a third trusted party.

If Kerberos cannot be used (various reasons apply), Windows will fall back to NTLM authentication.



How Does Authentication in AD Work? – Kerberos - Tickets
In Active Directory, when a client successfully authenticates to a Domain Controller, the client is given a ticket-granting-ticket.

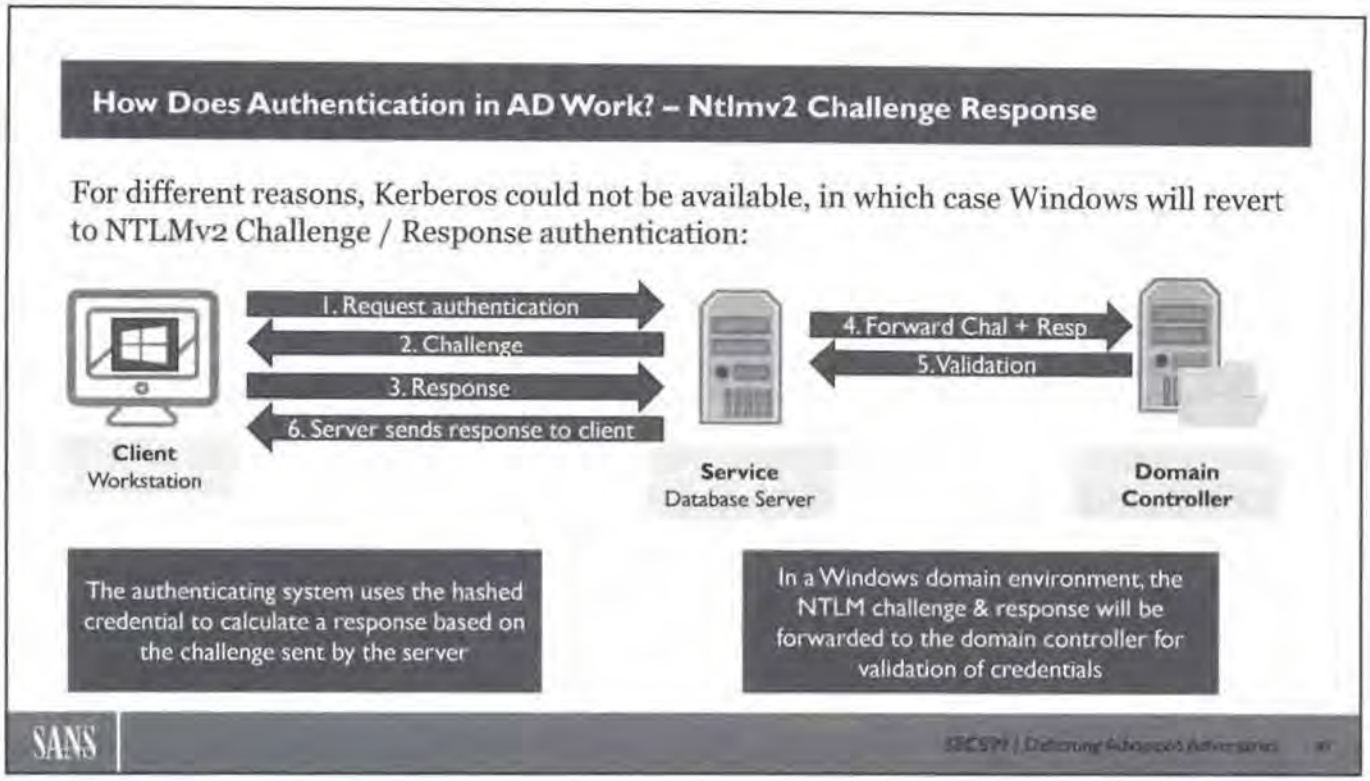
To authenticate, the user's workstation will send a request to the Authentication Server (AS) on the domain controller. This request includes the credentials (encrypted with keys obtained from the Key Distribution Center (KDC), also a service provided by a domain controller). The AS will decrypt and validate the credentials (lookup the NTLM hash) and if the NTLM hash matches, the client will be given a ticket-granting-ticket (TGT) by the Ticket Granting Service (TGS).

Kerberos works with tickets: cryptographically secured pieces of data that grant access to a service for a particular user and during a well-defined period. For example, you can receive a ticket to access a particular share on a file server.

A ticket-granting-ticket is the ticket a client receives first, and it is a special ticket: it is a ticket for the krbtgt service, and can thus be used to request other tickets. By default, a TGT is valid for 10 hours. When it expires, Windows will automatically and transparently request a new one.

Once a client has obtained the TGT, it can request access to other services, for example, a file share. To obtain this access, the client sends its TGT to the TGS along with the details of the service it wants to access. The TGS will validate the TGT, and then check Active Directory if the user is authorized to access the service. When this is the case, the TGS will issue a ticket to the client for the service it wants to access.

To access the service (for example a file share), the client will send the ticket to the service (the file server for example). Since the file server also trusts Kerberos (the TGS), it will accept tickets and grant access to the service once it has validated through cryptographic means that this is an authentic ticket issued by the TGS trusted by the service.



How Does Authentication in AD Work? – NTLMv2 Challenge Response
When Kerberos authentication is not possible, Windows will fall back to NTLM authentication.

This can even happen between machines that are members of the same domain, but when all necessary conditions to use Kerberos are not in place. For example, Kerberos works with so-called service names. If we don't have a name, Kerberos cannot be used. This is the case when we access a share on a file server by using the IP address of the server instead of its servername.

NTLM authentication is a 2-party authentication: the client and the server. It takes 3 steps:

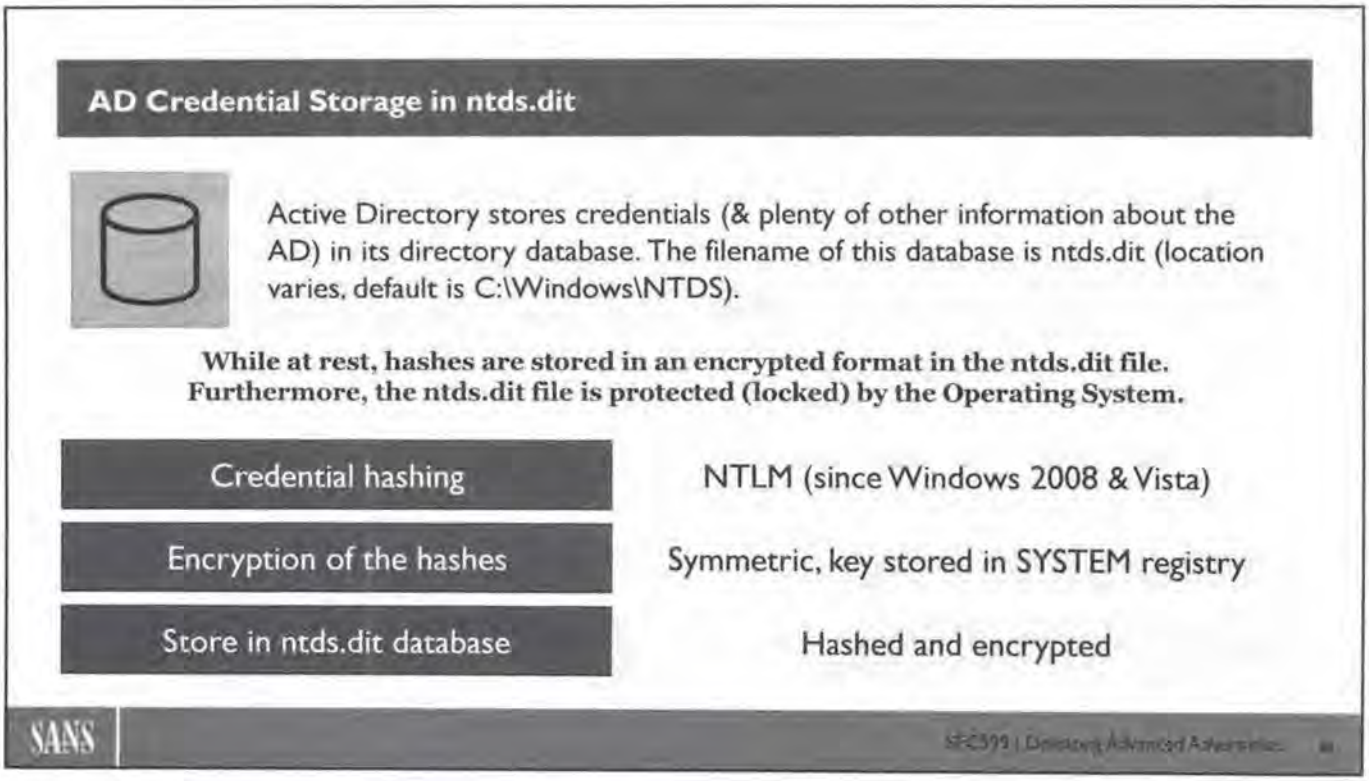
1. Negotiate
2. Challenge
3. Response

First, the client sends a negotiate packet to the server to request the authentication. There are different parameters and versions for NTLM, and the client has to inform the server what it is capable of. This is done with a negotiate packet. Next, (step 2) the server sends a challenge packet to the client. This challenge contains a so-called "nounce". A nounce is a random number of 16 bytes. Finally (step 3), the client sends the response to the server: it calculates a response based on its password and the nounce, and sends that to the server.

Using a nounce allows the 2 parties to perform authentication without having to send the password (cleartext or encrypted) over the network.

The server checks the credentials of the client by performing the same calculation as the client for the response, and if the response calculated by the server is the same as the response calculated by the client, then the client is authenticated to the server.

The server needs the credentials of the client to perform the calculation, in an Active Directory environment, the server obtains the credentials from a domain controller.



AD Credential Storage in ntds.dit
Active Directory has to store credentials in a database: the username and the password.

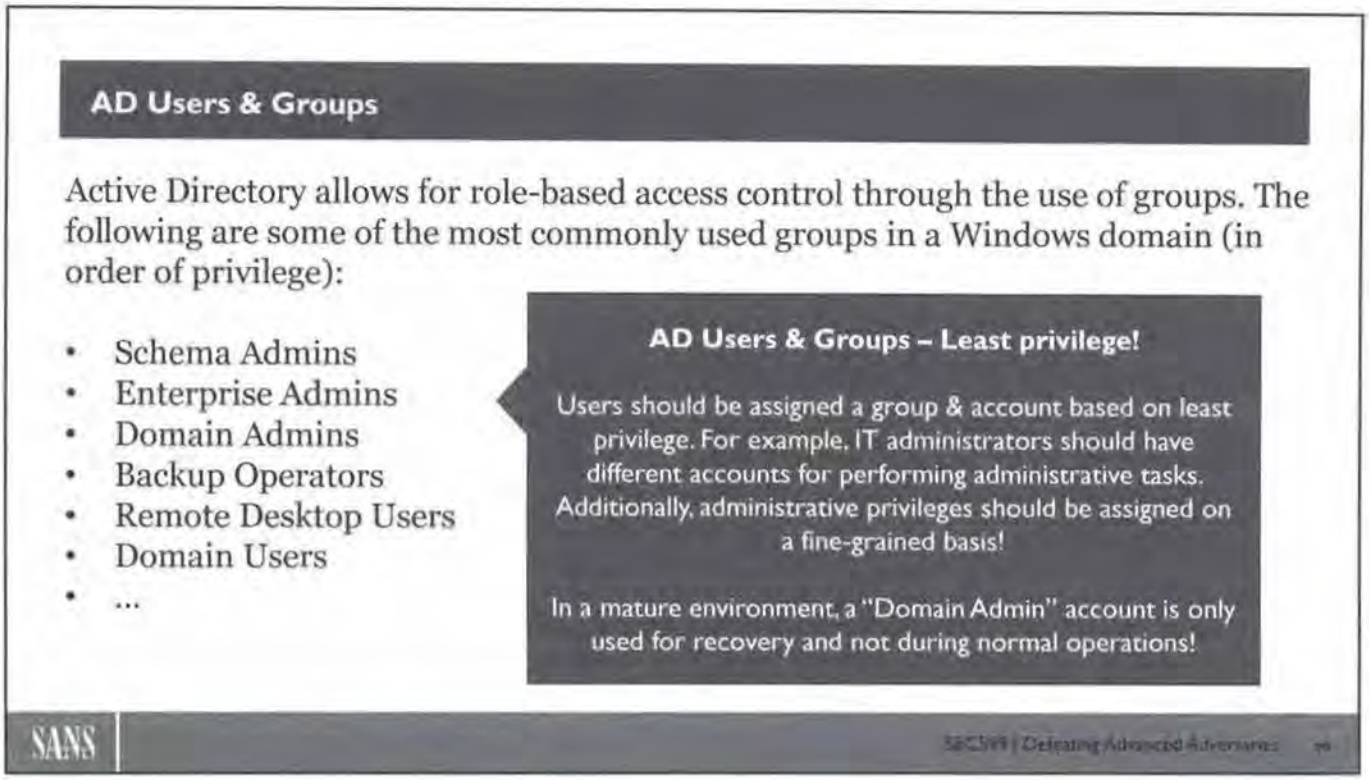
This database file is ntds.dit. This is a database in the ESE format (Extensible Storage Engine) from Microsoft, this format is also used by Microsoft Exchange for example.
By default, this file is stored in folder C:\Windows\NTDS on domain controllers.

To protect passwords, the hashed value of the passwords are stored in the ntds.dit database.
Windows uses a hashing function that Microsoft officially calls NTOWF. This stand for NTLM One-Way Function. However, in most literature on Windows passwords, the name NTLM will be used for the hashing function instead of NTOWF. We will do the same in this training: NTLM hashing function.

The NTLM hashing function is an MD4 hash of the little-endian UNICODE representation of the password. MD4 is a cryptographic hashing algorithm, it's design was later used to design the well-known MD5 hashing cryptographic hashing algorithm. MD4 (and MD5 too) is no longer a strong, cryptographic hashing algorithm, it has been broken.

On older Windows versions, another hash function will be used too; LM hashing (officially LMOWF). Storing passwords as LM hashes should be avoided at all costs (it is no longer used on modern Windows versions, and can be disabled on older versions). LM hashing is very weak.

To provide further protection, the hashes stored inside the NTDS database are also encrypted with a symmetric encryption cipher. The key used for encryption is unique to every domain controller and stored inside the SYSTEM registry hive.



Active Directory Attacks – Domain Admin Rights

Active Directory allows for role-based access control through the use of groups. The following are some of the most commonly used groups in a Windows domain (in order of privilege):

- Schema Admins: Members of this group can modify the Active Directory schema. By default, the Administrator account is a member of this group;
- Enterprise Admins: Members of this group have full control of all domains in the forest. By default, this group is a member of the Administrators group on all domain controllers in the forest.
- Domain Admins: Members of this group have full control of the domain. By default, this group is a member of the Administrators group on all domain controllers, all domain workstations, and all domain member servers at the time they are joined to the domain.
- Backup Operators: Members of this group can back up and restore files on the server, regardless of any permissions that protect those files. This is because the right to perform a backup takes precedence over all file permissions.
- Remote Desktop Users: Members of this group can log on remotely to a server.
- Domain Users: This group contains all domain users. By default, any user account that is created in the domain becomes a member of this group automatically.
- ...

Users should be assigned a group & account based on least privilege. For example, IT administrators should have different accounts for performing administrative tasks. Additionally, administrative privileges should be assigned on a fine-grained basis! Many organizations will use specific groups, based on least privilege (e.g. workstation administrators, web server administrators, database administrators, ...). In a mature environment, a "Domain Admin" account is only used for recovery and not during normal operations.

So, How About Local Users & Groups?

Next to domain-level users & groups, Windows endpoints (both servers & workstations) will also have a number of local accounts configured. The default accounts are "Administrator" & "Guest".

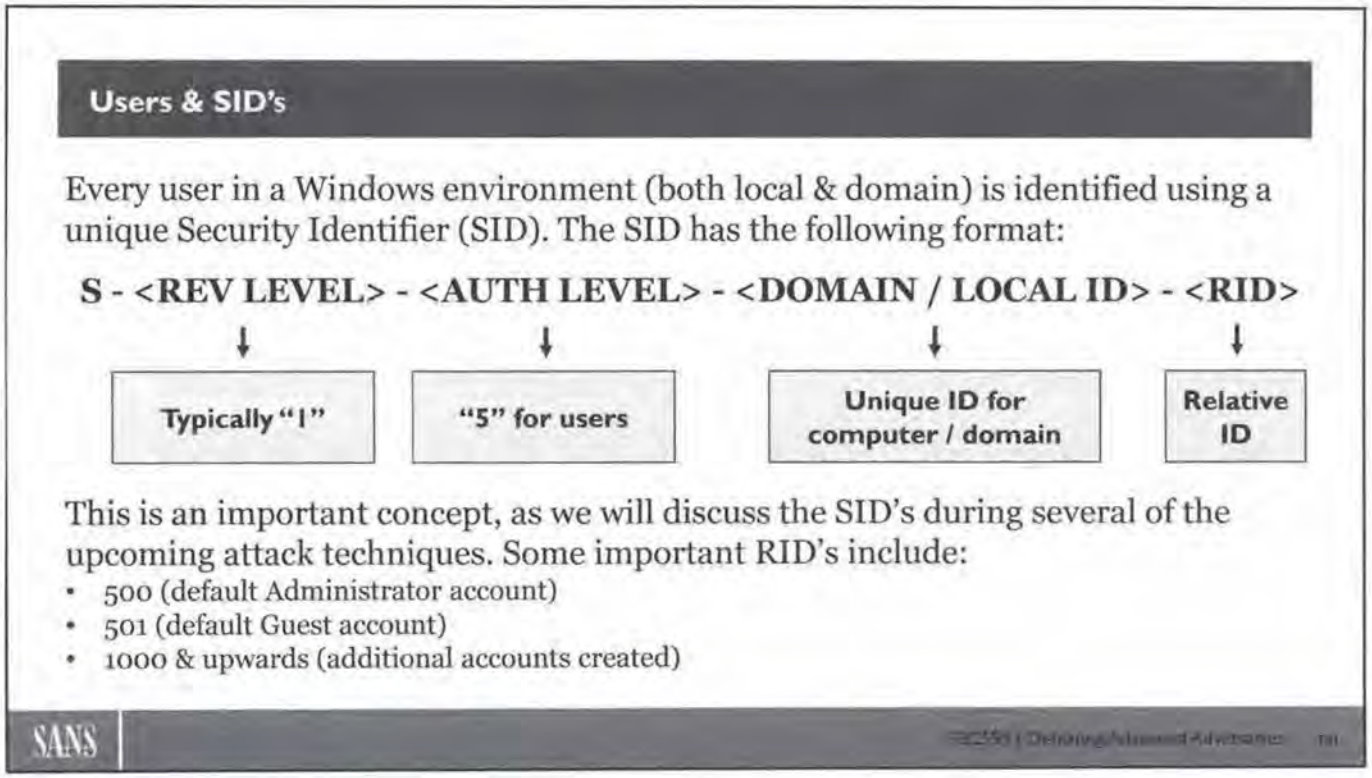
- Local accounts are difficult to manage at enterprise-level and should typically be avoided. Beware of forgotten local accounts!
- Adversaries often first obtain local administrator access, after they will use this to escalate to domain (administrative) privileges.
- Some organizations choose to set the same complex password for ALL local administrator accounts (for recovery). This is a very bad practice and should be avoided. Microsoft LAPS provides a solution.

SANS

So, How About Local Users & Groups?
Next to domain-level users & groups, Windows endpoints (both servers & workstations) will also have a number of local accounts configured. The default accounts are "Administrator" & "Guest".

Once an AD environment is set up, there's a number of items to take into account:

- Local accounts are difficult to manage at enterprise-level and should typically be avoided. Beware of forgotten local accounts! Even if local privileges have to be configured, it's better to implement & manage these centrally.
- Adversaries often first obtain local administrator access, after they will use this to escalate to domain (administrative) privileges. Again, it is vital we can limit administrative privileges as much as possible. There is typically no good reason for local end-users to have local administrator privileges.
- Some organizations choose to set the same complex password for ALL local administrator accounts (for recovery). This is a very bad practice and should be avoided. Microsoft LAPS provides a solution. Microsoft LAPS will generate a random password for the local administrator account for every machine in the network. This password is stored in the Active Directory.



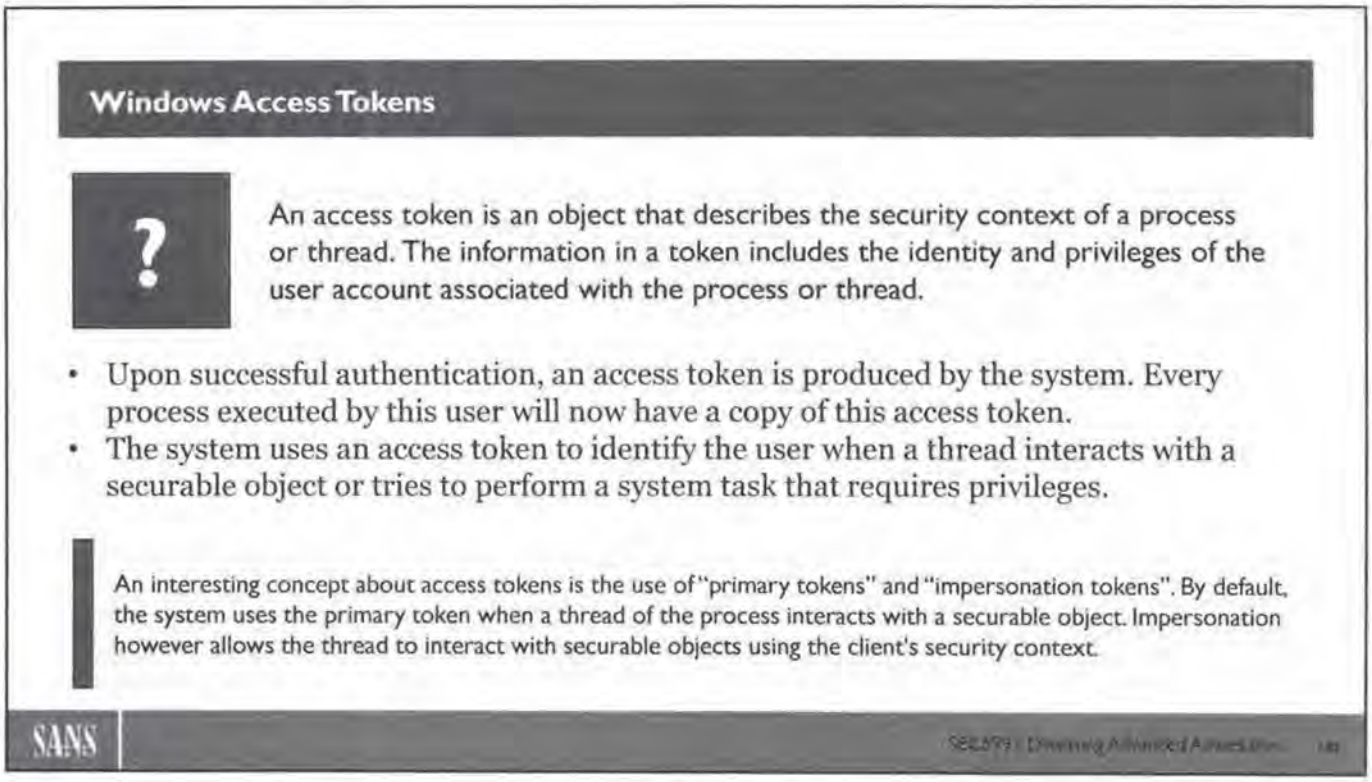
Users & SID's

Every user in a Windows environment (both local & domain) is identified using a unique Security Identifier (SID). The SID has the following format:

S - <REV LEVEL> - <AUTH LEVEL> - <DOMAIN / LOCAL ID> - <RID>

- <REV LEVEL> is typically 1
- <AUTH LEVEL> will be 5 for users
- <DOMAIN / LOCAL ID> will be a unique ID for a computer or a domain. All local users on the same machine will share the same ID. Furthermore, all domains users in the same domain will share the same ID.
- <RID> is a relative ID assigned to this specific user. Some well-known RID's include:
  - 500 (default Administrator account)
  - 501 (default Guest account)
  - 1000 & upwards (additional accounts created)

This is an important concept, as we will discuss the SID's during several of the upcoming attack techniques.



Windows Access Tokens
An access token is an object that describes the security context of a process or thread. The information in a token includes the identity and privileges of the user account associated with the process or thread.

- Upon successful authentication, an access token is produced by the system. Every process executed by this user will now have a copy of this access token.
- The system uses an access token to identify the user when a thread interacts with a securable object or tries to perform a system task that requires privileges.

An interesting concept about access tokens is the use of "primary tokens" and "impersonation tokens". By default, the system uses the primary token when a thread of the process interacts with a securable object. Impersonation, however, allows the thread to interact with securable objects using the client's security context. This opens up a number of attack vectors that can be used by adversaries!

Windows Access Tokens - Impersonation

As previously discussed, threads in a process have a security token they use by default. They can, however, also impersonate other user's security tokens. For example, a user can authenticate to a service and the service can impersonate the user's security token, allowing the service to perform actions on the user's behalf.

Introducing Single Sign-On:

- When a user authenticates to Windows in an AD environment, they can visit network resources without re-entering their credentials due to SSO
- Windows caches the credentials of the user so the user doesn't need to be prompted for his credentials again.
- The credentials are cached in the LSASS process. Windows will determine which credentials to use during SSO based on the security token of the process / thread

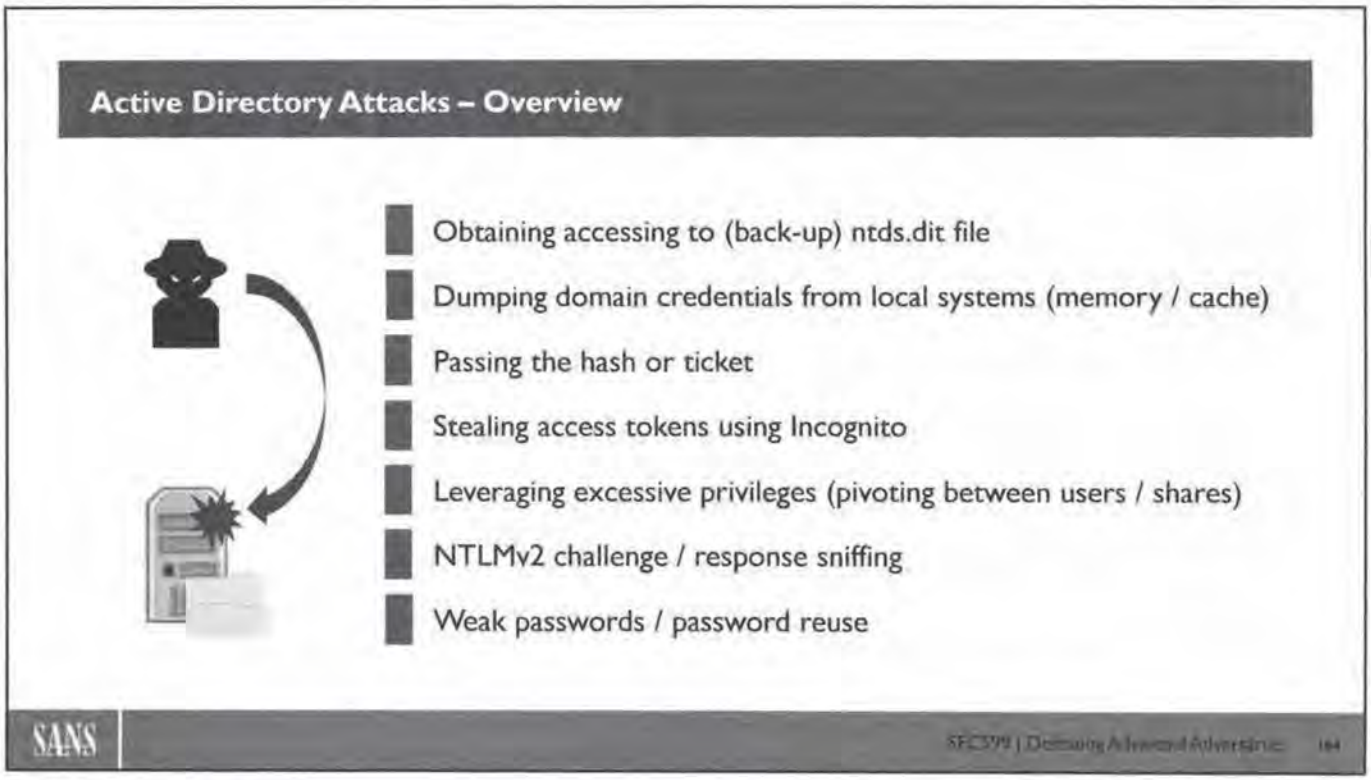
Windows Access Tokens – Impersonation

As previously discussed, threads in a process have a security token they use by default. They can, however, also impersonate other user's security tokens. For example, a user can authenticate to a service and the service can impersonate the user's security token, allowing the service to perform actions on the user's behalf.

This is for example especially useful during Single Sign-On:

- When a user authenticates to Windows in an AD environment, they can visit network resources without re-entering their credentials due to SSO
- Windows caches the credentials of the user so the user doesn't need to be prompted for his credentials again.
- The credentials are cached in the LSASS process. Windows will determine which credentials to use during SSO based on the security token of the process / thread

During the AD attacks overview, we will further discuss attacks against this mechanism!



Active Directory Attacks – Overview

Advanced adversaries will often attack the Active Directory infrastructure of their targets because they contain the "keys to the kingdom". Once they are inside, they will focus on Active Directory: if they obtain domain admin rights, they will have unlimited access to the resources of that domain. Once inside, they will also try to achieve persistence: obtain unconditional access to the administrative functions of Active Directory, even when credentials are changed.

There are many attacks possible with Active Directory, we will zoom in on some of the most common attack techniques:

- Obtaining access to (a back-up) ntds.dit file
- Dumping domain credentials from local systems (memory / cache)
- Passing the hash or ticket
- Stealing access tokens using Incognito
- Leveraging excessive privileges (e.g. pivoting between users / shares)
- NTLMv2 challenge / response sniffing
- Weak passwords / password reuse

AD Attack #1 – Obtaining Access to Back-Up NTDS.dit File (1)

As discussed before, credentials in the Active Directory are stored in the ntds.dit file (encrypted with system key). This file is only accessible with elevated privileges. Furthermore, it is **locked by the OS** while the domain controller is running.

Different techniques could be used to obtain access to the file:

- Should the adversary have already obtained administrative access to the domain, he could use tools such as the Volume Shadow Copy Service to create a read-only copy and steal the file;
- Badly secured backups of the domain controller drives (e.g. open network shares) could allow an adversary to extract the file without administrative privileges;
- Specialized, open-source, tools can be used to extract hashes from the ntds.dit file. These hashes can be used in subsequent Pass-the-Hash attacks or even be cracked using offline password crackers

SANS

AD Attack #1 – Obtaining Access to Back-Up NTDS.dit File (1)

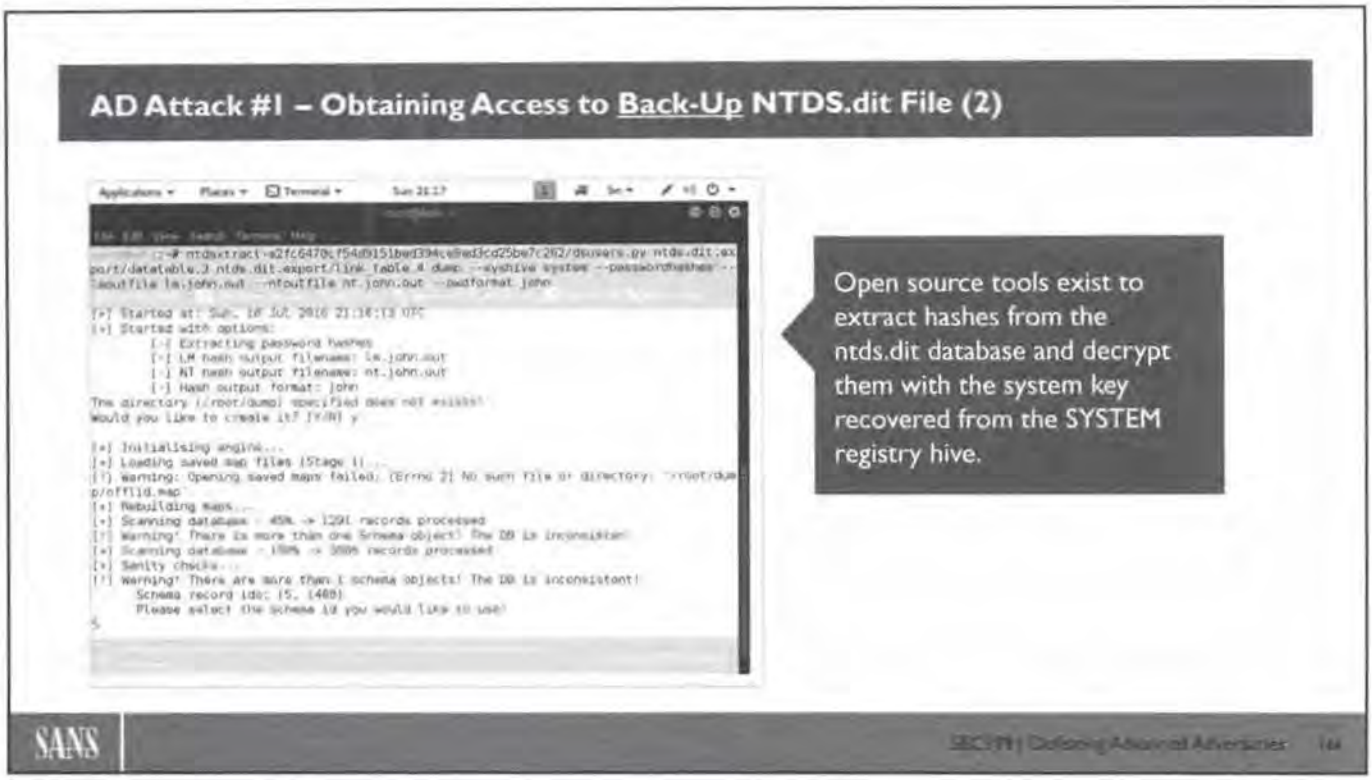
Advanced adversaries will often attack the Active Directory infrastructure of their targets because they contain the "keys to the kingdom". Once they are inside, they will focus on Active Directory: if they obtain domain admin rights, they will have unlimited access to the resources of that domain. Once inside, they will also try to achieve persistence: obtain unconditional access to the administrative functions of Active Directory, even when credentials are changed.

There are many attacks possible with Active Directory. Here we want to focus on attacks to obtain credentials and/or access. As we saw, the Active Directory database is stored inside a file called ntds.dit. And the hashes are protected by an encryption key stored in the system registry.

When an adversary obtains a copy of these files, he can extract hashes and recover passwords, amongst other things.

These files are present on a domain controller, but when the domain controller is up and running, these files are in use and cannot be copied. There is a workaround, however; shadow copies. Shadow copies (aka VSS, Volume Snapshot Service) is a Microsoft technology to create local backups of files. This technology can be used to recover backup copies of the ntds.dit and SYSTEM files.

adversaries don't always need access to a domain controller to obtain these files. When centralized backups are made of the domain controllers, these files will be found on the backup servers too. It is important to adequately protect these files, even in backups.



AD Attack #1 – Obtaining Access to Back-Up NTDS.dit File (2)

Once an adversary has obtained a copy of the ntds.dit and SYSTEM files, he can proceed to the extraction of hashes and recovery of passwords. There are several open source tools that can read these files and decrypt hashes: examples are ntdsxtract and secretsdump. For large databases (ten thousand and more users), ntdsxtract tends to be slow (can take several days), but secretsdump is much faster (a couple of hours). Both can output the NTLM hashes and LM hashes (when present) in different formats.

In this example, we output the NTLM and LM hashes in a file format that can be read into John the Ripper, a popular CPU based password cracker. A password cracker is a tool that takes hashes as input and tries to recover the cleartext password by performing dictionary attacks, brute force attacks, and blended attacks. A blended attack is a dictionary attack where password transformation rules are used to try out different combinations. For example, if a dictionary contains the password secret, then in a blended attack, variants like secret1 would also be tried. This is governed by a rule that appends 1 to each password in the dictionary, for example.

AD Attack #1 – Obtaining Access to Back-Up NTDS.dit File (3)

In this example, we extracted LM hashes and NTLM hashes in a format that can be fed to the password cracker John the Ripper.

AD Attack #1 – Obtaining Access to Back-Up NTDS.dit File (3)

In the example above, we can see the first 5 lines of the LM and NTLM hashes for password cracking with John the Ripper.

Each credential is stored in a line, and : is used as a field separator.

The first field is the username (We see Administrator, user01, user02, …), the second field is the LM hash, and the third field is the SID. The SID is the Security Identifier of a user (or computer). It is a unique number that never changes (a username can be changed). This is what Windows uses internally to identify users and computers.

We see that the LM hash is just a string of hexadecimal characters, while the NTLM hash is a string of hexadecimal characters prefixed with NT. This is how John the Ripper works: a string of 32 hexadecimal characters is assumed to be an LM hash, while other hashes of 32 hexadecimal characters (like NTML) need to be prefixed with a string that indicates their type: NT for NTLM.

John the Ripper is a popular password cracker that can perform dictionary and brute force using the CPU of the computer.

Another popular password cracker is Hashcat. Hashcat can use the GPU (Graphical Processing Unit, e.g. graphical card) of computers, and be much faster than John the Ripper because of the parallel computing features of GPUs.

AD Attack #2 – Dumping Domain Credentials from Local Systems

Once adversaries obtain administrative access to a local system (e.g. through one of the privilege escalation techniques we discussed before), he could now leverage this to steal local credentials:

Windows endpoints by default store the last 10 used domain credentials in DCC format. This is used to authenticate known users when a connection to the domain controller cannot be established. These can be extracted by for example Metasploit.

While domain users are authenticated to a system, their credentials (hashes and sometimes clear-text passwords) are available in memory (lsass.exe process). These can be extracted using well-known tools such as Mimikatz.

SANS

AD Attack #2 – Dumping Domain Credentials from Local Systems

Once adversaries obtain administrative access to a local system (e.g. through one of the privilege escalation techniques we discussed before), he / she could now leverage this to steal local credentials:

- Windows endpoints by default store the last 10 used domain credentials in DCC format. This is used to authenticate known users when a connection to the domain controller cannot be established. These can be extracted by for example Metasploit. As an important side note, it's important to understand that this is a dedicated format for caching (and can thus not be used / abused in the same format as an NTLM hash).

- While domain users are authenticated to a system, their credentials (hashes and sometimes clear-text passwords) are available in memory (lsass.exe process). These can be extracted using well-known tools such as Mimikatz.

Again, both of these attack strategies require local administrative access to a system that is already part of the domain.

AD Attack #2 – Dumping Domain Credentials from Local Systems (Mimikatz) (1)

Mimikatz is a free, open-source Windows tool built by Benjamin Delpy (@gentilkiwi) to extract credentials from Windows computers. Its second version is often referred to as "Kiwi".

"Mimikatz is a tool I've made to learn C and make somes experiments with Windows security. It's now well known to extract plaintexts passwords, hash, PIN code and kerberos tickets from memory. Mimikatz can also perform pass-the-hash, pass-the-ticket or build Golden tickets."

Due to its high reliability & flexibility, it is used by adversaries and penetration testers alike. Several variations have been created and it has been included as a module in the Metasploit Meterpreter attacking tool.

AD Attack #2 – Dumping Domain Credentials from Local Systems (Mimikatz) (1)

Mimikatz is a tool that has many features and functions, for example extracting hashes from the LSA process lsass.exe. It is a free, open-source Windows tool, developed by Benjamin Delpy (Twitter @gentilkiwi). It has many features:

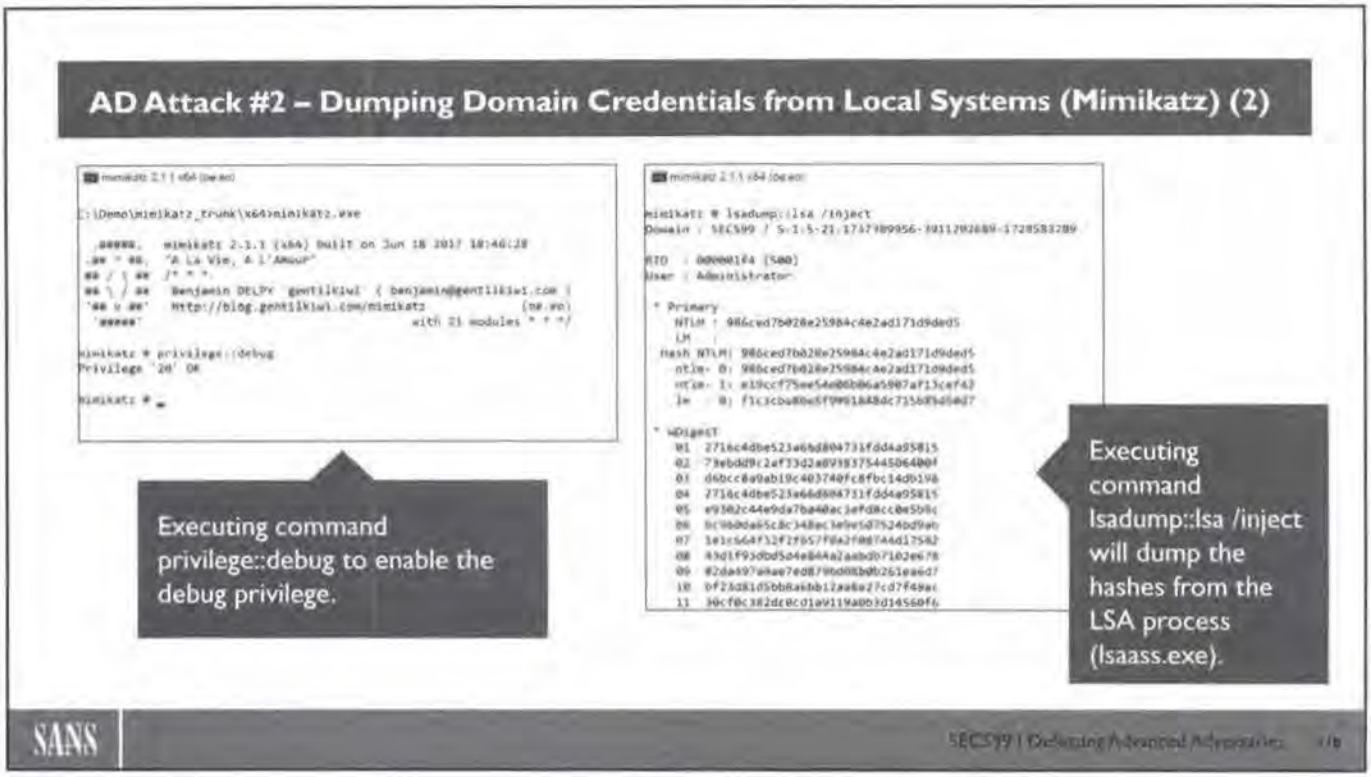
- Extracting hashes
- Extracting passwords
- Extracting tickets
- Executing Pass-the-Hash attacks
- Executing Pass-the-Ticket attacks
- Generating golden tickets
- …

Several of these features will be explained later. Because of all these features, and constant updates with new features and support for new Windows versions, Mimikatz has become the most popular credential tool used by red teams and adversaries.

Mimikatz has 3 components (in 32-bit and 64-bit versions):

1. Mimikatz.exe: this is the executable, and the console that interacts with the user
2. Mimilib.dll: this is the dll
3. Mimidrv.sys: this is the kernel driver, necessary for features that require access or modifications to kernel data

Because it is very powerful and open source, it has been transformed by hackers and malware authors for various purposes. A lot of anti-virus programs detect the mimikatz.exe executable. Because this poses a problem to pentesters, file-less versions have been developed, that launch directly into memory from various scripting platforms, like PowerShell.



AD Attack #2 – Dumping Domain Credentials from Local Systems (Mimikatz) (2)
When mimikatz is launched, a banner is displayed and the user can start to type commands. Commands are grouped into modules. A command has to be prefixed with the module name followed by :: (except for the standard module). In the example above, command privilege::debug is executed. privilege is the name of the module, and debug is the name of the command.

A normal user can access all the processes (including their memory) that run under his user account. He cannot access the memory of processes running under other users, like system for example. An administrator can access all the processes, including processes running under other accounts. This is possible because administrators have the debug privilege. Having this privilege is a mandatory requirement to access the memory of system processes, for example.

Besides having the privilege, the privilege also has to be "enabled" before one can use this privilege. Many tools (like debuggers) do this automatically, but with mimikatz, it is optional. The command privilege::debug has to be executed to enable the debug privilege.
Module lsadump contains several commands to extract information from the Local Security Authority process. The lsa command will extract hashes and other secrets from the LSA process. It requires admin rights, and the debug privilege must be enabled.

There are two methods to extract hashes from the LSA process. Option /patch will modify Windows code to be able to dump hashes, and option /inject will inject extra code into the LSA process to dump hashes.

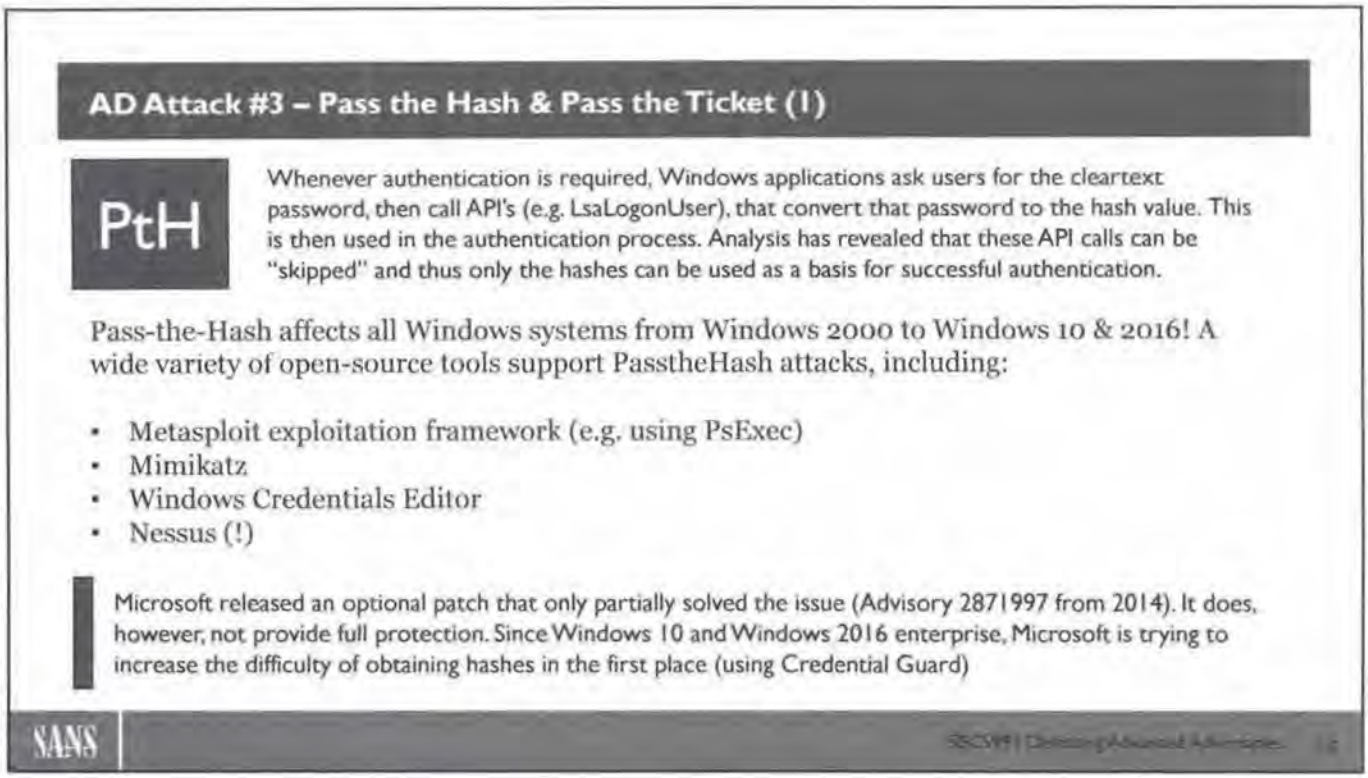
In the screenshot above, we see the use of Mimikatz's lsadump::lsa command with option /inject on a domain controller. This command will dump hashes for all users in the Active Directory domain managed by this domain controller.

The first line gives us the domain: SEC599, together with the SID of the domain. The Security ID of the domain is the string that starts with S-1-5-21- …

After that, Mimikatz dumps hashes for each user. The RID is the Relative ID and is appended to a base SID. In this example, it is 500 because the account is the Administrator account (local Administrator accounts and Domain Administrator accounts have RID 500). The SID of the Administrator account is the concatenation of the domain SID and the users' RID: S-1-5-21-…-500.

Then the primary credentials are dumped. In this example, we have the NTLM hash (986CED7B028E25984C4E2AD171D9DED5) and the LM hash (). The LM hash is empty because this is a Windows 2016 server domain controller: by default, storing LM hashes is turned off.

The following NTLM hashes are the password history hashes (used to enforce password rotation).



AD Attack #3 – Pass the Hash & Pass the Ticket (1)

If we go back to the explanation of NTLM authentication (challenge/response), we remember that it is actually the NTLM hash of the password that is used to calculate the response, and not the password itself.

This means that if we can obtain the hash of a password (and not the password itself), we still must be able to authenticate. A pass the hash attack is exactly this: a stolen hash (for example extracted from the Active Directory database) is used to authenticate.

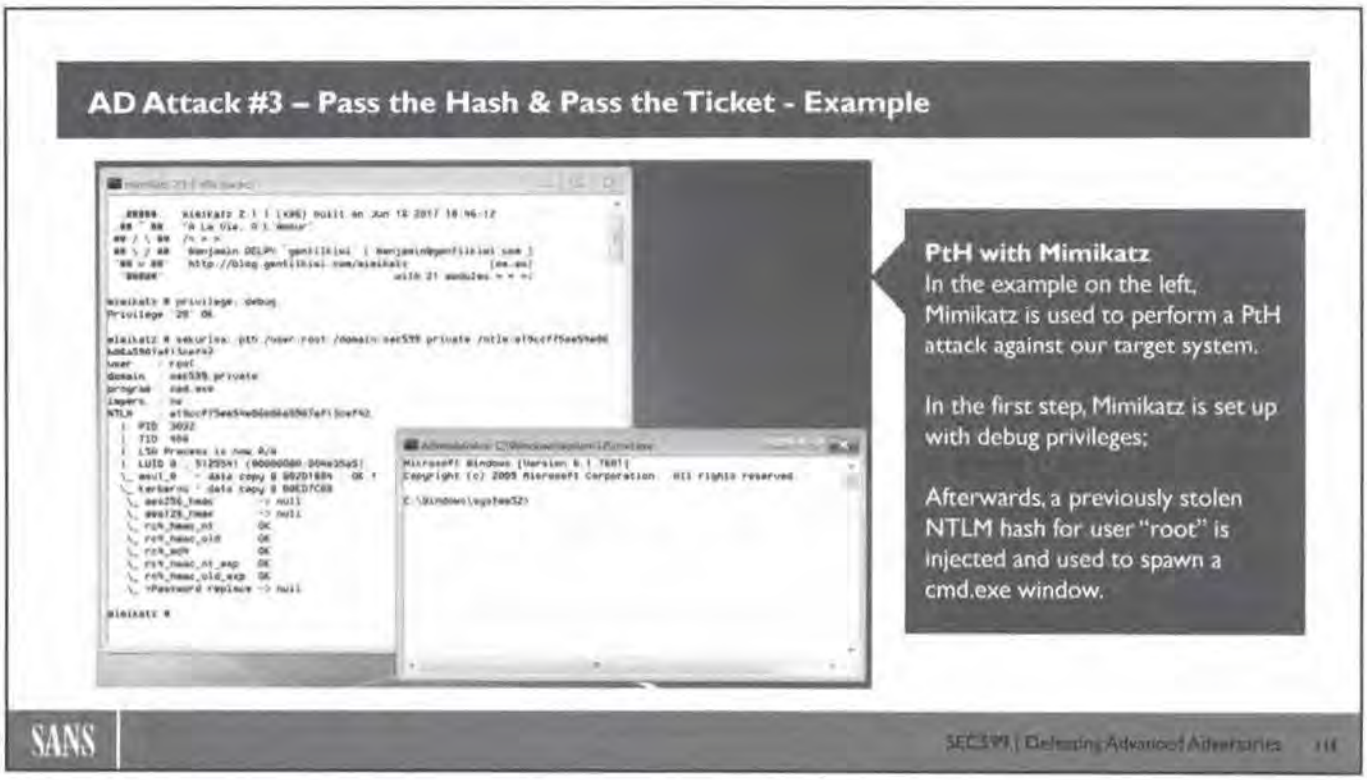
This cannot be done just using built-in Windows utilities or commands; there is no command (for example net use) that will take a hash as an argument instead of a password.

But there are adversary tools available that allow the use of a hash to authenticate (for example Windows Credential Editor and Mimikatz). These fall into two categories: tools that implement the NTLM authentication algorithm and take a hash as input, and tools that inject a hash into Windows memory for use by build-in Windows NTLM authentication. The advantage for an adversary to use a pass the hash attack is that the password is not required (e.g. time-consuming password cracking is not necessary).

Protecting against pass-the-hash attacks is virtually impossible: the hash is used in NTLM authentication and is the only secret necessary to calculate the response. It can only be protected against by preventing the use of NTLM authentication, or by making sure hashes cannot be obtained. Kerberos authentication with tickets is the authentication mechanism that replaces NTLM. Unfortunately, there are many cases where Kerberos authentication cannot work and then Windows falls back to NTLM authentication. It is possible to disable NTLM authentication via the registry so that only Kerberos authentication can be used, but in our experience, this is not a viable option for corporate networks. Corporate infrastructure has too many "legacy" systems and applications that require NTLM authentication.

Microsoft released an optional patch that only partially solved the issue (Advisory 2871997 from 2014). It does however not provide full protection, as it only protects against PtH attacks focused on local accounts (so not domain-level accounts) AND not the default RID-500 local administrator account.

With Windows 10 Enterprise and Windows 2016 Enterprise, Microsoft introduced a technology called Credential Guard: with Credential Guard properly implemented, typical hash extraction from the memory will be stopped. Note that this requires that all systems in a corporate environment use Windows 10/2016 enterprise without a single exception, otherwise hashes can be stolen from those systems that do not use Windows 10/2016 enterprise.



AD Attack #3 – Pass the Hash & Pass the Ticket – Example
Mimikatz is a tool that allows user to execute pass-the-hash attacks by injecting hashes into Windows memory.

In the example above, Mimikatz is executed as administrator and the debug privilege is enabled (privilege::debug command). This is necessary because Mimikatz will write into the LSA process to inject the hash. This cannot be done without administrative rights.

The Mimikatz command to start a pass-the-hash attack is sekurlsa::pth. It has 3 mandatory arguments: the username (root in our example), the domain name (sec599.private in our example) and the NTLM hash (E19CCF75EE54E06B06A5907AF13CEF42 in our example, which is the NTLM hash of P@ssw0rd).

With this information, Mimikatz will launch a command-line process (cmd.exe) while opening the LSA process memory to inject the hash we provided for the security tokens used by the cmd.exe program.

This cmd.exe program is running with the injected credentials, and if we would use a "net use *" command (without providing credentials) to connect to a remote share, the NTLM challenge/response would be executed with the NTLM hash we injected, and thus execute a pass-the-hash attack.

Programs started by this cmd.exe process will also inherit the injected credentials.

The sekurlsa::pth command can also be used to start other programs than cmd.exe.

Credential Guard will protect against pass-the-hash attacks with tools that inject the hash into memory because reading/writing the credentials in memory is no longer possible (cfr. next chapter). But of course, this requires that all your Windows machines run Windows 10/2016 Enterprise.

AD Attack #3 – Pass the Hash & Pass the Ticket (2)

PtT

In a pass the ticket attack, access is gained to a resource of a system (for example the administrative share) by using a Kerberos ticket that was generated or obtained from a compromised machine

PtT with Mimikatz
Pass-the-Ticket affects all Windows platforms relying on Kerberos. A good example of a tool that support Pass-the-Ticket attacks is Mimikatz!

In the screenshot on the left, we can see Mimikatz in use on a compromised machine, where it is attempting to export & store available tickets.

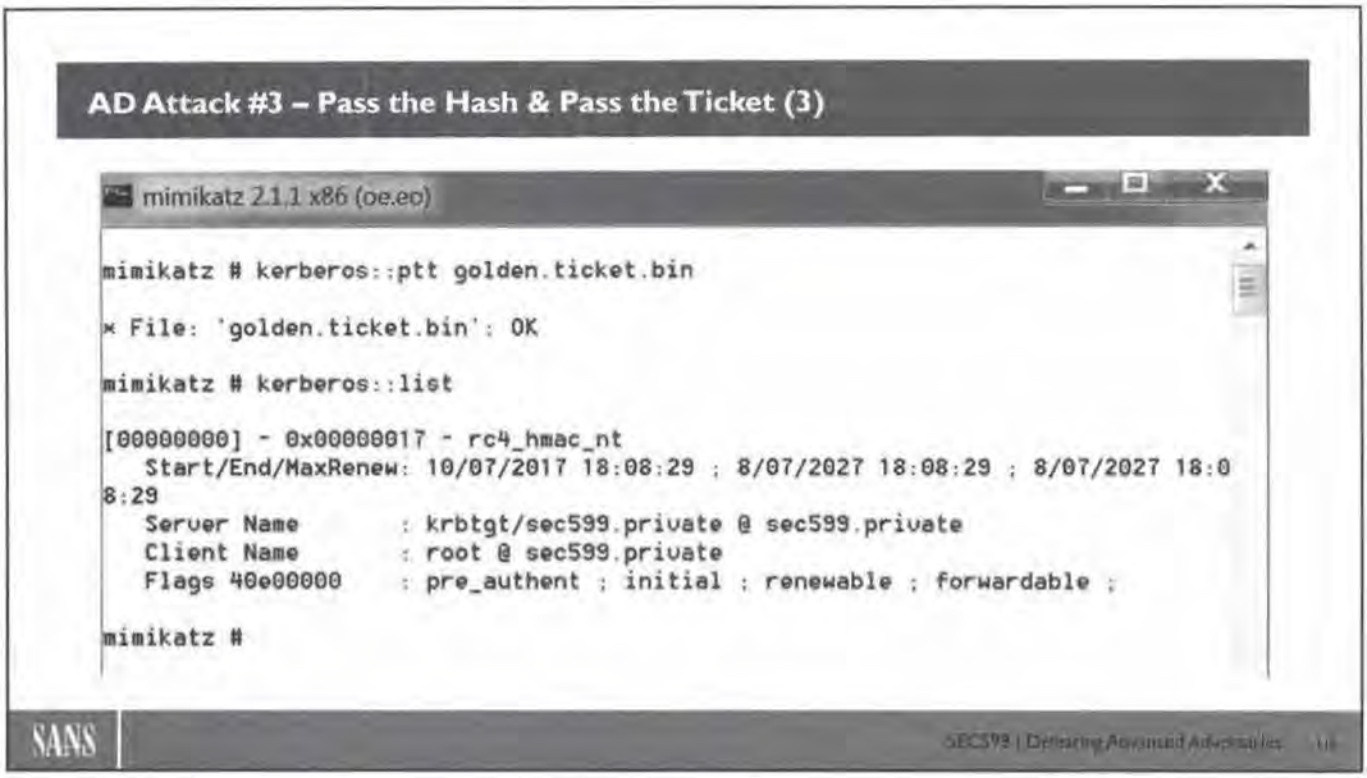
AD Attack #3 – Pass the Hash & Pass the Ticket (2)

Since tickets can be used to obtain access to a service or request other tickets, it should be no surprise that adversaries found out ways to steal and reuse tickets. Such an attack is called a pass-the-ticket attack. adversaries obtain existing tickets (preferably ticket-granting-tickets from administrative accounts) by extracting them from the memory of compromised machines, and then use them to gain access to other machines. There are open source attack tools available to extract tickets and execute pass-the-ticket attacks.

Abusing a ticket implies that we are abusing Kerberos authentication and authorization: the advantage for an adversary to use an attack with a ticket is that he does not require the password or the hash of the compromised account. Just a ticket.

With Mimikatz, we can display the TGT by using command kerberos::tgt. As you can see from the output, the service name is krbtgt for domain sec599.private (hence it is a TGT) and the client name is administrator at domain sec599.private. So, this is a very valuable ticket to steal: it is the TGT of the domain administrator. This ticket will provide access to all resources of the domain. From the start and end time, we can see that this ticket is valid for 10 hours. To obtain this ticket, the adversaries need to have (local) administrative access to a machine where the domain administrator is logged on. This ticket can be exported to a file, for example. For this we issue the command "kerberos::list /export": this will export all tickets to a file (one for each ticket) in the current directory.

We cannot select the name for the tickets, it is Mimikatz that decides the name based on metadata like the username. Tickets generated by Mimikatz use the extension .kirbi, but any extension can be used for a pass-the-ticket attack.



AD Attack #3 – Pass the Hash & Pass the Ticket (3)

Once we have exported the tickets, we can now also load them: Executing a pass the ticket attack with Mimikatz is easy. In the example above, we have a ticket in the file golden.ticket.bin. To use this ticket, we issue the command Kerberos::ptt golden.ticket.bin. This injects the ticket into Windows' memory, ready to be used when we attempt to connect to a service.

The Kerberos::list command can be used to list all the tickets that we have. We can see that the servername is krbtgt/sec599.private: this tells us that this is a ticket-granting ticket for the sec599.private domain. The user is root of the sec599.private domain.

If we look at the start and end dates, we see 2017 – 2027. That's an extremely long period for a ticket: 10 years (remember, the default lifetime of a TGT is 10 hours). That's because here, we used a Golden Ticket (more details on this later).

Now when we try to connect to a share for example, this injected ticket will be used to obtain a ticket to the share we want to access (this requires Kerberos, thus we need to use the fileserver name to access, and not its IP address, as this would result in NTLM authentication which does not work with tickets).

AD Attack #4 – Stealing Access Tokens Using Incognito (1)

Windows access tokens are at the heart of Microsoft's authentication & SSO model. They are managed by the Local Security Authority Subsystem Service (LSASS). Once the adversary obtains administrator privileges, they can steal available tokens in LSASS!

- Token-stealing techniques were explained in detail by Luke Jennings (2008 - MWR InfoSecurity). He also developed "Incognito", a tool that can facilitate the extraction of tokens from LSASS once local administrator privileges are obtained.
- These tokens can be stolen and subsequently reused against other systems in the network!
- As many others, Incognito has been built in in the Metasploit framework (as part of the Meterpreter)!

Want to know more? An interesting local privilege escalation technique in Windows will allow adversaries to escalate from "service accounts" to "NT AUTHORITY\SYSTEM", thereby "generating" & stealing tokens! The technique has been dubbed "Rotten Potato".

SANS

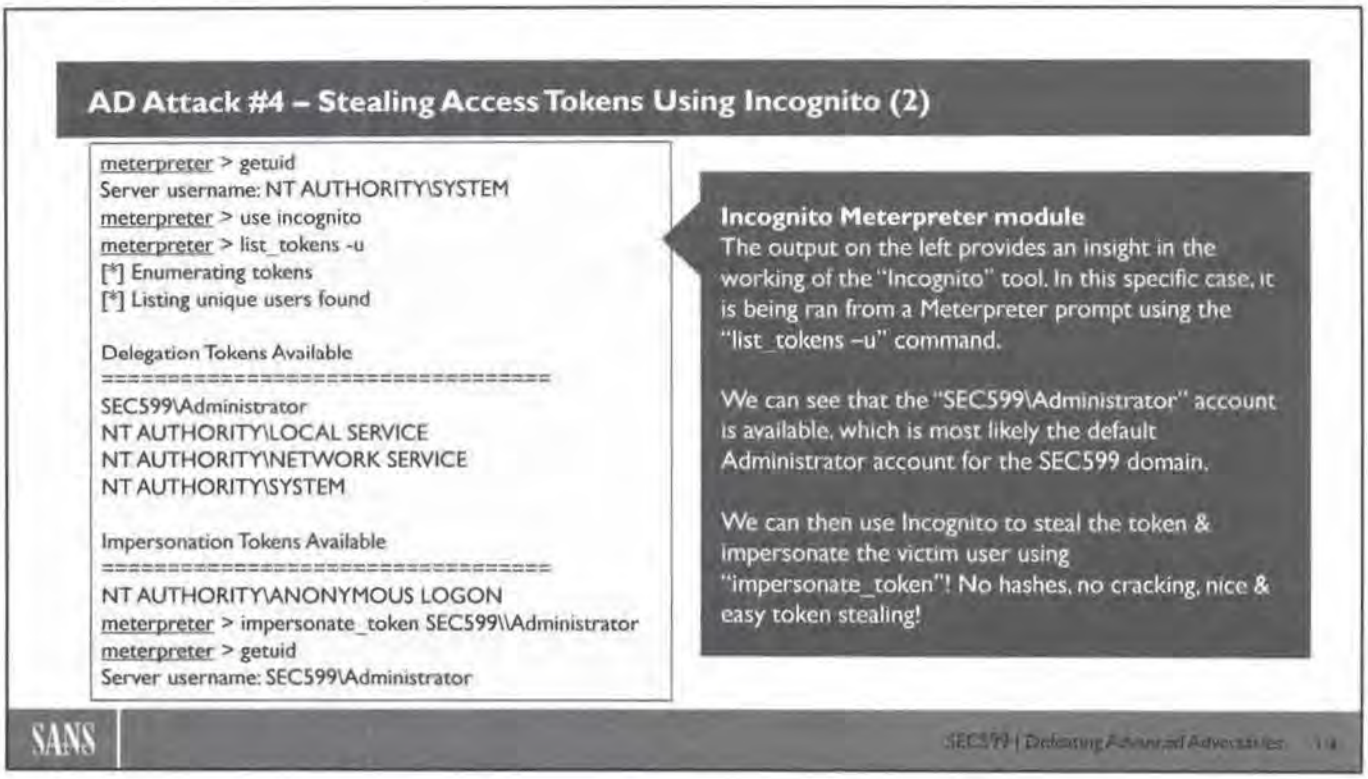
AD Attack #4 – Stealing Access Tokens Using Incognito (1)

Windows access tokens are at the heart of Microsoft's authentication & SSO model. Please refer to the previous sections of this course for additional information on how Windows Access Tokens are used by Microsoft. In short, tokens are managed by the Local Security Authority Subsystem Service (LSASS). Once the adversary obtains administrator privileges, they can steal available tokens in LSASS!

- Lule Jennings (MWR InfoSecurity) published an in-depth analysis of token-stealing techniques in 2008. He also developed "Incognito", a tool that can facilitate the extraction of tokens from LSASS once local administrator privileges are obtained;
- These tokens can be stolen and subsequently reused against other systems in the network!
- As many others, Incognito has been built in in the Metasploit framework (as part of the Meterpreter)!

Want to know more? An interesting local privilege escalation technique in Windows will allow adversaries to escalate from "service accounts" to "NT AUTHORITY\SYSTEM", thereby "generating" & stealing tokens! The technique has been dubbed "Rotten Potato". For additional information on the "Rotten Potato" attack, please refer to the following article:

https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/

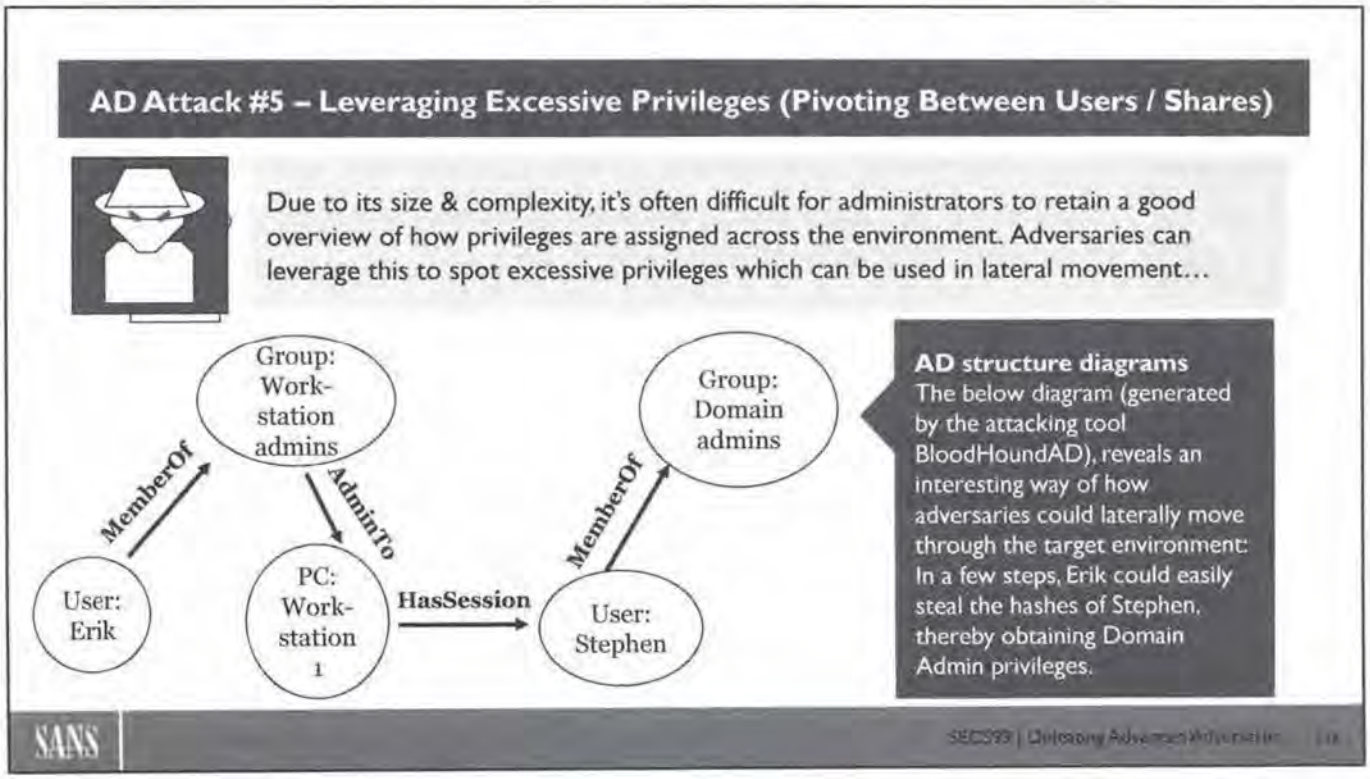


AD Attack #4 – Stealing Access Tokens Using Incognito (2)

The output on the slide provides an insight into the working of the "Incognito" tool. What we can see is the following:

- The adversary checks his current privileges on the system using the "getuid" command
- The adversary loads the incognito module in the meterpreter
- The available access tokens are listed using "list_tokens -u"
- The "SEC599\Administrator" token is found & impersonated using "impersonate_token"
- The adversary confirms his stolen privileges using the "getuid" command

No hashes, no cracking, nice & easy token stealing!



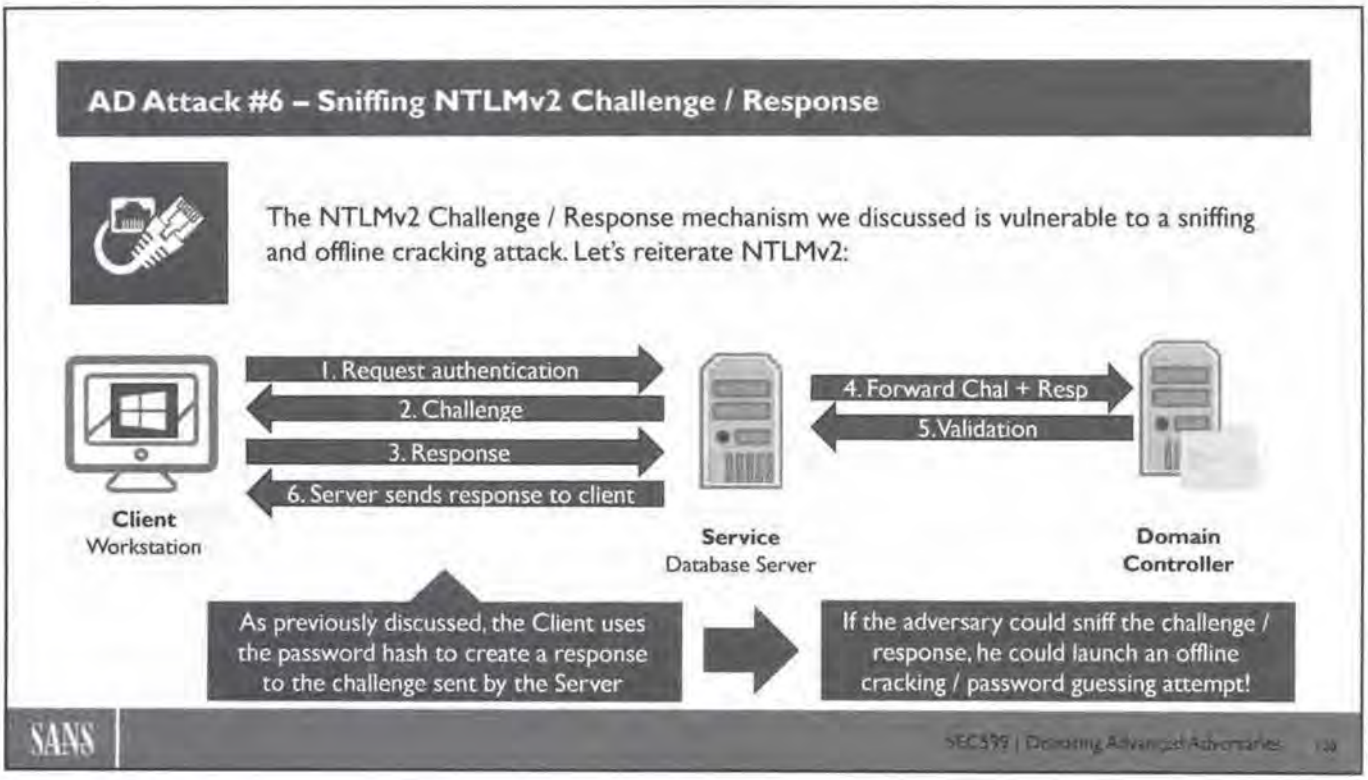
AD Attack #5 – Leveraging Excessive Privileges (Pivoting Between Users / Shares)

Active Directory Attacks – Domain Admin Rights

Due to its size & complexity, it's often difficult for administrators to retain a good overview of how privileges are assigned across the environment. Adversaries can leverage this to spot excessive privileges which can be used in lateral movement…

Once (limited) administrator privileges are obtained (e.g. on all workstations but not on servers), adversaries can start hopping from one system to the other in an attempt to steal credentials from different hops, thereby escalating privileges as they go along. An example would be a domain administrator that is authenticated to one of the workstations under the control of the adversary. The adversary could go to this workstation and dump the credentials from memory using Mimikatz.

A tool that facilitates this attack is BloodHoundAD, which generates a diagram of active sessions and relationships in the active directory. On the slide above, we can see an example of such a diagram: In a few steps, Erik could easily steal the hashes of Stephen, thereby obtaining Domain Admin privileges.



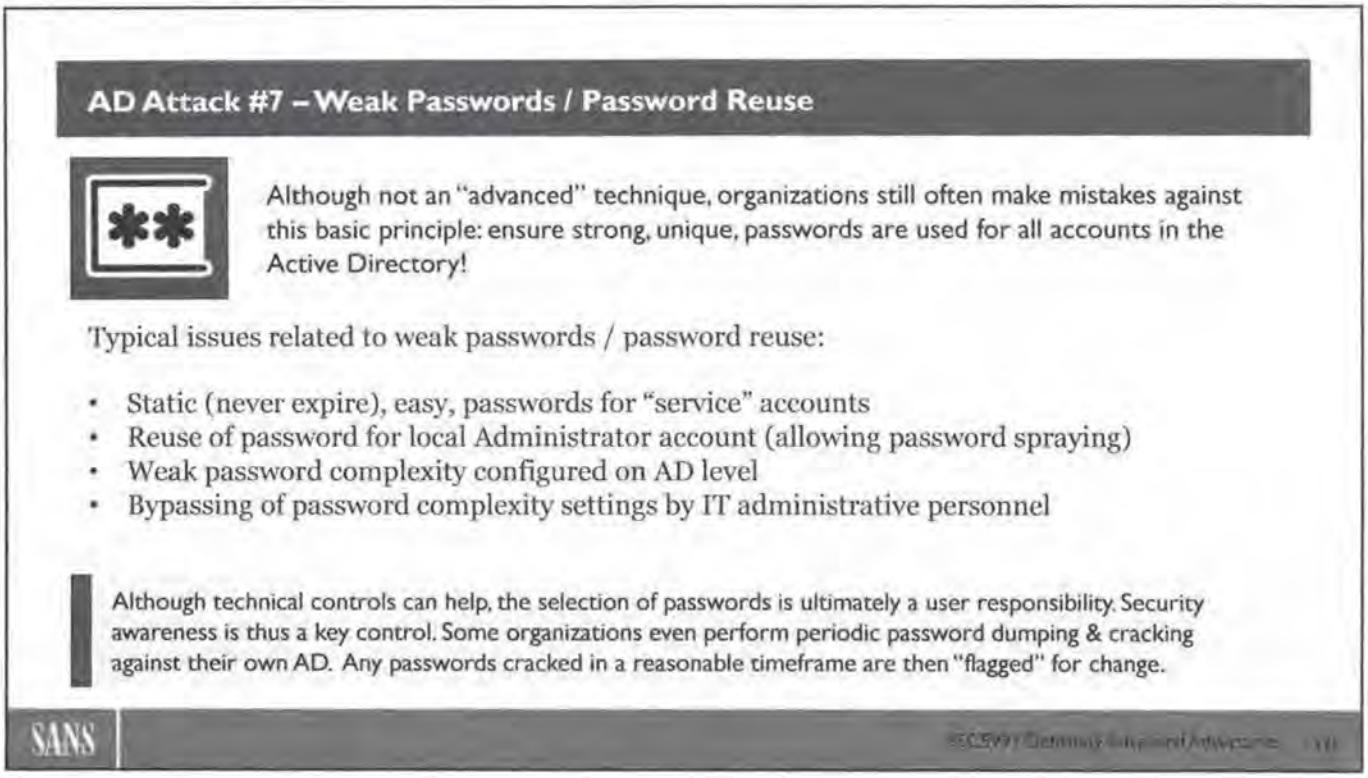
AD Attack #6 – Sniffing NTLMv2 Challenge / Response

Another attack technique involves the tricking of systems to send NTLMv2 challenge / responses to systems under control of the adversary. If you recall the NTLMv2 challenge / response method, you might remember that the response to an NTLMv2 challenge is created based upon the client challenge and the password hash of the user. Once an NTLMv2 challenge / response is sniffed, it can be fed to a password cracking tool such as Hashcat or John.

AD Attack #6 – Sniffing NTLMv2 Challenge / Response – Responder

Responder
Responder is (amongst others) an LLMNR, NBT-NS and MDNS poisoner. It will attempt to trick systems to connect / authenticate to the system it is running on. It will then attempt to sniff the authentication challenge (e.g. NTLMv2), which could be cracked by a password cracking tool.

AD Attack #6 – Sniffing NTLMv2 Challenge / Response - Responder
A tool that has implemented this attack technique is Responder. Responder is an LLMNR, NBT-NS and MDNS poisoner. It will attempt to trick systems on the network to connect / authenticate to the system it is running on. It will then attempt to sniff the NTLMv2 Challenge / Response, which could be cracked by a password cracking tool.



AD Attack #7 – Weak Passwords / Password Reuse

Although not an "advanced" technique, organizations still often make mistakes against this basic principle: ensure strong, unique, passwords are used for all accounts in the Active Directory! Throughout our careers as cyber security professionals, the course authors have observed the following typical issues on a frequent basis:

- "Technical" or "Service" accounts that are configured with a weak, static, password ("never expire"), often even with administrative privileges!
- Use of the local administrator account as a "recovery" account with the same password on all Windows hosts;
- Overall, weak password complexity settings configured on AD / domain level;
- Even if strong password complexity settings are configured, we've seen IT administrative personnel circumvent / bypass the controls for their own accounts!

As a conclusion, we'd like to note that, although technical controls can help, the selection of passwords is ultimately a user responsibility. This means security awareness is of paramount importance. If users are not convinced of the need / value of strong password selection, they will find a way to configure a weak password that meets your technical complexity controls. Even worse: they might select a strong password but then store it in an insecure way ("password.txt on the Desktop"). NIST has an excellent password standard that can be used to provide some insights in what a strong password looks like.

Some organizations have implemented offensive techniques themselves: they perform periodic password dumping & cracking against their own AD. Any passwords cracked in a reasonable timeframe are then "flagged" and users are forced to change them.

What's Next? – AD Persistence - Active Directory Attacks

Throughout this section, we've discussed a variety of techniques that are used by adversaries to move laterally through a Windows environment and obtain administrative access. Remember that our adversaries are persistent and would like to persist the access they achieve!

We will now discuss a number of techniques that are used to consolidate / persist administrative access to the AD:

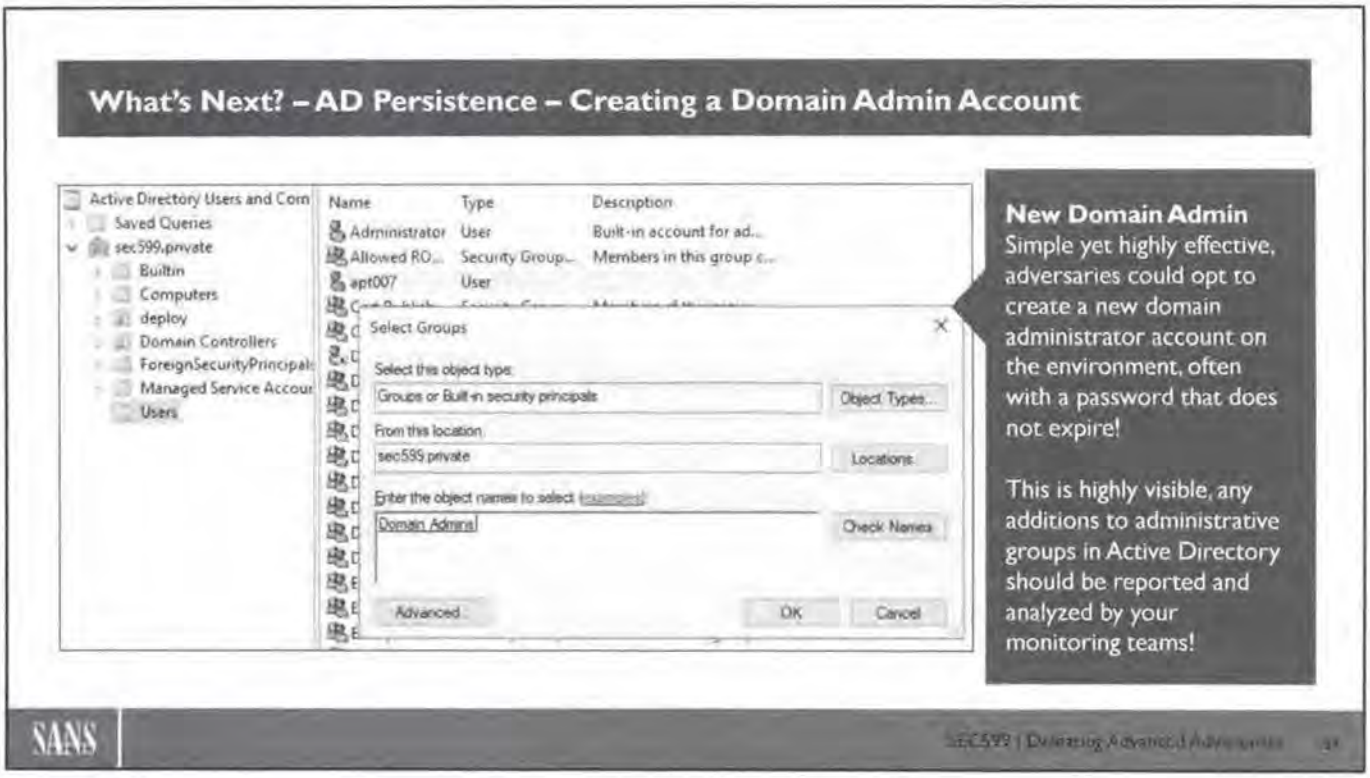
- Creation of a domain administrator account
- Creation of a golden ticket
- Use of DCSync
- Creation of a Skeleton Key

What's Next? – AD Persistence - Active Directory Attacks
Throughout this section, we've discussed a variety of techniques that are used by adversaries to move laterally through a Windows environment and obtain administrative access. Remember that our adversaries are persistent and would like to persist the access they achieve!

So, what is the next step in the adversary's toolbox? Typically, an adversary will attempt one of the following tricks to consolidate / persist administrative access to the AD:

- Simply creating a domain administrator account (preferably with strong credentials);
- Creating a Kerberos golden ticket that will allow long-term access to the environment;
- Using the DCSync attack;
- Creating a Skeleton key, which will allow the adversary to authenticate as any user in the environment.

Let's discuss these techniques in some more detail!



What's Next? – AD Persistence – Creating a Domain Admin Account
A simple attack to achieve persistence in Active Directory is to create a new domain admin user with a password that never expires.

Once an adversary has administrative rights to the domain, he can create a new domain administrator. This does not necessarily imply that the adversary must obtain domain administrator rights, but just obtain the right to create new users and assign them to groups. A domain administrator has these rights, be in Active Directory, these rights can also be delegated to administrative users that don't have domain admin rights. Because the domain administrator is such a powerful account, many organizations will only provide this account to a select group of security staff members.

Other users that need to perform common administrative tasks, like managing users, are only given the necessary rights to do this. But once a user has the right to create a new user and assign it to arbitrary groups, he can create a domain administrator.

This created account will give the adversary domain admin access to the domain as long that the account is not discovered and removed. It is therefore important to monitor your Active Directory infrastructure for the creation of new accounts with administrative rights.

What's Next? – AD Persistence – Golden Ticket

We already discussed "Pass-The-Ticket" attacks, where Kerberos tickets are stolen and abused by adversaries... The most damaging variant of this attack is the "Golden Ticket"!

- A Golden Ticket is a Ticket Granting Ticket generated with mimikatz that has several special properties.
- Mimikatz can generate the ticket with secret information (the NTLM hash of the krbtgt account) and readable information (for example domain name)
- The Golden Ticket refers to Willy Wonka's chocolate factory.

What's Next? – AD Persistence – Golden Ticket

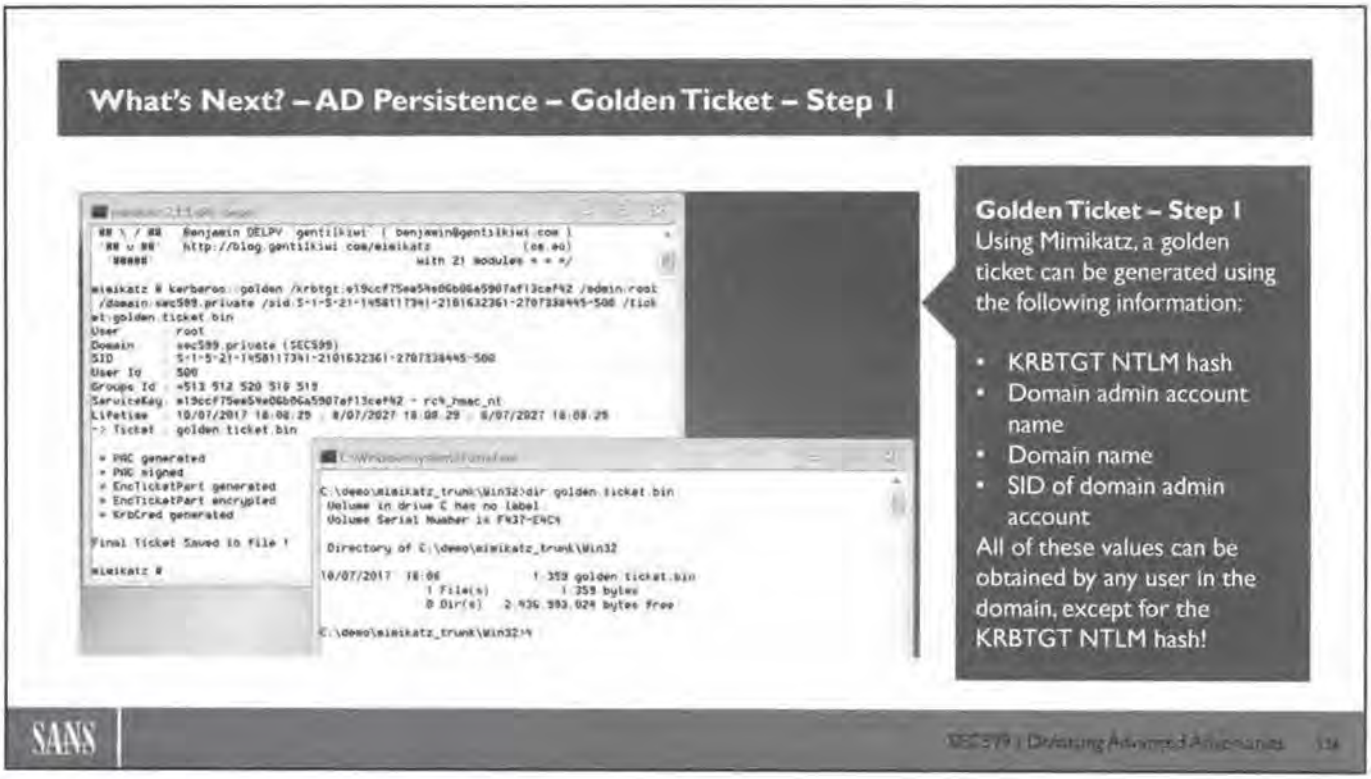
Another method to obtain a ticket for pass-the-ticket attacks is to use Mimikatz to generate a Golden Ticket. A Golden Ticket is a ticket-granting-ticket providing maximum access for a maximum period of time. It's the mother of all tickets, there is no ticket that provides more access.

As said, a Golden Ticket is not a ticket that the domain controller will generate, but it is a ticket the adversary generates purely from scratch, using tools like Mimikatz. The only secret information that is needed to create a Golden Ticket, is the NTLM hash of the krbtgt account.

When Active Directory is compromised, the NTLM hash of the krbtgt account can be extracted from memory (see our sekurlsa::lsa example). All the other information needed to create a Golden Ticket is not secret information but can be obtained readily. For example, the name of the domain will be needed. Although the domain name of your organization is not something you publicize, it will not be difficult for an adversary to obtain the name of your domain.

Remark that the NTLM hash of the krbtgt account can also be extracted from the Active Directory database, and its backups. So, it is primordial to protect this data. The only recourse one can have when a Golden Ticket has been generated is to change the password of the krbtgt account. Microsoft has scripts for this.

The name Golden Ticket refers to the Willi Wonka movie. In this movie, children can win a Golden Ticket that provides them full access to a fabulous chocolate factory.



What's Next? – AD Persistence – Golden Ticket – Step 1

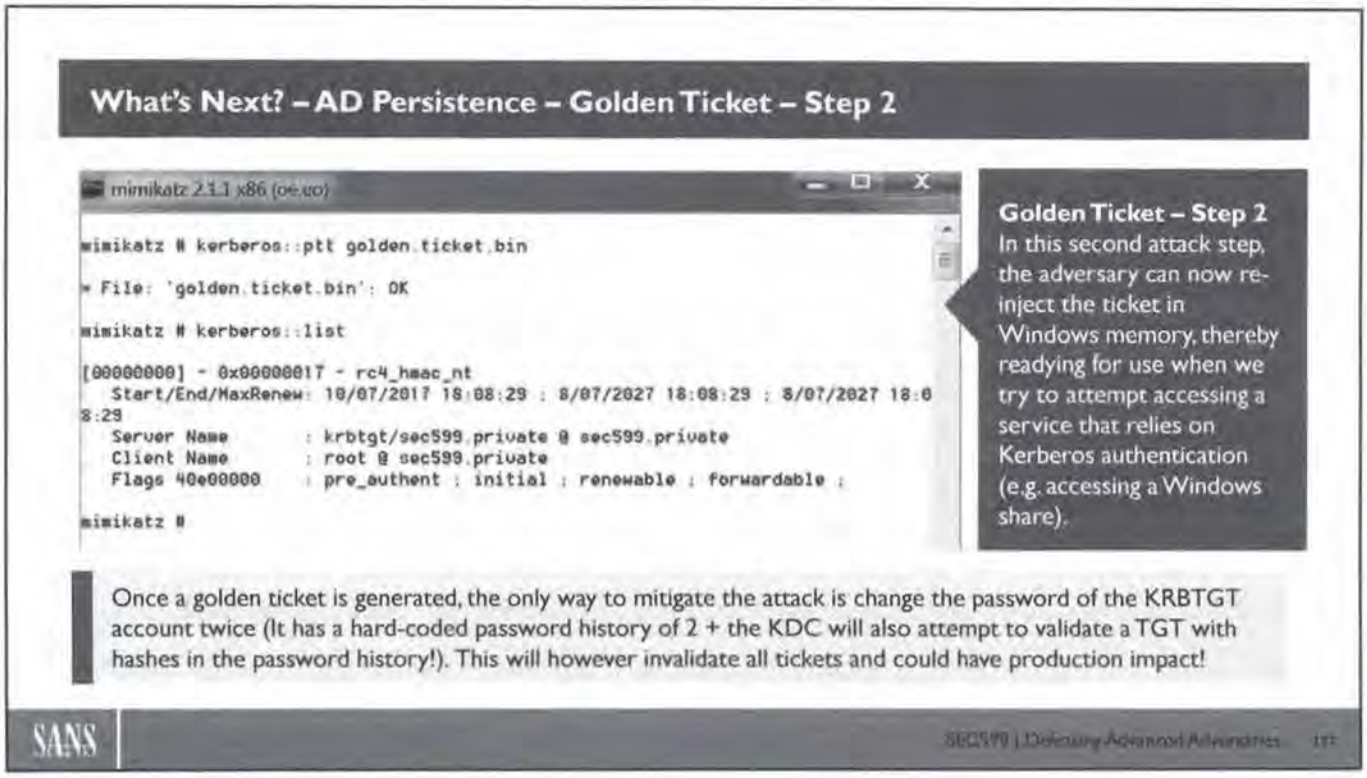
In this example, we explain what is needed to create a Golden Ticket with Mimikatz. Mimikatz's command Kerberos::golden can be used to generate a Golden Ticket. This is a pure "computational" command: it does not require administrative rights, and it does not require access to the domain. Generating a Golden Key can be done on any Windows computer in the world with normal access rights, provided that the necessary input values are known.

- The first input, the most difficult value to obtain, is the NTLM hash of the Kerberos account (krbtgt). As stated before, this hash can be obtained from compromised domain controllers or Active Directory databases.
- The second value is the name of the administrative account. In our example, that is root.
- The third value is the domain name: sec599.private in our example.
- And the fourth and last value is the SID of the admin account.

That is the only input that is needed to create a Golden Ticket. Remark that Mimikatz has many more options for this kerberos::golden command, but discussing these is not in scope (this is not an offensive course!). The only extra option we used here, it the /ticket: option to write the Golden Ticket to disk.

From the lifetime, you can see that this Golden Ticket is valid for 10 years! This means that as long that you do not change the password of the krbtgt account, the ticket will grant access to your complete Active Directory infrastructure for the next 10 years!

From the output, you can clearly see that the administrative account (user ID 500) is a domain admin (group ID 512) and an enterprise admin (group ID 519).

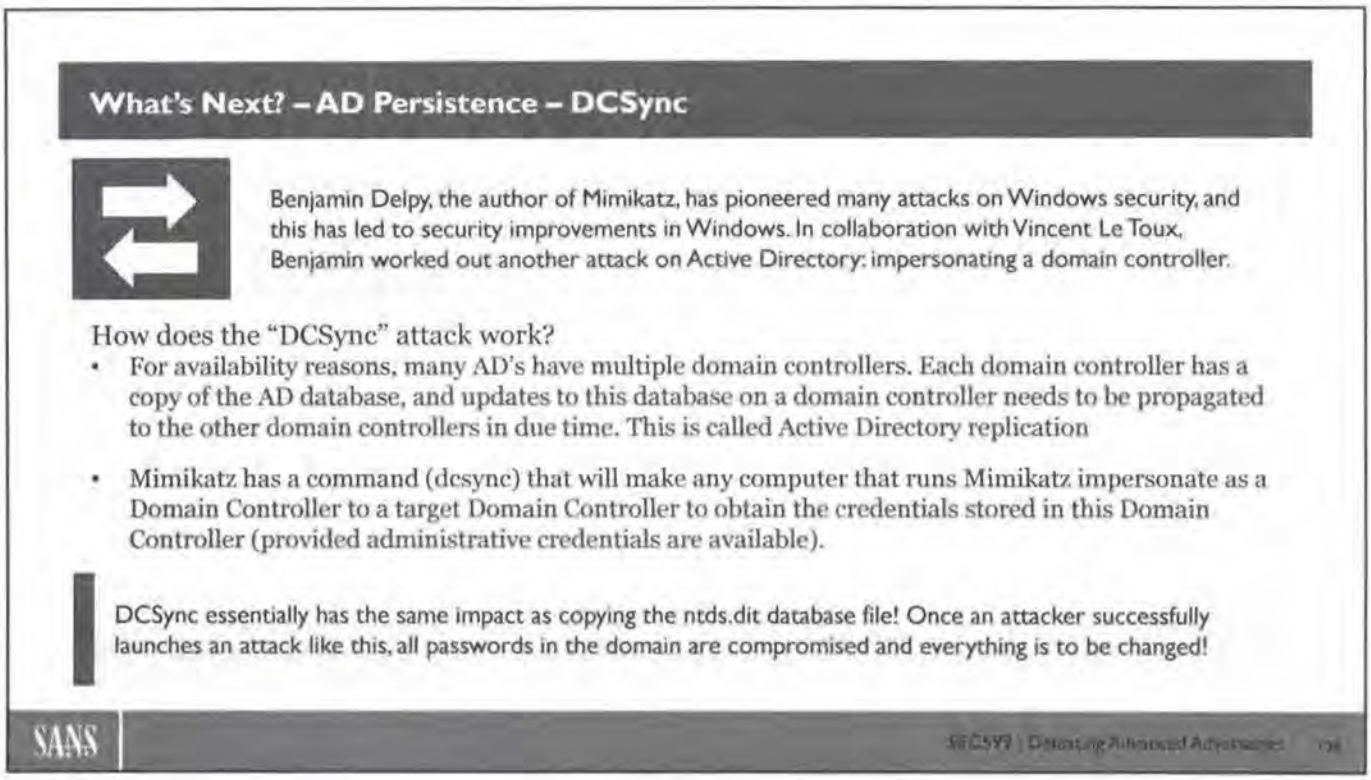


What's Next? – AD Persistence – Golden Ticket – Step 2
To use the previously generated ticket, we issue the Mimikatz command Kerberos::ptt golden.ticket.bin. This injects the ticket into Windows' memory, ready to be used when we attempt to connect to a service.

The Kerberos::list command can be used to list all the tickets that we have. We can see that the servername is krbtgt/sec599.private: this tells us that this is a ticket-granting ticket for the sec599.private domain. The user is root of the sec599.private domain.

If we look at the start and end dates, we see 2017 – 2027. That's an extremely long period for a ticket: 10 years (remember, the default lifetime of a TGT is 10 hours). That's because here, we used a Golden Ticket. Now when we try to connect to a share for example, this injected ticket will be used to obtain a ticket to the share we want to access (this requires Kerberos, thus we need to use the fileserver name to access, and not its IP address, as this would result in NTLM authentication which does not work with tickets).

Once a golden ticket is generated, the only way to mitigate the attack is to change the password of the KRBTGT account twice (It has a hard-coded password history of 2 + the KDC will also attempt to validate a TGT with hashes in the password history!). This will, however, invalidate all tickets and could have production impact!



What's Next? – AD Persistence – DCSync

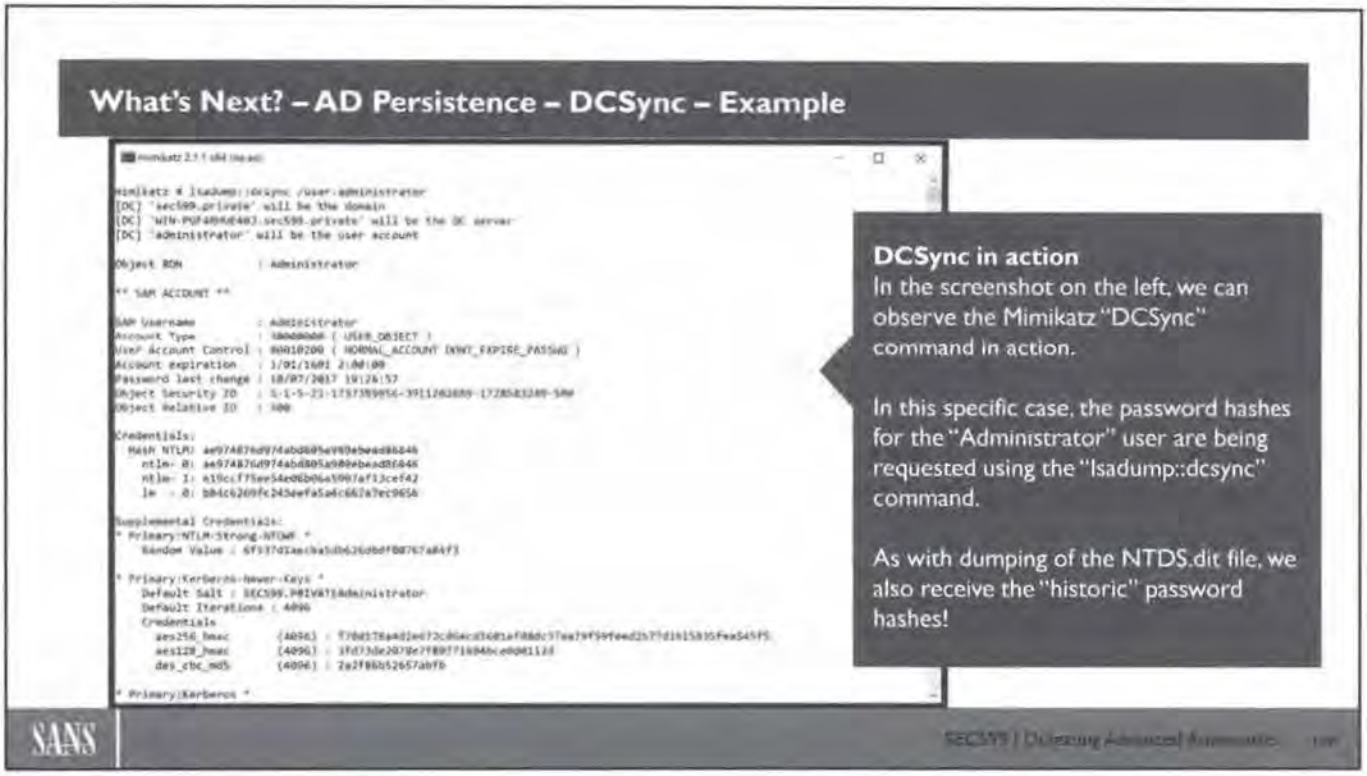
Benjamin Delpy, the author of Mimikatz, has pioneered many attacks on Windows security, and this has led to security improvements in Windows. In collaboration with Vincent Le Toux, Benjamin worked out another attack on Active Directory: impersonating a domain controller.

For availability reasons, administrators deploy more than one domain controller in an Active Directory infrastructure. Each domain controller has a copy of the Active Directory database, and updates to this database on a domain controller (for example the creation of a new user) needs to be propagated to the other domain controllers in due time. This is called Active Directory replication: a set of methods and protocols to synchronize the database of Active Directory domain controllers.

Vincent and Benjamin have worked out these methods and protocols for use by Mimikatz: Mimikatz has a command (dcsync) that will make any computer that runs Mimikatz to impersonate as a Domain Controller to a target Domain Controller to obtain the credentials stored in this Domain Controller.

Of course, normal users can not access this information. One needs domain admin rights to be able to participate in data replication. A Golden Ticket can provide these admin rights.

dcsync can dump the hashes of all users, or of a selected user.



What's Next? – AD Persistence – DCSync – Example

This example shows the dcsync command. By issuing "kerberos::dcsync /user:administrator" command, we request the credentials for user administrator to a domain controller. Command "kerberos::dcsync" would list the credentials of all users.

When this command is issued without extra options, Mimikatz selects the domain and the domain controller automatically, and extract the credentials from this domain controller via replication; the Directory Replication Service Remote (DRSR) protocol.

This is a very powerful attack: once an attacker has obtained domain admin credentials, it can use dcsync to connect to a domain controller and extracts the credentials of the krbtgt account. This data can then be used to create a Golden Ticket, and then it is game over: the only recourse you have is to change the krbtgt account password. This password never expires and is never changed, unless it is done manually. If you discover that the krbtgt NTLM hash has been compromised, you will have to change the password.

It is possible to detect and prevent a dcsync attack. DRSR network traffic should only occur between domain controllers. If you detect DRSR network traffic between a domain controller and a workstation, you know a dcsync attack took place.

If you segment your domain controllers in a dedicated network segment, with advanced firewalls as chokepoint between the other network segments, you can block DRSR traffic outside the domain controller network segment.

Persisting Administrative Access to the AD – Skeleton Key

A final AD persistence attack we would like to highlight is the "Skeleton Key" attack, which is also been added as a built-in module in Mimikatz. A skeleton key is a key that opens all the locks in a building. In the same way a Skeleton Key can "unlock" all systems in the domain!

How does the "Skeleton Key" attack work?

- The "Skeleton Key" essentially means that a backdoor is implanted on the domain controller;
- This backdoor runs in memory and adapts the running domain controller in such a way that a single password (the skeleton password) can be used to logon with any account;
- As it runs in memory, it does not persist by itself (but can, of course, be scripted or persisted using one of the persistence strategies we've seen above)
- This only affects one domain controller, so the adversary would have to target all domain controllers for maximum effect...

Persisting Administrative Access to the AD – Skeleton Key

The last AD persistence attack we want to illustrate is an attack to achieve persistence inside a domain. Of course, the Golden Ticket is the ultimate persistence mechanism, but this skeleton key attack also merits our attention.

A skeleton key is a special key that opens all the locks inside a building. Each door lock inside the building requires its own key to be opened, but all the locks can also be opened with a special, single key: the skeleton key. Mimikatz provides a skeleton key attack for Active Directory.

When the Mimikatz skeleton attack is executed on a domain controller, a bit of code is patched in memory so that all accounts can be logged in with a special password (mimikatz). This does not remove the existing passwords: it is now possible to log into an account using 2 different passwords: one can log in with the normal account password, or one can login with the skeleton key password.

By patching the code of the LSA process, the functions that validate credentials are modified: they still accept the normal password, but also accept the skeleton password.

This has to be done on a domain controller, it does not work on non-domain controllers. But this does not mean that an attacker can only login with the skeleton password on a domain controller: the password works on all domain members that use this compromised domain controller for authentication.

If there are more than one domain controllers, the attack needs to be executed on all domain controllers to be sure that the skeleton password will work in all cases.

Persisting Administrative Access to the AD – Skeleton Key – Example

Skeleton Key in action
In the screenshot on the left, we can observe Mimikatz installing a "skeleton key" backdoor on the domain controller.

Note the simplicity of the commands... This will now allow anyone to authenticate as any user in the domain with the skeleton key password ("mimikatz").

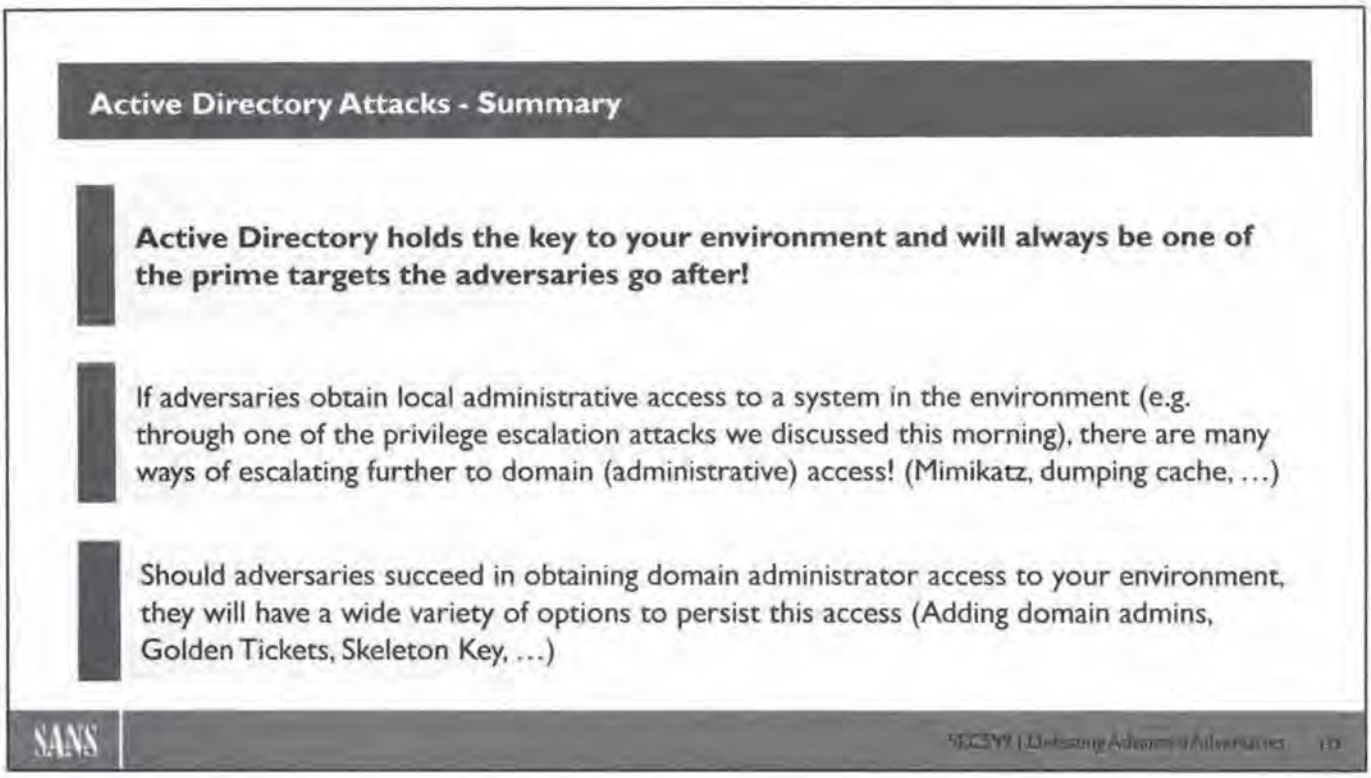
Might be a good idea to write a script to check successful authentication using the password "mimikatz", right? ☺

Persisting Administrative Access to the AD – Skeleton Key – Example
In the example above, we can see the skeleton key attack. Since this will patch the code of the LSA process in memory, administrative rights are required and the debug privilege needs to be enabled. An attacker just needs to run command "misc::skeleton" and voila, the skeleton key is installed on the domain controller.

When there are several domain controllers inside a domain, domain members connect to a domain controller for authentication via a kind of load-balancing scheme.

This means that to be 100% reliable, the skeleton key attack needs to be performed on all domain controllers. Otherwise, a domain member might authenticate to a domain controller that does not have the skeleton key patch applied in memory.

Since this is a patch in memory simply rebooting the domain controller removes the skeleton key. But of course, attackers can install an autorun entry for Mimikatz to run automatically when the domain controller boots.



Active Directory Attacks - Summary

In this section, we have seen a wide variety of attacks against Active Directory. It's important to understand that Active Directory holds the key to your environment and will always be one of the prime targets the adversaries go after!

To pull off these attacks, administrative credentials are needed. Depending on the attack, these can be machine administrator credentials or domain administrator credentials. To obtain these credentials, advanced adversaries have different avenues of attack. The credentials can be stolen from machines, backups, ... or they can be obtained by social engineering users. Another method is privilege escalation. For example, to dump the credentials from the Local Security Authority, the LSA process must be opened and its memory read. This requires the debug privilege, which administrators have. But the system account can also access the LSA process memory: a local privilege escalation vulnerability can be used to obtain elevation to the system account.

There are many tools to attack Active Directory (the domain and the domain controllers). A very popular tool is mimikatz. Mimikatz is reliable and powerful. Mimikatz is detected by many anti-virus programs, but because it is open source, it has been recompiled, repurposed or transformed into versions that are not detected by anti-virus. For example, in the Petya/Notpetya ransomare executable, credential stealing applications (32-bit and 64-bit) were stored as resources in the ransomware. These applications were repurposed versions of mimikatz.

Active Directory Attacks – Additional References

Some additional resources concerning Active Directory attacks:

- Mimikatz
  https://github.com/gentilkiwi/mimikatz
  https://github.com/gentilkiwi/mimikatz/wiki
  https://www.youtube.com/user/dist67/videos
- Active Directory
  https://github.com/gentilkiwi/mimikatz/wiki
- Kerberos
  https://en.wikipedia.org/wiki/Active_Directory

Active Directory Attacks - Additional References
Some additional resources concerning Active Directory attacks:

Mimikatz
https://github.com/gentilkiwi/mimikatz
https://github.com/gentilkiwi/mimikatz/wiki

Active Directory
https://github.com/gentilkiwi/mimikatz/wiki

Kerberos
https://en.wikipedia.org/wiki/Active_Directory

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

Active Directory Hardening & Segmentation

Now that we have an in-depth understanding of current AD attacks, let's see how we can prevent & detect them! We will zoom in on the following practical recommendations:

- Overall Active Directory architecture
- Implementing LAPS
- Implementing Protected Processes (Win8 & 2012)
- Implementing Credential Guard (Win10 & 2016)
- Privileged account management & IAM

After completing this section, we will also perform an exercise where we implement a number of controls that can help thwart AD lateral movement.

SANS

Active Directory Hardening & Segmentation

Now that we have an in-depth understanding of current AD attacks, let's see how we can prevent & detect them! We will zoom in on the following main categories of recommendations:

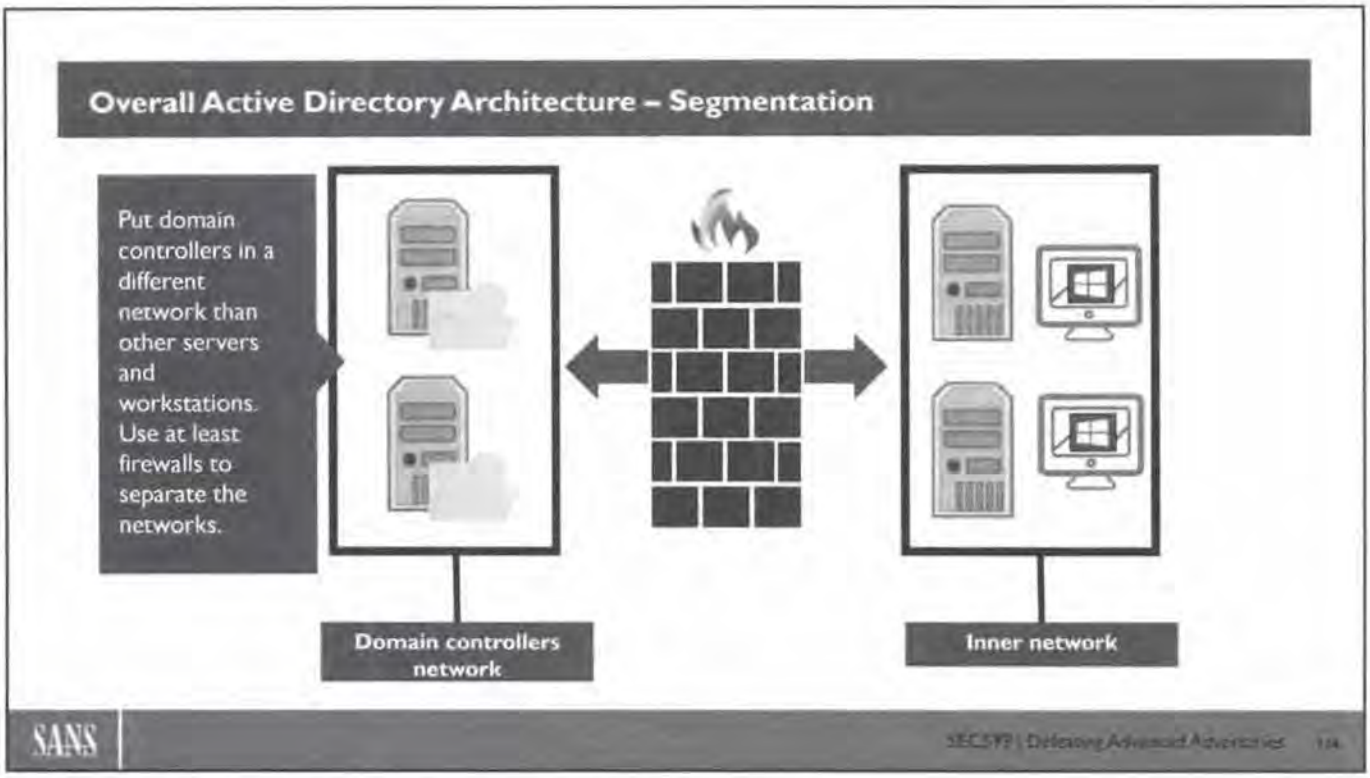
- First of all, domain controllers have to be connected to the network. By carefully designing the network segments that domain controllers will be placed in, we can prevent some attacks. Placing the domain controllers in the same, flat network as servers and workstations is a bad idea. Domain controllers should have their own LAN segment. Active Directory is also a hierarchical structure: this tree structure is present at many levels: forests, domains, organization units, … Carefully planning this structure can create a more robust Active Directory structure when security is involved.

- Second, it's important to address the issue of re-used local administrator passwords. Microsoft Local Administrator Password Solution (LAPS) provides an effective way of handling this issue by "generating" random local administrator passwords and storing them as part of the domain structure.

- CredentialGuard is an excellent new feature as of Windows 10 that will isolate the LSASS process, in an attempt to protect it from "hash dumping" attacks. We will discuss it in-depth in this upcoming section.

- As CredentialGuard is only available for Windows 10 (& upwards), we can consider implementing Protected Processes on older Windows versions such as Windows 8. These will attempt to achieve a similar result as CredentialGuard, but they can be bypassed by tools such as Mimikatz.

- Finally, we will discuss a number of matters related to privileged account management & IAM. As AD environments can become largely complex, it is important administrators retain a good overview of how the different privileges are assigned in their networks.



Overall Active Directory Architecture – Segmentation

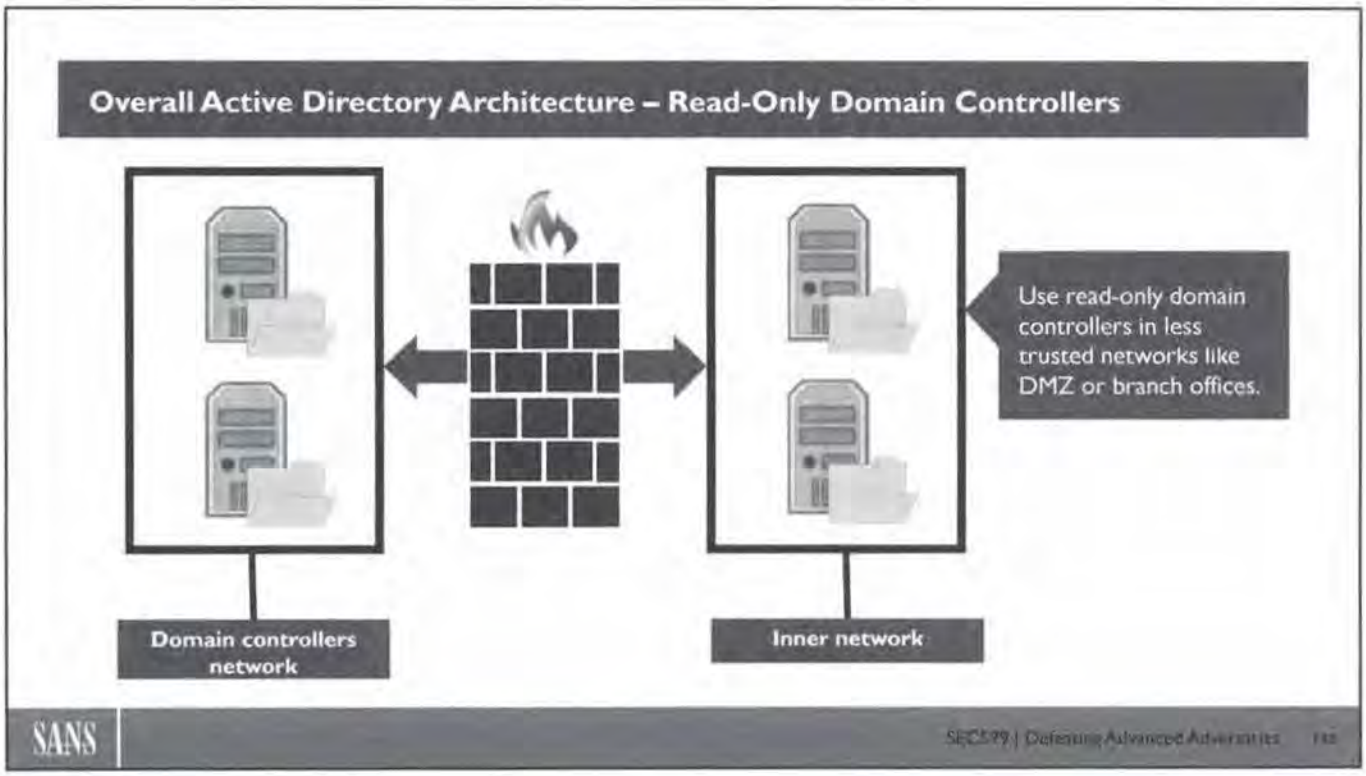
Domain controllers need to communicate with all their member machines, servers and workstations, and also, with other domain controllers.

But this does not imply that they should all be located in the same flat network. By putting domain controllers in one LAN segment, and workstations and servers in a different LAN segment (several LAN segments is better), one has more options to monitor and control network traffic between domain controllers and their members.

A simple solution is to put domain controllers into their own VLAN (Virtual LAN) segment. This should be the minimum: if possible, a physically separated LAN segment offers more security than a VLAN segment. Domain controllers of different domains should not be put into the same LAN segment, especially if they do not have the same level of trust, like DMZ and production.

Network devices to monitor and control the network traffic can be put in place between the segments. These should be at least firewalls that can block all traffic that is not "normal" traffic for domain controllers. Kerberos traffic, LDAP, NTLM, … is considered normal for example, but FTP would not be normal.

Network devices that can do deep inspection of network traffic, like IPS devices, are preferred if the firewalls used by your enterprise are not capable of doing deep inspection. This design will, for example, prevent a mimikatz dcsync attack: Active Directory data replication traffic between domain controllers is normal, but not between a domain controller and a Windows laptop.

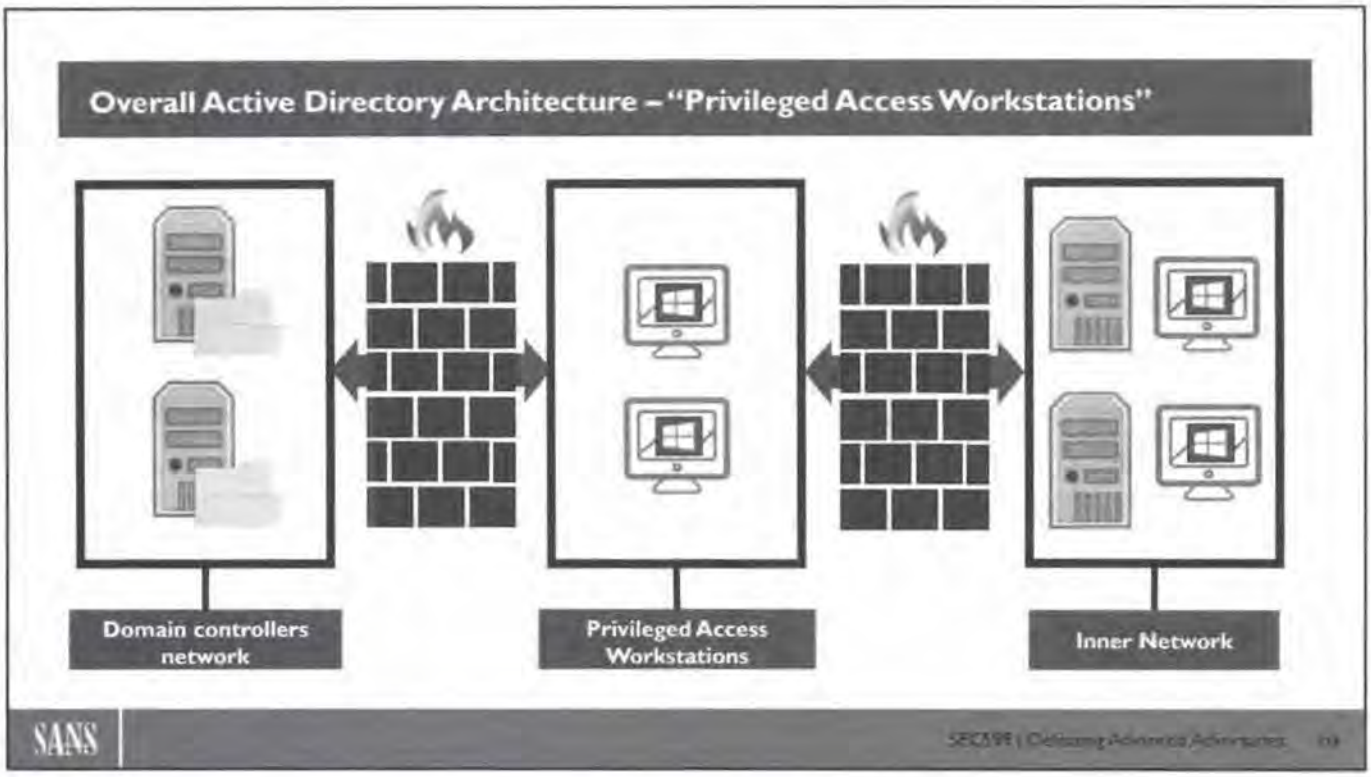


Overall Active Directory Architecture – Read-Only Domain Controllers
Further hardening at the network level can be implemented by using so-called read-only domain controllers.

Read-only domain controllers were introduced with Windows 2008. As their name implies, read-only domain controllers only keep a read-only version of the Active Directory database. When replicating Active Directory data with other domain controllers, they will only receive updates, they will never send out updates to the data.

Read-only domain controllers can be configured to cache only the credentials of users and computers that try to authenticate to them: these read-only domain controllers do not contain credentials. They will only accept authentication request from a pre-defined set of accounts and will delegate the authentication to full domain controllers.

This behavior makes them more suited to be used in environments that are less secure. For example, in DMZ or in branch offices that do not have a secure data center. If such a domain controller is compromised, then the credentials are not compromised.

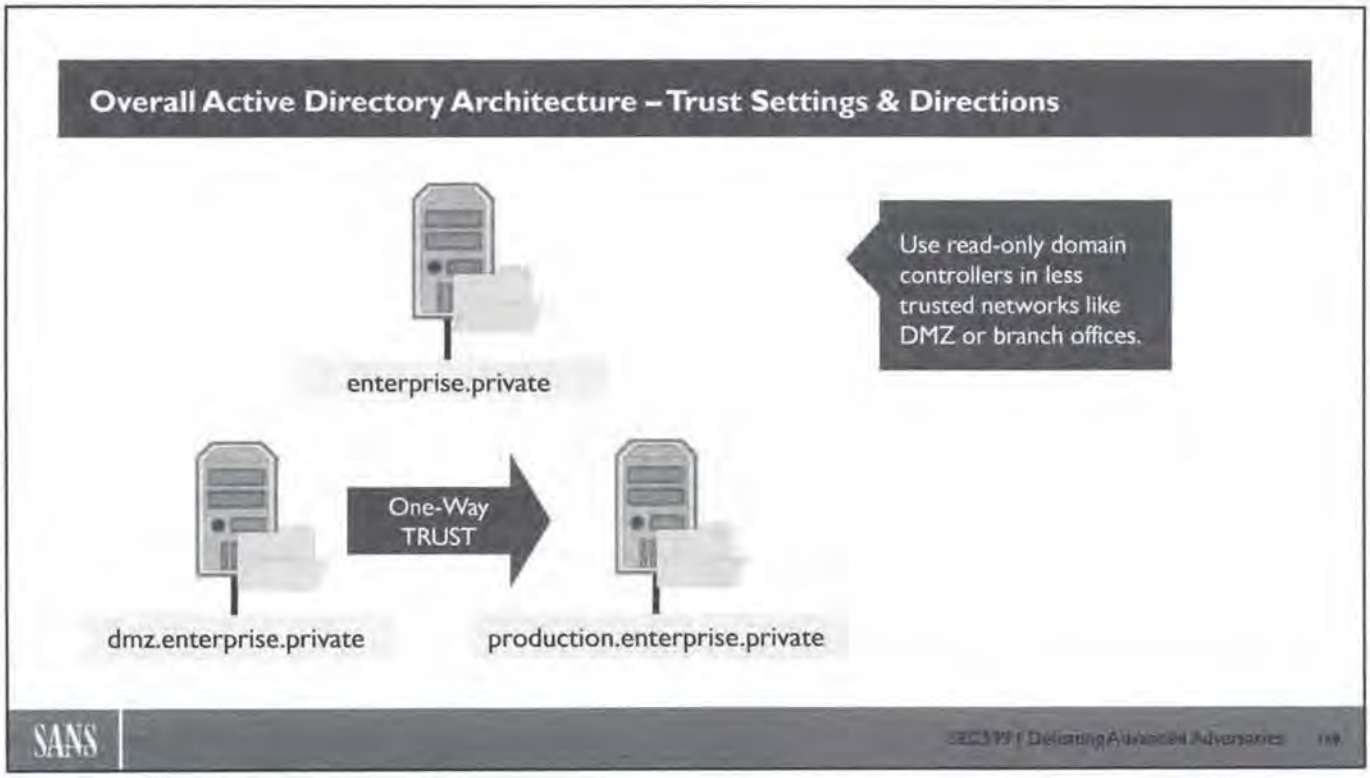


Overall Active Directory Architecture – "Privileged Access Workstations"
A practice recommended by Microsoft, this mainly relates to the use of highly privileged accounts that are used to administer the AD. Management of the Active Directory (& thus use of highly privileged accounts) should only be performed from a number of dedicated systems that can be seen as administrative "jump boxes". Direct administrative access to the Domain Controllers from normal "internal network zones" is not allowed.

These "jump boxes" are labeled "Privileged Access Workstations" by Windows. This type of system should be subject to additional security hardening such as the full implementation of Credential Guard. Furthermore, they should be dedicated to administrative functionality, and not run software such as email applications, web browsers, or productivity software such as Microsoft Office. In its "Security Best Practices for Active Directory", Microsoft describes three core principles for these administrative hosts:

- You should never administer a trusted system (that is, a secure server such as a domain controller) from a less-trusted host (that is, a workstation that is not secured to the same degree as the systems it manages).
- You should not rely on a single authentication factor when performing privileged activities; that is, user name and password combinations should not be considered acceptable authentication because only a single factor (something you know) is represented. You should consider where credentials are generated and cached or stored in administrative scenarios.
- Although most attacks in the current threat landscape leverage malware and malicious hacking, do not omit physical security when designing and implementing secure administrative hosts.

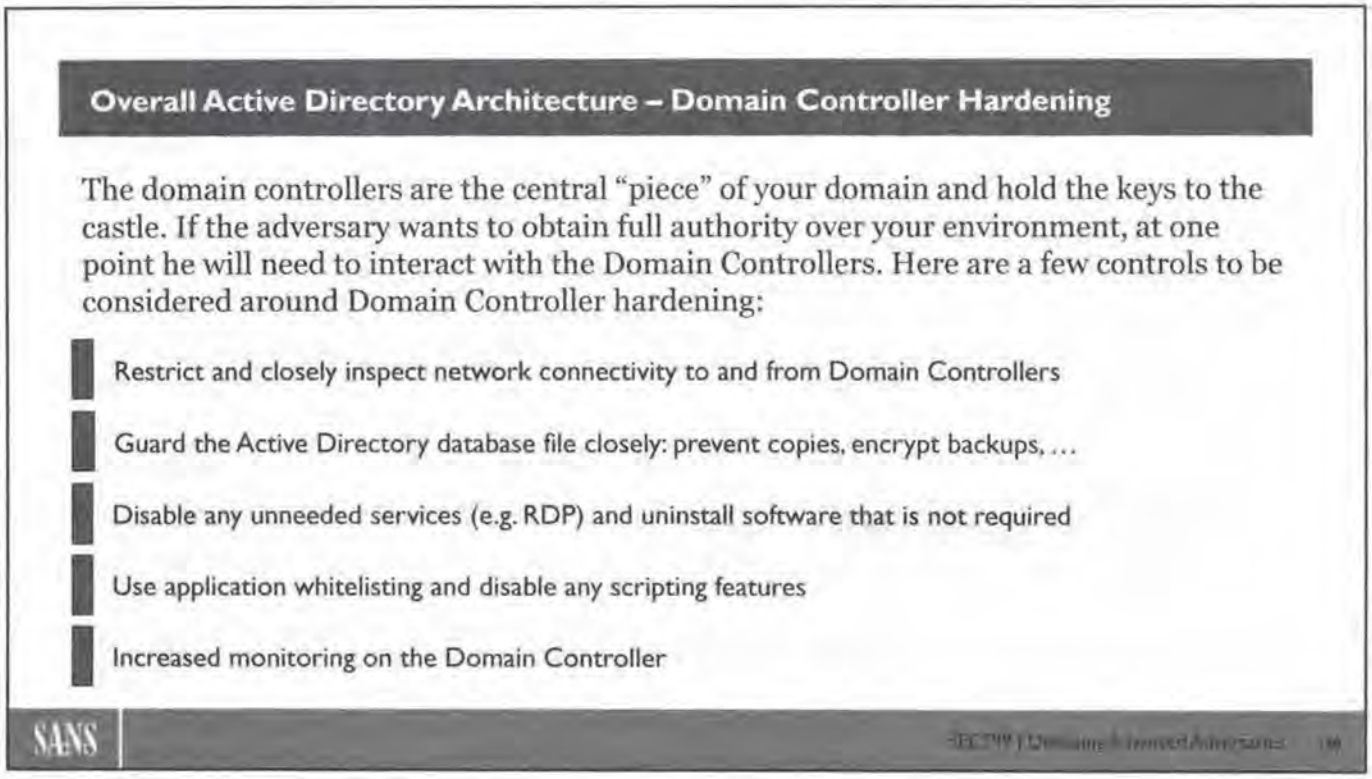
Sources:
- https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/implementing-secure-administrative-hosts
- https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/privileged-access-workstations



Overall Active Directory Architecture – Trust Settings & Directions
Active Directory has hierarchical structures at different levels. For example, domains can be put into a tree structure. In the example above, we have the domain enterprise.private at the root of the tree. dmz.enterprise.private and production.enterprise.private are branches of the tree.

Each domain requires at least one domain controller.

If a level of trust is required between dmz and production, then a trust relationship can be established where dmz trusts production. There should never be a trust of dmz by production. This is what we call a one-way trust!



Overall Active Directory architecture – Domain Controller Hardening
The domain controllers are the central "piece" of your domain and hold the keys to the castle. If the adversary wants to obtain full authority over your environment, at one point he will need to interact with the Domain Controllers. Here are a few controls to be considered around Domain Controller hardening:

- First of all, we should restrict and closely inspect network connectivity to and from Domain Controllers. An excellent example of this is Internet access. There are very few reasons why domain controllers should be able to (directly) access the Internet;
- Secondly, protect the active directory database file (ntds.dit) closely: this includes both in its original location, but also any replicated copies or backups that are spread in the environment. Monitor the use of ntdsutils in your network!
- Any unneeded services on the Domain Controller should be disabled. A good example of this is the Microsoft Remote Desktop Protocol (RDP). As we've seen in previous slides, management of the AD can be performed from "Privileged Access Workstations". Furthermore, there should be close to no "extra" software installed on the Domain Controller;
- Without exception, restrictive application whitelisting & script restrictions should be enforced on the domain controller;
- As the domain controller is the central component and holds "the keys to the castle", it should be subject to "increased monitoring" as a whole.

Introducing Protected Processes

In order to prevent hash dumping attacks aimed at the LSA process, Microsoft introduced "Protected Processes" as of Windows 8 & Windows Server 2012.

- Protected processes were first introduced in Windows Vista for DRM (Digital Rights Management) purposes, but were adapted for "security purposes" in Windows 8
- The screenshot on the right provides an example of the lsass.exe process running as a "protected process"
- Protected Processes are implemented in the Kernel software and can thus be defeated...

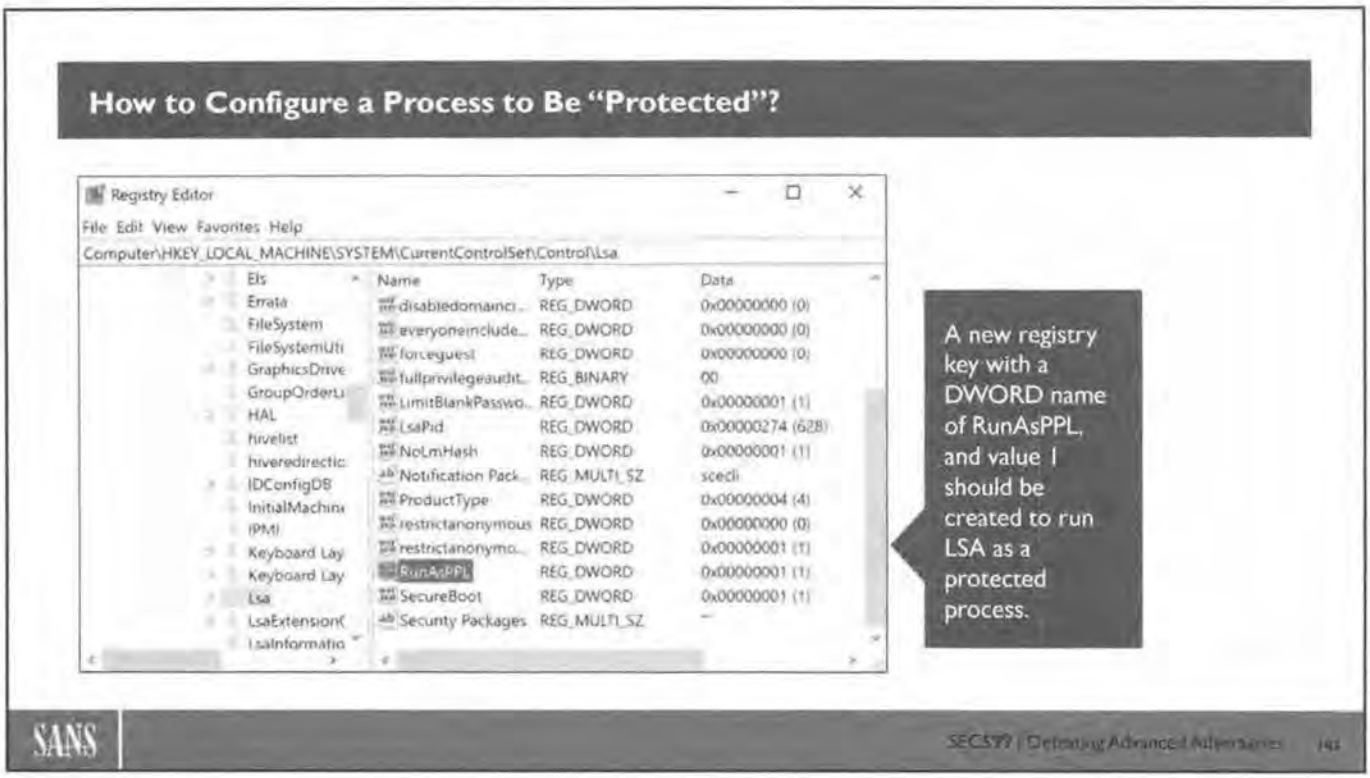
Introducing Protected Processes

The next two protection methods we will discuss not only apply to domain controllers, but to all Windows machines. The Local Security Authority is the process that handles the credentials. As we have seen in Active Directory attacks, many powerful tools are available to adversaries to extract credentials from the lsass.exe process.

With Windows Server 2012 (and Windows 8), the LSA can be configured to have its lsass.exe process hosted as a protected process. Under Windows, with the correct privileges, accounts can access the memory of any Windows process. This is not the case with protected processes. Protected processes (and their memory) cannot be accessed by other processes, regardless of the account, they run with. Protected processes were introduced with Windows Vista for DRM purposes (to make media players), but protected processes were repurposed for security boundaries when Windows 8 was introduced.

By running the lsass.exe process as a protected process, tools like Mimikatz cannot access the process to extract credentials. In the screenshot above, we can see that the lsass.exe process running on this machine is protected: Process Explorer's security tab indicates that the process is protected (PsProtectedSignerLsa-Light).

Protected Processes are implemented in the Kernel software and can thus be defeated... Mimikatz has a function to remove the protection from protected processes: this "converts" the process into a normal process. To do this, Mimikatz requires its kernel driver to be installed. Installation of this kernel driver can, however, be detected and responded to.

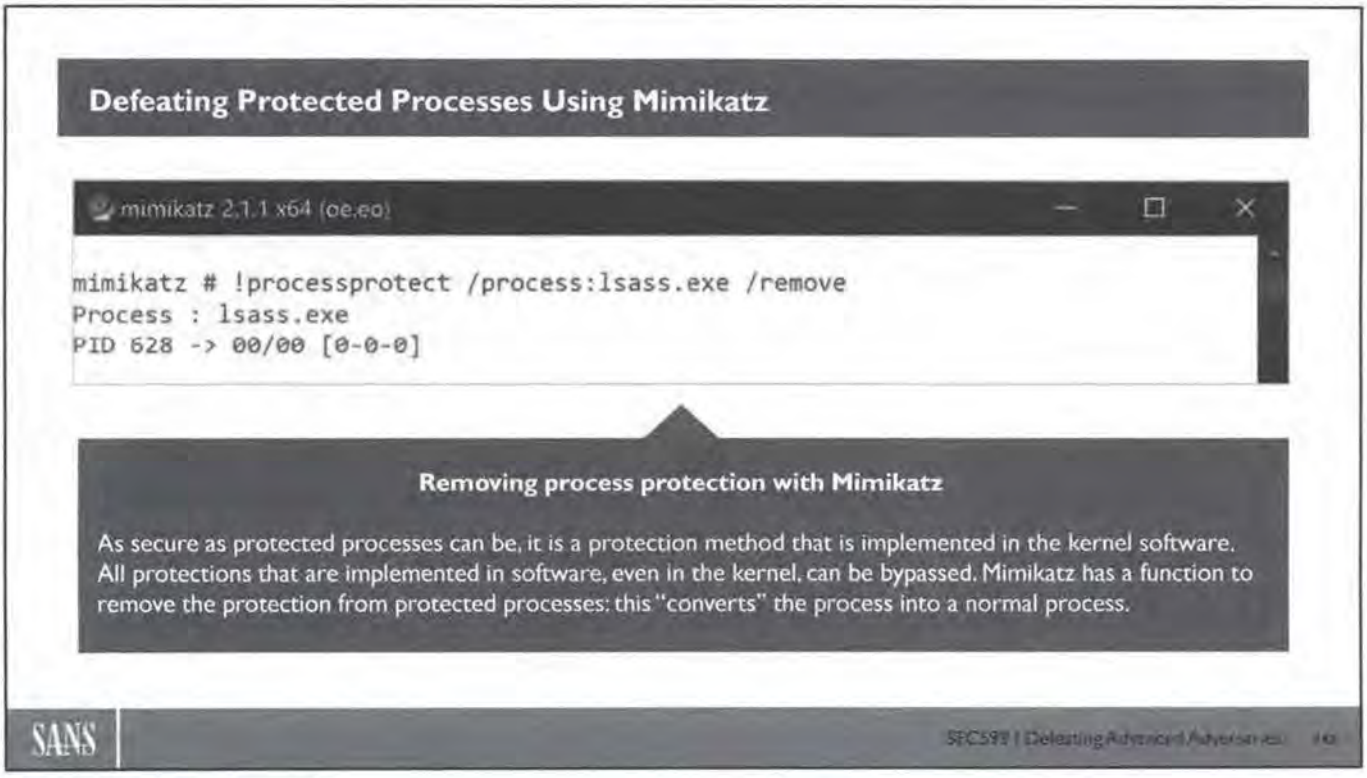


How to Configure a Process to Be "Protected"?
To configure Windows to run the lsass.exe process as a protected process, a change has to be made. By default, Windows will not run the lsass.exe process as a protected process.

A registry value has to be created and set: under the registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa, a double word value (DWORD) has to be created. The name of this DWORD is RunAsPPL, and its value should be set to 1.

Then Windows needs to be rebooted, after which the LSA process will run as a protected process.

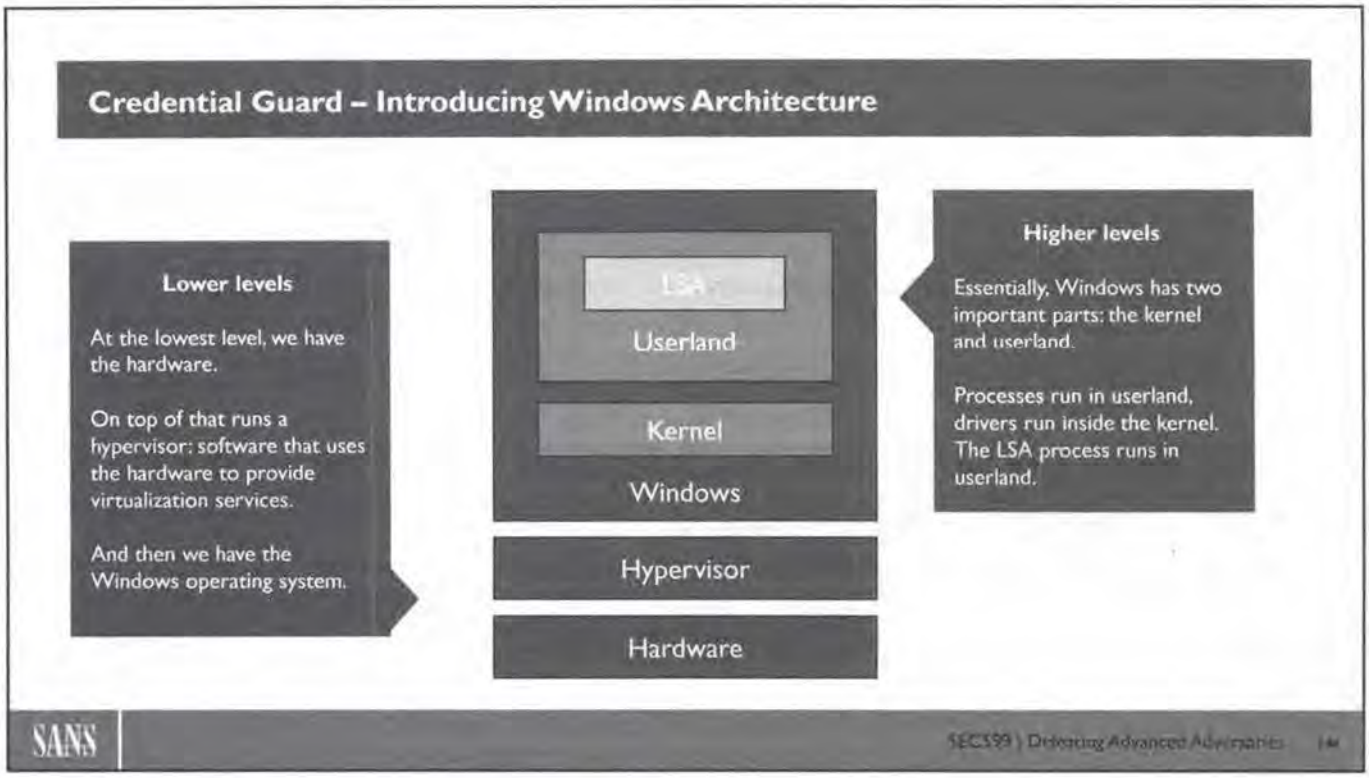
This requires at least Windows 2012 (or Windows 8).



Defeating Protected Processes Using Mimikatz
As secure as protected processes can be, it is a protection method that is implemented in the kernel software. All protections that are implemented in software, even in the kernel, can be bypassed. Mimikatz has a function to remove the protection from protected processes: this "converts" the process into a normal process. To do this, mimikatz requires its kernel driver to be installed.

This can be done with the command !+
The next step is to use the kernel command processprotect to remove the protection of the lsass.exe process: !processprotect /process:lsass.exe /remove.

When this is done, mimikatz (or other tools) can be used to extract the credentials, as the memory of the LSA process is no longer protected.



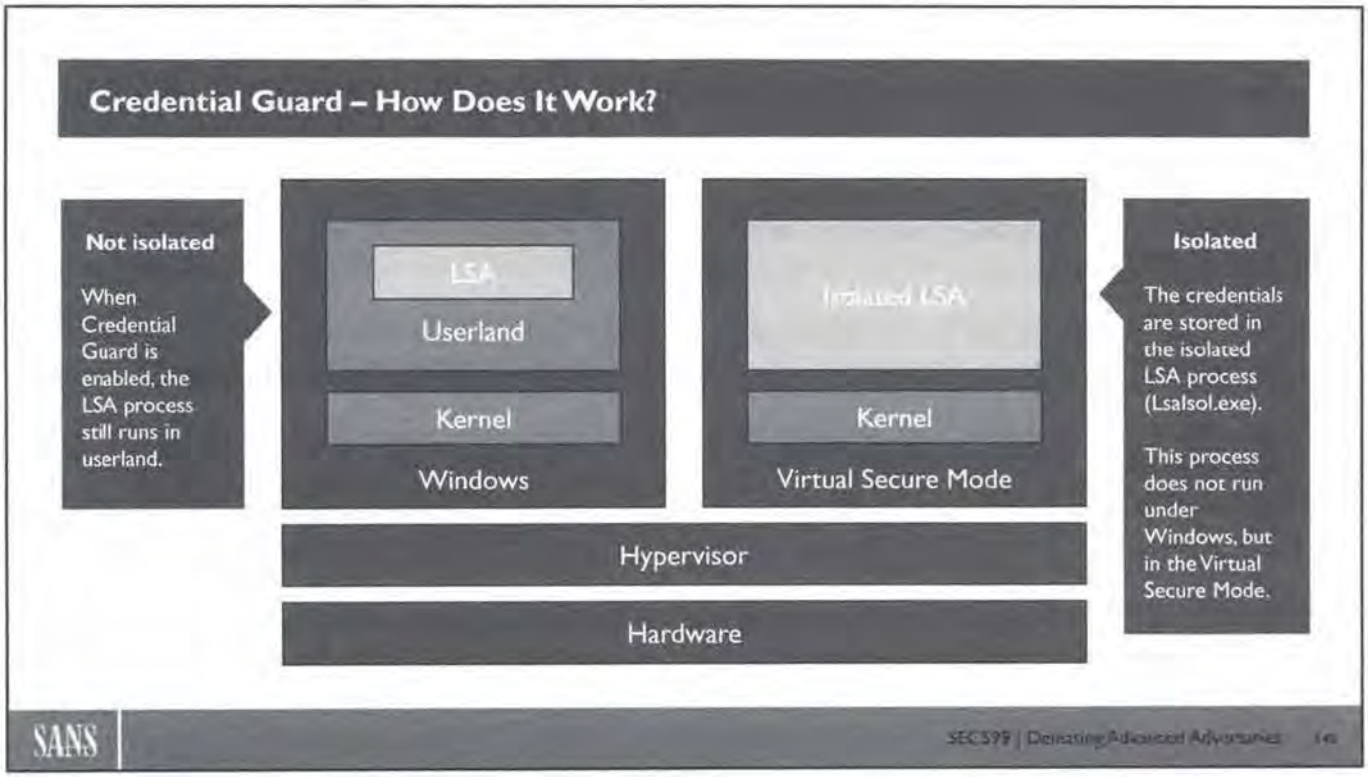
Credential Guard – Introducing Windows Architecture

As a pure software solution like protected processes can be bypassed, Microsoft developed a hardware-based solution: Credential Guard. In order to understand Credential Guard, we first need to understand the normal architecture of Windows.

The simplified diagram above describes this architecture.

At the lowest level, we have the hardware. On top of that runs a hypervisor: software that uses the hardware to provide virtualization services. And then we have the Windows operating system.

Essentially, Windows has two important parts: the kernel and userland. Processes run in userland, drivers run inside the kernel. The LSA process runs in userland.

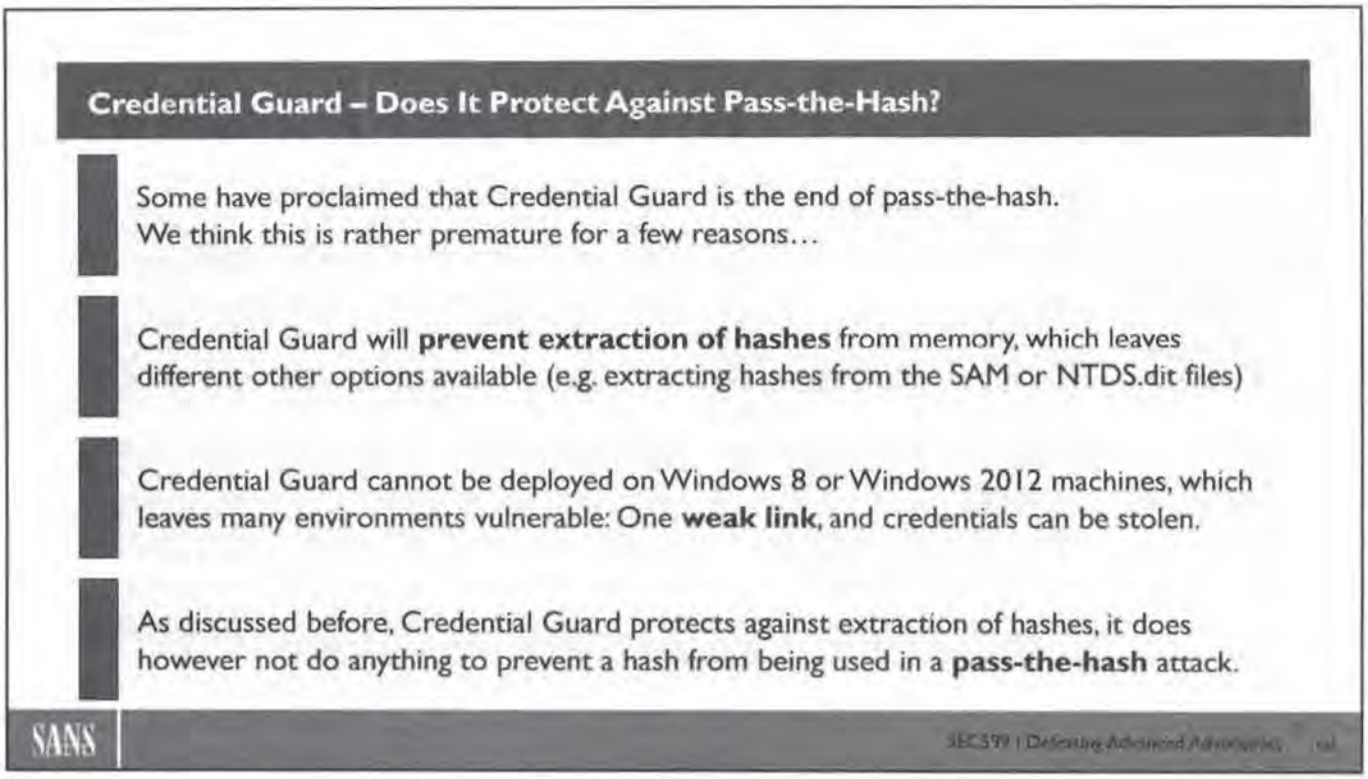


Credential Guard – How Does It Work?
Credential Guard was introduced with Windows Server 2016 and Windows 10, enterprise editions. It requires modern CPUs that provide virtualization functionality.

When Credential Guard is enabled, Windows still runs on top of the hypervisor and the hardware, and the LSA process still runs in userland. The difference, however, is that the credentials are no longer stored inside this LSA process (lsaas.exe).

With Credential Guard, the credentials are stored in the Isolated LSA process (Lsalsol.exe). This process does not run under Windows but in the Virtual Secure Mode. This is a separate, virtualized environment, that is separated from the other environments (like Windows) via hardware.

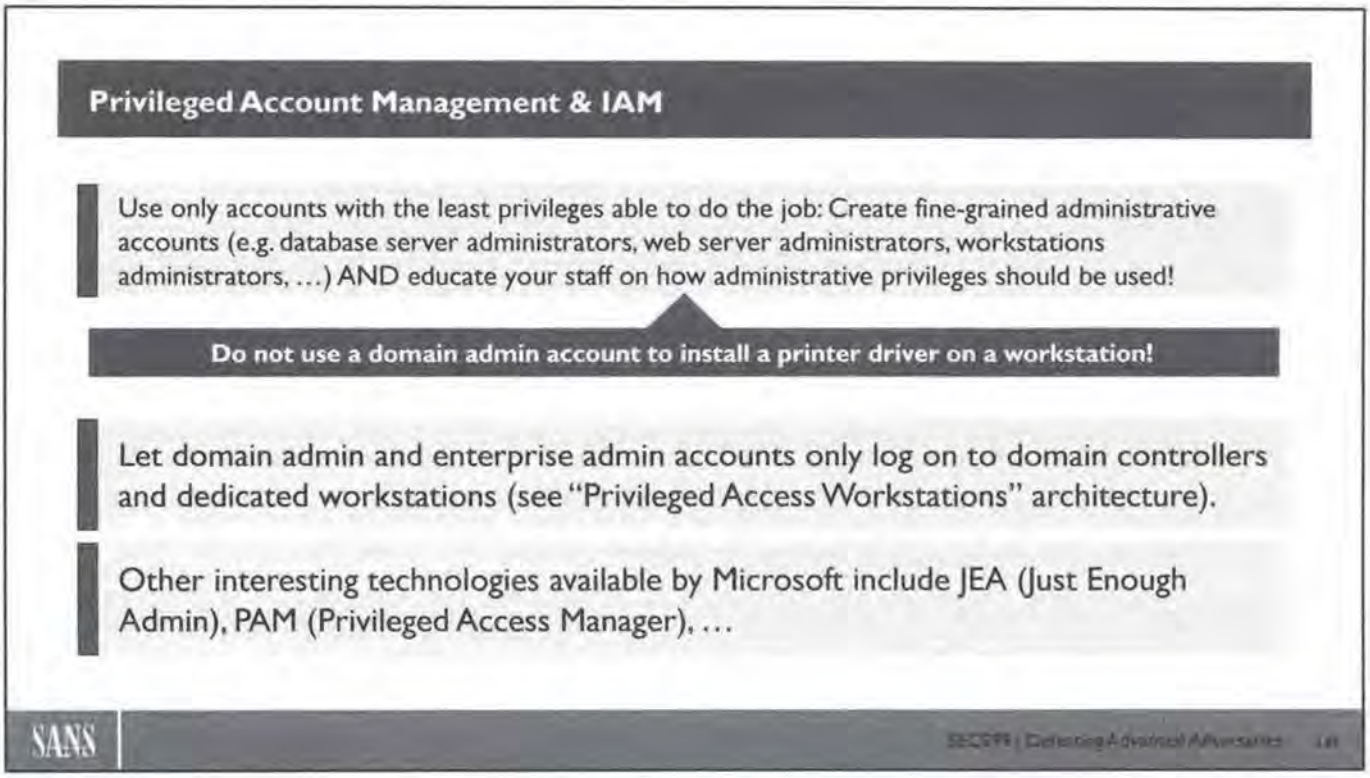
It is impossible for processes in the Windows environment to access processes in the Virtual Secure Mode environment, even by manipulating kernel data structures. All operations that require credentials, like checking NTML hashes, is done by the Isolated LSA upon request of the LSA. The credentials never leave the Isolated LSA.



Credential Guard – Does It Protect Against Pass-the-Hash?

Credential guard: the end of pass the hash? Some have proclaimed that Credential Guard is the end of pass-the-hash. We think this is rather premature for a few reasons:

- First of all, Credential Guard will prevent extraction of hashes from memory, which leaves different other options available (e.g. extracting hashes from the SAM or NTDS.dit files).
- Secondly, Credential Guard is only available for Windows 10 & Windows Server 2016, which leaves a potentially large group of vulnerable systems in your corporate environment. One Windows 8 or Windows Server 2012 machine and credentials can still be stolen from that machine.
- Credential Guard does not prevent using a hash for pass-the-hash attacks.
- As discussed before, Credential Guard protects against extraction of hashes, it does however not do anything to prevent a hash from being used in a pass-the-hash attack.



Privileged Account Management & IAM

Finally, we'd like to dedicate this slide to stress the importance of Identity & Access Management:

- Use only accounts with the least privileges able to do the job: Create fine-grained administrative accounts (e.g. database server administrators, web server administrators, workstations administrators, ...) AND educate your staff on how administrative privileges should be used! A classic mistake is to have helpdesk users wield Domain Administrator accounts to fix small issues on an employee's workstation. Use only accounts with the least privileges able to do the job.

- Let domain admin and enterprise admin accounts-only logon to domain controllers and dedicated workstations (see "Privileged Access Workstations" architecture).

Note that part of this will be to (again) raise employee security awareness: IT personnel should understand how to use (and how not to use) administrative privileges.

Active Directory Hardening & Segmentation – Additional Resources
Some additional resources concerning Active Directory Hardening & Segmentation:

Active Directory
https://github.com/gentilkiwi/mimikatz/wiki

Kerberos
https://en.wikipedia.org/wiki/Active_Directory

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.

Exercise – Hardening Windows to Stop Lateral Movement

The objective of the lab is to harden our Windows AD environment from the lateral movement strategies we just discussed. We will use a variety of techniques for this: disabling cached credentials & implementing CredentialGuard!

High-level exercise steps:

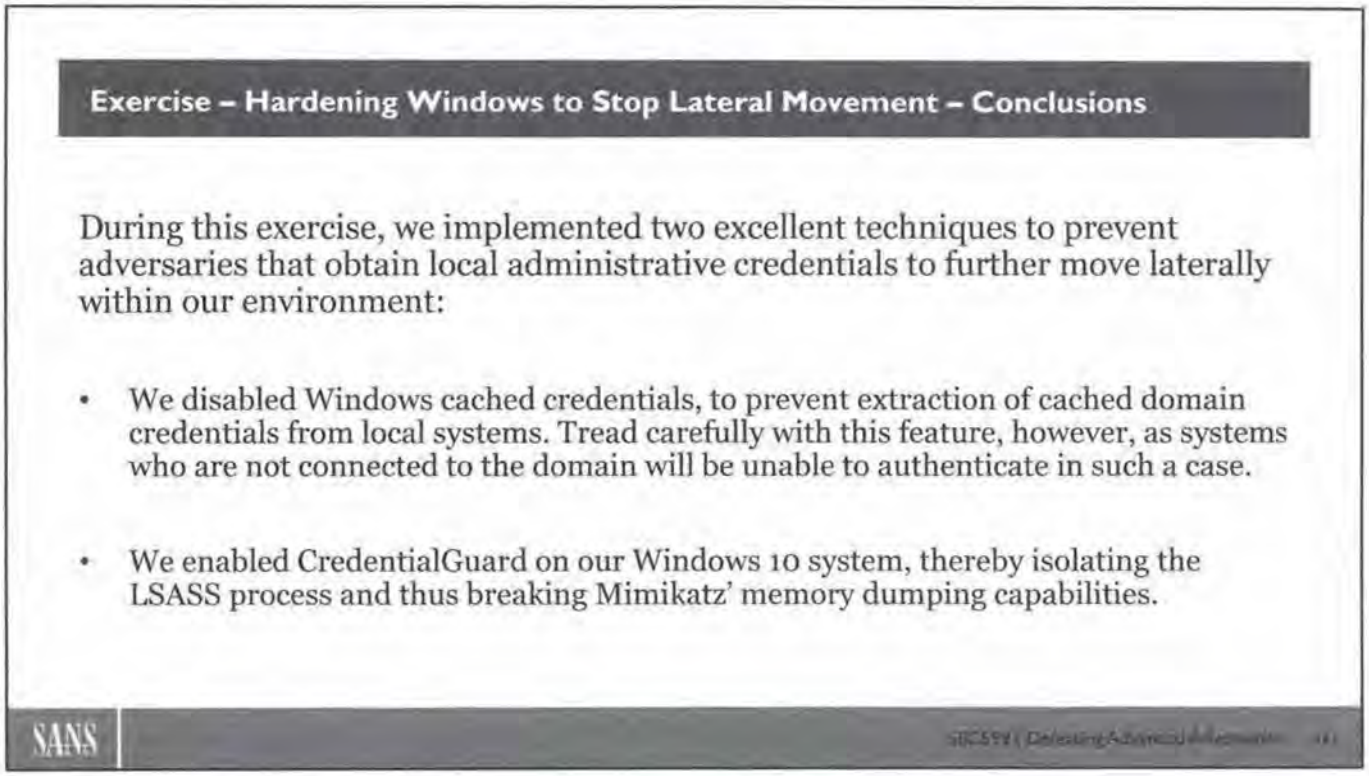
1. Stealing credentials from the cache & memory
2. Disabling cached credentials in Windows
3. Enabling enterprise guard throughout the environment
4. Confirming the fixes, we've added to our environment

Exercise – Hardening Windows to Stop Lateral Movement
The objective of the lab is to harden our Windows AD environment from the lateral movement strategies we just discussed. We will use a variety of techniques for this, including disabling cached credentials & implementing CredentialGuard!

Throughout the exercise, we will complete the following steps:

1. Stealing credentials from the cache & memory
2. Disabling cached credentials in Windows
3. Enabling enterprise guard throughout the environment
4. Confirming the fixes we've added to our environment

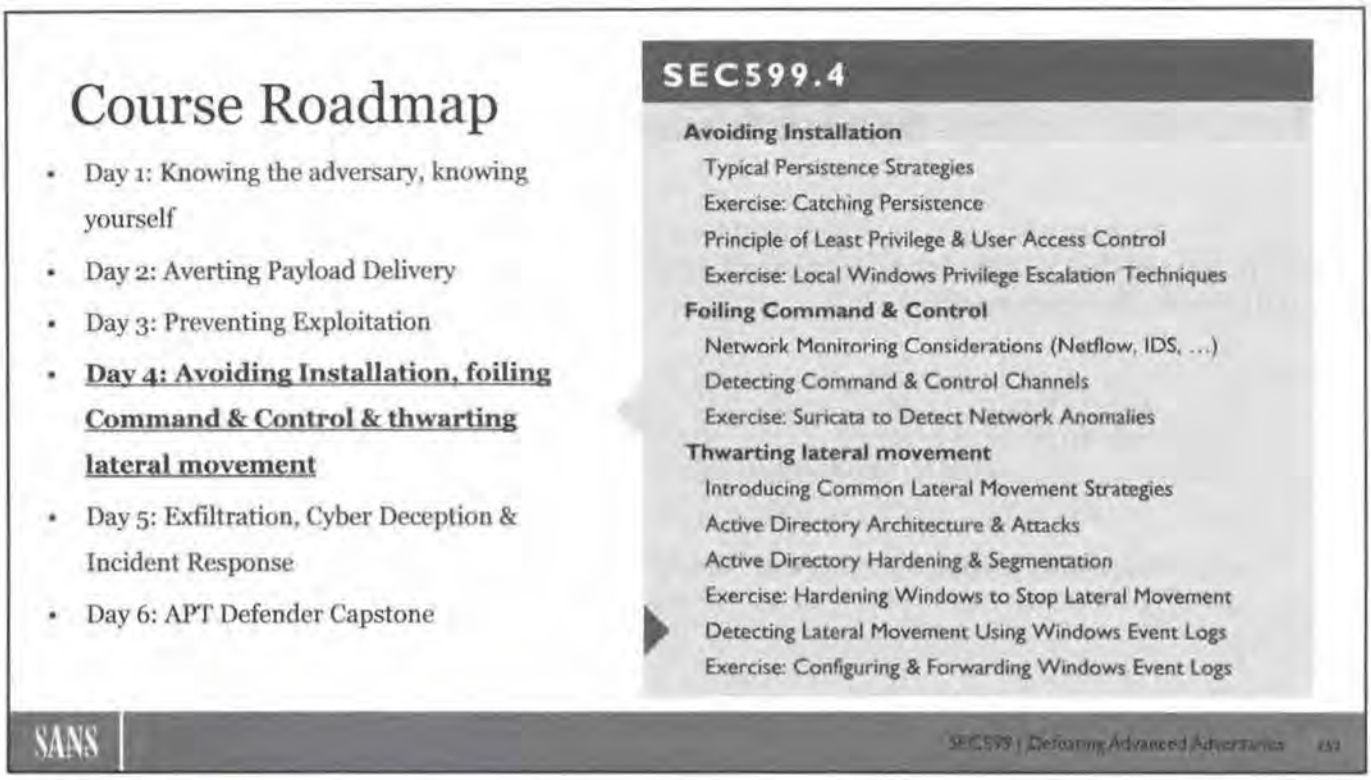
For additional guidance & details on the lab, please refer to the LODS workbook.



Exercise – Hardening Windows to Stop Lateral Movement – Conclusions

During this exercise, we implemented two excellent techniques to prevent adversaries that obtain local administrative credentials to further move laterally within our environment:

- We disabled Windows cached credentials, to prevent extraction of cached domain credentials from local systems. Tread carefully with this feature, however, as systems who are not connected to the domain will be unable to authenticate in such a case.
- We enabled CredentialGuard on our Windows 10 system, thereby isolating the LSASS process and thus breaking Mimikatz' memory dumping capabilities.



This page intentionally left blank.

Introducing Windows Event Logs

Windows event logs are log files kept by Windows to keep track of all sorts of events. They are typically stored locally in C:\Windows\System32\winevt\Logs\.

- Though not only designed for security investigations, Windows event logs are a GREAT way of obtaining visibility on what is happening on your endpoints;
- Up to Windows XP, the format used for Windows EventLog is EVT. Starting with Windows Vista, the format used for Windows XML EventLog is EVTX;
- Logs can be viewed locally with the event viewer (eventvwr.msc) or with various command line tools (wevtutil.exe, PsLogList, ...). Windows event logs can be forwarded & centralized to a central repository such as Splunk, Q-Radar, Arcsight, ELK, ...

Introducing Windows Event Logs

Since the introduction of Windows NT in 1993, the Windows operating system has kept logs of events that took place while the operating system was running. Events are very important for managing and troubleshooting Windows but are also very important for security. For example, to detect intrusions while they are happening, or for post-facto digital forensic investigations.

Applications and the Windows kernel can generate events. For example, Windows will generate several events when it is started, and many applications will generate events when they encounter an error.

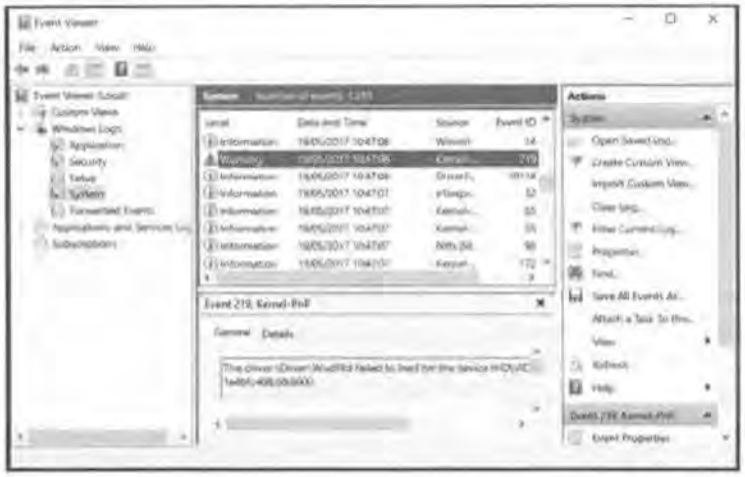
Windows events have a well-defined format and are stored in files called event logs, depending on their "source". For example, events that originate from the Windows kernel when Windows is started are stored in the System event log, while error events from applications (without dedicated event log) are stored in the Application event log.

By default, event logs are stored locally in folder C:\Windows\System32\winevt\Logs\. Up to Windows XP, a proprietary, binary file format was used to store event logs: the EVT format. Starting with Windows Vista, the format used for Windows XML EventLog is EVTX. This is a binary XML format. By switching to XML, Microsoft opens up its event logs for even more interoperability with third parties, but XML is a very verbose text format that would negatively impact performance. Therefore Microsoft opted for a binary XML format.

There are a couple of native Windows tools to view event logs. There is the event viewer, a graphical user interface tool that is a snap-in for the Microsoft Management Console (MMC). Wevutil.exe is a command line tool.

Event logs can also be viewed with PowerShell and many third-party tools, like Sysinternals' PsLogList.

The Event Viewer Interface



The event viewer (eventvwr.msc) is a graphical user interface tool to browse through event logs.

It is adequate for ad-hoc browsing, but not adapted to search through logs.

The filter functionality is for example extremely limited and does not allow us to combine complex conditions...

SANS | SEC599 | Defeating Advanced Adversaries 184

The Event Viewer Interface

The event viewer (eventvwr.msc) is a graphical user interface tool to browse through event logs. It is a snap-in for the MMC. In the screenshot above, we can see the various elements of the event viewer snap-in.

Left is a tree-view of all the different Windows logs. Originally, just a couple of logs existed on Windows NT (Application, Security, System): these can be found un Windows Logs. But since long, application and services can have dedicated Windows event logs. These can be found under "Applications and Services Logs".

Right to this tree-view, a list-view of all the events stored inside the event log is displayed. In the example above, we selected event log System in the tree-view, and thus the events of System are displayed in the list-view.

The list-view displays important properties of each event in columns, like the Level, the Date and Time, the Source, the Event ID, ...

When an event is selected, the details are displayed below the list-view: in the example above, we selected a Warning event entry with event ID 219 originating from the Kernel-PnP subsystem.

Event viewer is adequate for ad-hoc browsing, but it is not well adapted to search efficiently through a large number of events. Some form of automatic processing is required for efficient hunting.

XML Structure of Windows Event Logs

Wevtutil.exe is the built-in command line tool from Microsoft to manage event logs, and it can be used to review event logs from the command line. It can be used to export, archive, clear logs, … But in this example, we will show how to use it to query logs.

Wevutil.exe requires a command as an argument to instruct it what to do. In this example, where we want to query logs, we use command qe. This command requires the name of the log that we want to query, here we will use setup.

By executing this command, wevtutil will output all events in an XML format (text XML). This format is not very readable for humans but is appropriate for parsing by various tools. We can see that an event starts with the <event> XML tag and ends with </event>. The event ID of the first event (<EventID>) is 1.

Windows Event Logs Structure

Starting with Windows Vista, Microsoft overhauled its Windows event log systems. Because of the new flexibility to create dedicated event logs, a seemingly myriad number of event logs were created. Originally, there were only the following logs:

- Security: The Security log contains events such as valid and invalid logon attempts, as well as events related to resource use, such as creating, opening, or deleting files or other objects.
- Application: The Application log contains events logged by applications or programs.
- Setup: The Setup log contains events related to application setup.
- System: The System log contains events logged by Windows system components.

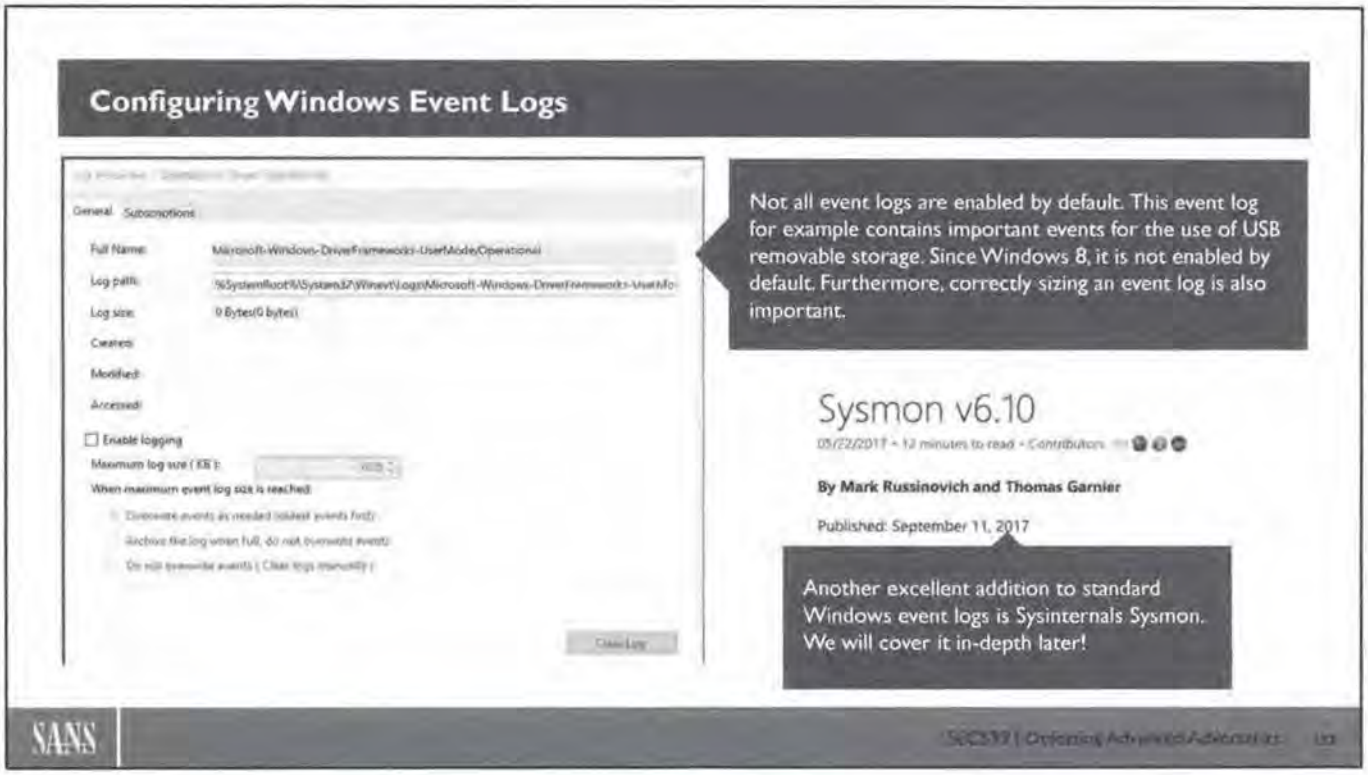
But since Windows Vista, many logs can be found under "Applications and Services Logs", but the traditionally most important ones remain.

Each event has an event ID, this is a number that indicates the type of event. For example, event 1104 means "The security Log is now full". Other event logs can have events with ID 1104, but they have a different meaning (if they exist). For example, an event log Microsoft-Windows-GroupPolicy, event 1104 means "Group Policy Preprocessing (WMI)".

An event has a "level", and this can be:

- Information
- Warning
- Error

It must be said that this level is not particularly useful for our purposes. For monitoring and detection purposes, a lot of "information" level events are important.



Configuring Windows Event Logs

When we look at the properties of an event log in event viewer, we can see information and options to configure the event log. We see the full name of the event log and the location where the log (.evtx file) is stored. More information includes the size of the log file and its timestamps.

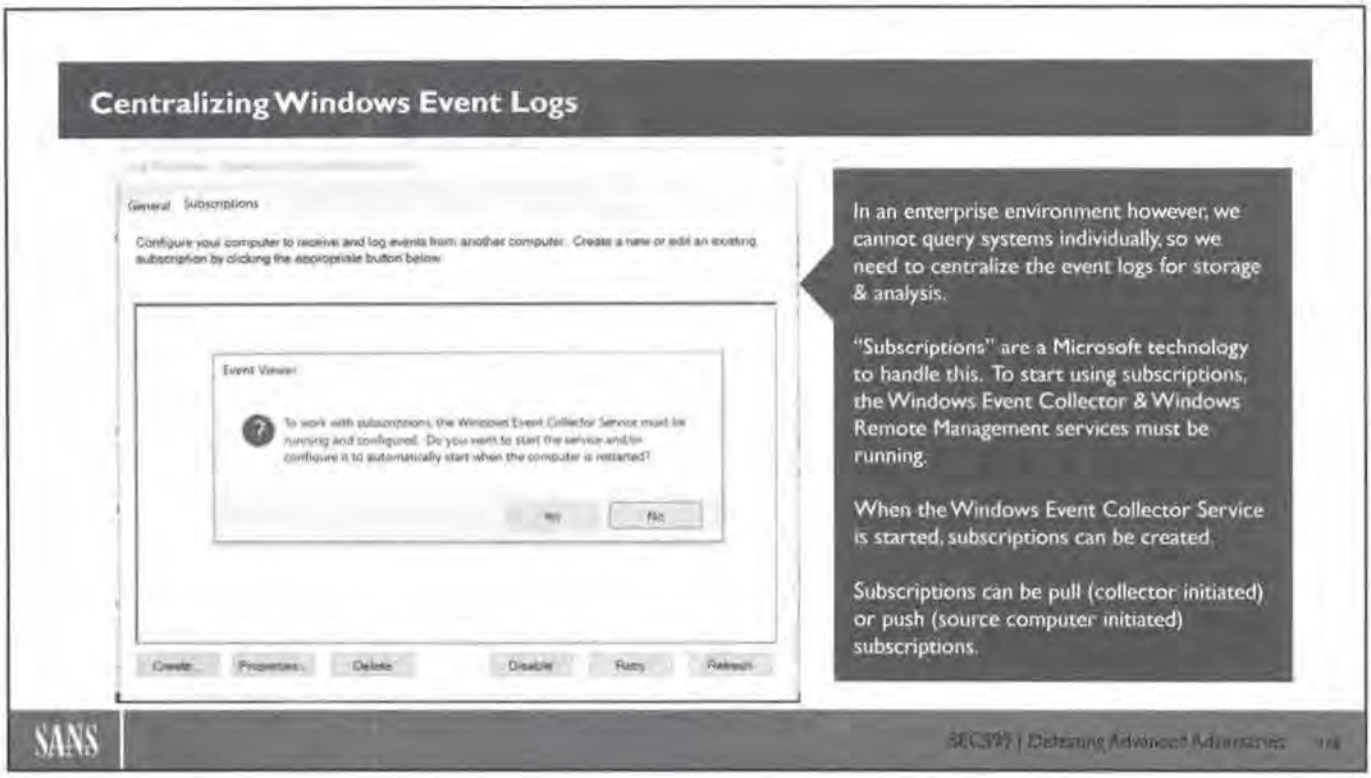
In the example above, the file size of event log DriverFrameworks-UserMode/Operational is zero, and it has no timestamps. That is because the event log is not enabled: the toggle Enable logging is not set.

We choose this event log as an example because it contains important events for the use of USB removable storage. For example, when a USB stick is inserted into the Windows machine, an event will be written to this event log. This event log is enabled by default on Windows 7, but since Windows 8, it is disabled by default. Another excellent addition to standard Windows event logs is Sysinternals Sysmon. We will cover it in-depth later!

We recommend enabling this event log on the workstations in your corporate environment. In this example, we illustrate this via the event log properties dialog in Windows event viewer, but this is something that you would typically configure via Active Directory GPOs.

Another important property is the maximum log size of an event log and the action that needs to be taken when this maximum would be exceeded. In this example, the maximum size would be just a bit more than 1MB (1028KB). The size of an event log must be tuned by taking into account the number of expected events, and the retention time needed.

By default, when the log grows to the maximum size, older events are dropped (the old .evt file format was a circular list). But an event log can be configured to archive older event logs, or just to stop logging (not recommended). When event logs are cleared manually (Clear Log button, this requires admin rights), an event indicating the log clearing is created.



Centralizing Windows Event Logs

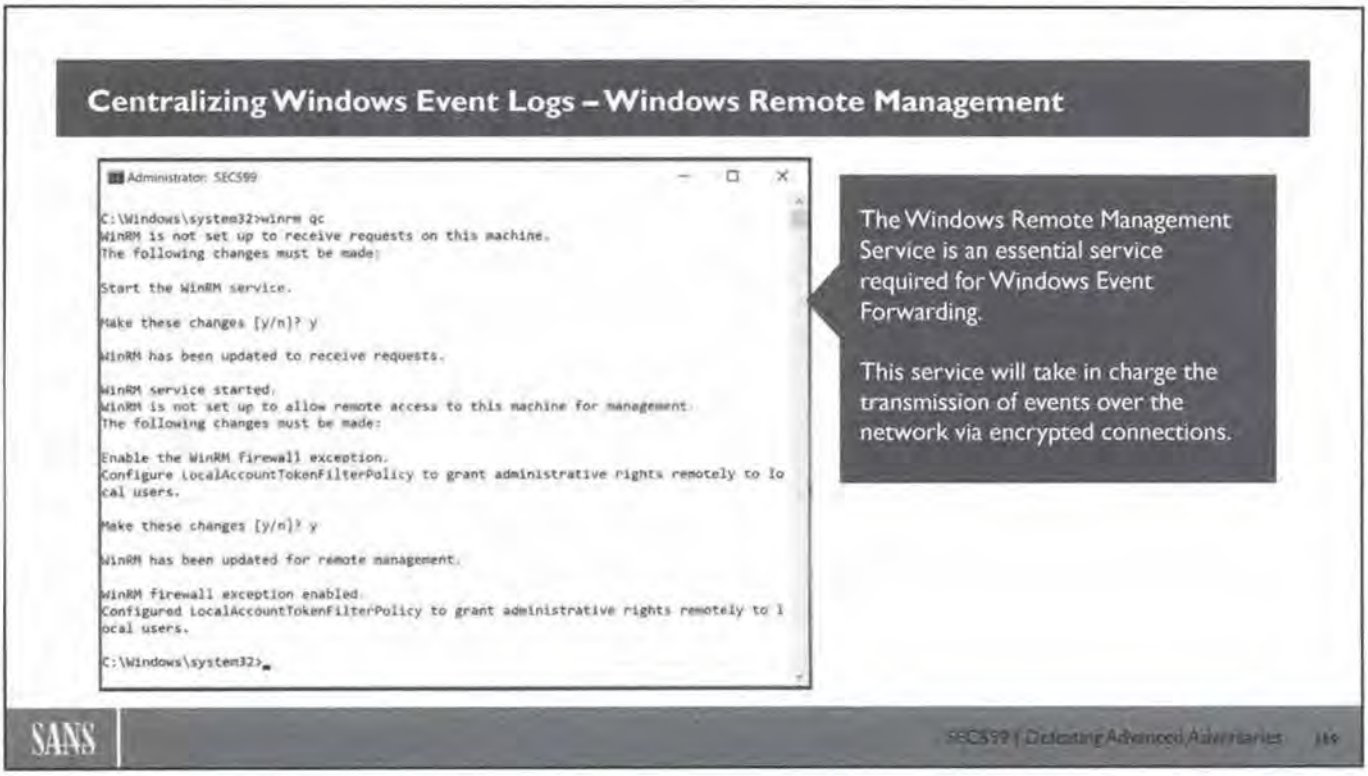
Event logs can be centralized. There are different third-party solutions for this, like Splunk, NXLog or Filebeat (ELK), but it can also be done using Microsoft's technology that comes out-of-the-box with Windows. This is called Windows Event Forwarding and is done with subscriptions.

Once the Windows Event Collector Service and Windows Remote Management Service are started, subscriptions can be created. Each subscription requires a name (to be chosen by the administrator), and an optional description. Subscriptions can be pull (collector initiated) or push (source computer initiated) subscriptions.

In a collector initiated subscription (pull), the source computers have to be selected. These have to be domain members. In a source computer initiated subscription (push), the source computers groups have to be selected. These groups can be domain members or non-domain members. In the case of non-domain members, certificates have to be provided for authentication and encryption.

The advantage of a source computer initiated subscription is that event logs from non-domain computers can be collected too. Forwarding all events from many computers would create too much data, therefor events can be filtered prior to forwarding.

Windows Event Forwarding requires no additional software and can be configured with a few simple steps. We will illustrate this via the eventvwr dialog, and with a command-line configuration, but of course, this configuration is not something you would do manually on each machine. This is something you would script with Active Directory GPOs.



Centralizing Windows Event Logs – Windows Remote Management
The Windows Remote Management Service is an essential service required for Windows Event Forwarding. This service will take in charge the transmission of events over the network via encrypted connections.

The Windows Remote Management Service not only requires to be started automatically when the computer starts, but it requires configuration of the firewall. This can be done with the winrm command.

Open an elevated command line prompt as administrator, and issue command "winrm qc". Reply yes to both questions that are asked.

Filtering Windows Event Logs

As you can imagine, Windows event logging can generate a lot of events & thus a lot of noise...

We can however implement filters that will ensure only events are being forwarded that meet certain conditions:

- Time when the event occurred;
- Event level (error, warning, information, ...)
- Event ID
- Username
- ...

SANS | SEC599 | Defeating Advanced Adversaries | 160

Filtering Windows Event Logs

As you can imagine, Windows event logging can generate a lot of events & thus a lot of noise… We can, however, implement filters that will ensure only events are being forwarded that meet certain conditions. Events can be filtered by many criteria:

- Time when the event occurred
- Event level (error, warning, …)
- Event ID
- Task category
- Keywords
- User
- Computer

We recommend filtering by event ID (and use the event IDs as discussed before in the NSA document). To configure filtering by event ID, the log or source must be selected.
Then it is possible to enter event IDs in the filter field. Event IDs have to be separated by a comma (,). A range of event IDs can be specified by separating the minimum and maximum event ID by a dash (-).

All the events specified will be included. To exclude particular events, their event IDs have to be "negative", for example -1104. Only events that match the criteria defined in this dialog will be forwarded.

Alternatives for Event Forwarding – Winlogbeat & Nxlog

Another approach preferred by some organizations is the addition of host-based agents for Windows event forwarding. We will shortly discuss two solutions, recognizing that many others exist:

Winlogbeat is lightweight log agent that can be installed on Windows systems to transfer the event log data towards Elasticsearch server. This agent runs as Windows service and you can specify which event logs to monitor by editing the "winlogbeat.yml" file.

NXLog is an opensource multi-platform log agent. This agent is similar to syslog-ng and rsyslog with the exception that it supports multiple platforms, including windows event logs.

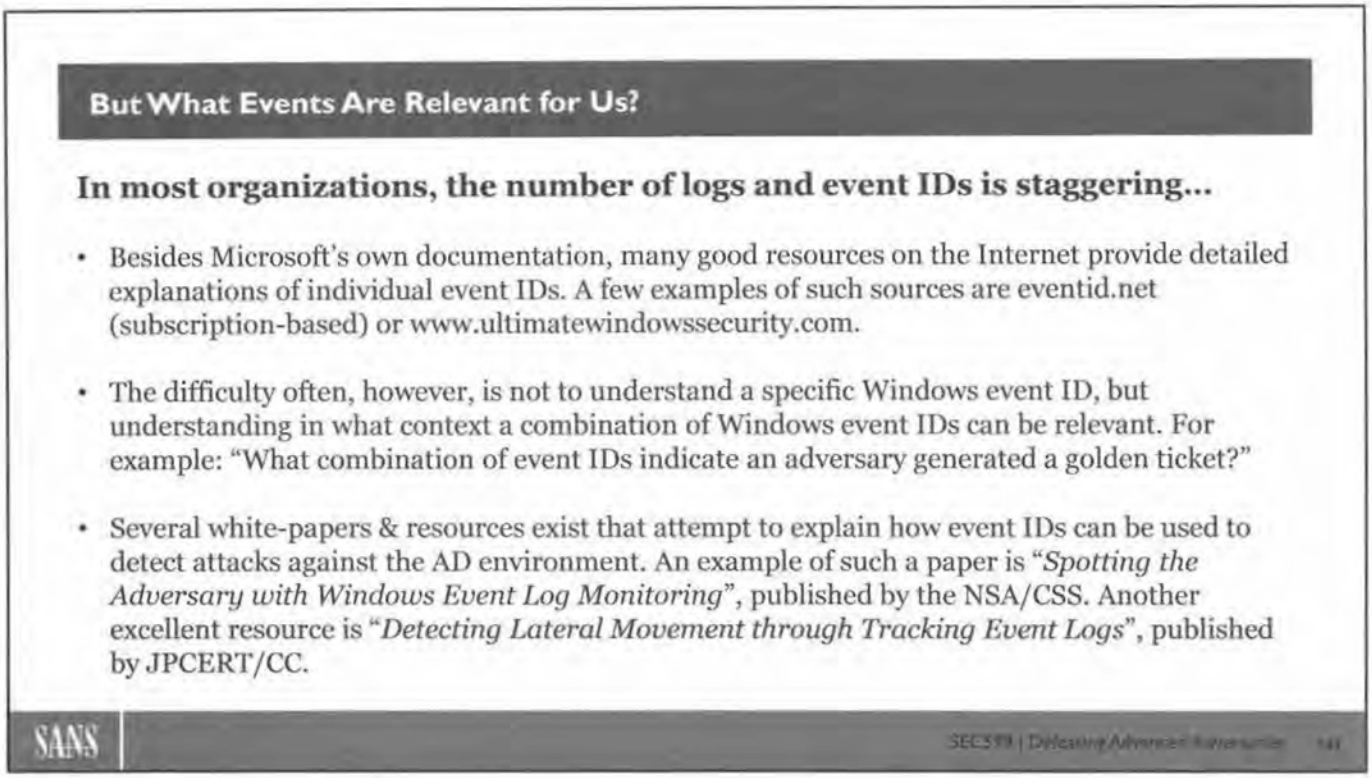
In large corporate environments host-based agents are preferred for implementation to standardize logging capabilities and configurations on multiple platforms.

Alternatives for Event Forwarding – Winlogbeat & NXLog

Another approach preferred by some organizations is the addition of host-based agents for Windows event forwarding. The advantage of these is that they often provide additional functionality to allow sending of Windows event logs to non-Microsoft systems such as an Elasticsearch cluster. We will shortly discuss two solutions, recognizing that many others exist:

- Winlogbeat is lightweight log agent that can be installed on Windows systems to transfer the event log data towards Elasticsearch server. This agent runs as a Windows service, and you can specify which event logs to monitor by editing the "winlogbeat.yml" file.
- NXLog is an opensource multi-platform log agent. This agent is similar syslog-ng and rsyslog with the exception that it supports multiple platforms, including Windows event logs.

In large corporate environments, host-based agents are preferred for implementation to standardize logging capabilities and configurations on multiple platforms.



But What Events Are Relevant for Us?
As stated before, there are a lot of Windows event logs, and therefore even more, event IDs. It requires a lot of practice and experience to understand the most important event IDs and to know how to respond adequately.

To help with the understanding of Windows event logs, several good resources are available on the Internet. Microsoft own documentation is a bit lacking when it comes to event logs. Not all event IDs are clearly documented, and explain exactly what they mean.

This situation has led to the rise of dedicated resources for Windows event logs. There are several websites that try to define as much Windows event IDs as possible. One of these sites is EventId.net. This is a subscription based service. Another one is https://www.ultimatewindowssecurity.com/securitylog/default.aspx, that will provide some information without registration.

The difficulty often, however, is not to understand a specific Windows event ID, but understanding in what context a combination of Windows event IDs can be relevant. For example: "What combination of event IDs indicate an adversary generated a golden ticket?" Third parties have also collected all their knowledge and recommendations on Windows event logs in a single resource. An excellent example is the document "Spotting the Adversary with Windows Event Log Monitoring" produced by the NSA/CSS.

eventid.net & www.ultimatewindowssecurity.com
Next to the official documentation provided by Windows, sources like eventid.net & www.ultimatewindowssecurity.com provide in-depth details on types of Windows event IDs.

EventId.net is a subscription-based website with tons of information. The number of an event ID can be typed in the form above, and then all sources that have an Event ID with a particular number will be listed. Searching can be limited to a particular source. Besides the source and explanation of the event ID, users can post comments. Some information is available for free without registration, but buying a subscription opens up much more information.

On the other hand, www.ultimatewindowssecurity.com, which is illustrated on the slide, provides in-depth information without any subscription being required. Both are excellent sources of additional information!

Interesting Windows Event IDs

Michael Gough (@HackerHurricane) created a presentation in 2015 where he listed six highly interesting Windows event IDs everyone should monitor. He labelled them the "sexy six". He advises the following steps to be taken:

- Enable command line logging!
- Enable Advanced Audit Policy in Windows
- Configure the following event ID's to be logged:

Event ID	Description	Event ID	Description
4688	A new process was executed	**5156**	Windows firewall network connection
4624	Successful authentication	**7045**	A new service was added to an endpoint
5140	A network share was accessed	**4663**	File & Registry auditing

Next to these "six", here are a few others we have found to be very useful:

4648	Logon using "explicit credentials"	**4728** **4732** **4756**	A used was added to a "privileged" group

SANS

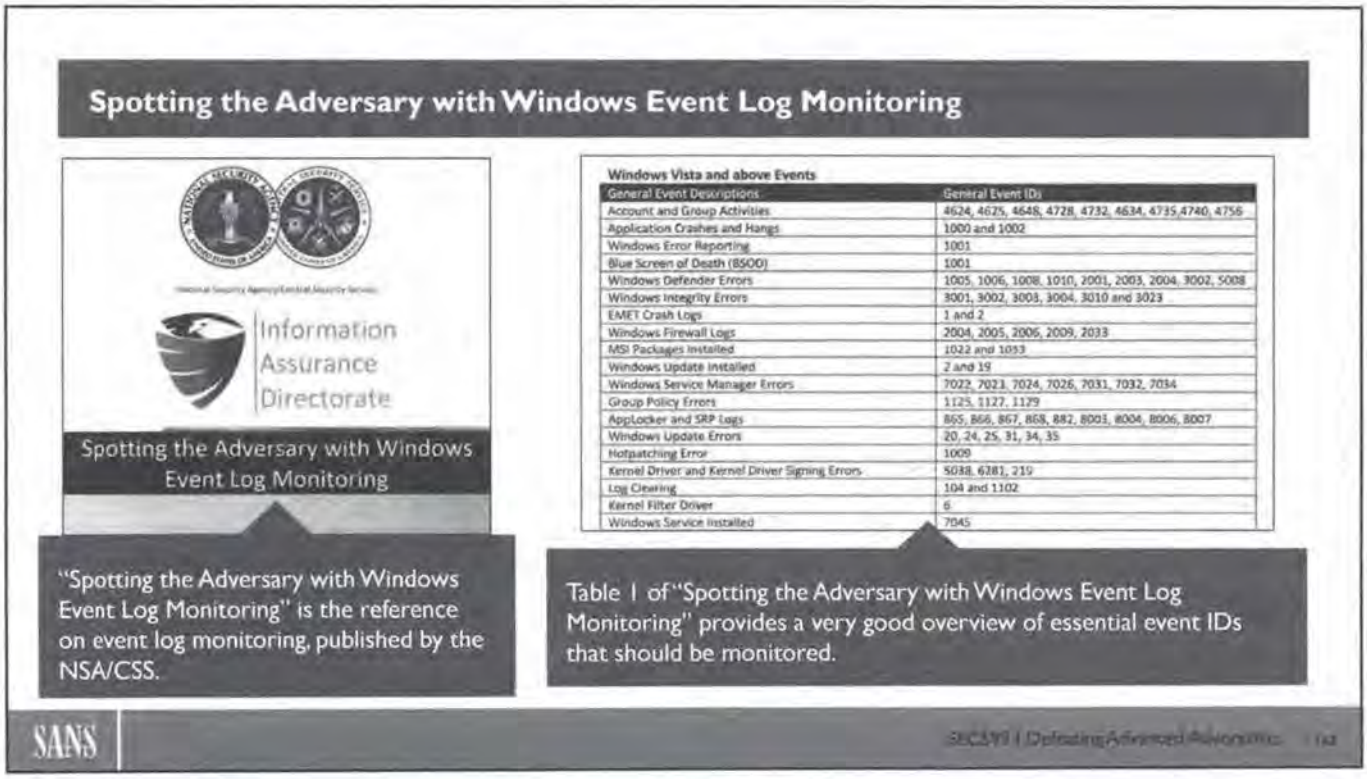
Interesting Windows Event IDs

Michael Gough (@HackerHurricane) created a presentation in 2015 where he listed six highly interesting Windows event IDs everyone should monitor. He labeled them the "sexy six". He advises the following steps to be taken:

- Enable command line logging!
- Enable Advanced Audit Policy in Windows
- Configure the following event ID's to be logged:
  - 4688: A new process was executed
  - 4624: Successful authentication
  - 5140: A network share was accessed
  - 5156: Windows firewall network connection
  - 7045: A new service was added to an endpoint
  - 4663: File & registry auditing

Next to these "six", here are a few others we have found to be very useful:

- 4648: Logon using "explicit credentials". When accessing remote systems in a Windows environment, there is typically no explicit credentials being used (due to Windows access tokens & SSO). Whenever authentication is performed using "explicit credentials" (i.e. explicitly providing a password), this event ID will trigger;
- 4728, 4732, 4756: A user was added to a privileged group. This can be useful in later stages of the APT Attack Cycle, should adversaries be creating additional administrative accounts.



Spotting the Adversary with Windows Event Log Monitoring
"Spotting the Adversary with Windows Event Log Monitoring" is the reference on event log monitoring, published by the NSA/CSS. This lengthy document (around 50 pages) is a must read for blue teams. It was last reviewed 16 July 2015.

The following topics are covered in detail:
- Deployment
- Hardening Event Collection
- Recommended Events to Collect
- Event Log Retention
- Final Recommendations

Deployment will not only cover the configuration of Windows event logs but also centralization of these logs using Microsoft's publisher/subscriber model. It must be said that this is not the only way to centralize event logs. There are many other systems, for example, Splunk comes to mind.

https://www.iad.gov/iad/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm

Table 1 of "Spotting the Adversary with Windows Event Log Monitoring" provides a very good overview of essential event IDs that should be monitored.

The event IDs themselves are not explained in much detail in this document, however.

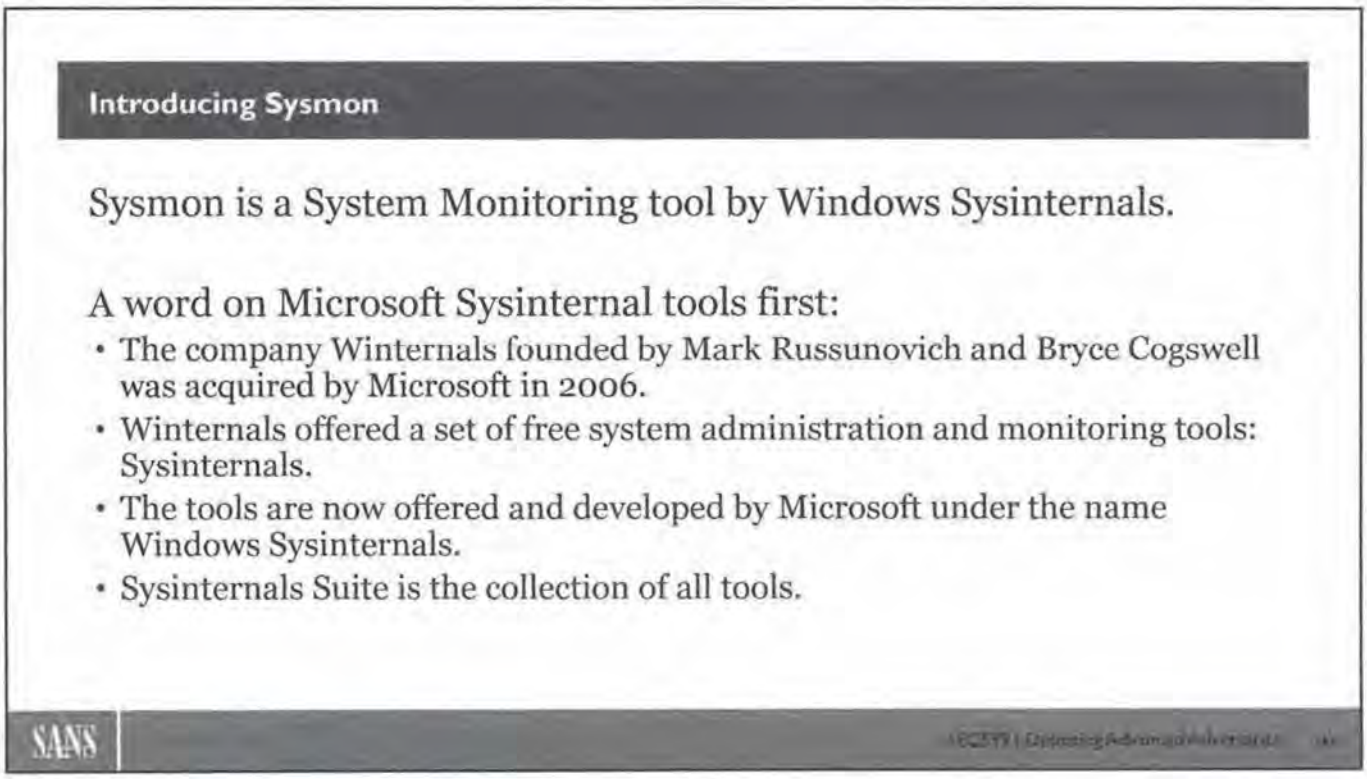
Let's illustrate this with event ID 4648. This ID is mentioned first in table 1 (under account and group activities), and second in table 9. From table 9, we know that this event is from the Security event log. The description of this event is "Account login with explicit credentials". However, the document does not provide more information about this event, and why it is important.

If we turn to eventid.net, we get more information:

"This event is generated when a process attempts to log on an account by explicitly specifying that account's credentials. This most commonly occurs in batch-type configurations such as scheduled tasks, or when using the RUNAS command."

And there is also a user comment that explains in which circumstances the user saw this event.

From this information, we can indeed conclude that this is an important event ID to monitor: with single-sign-on, users on Windows don't have to provide explicit credentials to access a remote service. If this happens, it means that a user could not use single-sign-on. This can indicate an attack, where the adversary is impersonating another user.



Introducing Sysmon

Sysmon (System Monitoring) is a tool by Windows Sysinternals. Before we dive into this tool, a bit of history can help us understand where these tools come from.

Winternals was a company founded by Mark Russinovich and Bryce Cogswell. This company provided Windows administration tools as paying products (Winternals) and free products (Sysinternals). In 2006, Microsoft acquired Winternals, Mark Russinovich became a Microsoft Technical Fellow, and the Sysinternal tools were rebranded as Windows Sysinternals.

Mark Russinovich still maintains and develops new tools for the Sysinternals Suite (the collection of Windows Sysinternals administration and system tools). These tools have a shorter development life-cycle than other Microsoft products. It is not unusual to have several updates per year for tools like Sysmon.

Sysmon

Sysmon is short for System Monitoring.

Sysmon installs a Windows service and a device driver.

These components monitor activity on a system:

- Creation and termination of processes
- Loading of executable images
- Network connection establishing
- ...

Activity is recorded with events.

Sysmon
Sysmon is a system monitoring tool that is part of the Sysinternals Suite.

It is a tool that was originally developed for Microsoft. It is deployed on many of their servers and workstations to monitor system activity. In case of incidents, Sysmon provides a valuable log of system activities that can help forensic investigators to reconstruct an incident.

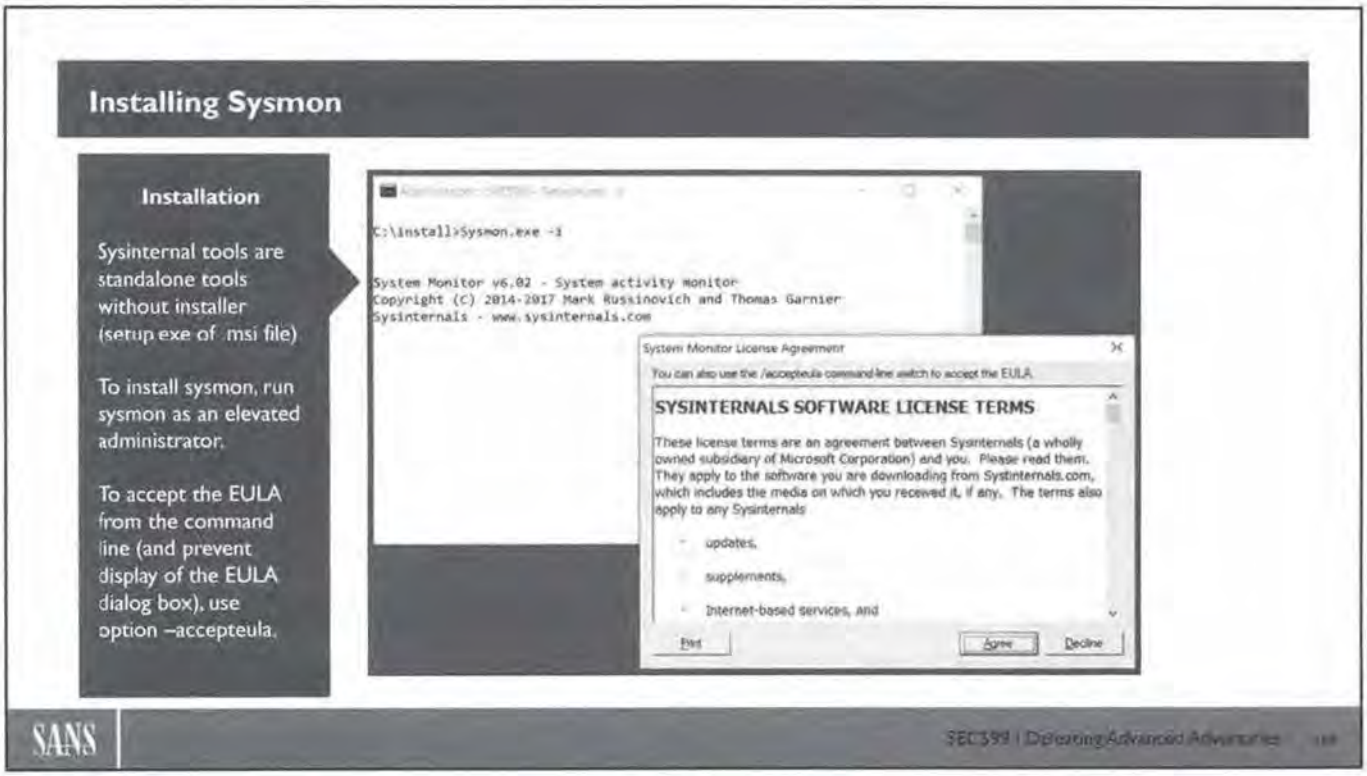
Sysinternal tools are stand-alone tools that don't come with an installer (like setup.exe or install.msi). For Sysmon, there is Sysmon.exe and Sysmon64.exe.

Sysmon.exe is a 32-bit version that embeds the 64-bit version too.
Sysmon64.exe is a 64-bit version only, it is provided for Windows systems that only support 64-bit executables, and not 32-bit (Windows Servers without 32-bit subsystem).

If the 32-bit version is executed on a 64-bit OS, it will extract the 64-bit version and run that instead.

When Sysmon is installed on a Windows machine, it installs a Windows service and a device driver. These components are necessary to detect and record system activities like the creation and termination of processes, loading of executable images, creation of network connections, loading of drivers, ...

Al this activity is logged in a dedicated Windows event log.



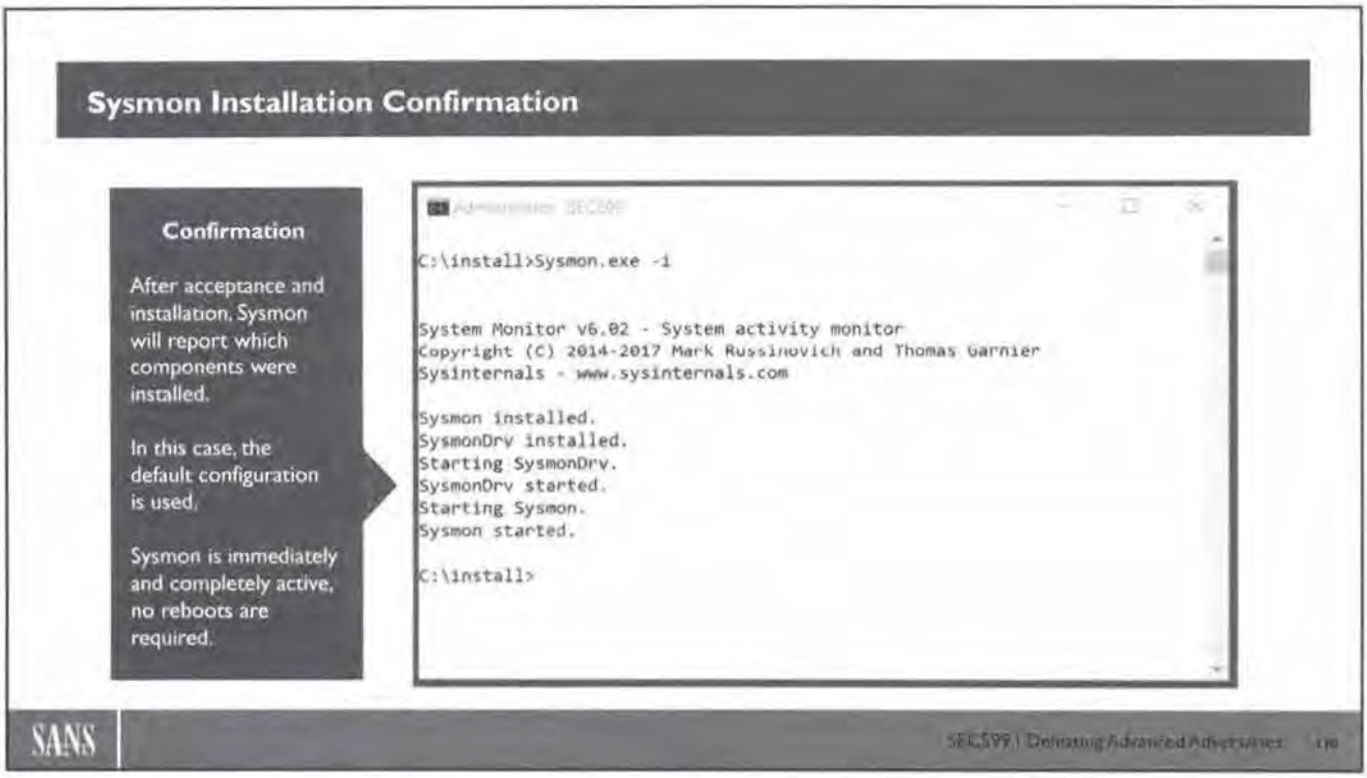
Installing Sysmon

To install Sysmon on a machine, start command "sysmon.exe –I" from an elevated, administrative command prompt.

Before the installation of the Service and driver takes places, a dialog with the EULA will be presented to the user. This EULA has to be accepted to proceed with the installation.

This EULA is presented the first time sysmon is executed. This is the case for all Sysinternals tools: after acceptance of the EULA, the dialog is no longer displayed.

As this dialog hinders unattended deployment of a tool like sysmon.exe, the EULA can also be accepted by providing option –accepteula on the command line. With this option, the dialog box with the EULA will never be displayed. Remark that typing –accepteula is considered equivalent to clicking on the Agree button: you accept the EULA.

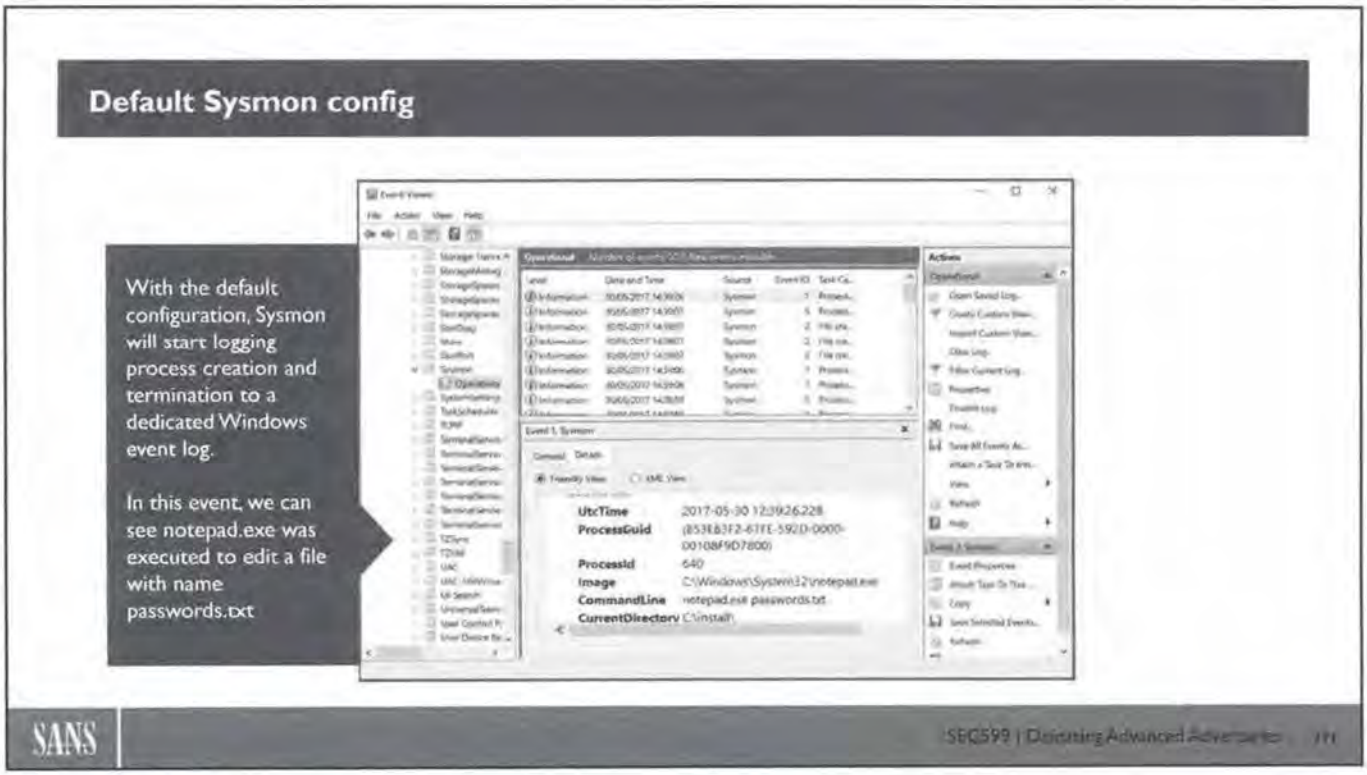


Sysmon Installation Confirmation

After accepting the EULA, Sysmon will install its service (Sysmon) and its driver (SysmonDrv). In the screenshot above, we can see that installation was successful: the service and the driver were installed and started. No reboot is required.

Without configuration file or configuration option, Sysmon will be installed with the default configuration and will run with these settings until they are changed.

Sysmon can be deployed as an application with group policies on all the machines in your environment. If you need custom configurations (which we recommend), it can be easier to package sysmon and the configuration file in an MSI file and deploy that through group policies.



Default Sysmon config

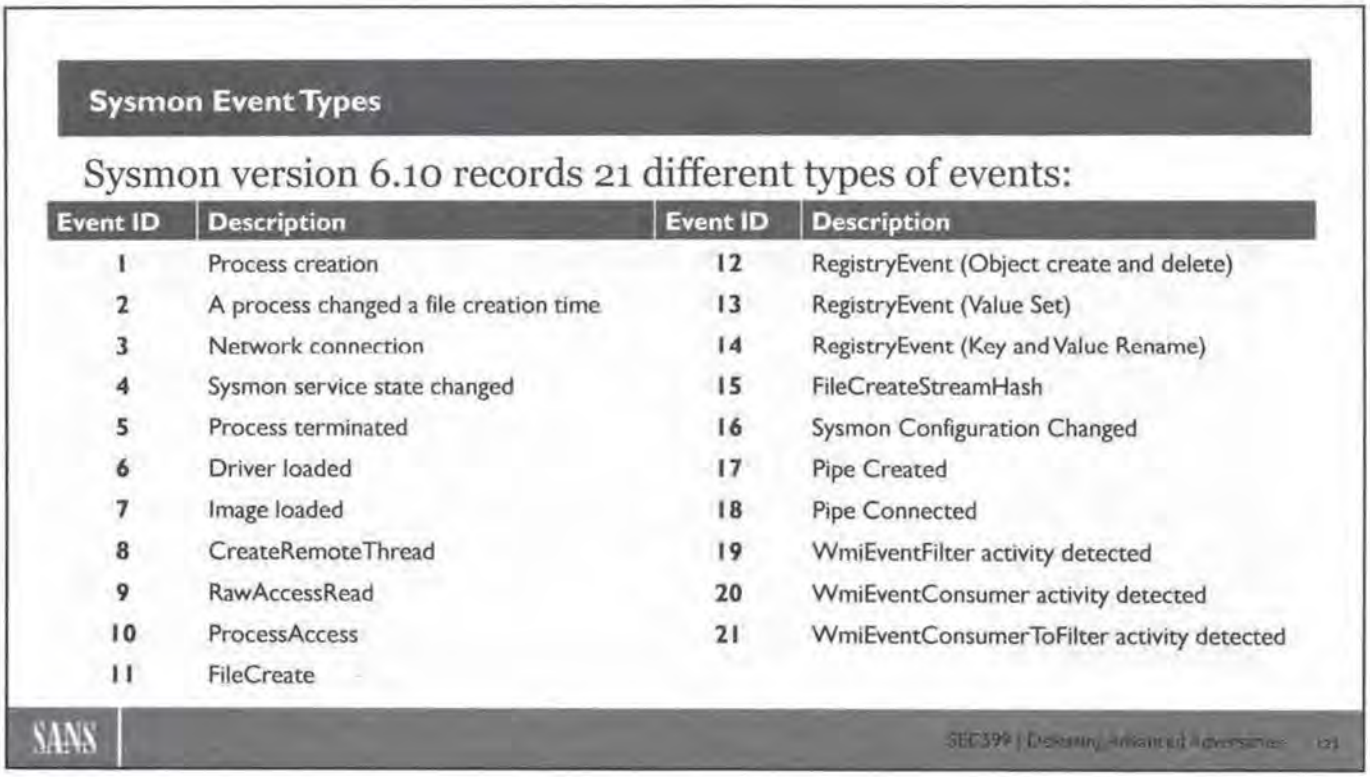
Once Sysmon is installed, it will start monitoring system activity and create events in a dedicated Windows event log (Sysmon / Operational). These events contain a lot of information that can help us reconstruct what happened on a machine.

In the example above, we see a Process Creation event (Event ID 1): this event is created each time a new program is launched. The program that was launched is notepad.exe.

From the commandline property, we can see what file was edited with notepad.exe: passwords.txt

Sysmon not only records the Process ID, but also a Process GUID. Process IDs are a 16-bit integer. Because of this limitation (65536 possible combinations), Process IDs can be reused: for example, after a couple of days, depending on the amount of activity of your system, a new process can be created that will also have the PID 640.

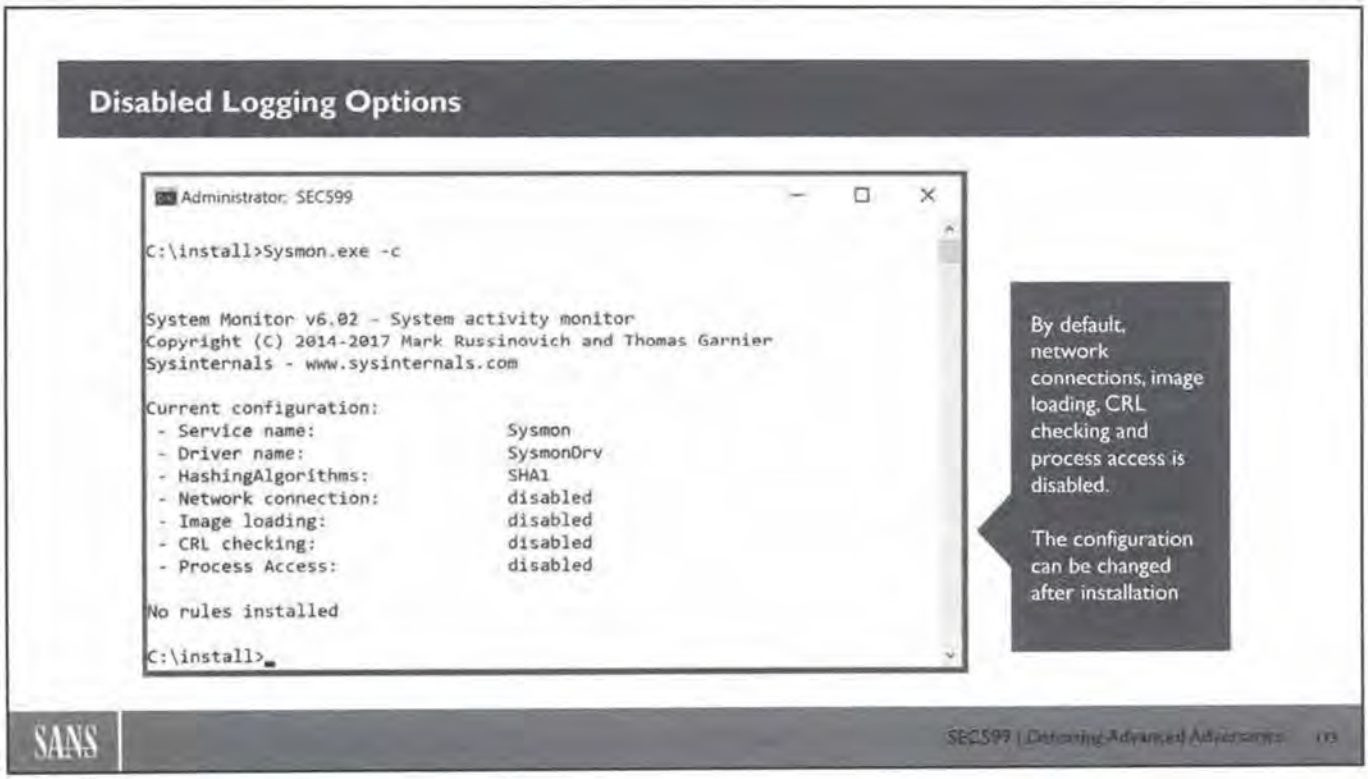
Since this can be a problem for log correlation, Sysmon will create a unique Process GUID per process. Another useful piece of information in a Process Creation event is the cryptographic hash of the content of the executable (notepad.exe in our case). By default, this is a SHA1 hash, but Sysmon can be configured to use more hashes too.



Sysmon Event Types

Sysmon version 6.10 records 21 different types of events (excluding event ID 255: internal error). These events monitor objects like processes, files, registry objects, … But also, events to monitor changes to Sysmon itself (IDs 4 and 16) can be logged. This can indicate tampering attempts.

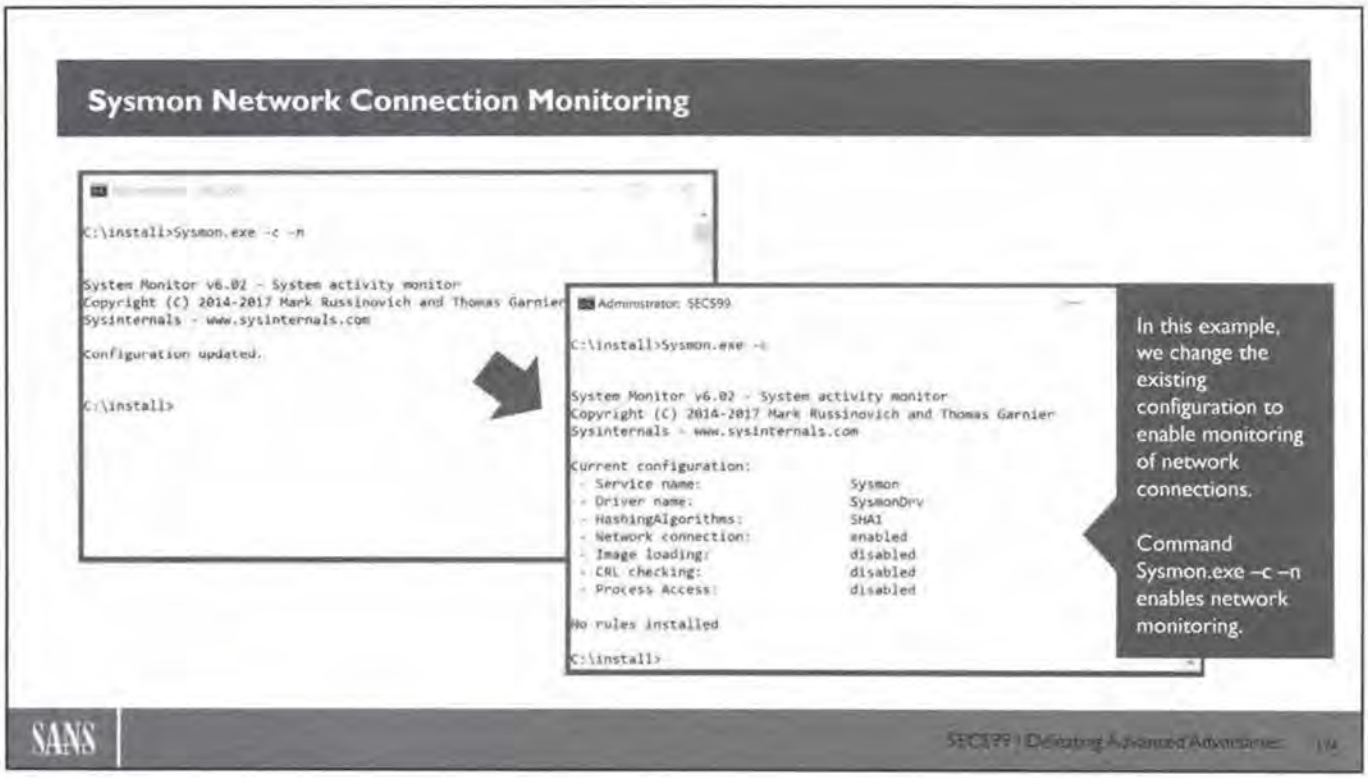
Other event IDs that can be indicative of tampering by malicious actors are changes in file creation times (ID 2), creation of remote threads (ID 8), often used for code injection, opening of processes (ID 10), …



Disabled Logging Options

By default, Sysmon will not generate all these events. Sysmon needs to be configured, but it can take up too many resources if it is configured to record all events.

As an example, we will enable the logging of network activity. By default, Network connection logging is disabled, as can be seen in the screenshot above: the configuration can be retrieved with option –c.



Sysmon Network Connection Monitoring

To configure network connection monitoring interactively, we can issue the following command (as an elevated administrator):

Sysmon.exe –c –n

This will update the configuration and apply it immediately.

To check that the configuration was updated as we desired, we can use command Sysmon.exe –c to check the updated configuration.

Remark that this interactive configuration method cannot scale. For enterprise deployments, configuration files can be used.

Monitoring a Malware Simulation

Network connection

This is a (simulated) example of malware with network activity.

The program malware.exe downloads from http://www.example.com.

This is logged with an event for a TCP connection with the source and destination IP addresses and ports.

SANS SEC599 | Defeating Advanced Adversaries

Monitoring a Malware Simulation
This is a network connection event (ID 3) created by a simulated malware.

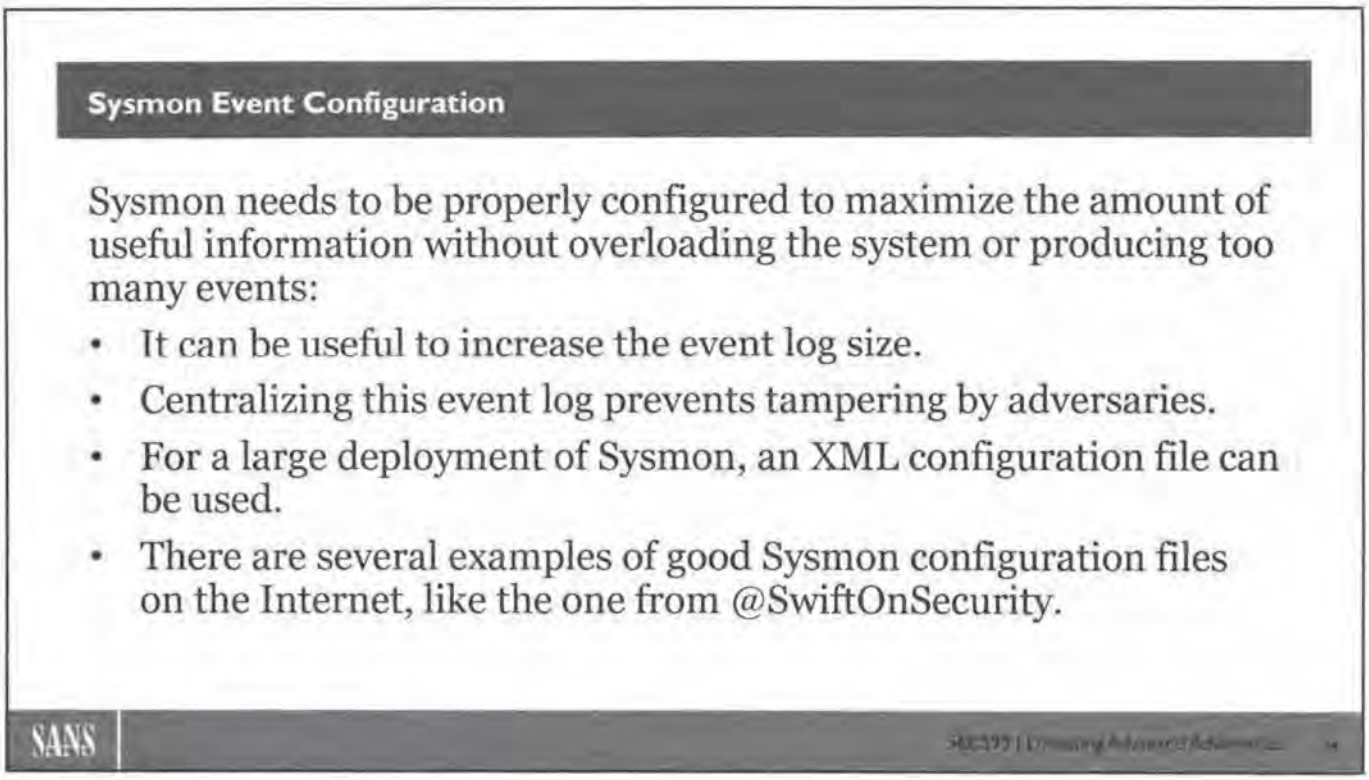
Malware.exe is an executable that will download a file via http from www.example.com when it is executed.

In the Sysmon event log, we can see the process creation and termination events, followed by the network connection event.

Data in the event allow us to identify the program (via Process ID, Process GUID, and image) and what connection is established. We can see it is a TCP connection from a private IPv4 address to a public IPv4 address on port 80.

So, the network level we see is TCP, not HTTP. We do not see the domain name, and not the content of the network connection.

In a corporate environment, many of the connections from workstations will be to the corporate proxy. Time correlation with the proxy logs allow us to know more about the established connection.



Sysmon Event Configuration

As Sysmon can create a huge amount of events, that can impact performance of the machine, or flood the event log with events thereby flushing out older events, a system needs to be configured properly for Sysmon to operate well.

By configuring Sysmon via a configuration file, filtering can be applied. Filtering denotes which type of events need to be included or excluded. For example, one could configure Sysmon to not log network connections established by Internet Explorer, but to do log network connections established by Word. This will already significantly reduce the amount of events.

Increasing the size of the Windows event logs is a good idea to increase the retention period (once an event log is full, old events are discarded to make a place for new events). Centralizing event logs can also alleviate the lack of space, and is also a solution to combat log tampering.

The Sysmon configuration file is an XML file. This is very useful to deploy and configure Sysmon on many systems.

Good examples of Sysmon configuration files can be found on the Internet, like Internet security legend @SwiftOnSecurity. This configuration file has been deployed on thousands of systems and has been in use for more than a year.

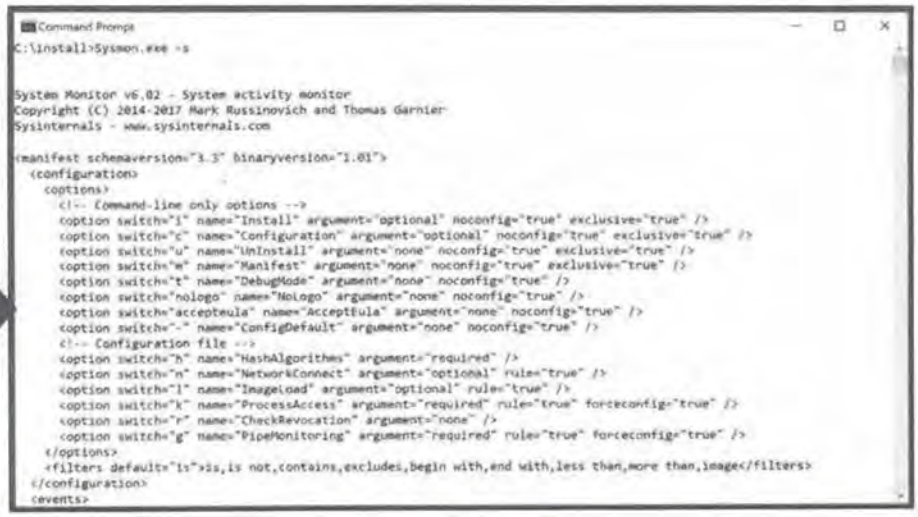
Sysmon Configuration File

XML file

Sysmon can be configured with a configuration file.

This file is provided with option –c. It is an XML file.

The configuration schema can be dumped with option –s. This XML file can be the start of your configuration.



Sysmon Configuration File

To start creating your own configuration file, you can use Sysmon to produce an XML file with the schema (all possible configuration options).

This can be done with command Sysmon.exe –s.

The output can be redirected to a file, which can then be edited and tested.

Detecting Lateral Movement Through Tracking Event Logs (1)

Next to the NSA's interesting article on Windows event logs, there is a highly useful paper that was released by JPCert in June 2017

The document aims to provide a good overview of common attack techniques & tools and how they can be detected!

https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf

Detecting Lateral Movement Through Tracking Event Logs (1)
Next to the NSA's interesting article on Windows event logs, there is a highly useful paper that was released by JPCert in June 2017. The document aims to provide a good overview of common attack techniques & tools and how they can be detected! We will zoom in on the document by walking through a specific example of such a tool: Mimikatz!

The full paper can be found at the following link:

https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf

Detecting Lateral Movement Through Tracking Event Logs (2)

We will now take an example of an attack tool that is further explained in the JPCERT's paper: Mimikatz!

3.3.4. Mimikatz (Obtaining Password Hash)

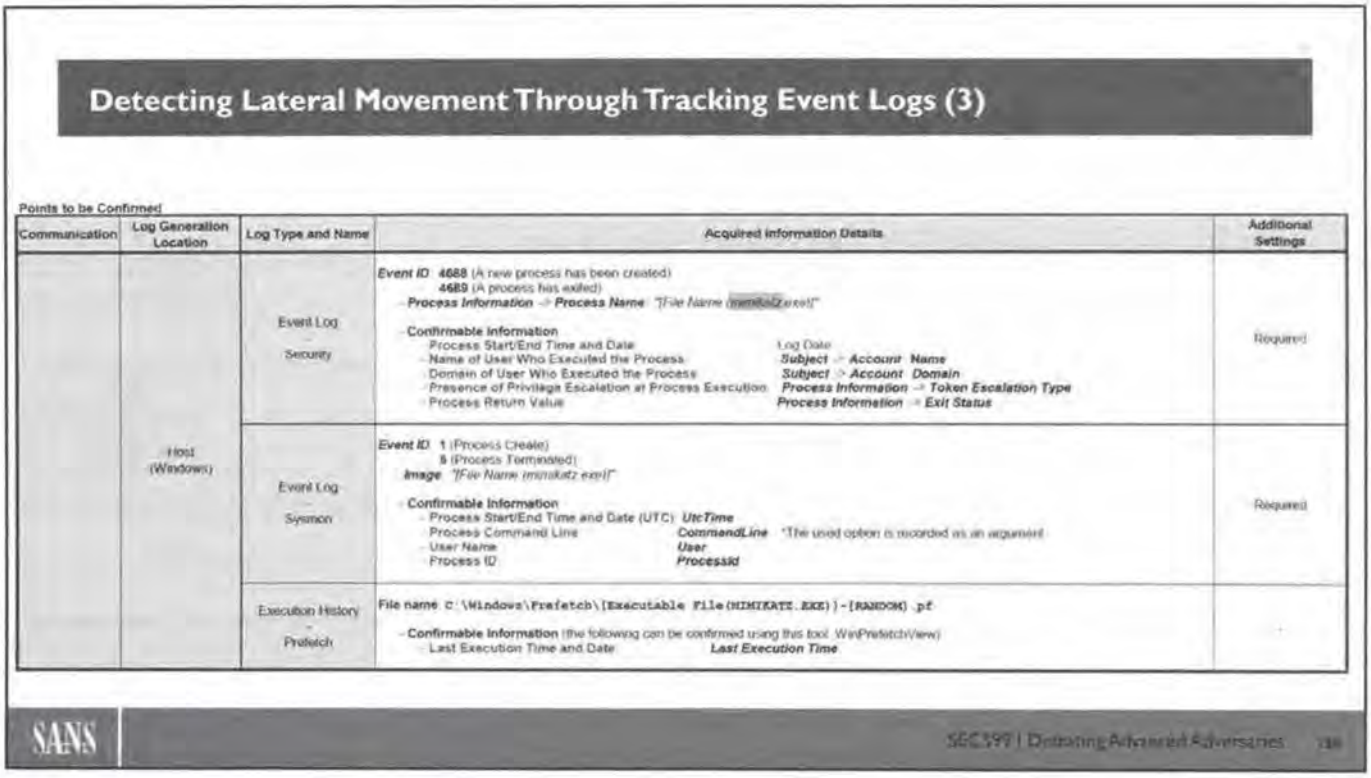
Basic Information

Tool	Tool Name	mimikatz > sekurlsa::logonpasswords mimikatz > lsadump::sam
	Category	Password and Hash Dump
	Tool Overview	Steals recorded authentication information
	Example of Presumed Tool Use During an Attack	This tool is executed to acquire passwords or escalate the privileges to the domain Administrator privileges.
Operating Condition	Authority	Administrator
	Targeted OS	Windows
	Domain	Not required
	Communication Protocol	-
	Service	-
Information Acquired from	Standard Settings	- Execution history (Prefetch)
	Additional Settings	- Execution history (Sysmon / audit policy)
Evidence That Can Be Confirmed When Execution is Successful		The successful execution of the tool cannot be determined from event logs or execution history.

Legend
- Acquirable Information
- **Event ID/Item Name**
- ***Field Name***
- "Field Value"

Detecting Lateral Movement Through Tracking Event Logs (2)

We will now take an example of an attack tool that is further explained in the JPCERT's paper: Mimikatz! In this first section, the paper will describe the typical purpose of the tool and the type of forensic artifacts that can be used to detect use of the tool.



Detecting Lateral Movement Through Tracking Event Logs (3)

Below the initial table (as described in the previous slide), a more detailed section is included, where it is illustrated what event IDs are required to successfully detect use of the tools. In this example, you will notice the following event IDs are being used:

Windows event logs

- 4688: A new process has been created
- 4689: A process has exited

Sysmon

- 1: Process create
- 5: Process Terminated

Furthermore, it also uses the name of the tool (Mimikatz)… This is a rather basic control, as adversaries could perfectly rename the tool. We could however also monitor for use of the typical command line arguments passed to Mimikatz ("sekurlsa", …)

Be Careful... Killing Your Windows Visibility Softly with Invoke-Phant0m

As a final remark, we'd like to point out that Windows event logs are an excellent means of understanding what is happening in our environment!

After successful configuration of a centralized monitoring setup, it's however important to monitor whether or not your event IDs are still arriving as expected...

To illustrate this issue, consider the Invoke-Phantom PowerShell script by penetration tester Halil Dalabasmaz:

"This script walks thread stacks of Event Log Service process (specific svchost.exe) and identify Event Log Threads to kill Event Log Service Threads. So the system will not be able to collect logs and at the same time the Event Log Service will appear to be running."

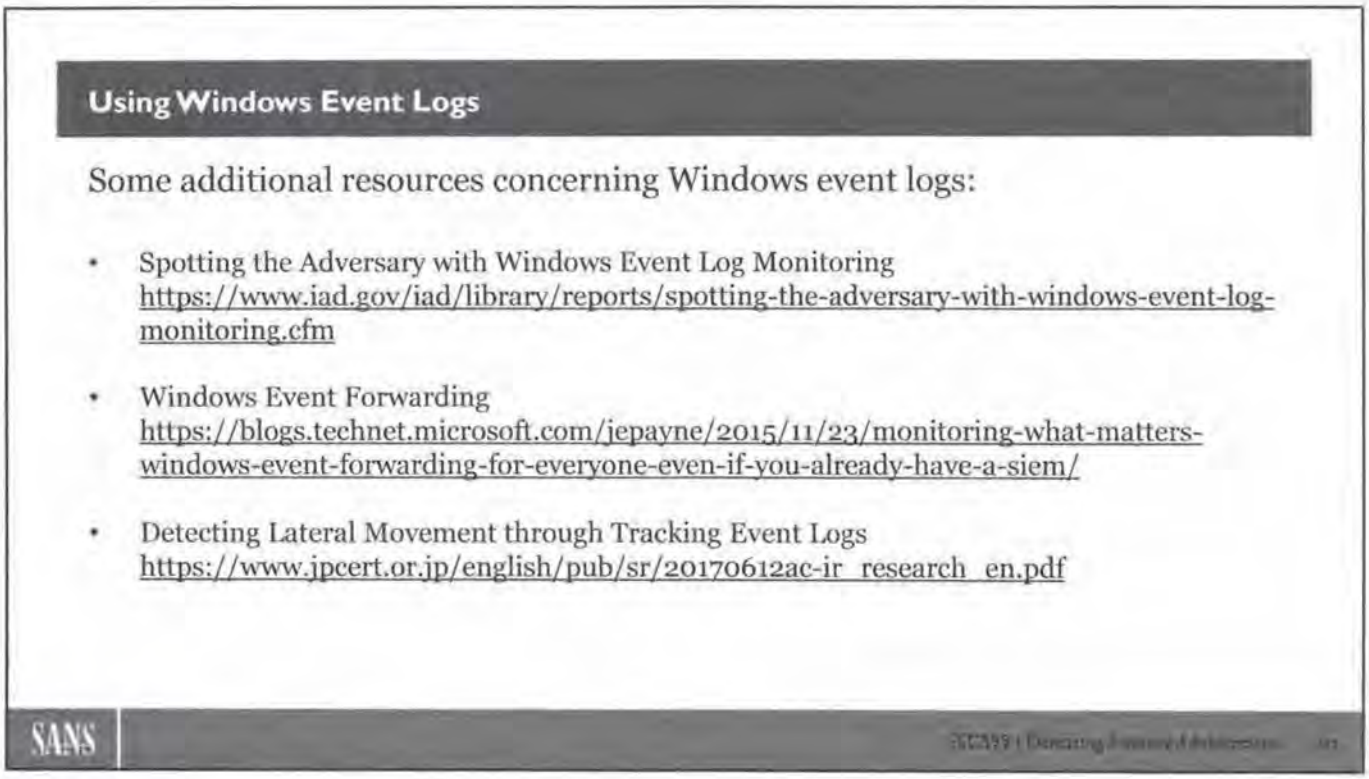
SANS

Be Careful... Killing Your Windows Visibility Softly with Invoke-Phant0m
As a final remark, we'd like to point out that Windows event logs are an excellent means of understanding what is happening in our environment!

After successful configuration of a centralized monitoring setup, it's important to monitor whether or not your event IDs are still arriving as expected. To illustrate this issue, consider the Invoke-Phant0m PowerShell script by penetration tester Halil Dalabasmaz:

"This script walks thread stacks of Event Log Service process (specific svchost.exe) and identify Event Log Threads to kill Event Log Service Threads. So the system will not be able to collect logs and at the same time the Event Log Service will appear to be running."

You can find the script on Halil's github page: https://github.com/hlldz/Invoke-Phant0m.



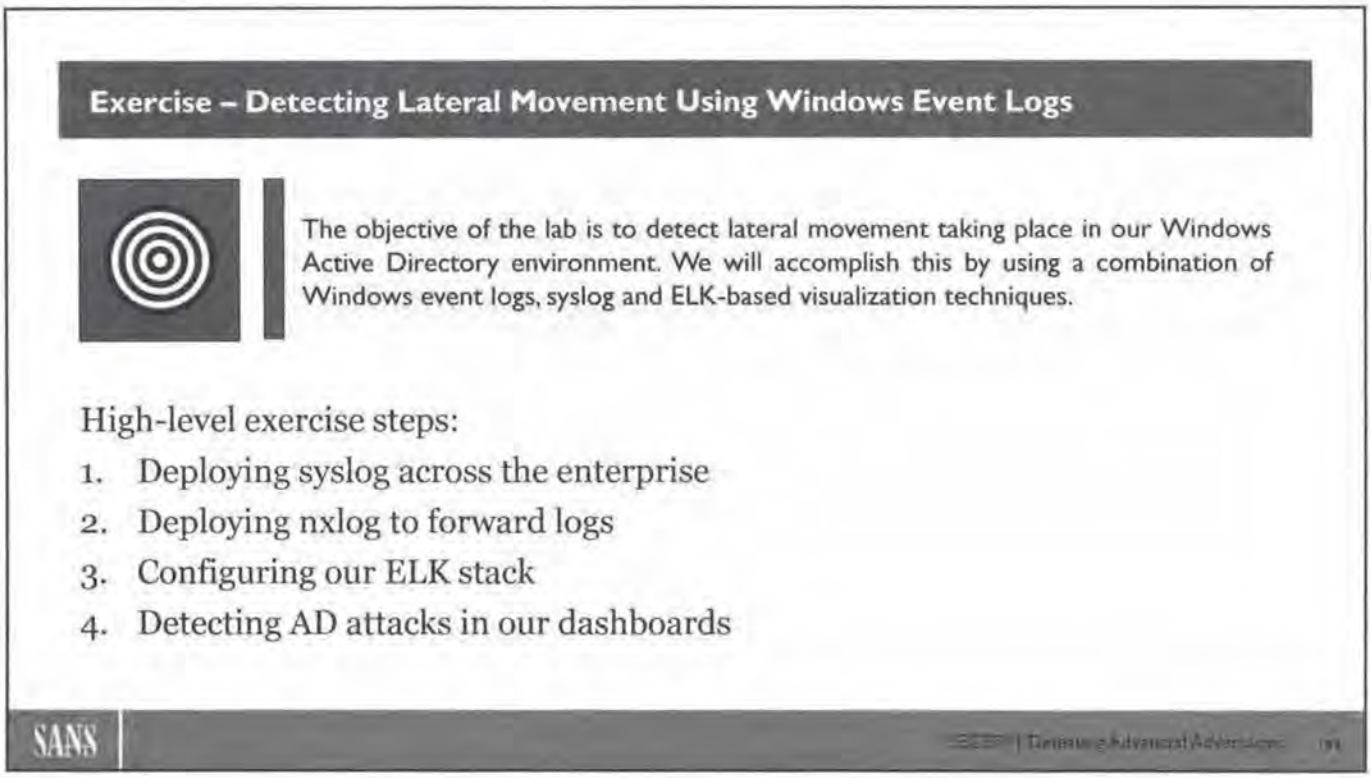
Using Windows Event Logs

Spotting the Adversary with Windows Event Log Monitoring
https://www.iad.gov/iad/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm

Windows Event Forwarding
https://blogs.technet.microsoft.com/jepayne/2015/11/23/monitoring-what-matters-windows-event-forwarding-for-everyone-even-if-you-already-have-a-siem/

Detecting Lateral Movement through Tracking Event Logs
https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf

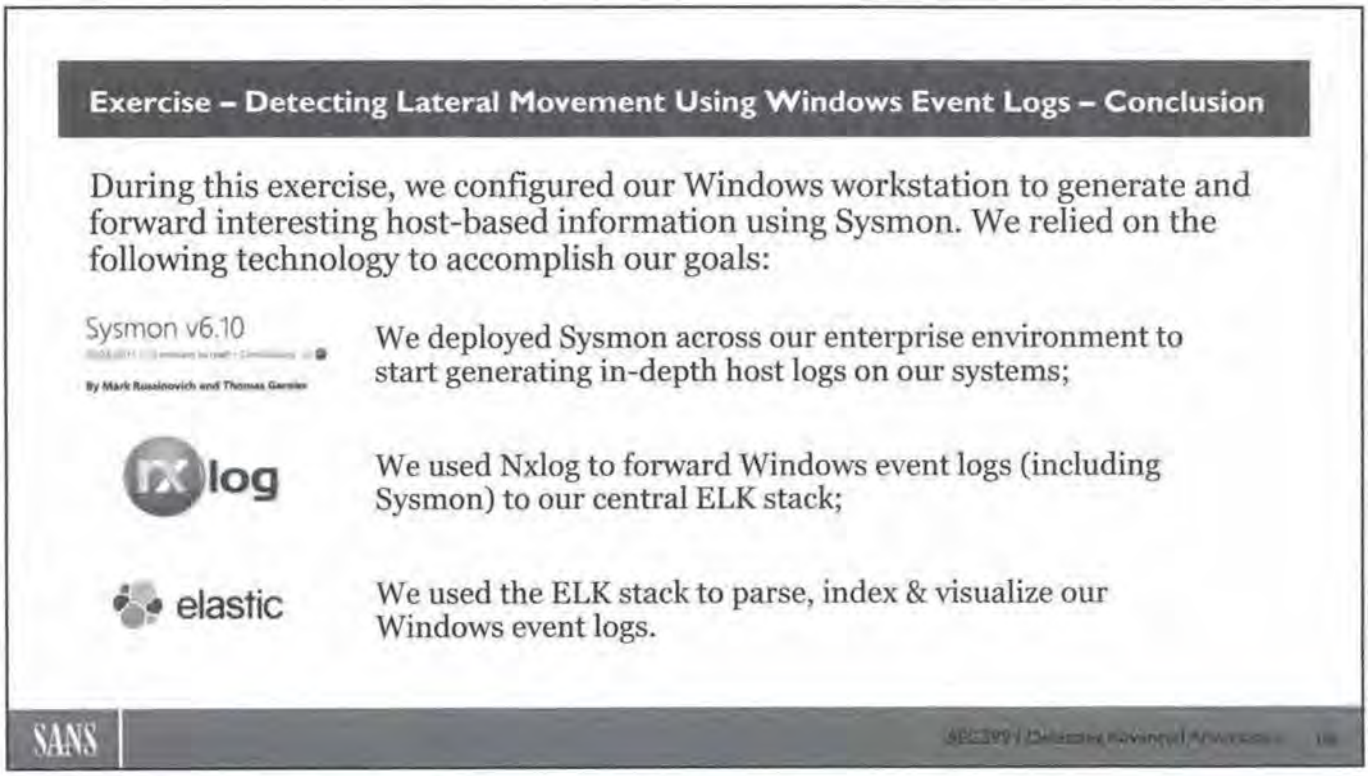
This page intentionally left blank.



Exercise – Detecting Lateral Movement Using Windows Event Logs
The objective of the lab is to detect lateral movement taking place in our Windows Active Directory environment. We will accomplish this by using a combination of Windows event logs, syslog and ELK-based visualization techniques.

Throughout the exercise, we will complete the following steps:
1. Reviewing the syslog configuration
2. Deploying syslog across the enterprise
3. Forwarding event logs to our ELK stack
4. Dashboarding Windows event logs in Kibana

For additional guidance & details on the lab, please refer to the LODS workbook.



Exercise – Detecting Lateral Movement Using Windows Event Logs – Conclusions
During this exercise, we configured our Windows workstation to generate and forward interesting host-based information using Sysmon. We relied on the following technology to accomplish our goals:

- We deployed Sysmon across our enterprise environment to start generating in-depth host logs on our systems;
- We used Nxlog to forward Windows event logs (including Sysmon) to our central ELK stack;
- We used the ELK stack to parse, index & visualize our Windows event logs.

Course Roadmap

- Day 1: Knowing the adversary, knowing yourself
- Day 2: Averting Payload Delivery
- Day 3: Preventing Exploitation
- **Day 4: Avoiding Installation, foiling Command & Control & thwarting lateral movement**
- Day 5: Exfiltration, Cyber Deception & Incident Response
- Day 6: APT Defender Capstone

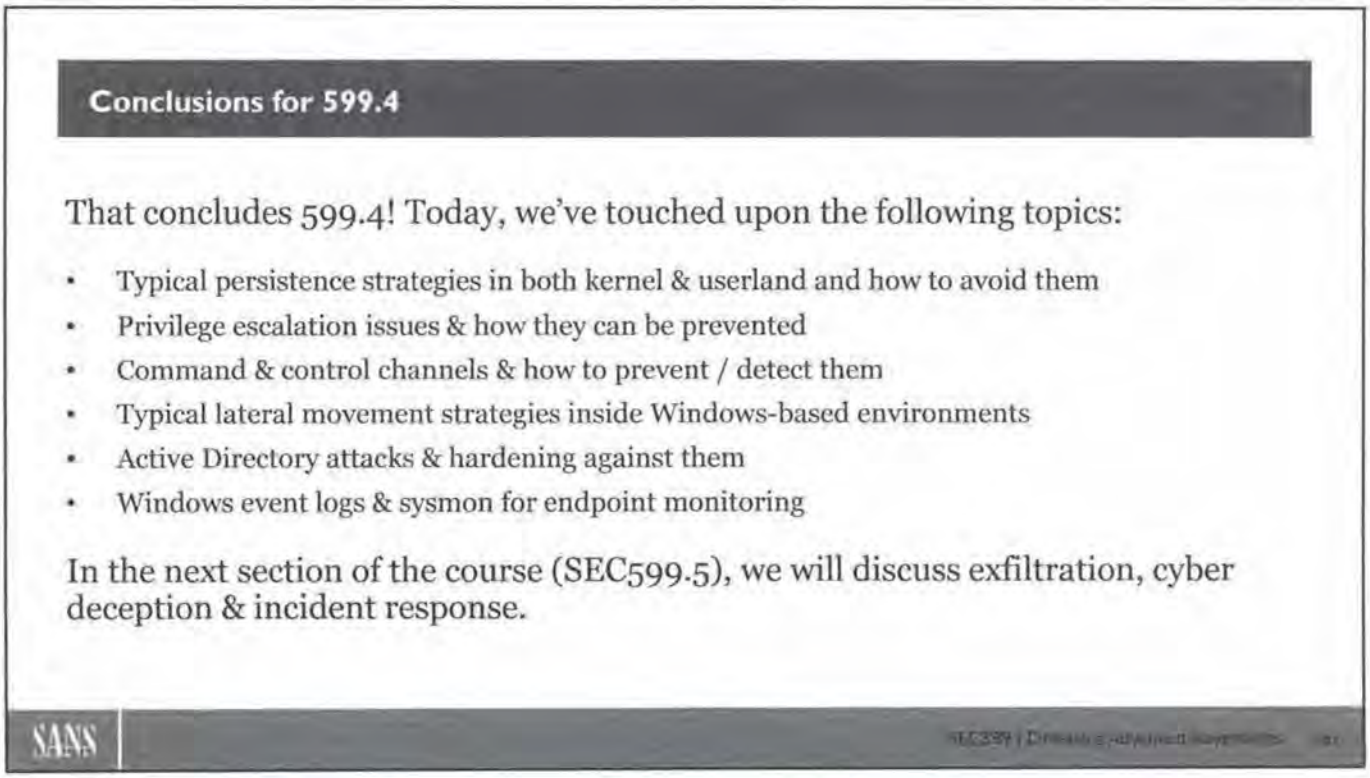
SEC599.4

Avoiding Installation
- Typical Persistence Strategies
- Exercise: Catching Persistence
- Principle of Least Privilege & User Access Control
- Exercise: Local Windows Privilege Escalation Techniques

Foiling Command & Control
- Network Monitoring Considerations (Netflow, IDS, ...)
- Detecting Command & Control Channels
- Exercise: Suricata to Detect Network Anomalies

Thwarting lateral movement
- Introducing Common Lateral Movement Strategies
- Active Directory Architecture & Attacks
- Active Directory Hardening & Segmentation
- Exercise: Hardening Windows to Stop Lateral Movement
- Detecting Lateral Movement Using Windows Event Logs
- Exercise: Configuring & Forwarding Windows Event Logs

This page intentionally left blank.



Conclusions for 599.4

So far 599.4! Today, we focused on persistence, privilege escalation, command & control, and lateral movement. Amongst others, we touched upon the following topics:

- Typical persistence strategies in both kernel & userland and how to avoid them
- Privilege escalation issues & how they can be prevented
- Command & control channels & how to prevent / detect them
- Typical lateral movement strategies inside Windows-based environments
- Active Directory attacks & hardening against them
- Windows event logs & sysmon for endpoint monitoring

In the next section of the course (SEC599.5), we will discuss exfiltration, cyber deception & incident response.

Course Resources and Contact Information

AUTHOR CONTACT
Erik Van Buggenhout
evanbuggenhout@nviso.be
Stephen Sims
ssims@sans.org

SANS INSTITUTE
8120 Woodmont Ave., Suite 310
Bethesda, MD 20814
301.654.SANS (7267)

CYBER DEFENSE CONTACT
Stephen Sims
ssims@sans.org

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

This page intentionally left blank.