APPLIED
INCIDENT
RESPONSE

Analyst Reference

Lateral Movement Analysis

Version 20200104

Introduction

This document is designed to help network defenders understand some of the tools and tactics used by attackers in order to inform security and threat hunting activities. Modern attackers frequently use client-side attacks to achieve an initial foothold within a network, but once that foothold is gained, they often use existing tools and accounts to move laterally, or pivot, throughout the victim network. By understanding the attackers' methodology as well as the evidence that such attacks leave behind, defenders will be better suited to seek out attacker activity and recognize evidence of such attacks when it is encountered.

This document is broken up into sections based on attacker methodologies and tools. Not all of these techniques will work in all environments, and savvy attackers will make note of the technologies already in use within a victim network and chose attack methods that will be most likely to blend in with normal network activity to evade detection. Within each of these categories, we outline methods used by attackers as well as highlight indicators that can be used to detect such activity within your network.

Table of Contents

Credential Theft

Attack Methodology

Client-side attacks through phishing, rogue Wi-Fi access points, USB drops, and even physical attacks on unattended devices have become a common attack vector. As IT teams have improved their ability to secure network perimeters, the use of external attacks to exploit vulnerabilities has grown increasingly difficult for attackers. As a result, attacks routinely target clients through phishing emails or social media posts designed to entice a user to click on a link, executable or macro to run malicious code. In addition to common phishing techniques, advanced threat actors proactively target hotel Wi-Fi infrastructure (such as hotels and cafes) to perform man-in-the-middle attacks and introduce malicious software onto unsuspecting business traveler's phones and laptops, or steal credentials as they pass through the rogue Wi-Fi access points to later gain access to corporate network resources. BYOD programs that allow users unrestricted use of a personal device that is then allowed to connect to a secure, internal network have only increased the effectiveness of client-side attacks.

Once a client system is compromised, attackers will set to work to collect credentials from that machine. Using tools such as Metasploit's Meterpreter, Mimikatz, Mimikittenz, Windows Credential Editor, key loggers, sniffers and others, attackers are able to capture credentials such as user names and passwords from client systems or off the wire when the client returns to the corporate network.

Keep in mind that credentials do not need to be full username and cleartext password pairs. Attackers can steal hashed representations of passwords, load them into memory, and allow Windows passthrough authentication to handle authentication to other systems as an arbitrary user through pass-the-hash attacks. Even with multifactor authentication in use, after a user has authenticated an attacker can steal authenticated Kerberos service tickets to impersonate the user and access remote systems. These service tickets are issued with a 10-hour validity by default, giving an attacker ample time to leverage completed authentication to access networked resources. Attackers who compromise your domain controller may be able to impersonate the Kerberos service itself, granting themselves a so called "Golden Ticket" capable of issuing credentials for any resource in the Kerberos realm.

While many of the attacks mentioned above focus on Windows, don't forget that other operating systems are not immune to similar attacks. Whether using RDP or ssh, an attacker with valid credentials is able to wreak havoc within your internal network.

Defensive Considerations

Client-side attacks are rampant, so focusing on defending against them is a good place to start. While no solution provides 100% effectiveness, we should strive to make our clients as hard to breach as possible, and minimize damage in the event of a compromise. Using endpoint security products to detect and defeat attacks is a good place to start, but don't place too much faith in any one solution. No matter how "Next generation" your security product may be, attackers are already working on ways to bypass it. Layered security best practices still apply, so combining antivirus, white listing, and heuristic detection with strong auditing policies is the best approach. Also, be sure to add good patch management and network security controls to the environment to help protect your end points. If you allow endpoints to physically leave your environment, recognize the risks of them returning with malicious software. BYOD programs should treat each employee-owned device as likely hostile, and

only allow access to restricted segments of the network, and those segments should be heavily monitored with threat hunting technologies.

Give users accounts with the least privilege necessary to perform tasks. This includes any administrator accounts you may use within your environment. Don't use your Enterprise Administrator account for common admin functions. Instead create administrative accounts with specific organizational units or tasks in mind and use the account with the least privilege needed to accomplish each task. Remember, if the box you log onto has been compromised, the attacker can scrape your administrative credentials out of memory, giving them access to anything that your account can access. Consider the use of secure administrative workstations for privileged account access rather than using your daily-use workstation for administrative tasks. Attackers will target administrators' computers knowing that key stroke loggers and RAM scraping tools on those systems may lead to credentials for elevated accounts. By using a hardened, restricted-access system with additional inbound connection controls to perform your administrative logons, you make it that much harder for an attacker to obtain the privileged credentials he is seeking.

Aside from protecting your credentials, be sure that you also store any hashed representations of passwords securely. Don't allow your web apps to use weak password hashing algorithms such as unsalted MD5 to store credentials. Highly publicized losses of millions of user credentials are all too common these days. Don't be that company.

If credentials are lost, be sure you are monitoring the inside of your network to detect their use. Workstations usually logon to a handful of servers in your environment. When you see a workstation suddenly authenticating to other workstations or to servers that it has never accessed before, that should be a clue. Too many organizations simply are not monitoring internal user activity, and attackers are heavily leveraging this blind spot.

Auditing Account Logon and Logon Events

While our book, Mastering Windows Network Forensics and Investigations, Second Edition, provides a detailed account of how to track user activity throughout a Windows environment, this section will cover the highlights to get you started. Account Logon is the Microsoft term for authentication. Logon is the term used to refer to an account gaining access to a resource. Both Account Logon and Logon events will be recorded in the Security event log. Authentication (account logon) of domain accounts is performed by a domain controller within a Windows network. Local accounts (those that exist within a local SAM file rather than as a part of Active Directory) are authenticated by the local system where they exist. Account logon events will be logged by the system that performs the authentication. Auditing of Account Logon and Logon events is easily set by Group Policy. While Microsoft continues to enable more logging by default as new versions of Windows are released, administrators should review their audit policies on a regular basis to ensure that all systems are generating adequate logs. The ability to store event logs on remote systems (either using the native Microsoft remote logging features or third-party SIEM or other tools) helps safeguard logs from alteration or destruction.

The domain controllers in your network should therefore be able to provide a fairly centralized accounting of which accounts where authenticated throughout the domain. Remember that to get a full picture, you will need to query each of your DCs since the one that performs the authentication creates the associated event log. On the other hand, if you find that member servers or workstations are performing their own authentication, that is a good indicator that local user accounts are being used. As this is not normally done in most environments, account logon events on non-domain controllers can often be an indicator of compromise. By contrast, logon event logs are generated by the system that is being accessed, so logon events will be generated by systems all across the network, providing another reason to aggregate logs to a central location.

Event IDs that are of particular interest on your domain controllers include:

- 4768 – The issuance of a Ticket Granting Ticket (TGT) shows that a particular user account was authenticated by the domain controller.
- 4769 – A service ticket was issued to a particular user account for a specified resource. This event will show the source IP of the system that made the request, the user account used, and the service to be accessed. These events provide a useful source of evidence as they track authenticated user access across the network.
- 4776 – While less common in a domain environment, NTLM may still be used for authentication. Additionally, many attack tools downgrade authentication attempts to NTLM when authenticating. While these types of authentication do frequently occur with legitimate traffic, such as some authentication requests originating by IP address rather than computer name, their presence may also indicate a non-standard tool being used to authenticate.

On member servers and workstations, Event IDs of note include:

- 4624 – A logon to a system has occurred. Type 2 indicates an interactive (local) logon, while a Type 3 indicates a remote or network logon.
- 4625 - A failed logon attempt.
- 4672 – This Event ID is recorded when certain privileges associated with elevated or administrator access are granted to a logon. As with all logon events, the event log will be generated by the system being accessed.
- 4776 – An NTLM-based authentication has occurred. When found on a non-domain controller this indicates the use of a local user account. Since most domains are designed to use domain rather than local user accounts, the presence of this Event ID on member servers or client workstations is frequently suspicious.
- 4634/4647 – User logoff is recorded by Event ID 4634 or Event ID 4647. The lack of an event showing a logoff should not be considered overly suspicious, as Windows is inconsistent in logging event 4634 in many cases.

If a new account is created, Event ID 4720 will be created on a domain controller for a domain account or on the local system for a local account.

In addition to account logon and logon events, analysts should look for evidence of access to network shares. Attackers frequently leverage valid credentials to remotely access data through

user created or administrative shares. On the system being accessed Event ID 5140, a network share object was accessed, will show when a shared folder or other shared object is accessed. The event entry provides the account name and source address of the account that accessed the object. The system initiating the access may show evidence of the connections in the registry key NTUSER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2.

Remote Desktop Protocol

Attack Methodologies

Despite being a subset of credential theft, there are several additional considerations to be considered when credentials are leveraged to gain access to a system over Remote Desktop Protocol (RDP). If RDP is routinely used by your administrators, help desk, or others it provides an attractive attack vector for adversaries who are trying to blend in with standard network activity. Once on a client system, the attacker can simply leverage the built in Microsoft tools to allow for remote access to other systems using RDP and valid credentials.

An attacker can use the Microsoft Remote Desktop Connection (*mstsc.exe*) tool to access a victim system. While you hopefully are not exposing the default port 3389 to the Internet, port forwarding set up as a pivot on other, Internet accessible systems does make this possible from either external or internal hosts. Depending on the bandwidth available, this GUI-based approach may be less than ideal to the attacker, but if this is a method commonly used in your environment attackers are likely to use it to blend in with normal network traffic.

Defensive Options

Remember that RDP leverages standard Microsoft authentication to control access to resources, therefore the logs associated with account logon and logon events outlined above will apply to RDP connections. Aside from account logon events, other event log entries that may be of use in detecting and tracking malicious use of RDP within your environment include:

- 4624 – The logon event will show either a Type 10 or Type 3 when RDP is used, depending on the versions of Windows used and their specific configuration.
- 4778 – This event is logged when a session is reconnected to a Windows station. This can occur locally when the user context is switched via fast user switching. It can also occur when a session is reconnected over RDP. To differentiate between RDP versus local session switching, look at the Session Name field within the event description. If local, the field will contain Console, and if remote, it will begin with RDP. For RDP sessions, the remote host information will be in the Network Information section of the event description.
- 4779 – This event is logged when a session is disconnected. This can occur locally when the user context is switched via fast user switching. It can also occur when a session is reconnected over RDP. A full logoff from an RDP session is logged with Event ID 4637 or 4647 as mentioned earlier. To differentiate between RDP versus local session switching, look at the Session Name field within the event description. If local, the field will contain Console, and if remote, it will begin with RDP. For RDP sessions, the remote host information will be in the Network Information section of the event description.

On the machine receiving the connection, additional RDP specific logs may be found. The %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-TerminalServices-LocalSessionManager%4Operational log may contain the IP address and logon user name of the originating computer in Event IDs 21, 22 or 25. Event ID 41 may also contain the logon user name. The %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-TerminalServices-RemoteConnectionManager%4Operational log may record Event ID 1149 that contains the initiating IP address and logon user name. Finally, the %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-

RemoteDesktopServices-RdpCoreTS%4Operational log may record Event ID 131 containing the initiating IP address and logon user name.

On the system that initiates an RDP session, additional evidence may be available. This can be important when a compromised host is used as a jump off point for the attacker to connect to other systems. The %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-TerminalServices-RDPClient%4Operational log records the systems to which the host-initiated connections using Event IDs 1024 and 1102. You may also find evidence of connections on the initiating system in the NTUSER.DAT\Software\Microsoft\Terminal Server Client\Server registry key.

Additional information about RDP Sessions can be found in the %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-TerminalServices-LocalSessionManager%4Operational log file. Event ID 21 in this log shows session logon events, both local and remote, including the IP from which the connection was made if remote. Event ID 24 in this log shows session disconnection, including the IP from which the connection was made if remote. For local logons, the Source Network Address field of the event description will read LOCAL rather than provide the remote IP.

You can also configure network security monitoring solutions to watch for activity on port 3389, the default RDP port. For example, Zeek provides logging of RDP activity observed on the wire.

at/schtasks

Attack Methodologies

Malicious attackers can leverage the built-in Windows *at* and *schtasks* commands to both expand their influence and maintain persistence within a victim environment. The *at* command, while deprecated in the latest versions of Windows, is still in use on older Windows versions. The command allows for a process to be executed on regular intervals in the future on either a local or remote machine. The syntax is:

at [\\targetIP] [HH:MM][A|P] [command]

Where *targetIP* that specifies a remote system to be named, the time with AM or PM designated, and the command to be executed are taken as options.

Similarly, newer Windows systems support the *schtasks* command, albeit it with a slightly more involved syntax:

schtasks /create /tn [taskname] /s [targetIP] /u [user] /p [password] /sc [frequency] /st [starttime] /sd [startdate] /tr [command]

Once again, this command allows for the execution of a process on a local or remote system at a designated time. Additionally, if run with administrator credentials, the */ru SYSTEM* option allows for the specific program to be executed with System-level permissions. Attackers with valid credentials are therefore able to schedule commands to run on systems as a means of exfiltrating data, maintaining persistent access, expanding control, or other tasks as they see fit.

Defensive Considerations

Administrators should routinely use the *schtasks* command to check for processes that are scheduled to run. The *schtasks* command will list items scheduled both with *schtasks* and with *at* whereas the *at* command will not show jobs scheduled with *schtasks*. You can output the list of jobs to a comma-separated value file with the following syntax:

Schtasks /query /fo csv > scheduled_tasks.csv

You can optionally include the */v* switch to enable verbose output if you need additional detail.

PowerShell can be leveraged to query remote systems in an automated fashion. The PowerShell cmdlet *Get-ScheduledTask* will show the path, name and state of scheduled tasks when run.

On the system where the task is scheduled, additional details can be found in the %SystemRoot%\System32\Tasks folder. Each task created with schtasks creates an XML file with the same name as the task in this location. Within these XML files a number of fields are useful. Under the "RegistrationInfo" section, the "Author" field shows the account used to schedule the task and the "Date" field shows the local system date and time when the task was registered. In the "Principals" section, the "UserID" field shows the user context under which the task will execute. The Triggers section provides details on when the task will run and the Exec field under the Actions section details what will be run.

See our Event Log Analyst Reference for additional details on logs that may be generated by use of scheduled tasks. Also, any use of authenticated credentials to schedule tasks on remote systems will leave the associated account logon and logon events as previously discussed.

SC

Attack Methodologies

The Service Controller command, *sc*, is able to create, stop, and start services. Services are processes that run outside of the context of a logged-on user, allowing them to start automatically at system boot time. By running a process as a service, the malicious actor can ensure persistence of the service on the system, including allowing for automating restarting or other actions should the process stop execution. Once again, the *sc* command allows for these actions to be taken on the local system or on remote systems with appropriate credentials. The syntax to establish an initial, authenticated connection to a remote system is:

net use *[targetIP] [password] /u:[Admin_User*

Where *targetIP* identifies the remote system and the *Admin_User* is the username of an account with administrator privileges on the target system. To create a service on the remote system, the syntax is:

sc *[targetIP] create [svcname] binpath= [executable]*

Note that the executable must be packaged in a way that provides for the necessary service control options, including responding back to the OS when it has successfully started. Without this acknowledgement, Windows will kill the service after about 30 seconds. Any arbitrary executable can be packaged for use as a service through the use of the free *ServifyThis* tool made available on GitHub by InGuardians (https://github.com/inguardians/ServifyThis). The service can then be started with:

sc *[targetIP] start [svcname]*

Defensive Considerations

As with the *schtasks* command, administrators should establish a baseline of what services are running on each of their systems. This can be accomplished using *WMIC* or PowerShell either as a local script that runs at set intervals, or as a centralized script that queries remote systems and stores this type of data in a central location. Having historical records of ports, processes, services, etc. that are in use on each system (or at least on critical systems) gathered by such scripts provides a great investigative reference once an incident is declared or when threat hunting.

Event ID 7045 records the creation of a new service on the system, including the path to the executable service file name. This event, recorded in the System event log, can be useful in identifying the creation of malicious services on victim systems.

When a service is created, the path to its associated executable is stored in the registry. Examining the entries in the registry under HKLM\SYSTEM\CurrentControlSet\Services for unusual entries can help locate malicious services. The ImagePath key will specify the location of the associated executable on disk for each service.

Any use of authenticated credentials to modify services on remote systems will also leave the associated account logon and logon events.

psexec

Attack Methodologies

psexec is an administration tool that leverages SMB to remotely execute commands on other systems. While not a native Windows binary, it is provided by Sysinternals, which is owned by Microsoft. As a result, many administrators use it within their environment, so finding it running inside a network may not be unusual. The command allows remote execution of programs over an encrypted network connection when provided with the necessary credentials. If the executable to be run is not already on the target system, it can be copied by *psexec* to the target and then executed. The Sysinternals version of the tools runs at the command line with the following syntax:

psexec *[targetIP] [-d] [-e] [-u user] [–p password] [command]*

Where the *targetIP* is the remote system and the *command* is any executable on the system. A common technique is to use cmd.exe as the remote command to grant a remote shell to the attacker. The *-c* switch can be added to copy the executable to the target first if the desired executable is not already on the target or is not in a location found in the system path on the target. The *-d* switch is used to execute the specified command in a non-interactive manner and detach without waiting for the created process to terminate. The *-e* switch can optionally be used to disable creation of a user profile on the remote system. The *-s* switch can be used to run the remote process in the context of the System account.

When used for legitimate system administration tasks, note that the use of the *-u* switch causes the remote system to treat the logon as interactive, which results in caching of the logon credential in RAM. If the remote system is already compromised, this can expose the credential to an attacker.

The *Metasploit* team has also added a version of *psexec* as an exploit module in the *Metasploit Framework*. The *psexec* module requires a valid credential to access the remote system, but can accept either a cleartext password or a password hash representation to facilitate pass-the-hash attacks. In the absence of a valid credential, the module will attempt to logon as Guest.

Defensive Considerations

The *psexec Metasploit* exploit module uses valid administrator credentials to copy an executable to the target system, create a service to load the desired payload, delete the service it created, and then delete the uploaded executable. By default, both the executable and the service are given a random string of characters as a name; however, arbitrary names can be specified by the attacker. The creation of the service generates an Event ID 7045 in the System event log, complete with the name of the service created (Service Name field) and the executable that was used to create it (Service File Name field). The executable may be uploaded with a random or explicitly provided name, or the Service File Name may be PowerShell run with a long, Base64-encoded command. If enabled, Event ID 4697 will also be logged in the Security event log recording the service being installed on the system.

By default, the Sysinternals version of *psexec* will also install itself as a service with a service name of PSEXESVC and an associated executable of psexesvc.exe written to disk, making it easy to spot in the event 7045 System event log records (and possibly Event ID 4697 in the Security event log if enabled as described in Chapter 8). However, the name of the service and its associated executable can be changed to any arbitrary name using the -r switch when *psexec* is run. Unlike the Metasploit version, the Sysinternals version of *psexec* does not automatically delete the service upon completion. By default,

the Sysinternals *psexec* will also cause a user profile to be created on the remote system if one does not already exist for the associated user credential. This can be avoided if the attacker uses the -e switch when initiating the connection, but since the attacker may omit that switch, checking for the presence of unusual user profiles can be a useful indicator of unauthorized activity. When the session ends, you may see an event 7036 in the System event log showing the PSEXESVC service entering a stopped state.

On the system initiating the connection, when the -u switch is used, an event 4648 is recorded, showing the account initiating the use of the credential in the Subject section, the credential provided with the *-u* switch in the Account Whose Credentials Were Used section, and the remote system targeted in the Target Server section. If the Sysinternals version of *psexec* was used, you may find evidence of it being used in the registry. In addition to the standard forensic artifacts showing program execution, this utility also writes to a registry key in NTUSER.DAT\Software\Sysinternals\PsExec where it sets the EulaAccepted value to 1. This key is named PsExec even if the attacker named the tool something else in an attempt to conceal its execution.

Since valid credentials are used, the account logon and logon events discussed previously also apply to this attack vector. If the attacker uses the currently logged-on user's credentials, Windows will record the access on the remote system with event 4624, Type 3 (Network logon). However, if the attacker explicitly provides a different credential to *psexec* with the -u switch, Windows treats this as an interactive logon (event 4624, Type 2) on the remote system and will also log an event 4648 showing the PSEXESVC.exe process using an explicit credential for the user specified in the *-u* switch.

Other tools implementing variations on the PsExec idea exist, such as CSExec, PAExec, and RemCom. You can look for Event ID 7045 (new service was installed) and Event ID 7036 (service was started/stopped) in the System event log and Event ID 4697 in the Security event log to try to identify the malicious services associated with these types of attacks.

Windows Management Instrumentation

Attack Methodologies

Windows Management Instrumentation (WMI) is a Microsoft-provided platform for simplifying administrative tasks, and we looked at ways in which network defenders can leverage WMI in Chapter 4. Windows Management Instrumentation Command-line utility (WMIC) is a command-line tool that can execute WMI commands on local or remote systems. WMIC uses Distributed Component Object Model (DCOM) to connect to remote systems to execute commands, meaning that in environments with WinRM and/or PowerShell remoting disabled, WMIC provides an attractive option to attackers. PowerShell is likewise able to access and manipulate systems using WMI. The older Get-WmiObject cmdlet uses DCOM to access remote systems and the newer Get-CimInstance cmdlet uses PowerShell remoting, providing flexibility for accessing WMI on remote systems for both attackers and defenders.

The scope of WMI is truly massive, allowing attackers to perform a variety of actions on remote systems including listing files, enabling/unlocking user accounts, gathering system information, starting and stopping services, creating or stopping processes, and many other tasks. Once again, use of WMI requires that authenticated access be made to the target system, so account logon and logon events may provide a useful indicator of suspicious activity. WMIC does not encrypt its network traffic when run against a remote system, making network security monitoring a viable approach to detect malicious use of WMIC.

In addition to running single commands, WMI can be used as a persistence mechanism using WMI subscriptions. WMI subscriptions allow for an event to be triggered when a specific condition is met. This is a common persistence technique used by PowerShell-based, post-exploitation frameworks. The action that will occur is referred to as an event consumer. The event that triggers the action is called an event filter. An event filter is connected to its associated event consumer through a filter to consumer binding. When the event described by the filter is triggered (its condition is true), the filter sends certain events (those that matched its filter) to the event consumer to take its defined action on those events. WMI subscriptions can be created using PowerShell or through Managed Object Format (MOF) files using mofcomp.exe. Once created, a WMI subscription is stored in the WMI database, located in the %SystemRoot%\wbem\Repository folder.

Defensive Considerations

Sysmon can be configured to monitor WMIEventFilter activity (event 19), WMIEventConsumer activity (event 20), and WMIEventConsumerToFilter activity (event 21). In addition to Sysmon logs, the %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-WMI-Activity%4Operational log also records information about use of WMI. Event IDs 5860 and 5861 record details of event consumers. Searching through these for encoded PowerShell commands or other unusual entries may help identify malicious use of WMI. Tools like Autoruns (discussed in Chapter 3) can also help identify the use of WMI subscriptions as a persistence mechanism.

You can query a live system for information regarding WMI subscriptions using PowerShell. The Kansa framework (discussed in Chapter 4) does this very effectively, providing the capability to stack results from different systems for comparison. In addition, PowerShell scripts can help generate notifications of WMI activity that can be fed into a SIEM solution for enhanced detection. Matt Graeber has written some scripts that serve as a good starting point for such endeavors, and these were expanded upon by Timothy Parisi and Evan Pena. You can read more about these options at here.

If analyzing an offline image, WMI subscriptions are stored in the WMI database, located in the %SystemRoot%\wbem\Repository folder, which can be parsed using the open-source python-cim tool.

WMIC does not encrypt its network traffic when run against a remote system, so network detection is a good approach to monitor your network for malicious use of WMI

Once again, use of *WMIC* requires that an authenticated access be made to the target system, so using audit logon and logon events to identify unusual system access is useful indicator for suspicious WMI activity. In addition, use of *WMIC* is not constrained to malicious actors. Defenders should leverage *WMIC* to help automate creation of system baselines, detection of specific indicators of compromise, and other security tasks to take advantage of the power that WMI provides.

WinRM

Attack Methodologies

Windows Remote Management (WinRM) allows for commands to be sent to remote Windows computers over HTTP or HTTPS by leveraging the Web Services for Management protocol. WinRM runs as a service under the Network Service account, and as native Microsoft components, use of these tools will bypass many whitelisting solutions providing another attractive option for attackers. Use of the Windows Remote Services command, *winrs*, allows for execution of arbitrary commands on remote systems. A simple command shell can be returned with the following syntax:

winrs -r:http://target_host "cmd"

Where *target_host* is the remote system on which the *cmd.exe* should execute. An interactive shell is returned to the user running this command.

Defensive Considerations

WinRM utilizes TCP port 5985 for HTTP traffic and TCP port 5986 for HTTPS by default. Configuring network security monitoring tools to look for unusual activity on these ports is therefore advisable. In addition, specific Event IDs can be of use when trying to identify malicious use of WinRM. These events will be logged in the %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-WinRM%4Operational log file.

When a connection is initiated using WinRM, Event ID 6 will be generated. This event will include the remote destination to which the connection was attempted. Therefore, the appearance of Event ID 6 on local workstations or other computers where administrative tasks are not frequently done may be suspicious.

Additionally, Event ID 91 will be logged on the system where the connection is received. This log will include the user field which shows the account used to authenticate the connection. Once again, the standard account logon and logon events can also be leveraged to help fill in additional gaps regarding the systems, accounts, and times involved in such activity.

PowerShell

Attack Methodologies

PowerShell is another example of a tool that provides a great deal of capability, but whether that capability is used for good or evil is entirely at the discretion of the user. PowerShell's Remoting feature makes it the ideal mechanism to move throughout a network. Any action that can be taken on a Windows system can be taken through PowerShell, without the need for additional malware to be installed. Empire (www.powershellempire.com) and similar projects leverage this fact into complete post-exploitation kits that allow an attacker to maintain almost unparalleled control over a victim network, all using PowerShell scripts. PowerShell has become a favorite attack mechanism for adversaries of all flavors. Although Empire is no longer officially supported, it is still available for download, and many similar projects continue to advance the use of PowerShell and the underlying .NET Framework for post-exploitation command and control. Covenant (https://github.com/cobbr/Covenant) and the Faction C2 Framework (www.factionc2.com) are examples of projects that continue to be actively developed.

In addition to PowerShell Remoting, many older PowerShell cmdlets support the -ComputerName parameter to execute the cmdlet on a remote system. These cmdlets generally use DCOM to accomplish this (though the mechanism may vary) rather than use PowerShell Remoting.

Defensive Considerations

PowerShell Remoting is enabled by default for members of the Administrators group and Remote Management Users group on Windows Server 2012 and later. Disabling PowerShell Remoting might discourage its use by attackers, but it also disables one of the most useful tools in the administrator's arsenal for daily administrative tasks as well as baselining and incident handling. Just as attackers have scripts like Empire to help do their jobs, so to do defenders have frameworks like Kansa to help do theirs. Kansa allows defenders to gather data from collections of systems, stack the results, and look for deviations from the norm. Kansa can be a powerful tool in preparing for and responding to incidents. An update to the Kansa framework, ARTHIR, is also an outstanding option for incident response.

Microsoft continues to increase the logs available surrounding PowerShell to help combat its nefarious use. Once again, these logging facilities must be enabled via Group Policy, specifically at Computer Configuration > Policies > Administrative Templates > Windows Components > Windows PowerShell. There are three basic categories of logging that may be available, depending on the version of Windows in question.

- Module Logging
  - Logs pipeline execution events;
  - Logs to event logs.
- Script Block Logging
  - Captures de-obfuscated commands sent to PowerShell;
  - Captures the commands only, not the resulting output;
  - Logs to event logs.
- Transcription
  - Captures PowerShell input and output;
  - Will not capture output of outside programs that are run, only PowerShell;

    - Logs to text files.

Once enabled, these logs can provide a wealth of information concerning the use of PowerShell on your systems. If you routinely run lots of PowerShell scripts, this can produce a large volume of data, so be sure to test and tune the audit facilities to strike a balance between visibility and load before deploying such changes in production.

PowerShell event log entries appear in different event logs. Inside of %SystemRoot%\System32\winevt\Logs\Microsoft-Windows-PowerShell%4Operational you will find two events of particular note:

- 4103
    - Shows pipeline execution from the module logging facility;
    - Includes the user context used to run the commands;
    - Hostname field will show "Console" if executed locally or will show if run from a remote system;
    - Can correlate account logon and logon events to determine further information about the source of a remote connection.
- 4104
    - Shows script block logging entries;
    - Logs full details of each block only on first use to conserve space;
    - Will show as a "Warning" level event if Microsoft deems the activity "Suspicious."

Additional entries are located in the %SystemRoot%\System32\winevt\Logs\Windows PowerShell log, as follows:

- Event 400
    - Indicates the start of command execution or session;
    - Hostname field shows if (local) console or remote session caused the execution.
- Event 800
    - Shows pipeline execution details;
    - UserID shows account used;
    - Hostname field shows if (local) console or remote session caused the execution;
    - Since many malicious scripts encode options with Base64, check the HostApplication field for options encoded with -enc switch.

You may also find a PowerShell history file per user, located at %HOMEPATH%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt. You can confirm the location on a system by running the Get-PSReadLineOption cmdlet and checking the HistorySavePath. The MaximumHistoryCount will show the number of lines that will be stored in the ConsoleHost_history.txt file before it starts overwriting older entries (the default is 4096). This can provide a great source of information regarding PowerShell use.

Remember that PowerShell remoting uses WinRM to establish connections to remote machines. As a result, the same detection methods used for WinRM also apply to PowerShell Remoting. Note that regardless of whether HTTP or HTTPS is used in the WinRM transfer, PowerShell encrypts all remoting commands with AES-256 after the initial authentication.

The Australian Signals Directorate has published an excellent guide to securing PowerShell, including configuration and use of PowerShell logging. The guide is freely available here. Remember that PowerShell Remoting requires authenticated access, so look for the associated Account Logon and Logon events as well.

Network defenders can help secure PowerShell in their environment by creating constrained endpoints with restricted PowerShell session configurations, which allows more granular control over which users can use PowerShell remoting and which cmdlets they can run (additional information can be found here). Similarly, the Just Enough Administration PowerShell security controls can be used to provide necessary access while minimizing the risk of abuse. See here for additional details. Consider applying host-based firewall rules to allow inbound PowerShell Remoting sessions only from trusted administration workstations rather than from all arbitrary systems. Finally, implement outbound restrictions to keep systems that don't need to initiate outbound PowerShell Remoting sessions from initiating outbound connections on TCP ports 5985 and 5986. Doing so can greatly reduce the malicious use of PowerShell for lateral movement while still allowing network administrators and incident handlers to leverage PowerShell in their daily operations.

General Defensive Measures

Preventive defensive strategies designed to keep attackers from penetrating the network are no longer adequate defense. Active defense measures within the network, threat hunting for intruders, and countermeasures located within your environment are the new baseline for adequate security. All IT security staff must change their mindset to realize that protecting the network perimeter in order to fend off evil is a doomed enterprise. Malicious actors will gain influence over user end points, inside staff will make mistakes, and malicious actors will occasionally emerge from the ranks of our own users. With these realities in mind, we must shift the defensive strategy from one of only prevention to one of prevention, detection and mitigation.

Networks must be designed not only to attempt to keep attackers out, but also to make their lives as difficult as possible once they do gain an internal foothold. Networks should be segmented with strong barriers in place between different logical groupings of resources. An attacker who finds himself on the inside of your network should have to fight for every inch of territory that she seeks to own, not be given free reign over a flat network with no further impediments. Ideally, private VLANs isolating each workstation from all others, and allowing access only to the handful of resources needed by each (such as email, proxy, file servers, and possibly a default gateway) would be implemented so that an intruder's ability to further pivot throughout a network is minimized. While such extreme isolation may not be feasible, segmenting networks at a number of different points to protect critical resources is a must.

Internal networks should be forensically ready, generating adequate logging events, aggregating those logs on secure servers, and maintaining them for sufficient time to facilitate effective incident investigations. Ensuring logging of critical events, including process tracking and command line auditing, is vital to a forensically ready network. Asset inventories, network diagrams, change management procedures, and incident response plans should be current and up to date. Security teams should use automation scripts to gather and update system baselines to be used as comparisons during incident analysis. Network security monitoring should be in place to detect and deter malicious internal activity, and end points should be secured through layered defenses including antivirus, white listing, and heuristic detection mechanisms. Honey items or canaries such as heavily monitored files, digitally watermarked files, honey pot systems, honey credentials, and other mechanisms can be used to further detect malicious actors roving about areas of the network where legitimate users should not be.

Only by making our environments as inhospitable to attackers as possible will we be able to continue to deter attack, detect intruders, and secure our assets from the multitude of adversaries that we face.

Additional References

The links below provide additional information about many of the topics discussed in this document. Some were consulted as research for this document and others offer more detailed information about specific topics discussed. We have no affiliation with the publishers of the below links, but we thank them for their information sharing and assistance to the information security community.

http://blog.commandlinekungfu.com/

https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx

https://help.rapid7.com/metasploit/Content/home.html

https://www.powershellempire.com/?page_id=83

https://github.com/davehull/Kansa

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

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

https://4sysops.com/archives/sign-your-powershell-scripts-to-increase-security/

https://blogs.technet.microsoft.com/heyscriptingguy/2012/07/23/an-introduction-to-powershell-remoting-part-one/

https://docs.microsoft.com/en-us/powershell/scripting/setup/winrmsecurity?view=powershell-5.1

https://msdn.microsoft.com/en-us/library/dd163506.aspx

https://github.com/sans-blue-team/DeepBlueCLI