Persistence, Tactic TA0003 - Enterprise

Archived: 2026-04-05 13:06:38 UTC

T1098 Account Manipulation Adversaries may manipulate accounts to maintain and/or elevate access to victim systems. Account manipulation may consist of any action that preserves or modifies adversary access to a compromised account, such as modifying credentials or permission groups. These actions could also include account activity designed to subvert security policies, such as performing iterative password updates to bypass password duration policies and preserve the life of compromised credentials. .001 Additional Cloud Credentials Adversaries may add adversary-controlled credentials to a cloud account to maintain persistent access to victim accounts and instances within the environment. .002 Additional Email Delegate Permissions Adversaries may grant additional permission levels to maintain persistent access to an adversary-controlled email account. .003 Additional Cloud Roles An adversary may add additional roles or permissions to an adversary-controlled cloud account to maintain persistent access to a tenant. For example, adversaries may update IAM policies in cloud-based environments or add a new global administrator in Office 365 environments. With sufficient permissions, a compromised account can gain almost unlimited access to data and settings (including the ability to reset the passwords of other admins).
.004 SSH Authorized Keys Adversaries may modify the SSH `authorized_keys` file to maintain persistence on a victim host. Linux distributions, macOS, and ESXi hypervisors commonly use key-based authentication to secure the authentication process of SSH sessions for remote management. The `authorized_keys` file in SSH specifies the SSH keys that can be used for logging into the user account for which the file is configured. This file is usually found in the user's home directory under `<user-home>/.ssh/authorized_keys` (or, on ESXi, `/etc/ssh/keys-<username>/authorized_keys`). Users may edit the system's SSH config file to modify the directives `PubkeyAuthentication` and `RSAAuthentication` to the value `yes` to ensure public key and RSA authentication are enabled, as well as modify the directive `PermitRootLogin` to the value `yes` to enable root authentication via SSH. The SSH config file is usually located under `/etc/ssh/sshd_config`. .005 Device Registration Adversaries may register a device to an adversary-controlled account. Devices may be registered in a multifactor authentication (MFA) system, which handles authentication to the network, or in a device management system, which handles device access and compliance. .006 Additional Container Cluster Roles An adversary may add additional roles or permissions to an adversary-controlled user or service account to maintain persistent access to a container orchestration system. For example, an adversary with sufficient permissions may create a RoleBinding or a ClusterRoleBinding to bind a Role or ClusterRole to a Kubernetes account. Where attribute-based access control (ABAC) is in use, an adversary with sufficient permissions may modify a Kubernetes ABAC policy to give the target account additional permissions. .007 Additional Local or Domain Groups An adversary may add additional local or domain groups to an adversary-controlled account to maintain persistent access to a system or domain. T1197 BITS Jobs Adversaries may abuse BITS jobs to persistently execute code and perform various background tasks. Windows Background Intelligent Transfer Service (BITS) is a low-bandwidth, asynchronous file transfer mechanism exposed through Component Object Model (COM). BITS is commonly used by updaters, messengers, and other applications preferred to operate in the background (using available idle bandwidth) without interrupting other networked applications. File transfer tasks are implemented as BITS jobs, which contain a queue of one or more file operations. T1547 Boot or Logon Autostart Execution Adversaries may

configure system settings to automatically execute a program during system boot or logon to maintain persistence or gain higher-level privileges on compromised systems. Operating systems may have mechanisms for automatically running a program on system boot or account logon. These mechanisms may include automatically executing programs that are placed in specially designated directories or are referenced by repositories that store configuration information, such as the Windows Registry. An adversary may achieve the same goal by modifying or extending features of the kernel. .001 Registry Run Keys / Startup Folder Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key. 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. .002 Authentication Package Adversaries may abuse authentication packages to execute DLLs when the system boots. Windows authentication package DLLs are loaded by the Local Security Authority (LSA) process at system start. They provide support for multiple logon processes and multiple security protocols to the operating system. .003 Time Providers Adversaries may abuse time providers to execute DLLs when the system boots. The Windows Time service (W32Time) enables time synchronization across and within domains. W32Time time providers are responsible for retrieving time stamps from hardware/network resources and outputting these values to other network clients. .004 Winlogon Helper DLL Adversaries may abuse features of Winlogon to execute DLLs and/or executables when a user logs in. Winlogon.exe is a Windows component responsible for actions at logon/logoff as well as the secure attention sequence (SAS) triggered by Ctrl-Alt-Delete. Registry entries in `HKLM\Software[\Wow6432Node\]\Microsoft\Windows NT\CurrentVersion\Winlogon\` and `HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\` are used to manage additional helper programs and functionalities that support Winlogon. .005 Security Support Provider Adversaries may abuse security support providers (SSPs) to execute DLLs when the system boots. Windows SSP DLLs are loaded into the Local Security Authority (LSA) process at system start. Once loaded into the LSA, SSP DLLs have access to encrypted and plaintext passwords that are stored in Windows, such as any logged-on user's Domain password or smart card PINs. .006 Kernel Modules and Extensions Adversaries may modify the kernel to automatically execute programs on system boot. Loadable Kernel Modules (LKMs) are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system. .007 Re-opened Applications Adversaries may modify plist files to automatically run an application when a user logs in. When a user logs out or restarts via the macOS Graphical User Interface (GUI), a prompt is provided to the user with a checkbox to "Reopen windows when logging back in". When selected, all applications currently open are added to a property list file named `com.apple.loginwindow.[UUID].plist` within the `~/Library/Preferences/ByHost` directory. Applications listed in this file are automatically reopened upon the user’s next logon. .008 LSASS Driver Adversaries may modify or add LSASS drivers to obtain persistence on compromised systems. The Windows security subsystem is a set of components that manage and enforce the security policy for a computer or domain. The Local Security Authority (LSA) is the main component responsible for local security policy and user authentication. The LSA includes multiple dynamic link libraries (DLLs) associated with various other security functions, all of which run in the context of the LSA Subsystem Service (LSASS) lsass.exe process. .009 Shortcut Modification Adversaries may create or modify shortcuts that can execute a program during system boot or user login. Shortcuts or symbolic links are used to reference other files or programs that will be opened or executed when the shortcut is clicked or executed by a system startup process. .010 Port Monitors Adversaries may use port monitors to run an adversary supplied DLL during system boot for

persistence or privilege escalation. A port monitor can be set through the `AddMonitor` API call to set a DLL to be loaded at startup. This DLL can be located in `C:\Windows\System32` and will be loaded and run by the print spooler service, `spoolsv.exe`, under SYSTEM level permissions on boot. .012 Print Processors Adversaries may abuse print processors to run malicious DLLs during system boot for persistence and/or privilege escalation. Print processors are DLLs that are loaded by the print spooler service, `spoolsv.exe`, during boot. .013 XDG Autostart Entries Adversaries may add or modify XDG Autostart Entries to execute malicious programs or commands when a user's desktop environment is loaded at login. XDG Autostart entries are available for any XDG-compliant Linux system. XDG Autostart entries use Desktop Entry files (`.desktop`) to configure the user's desktop environment upon user login. These configuration files determine what applications launch upon user login, define associated applications to open specific file types, and define applications used to open removable media. .014 Active Setup Adversaries may achieve persistence by adding a Registry key to the Active Setup of the local machine. Active Setup is a Windows mechanism that is used to execute programs when a user logs in. The value stored in the Registry key will be executed after a user logs into the computer. These programs will be executed under the context of the user and will have the account's associated permissions level. .015 Login Items Adversaries may add login items to execute upon user login to gain persistence or escalate privileges. Login items are applications, documents, folders, or server connections that are automatically launched when a user logs in. Login items can be added via a shared file list or Service Management Framework. Shared file list login items can be set using scripting languages such as AppleScript, whereas the Service Management Framework uses the API call `SMLoginItemSetEnabled`. T1037 Boot or Logon Initialization Scripts Adversaries may use scripts automatically executed at boot or logon initialization to establish persistence. Initialization scripts can be used to perform administrative functions, which may often execute other programs or send information to an internal logging server. These scripts can vary based on operating system and whether applied locally or remotely. .001 Logon Script (Windows) Adversaries may use Windows logon scripts automatically executed at logon initialization to establish persistence. Windows allows logon scripts to be run whenever a specific user or group of users log into a system. This is done via adding a path to a script to the `HKCU\Environment\UserInitMprLogonScript` Registry key. .002 Login Hook Adversaries may use a Login Hook to establish persistence executed upon user logon. A login hook is a plist file that points to a specific script to execute with root privileges upon user logon. The plist file is located in the `/Library/Preferences/com.apple.loginwindow.plist` file and can be modified using the `defaults` command-line utility. This behavior is the same for logout hooks where a script can be executed upon user logout. All hooks require administrator permissions to modify or create hooks. .003 Network Logon Script Adversaries may use network logon scripts automatically executed at logon initialization to establish persistence. Network logon scripts can be assigned using Active Directory or Group Policy Objects. These logon scripts run with the privileges of the user they are assigned to. Depending on the systems within the network, initializing one of these scripts could apply to more than one or potentially all systems. .004 RC Scripts Adversaries may establish persistence by modifying RC scripts, which are executed during a Unix-like system's startup. These files allow system administrators to map and start custom services at startup for different run levels. RC scripts require root privileges to modify. .005 Startup Items Adversaries may use startup items automatically executed at boot initialization to establish persistence. Startup items execute during the final phase of the boot process and contain shell scripts or other executable files along with configuration information used by the system to determine the execution order for all startup items. T1671 Cloud Application Integration Adversaries may achieve persistence by leveraging OAuth application integrations in a software-as-a-service environment. Adversaries may create a

custom application, add a legitimate application into the environment, or even co-opt an existing integration to achieve malicious ends. T1554 Compromise Host Software Binary Adversaries may modify host software binaries to establish persistent access to systems. Software binaries/executables provide a wide range of system commands or services, programs, and libraries. Common software binaries are SSH clients, FTP clients, email clients, web browsers, and many other user or server applications. T1136 Create Account Adversaries may create an account to maintain access to victim systems. With a sufficient level of access, creating such accounts may be used to establish secondary credentialed access that do not require persistent remote access tools to be deployed on the system. .001 Local Account Adversaries may create a local account to maintain access to victim systems. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service. .002 Domain Account Adversaries may create a domain account to maintain access to victim systems. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover user, administrator, and service accounts. With a sufficient level of access, the `net user /add /domain` command can be used to create a domain account. .003 Cloud Account Adversaries may create a cloud account to maintain access to victim systems. With a sufficient level of access, such accounts may be used to establish secondary credentialed access that does not require persistent remote access tools to be deployed on the system. T1543 Create or Modify System Process Adversaries may create or modify system-level processes to repeatedly execute malicious payloads as part of persistence. When operating systems boot up, they can start processes that perform background system functions. On Windows and Linux, these system processes are referred to as services. On macOS, launchd processes known as Launch Daemon and Launch Agent are run to finish system initialization and load user specific parameters. .001 Launch Agent Adversaries may create or modify launch agents to repeatedly execute malicious payloads as part of persistence. When a user logs in, a per-user launchd process is started which loads the parameters for each launch-on-demand user agent from the property list (.plist) file found in `/System/Library/LaunchAgents`, `/Library/LaunchAgents`, and `~/Library/LaunchAgents`. Property list files use the `Label`, `ProgramArguments`, and `RunAtLoad` keys to identify the Launch Agent's name, executable location, and execution time. Launch Agents are often installed to perform updates to programs, launch user specified programs at login, or to conduct other developer tasks. .002 Systemd Service Adversaries may create or modify systemd services to repeatedly execute malicious payloads as part of persistence. Systemd is a system and service manager commonly used for managing background daemon processes (also known as services) and other system resources. Systemd is the default initialization (init) system on many Linux distributions replacing legacy init systems, including SysVinit and Upstart, while remaining backwards compatible. .003 Windows Service Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. When Windows boots up, it starts programs or applications called services that perform background system functions. Windows service configuration information, including the file path to the service's executable or recovery programs/commands, is stored in the Windows Registry. .004 Launch Daemon Adversaries may create or modify Launch Daemons to execute malicious payloads as part of persistence. Launch Daemons are plist files used to interact with Launchd, the service management framework used by macOS. Launch Daemons require elevated privileges to install, are executed for every user on a system prior to login, and run in the background without the need for user interaction. During the macOS initialization startup, the launchd process loads the parameters for launch-on-demand system-level daemons from plist files found in `/System/Library/LaunchDaemons/` and `/Library/LaunchDaemons/`. Required Launch Daemons parameters include a `Label` to identify the task, `Program` to provide a path to the executable, and `RunAtLoad` to specify

when the task is run. Launch Daemons are often used to provide access to shared resources, updates to software, or conduct automation tasks. .005 Container Service Adversaries may create or modify container or container cluster management tools that run as daemons, agents, or services on individual hosts. These include software for creating and managing individual containers, such as Docker and Podman, as well as container cluster node-level agents such as kubelet. By modifying these services, an adversary may be able to achieve persistence or escalate their privileges on a host. T1546 Event Triggered Execution Adversaries may establish persistence and/or elevate privileges using system mechanisms that trigger execution based on specific events. Various operating systems have means to monitor and subscribe to events such as logons or other user activity such as running specific applications/binaries. Cloud environments may also support various functions and services that monitor and can be invoked in response to specific cloud events. .001 Change Default File Association Adversaries may establish persistence by executing malicious content triggered by a file type association. When a file is opened, the default program used to open the file (also called the file association or handler) is checked. File association selections are stored in the Windows Registry and can be edited by users, administrators, or programs that have Registry access or by administrators using the built-in assoc utility. Applications can modify the file association for a given file extension to call an arbitrary program when a file with the given extension is opened. .002 Screensaver Adversaries may establish persistence by executing malicious content triggered by user inactivity. Screensavers are programs that execute after a configurable time of user inactivity and consist of Portable Executable (PE) files with a .scr file extension. The Windows screensaver application scrnsave.scr is located in `C:\Windows\System32\` , and `C:\Windows\sysWOW64\` on 64-bit Windows systems, along with screensavers included with base Windows installations. .003 Windows Management Instrumentation Event Subscription Adversaries may establish persistence and elevate privileges by executing malicious content triggered by a Windows Management Instrumentation (WMI) event subscription. WMI can be used to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Examples of events that may be subscribed to are the wall clock time, user login, or the computer's uptime. .004 Unix Shell Configuration Modification Adversaries may establish persistence through executing malicious commands triggered by a user's shell. User Unix Shells execute several configuration scripts at different points throughout the session based on events. For example, when a user opens a command-line interface or remotely logs in (such as via SSH) a login shell is initiated. The login shell executes scripts from the system (`/etc`) and the user's home directory (`~/`) to configure the environment. All login shells on a system use /etc/profile when initiated. These configuration scripts run at the permission level of their directory and are often used to set environment variables, create aliases, and customize the user's environment. When the shell exits or terminates, additional shell scripts are executed to ensure the shell exits appropriately. .005 Trap Adversaries may establish persistence by executing malicious content triggered by an interrupt signal. The `trap` command allows programs and shells to specify commands that will be executed upon receiving interrupt signals. A common situation is a script allowing for graceful termination and handling of common keyboard interrupts like `ctrl+c` and `ctrl+d` . .006 LC_LOAD_DYLIB Addition Adversaries may establish persistence by executing malicious content triggered by the execution of tainted binaries. Mach-O binaries have a series of headers that are used to perform certain operations when a binary is loaded. The LC_LOAD_DYLIB header in a Mach-O binary tells macOS and OS X which dynamic libraries (dylibs) to load during execution time. These can be added ad-hoc to the compiled binary as long as adjustments are made to the rest of the fields and dependencies. There are tools available to perform these changes. .007 Netsh Helper DLL Adversaries may establish persistence by executing malicious content triggered by Netsh Helper DLLs. Netsh.exe (also referred to as Netshell) is a command-line scripting utility used to interact

with the network configuration of a system. It contains functionality to add helper DLLs for extending functionality of the utility. The paths to registered netsh.exe helper DLLs are entered into the Windows Registry at `HKLM\SOFTWARE\Microsoft\Netsh` . .008 Accessibility Features Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by accessibility features. Windows contains accessibility features that may be launched with a key combination before a user has logged in (ex: when the user is on the Windows logon screen). An adversary can modify the way these programs are launched to get a command prompt or backdoor without logging in to the system. .009 AppCert DLLs Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppCert DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the `AppCertDLLs` Registry key under `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\` are loaded into every process that calls the ubiquitously used application programming interface (API) functions `CreateProcess` , `CreateProcessAsUser` , `CreateProcessWithLoginW` , `CreateProcessWithTokenW` , or `WinExec` . .010 AppInit DLLs Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppInit DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the `AppInit_DLLs` value in the Registry keys `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows` or `HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows` are loaded by user32.dll into every process that loads user32.dll. In practice this is nearly every program, since user32.dll is a very common library. .011 Application Shimming Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by application shims. The Microsoft Windows Application Compatibility Infrastructure/Framework (Application Shim) was created to allow for backward compatibility of software as the operating system codebase changes over time. For example, the application shimming feature allows developers to apply fixes to applications (without rewriting code) that were created for Windows XP so that it will work with Windows 10. .012 Image File Execution Options Injection Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by Image File Execution Options (IFEO) debuggers. IFEOs enable a developer to attach a debugger to an application. When a process is created, a debugger present in an application's IFEO will be prepended to the application's name, effectively launching the new process under the debugger (e.g., `C:\dbg\ntsd.exe -g notepad.exe`). .013 PowerShell Profile Adversaries may gain persistence and elevate privileges by executing malicious content triggered by PowerShell profiles. A PowerShell profile (`profile.ps1`) is a script that runs when PowerShell starts and can be used as a logon script to customize user environments. .014 Emond Adversaries may gain persistence and elevate privileges by executing malicious content triggered by the Event Monitor Daemon (emond). Emond is a Launch Daemon that accepts events from various services, runs them through a simple rules engine, and takes action. The emond binary at `/sbin/emond` will load any rules from the `/etc/emond.d/rules/` directory and take action once an explicitly defined event takes place. .015 Component Object Model Hijacking Adversaries may establish persistence by executing malicious content triggered by hijacked references to Component Object Model (COM) objects. COM is a system within Windows to enable interaction between software components through the operating system. References to various COM objects are stored in the Registry. .016 Installer Packages Adversaries may establish persistence and elevate privileges by using an installer to trigger the execution of malicious content. Installer packages are OS specific and contain the resources an operating system needs to install applications on a system. Installer packages can include scripts that run prior to installation as well as after installation is complete. Installer scripts may inherit elevated permissions when executed. Developers often use these scripts to prepare the environment for installation, check requirements, download dependencies, and remove files after installation. .017 Udev Rules

Adversaries may maintain persistence through executing malicious content triggered using udev rules. Udev is the Linux kernel device manager that dynamically manages device nodes, handles access to pseudo-device files in the `/dev` directory, and responds to hardware events, such as when external devices like hard drives or keyboards are plugged in or removed. Udev uses rule files with `match keys` to specify the conditions a hardware event must meet and `action keys` to define the actions that should follow. Root permissions are required to create, modify, or delete rule files located in `/etc/udev/rules.d/`, `/run/udev/rules.d/`, `/usr/lib/udev/rules.d/`, `/usr/local/lib/udev/rules.d/`, and `/lib/udev/rules.d/`. Rule priority is determined by both directory and by the digit prefix in the rule filename. .018 Python Startup Hooks Adversaries may achieve persistence by leveraging Python's startup mechanisms, including path configuration (`.pth`) files and the `sitecustomize.py` or `usercustomize.py` modules. These files are automatically processed during the initialization of the Python interpreter, allowing for the execution of arbitrary code whenever Python is invoked. T1668 Exclusive Control Adversaries who successfully compromise a system may attempt to maintain persistence by "closing the door" behind them – in other words, by preventing other threat actors from initially accessing or maintaining a foothold on the same system. T1133 External Remote Services Adversaries may leverage external-facing remote services to initially access and/or persist within a network. Remote services such as VPNs, Citrix, and other access mechanisms allow users to connect to internal enterprise network resources from external locations. There are often remote service gateways that manage connections and credential authentication for these services. Services such as Windows Remote Management and VNC can also be used externally. T1574 Hijack Execution Flow Adversaries may execute their own malicious payloads by hijacking the way operating systems run programs. Hijacking execution flow can be for the purposes of persistence, since this hijacked execution may reoccur over time. Adversaries may also use these mechanisms to elevate privileges or evade defenses, such as application control or other restrictions on execution. .001 DLL Adversaries may abuse dynamic-link library files (DLLs) in order to achieve persistence, escalate privileges, and evade defenses. DLLs are libraries that contain code and data that can be simultaneously utilized by multiple programs. While DLLs are not malicious by nature, they can be abused through mechanisms such as side-loading, hijacking search order, and phantom DLL hijacking. .004 Dylib Hijacking Adversaries may execute their own payloads by placing a malicious dynamic library (dylib) with an expected name in a path a victim application searches at runtime. The dynamic loader will try to find the dylibs based on the sequential order of the search paths. Paths to dylibs may be prefixed with `@rpath`, which allows developers to use relative paths to specify an array of search paths used at runtime based on the location of the executable. Additionally, if weak linking is used, such as the `LC_LOAD_WEAK_DYLIB` function, an application will still execute even if an expected dylib is not present. Weak linking enables developers to run an application on multiple macOS versions as new APIs are added. .005 Executable Installer File Permissions Weakness Adversaries may execute their own malicious payloads by hijacking the binaries used by an installer. These processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself, are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM. .006 Dynamic Linker Hijacking Adversaries may execute their own malicious payloads by hijacking environment variables the dynamic linker uses to load shared libraries. During the execution preparation phase of a program, the dynamic linker loads specified absolute paths of shared libraries from various environment variables and files, such as `LD_PRELOAD` on Linux or `DYLD_INSERT_LIBRARIES` on macOS. Libraries specified in environment

variables are loaded first, taking precedence over system libraries with the same function name. Each platform's linker uses an extensive list of environment variables at different points in execution. These variables are often used by developers to debug binaries without needing to recompile, deconflict mapped symbols, and implement custom functions in the original library. .007 Path Interception by PATH Environment Variable Adversaries may execute their own malicious payloads by hijacking environment variables used to load libraries. The PATH environment variable contains a list of directories (User and System) that the OS searches sequentially through in search of the binary that was called from a script or the command line. .008 Path Interception by Search Order Hijacking Adversaries may execute their own malicious payloads by hijacking the search order used to load other programs. Because some programs do not call other programs using the full path, adversaries may place their own file in the directory where the calling program is located, causing the operating system to launch their malicious software at the request of the calling program. .009 Path Interception by Unquoted Path Adversaries may execute their own malicious payloads by hijacking vulnerable file path references. Adversaries can take advantage of paths that lack surrounding quotations by placing an executable in a higher level directory within the path, so that Windows will choose the adversary's executable to launch. .010 Services File Permissions Weakness Adversaries may execute their own malicious payloads by hijacking the binaries used by services. Adversaries may use flaws in the permissions of Windows services to replace the binary that is executed upon service start. These service processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM. .011 Services Registry Permissions Weakness Adversaries may execute their own malicious payloads by hijacking the Registry entries used by services. Flaws in the permissions for Registry keys related to services can allow adversaries to redirect the originally specified executable to one they control, launching their own code when a service starts. Windows stores local service configuration information in the Registry under `HKLM\SYSTEM\CurrentControlSet\Services`. The information stored under a service's Registry keys can be manipulated to modify a service's execution parameters through tools such as the service controller, sc.exe, PowerShell, or Reg. Access to Registry keys is controlled through access control lists and user permissions. .012 COR_PROFILER Adversaries may leverage the COR_PROFILER environment variable to hijack the execution flow of programs that load the .NET CLR. The COR_PROFILER is a .NET Framework feature which allows developers to specify an unmanaged (or external of .NET) profiling DLL to be loaded into each .NET process that loads the Common Language Runtime (CLR). These profilers are designed to monitor, troubleshoot, and debug managed code executed by the .NET CLR. .013 KernelCallbackTable Adversaries may abuse the `KernelCallbackTable` of a process to hijack its execution flow in order to run their own payloads. The `KernelCallbackTable` can be found in the Process Environment Block (PEB) and is initialized to an array of graphic functions available to a GUI process once `user32.dll` is loaded. .014 AppDomainManager Adversaries may execute their own malicious payloads by hijacking how the .NET `AppDomainManager` loads assemblies. The .NET framework uses the `AppDomainManager` class to create and manage one or more isolated runtime environments (called application domains) inside a process to host the execution of .NET applications. Assemblies (`.exe` or `.dll` binaries compiled to run as .NET code) may be loaded into an application domain as executable code. T1525 Implant Internal Image Adversaries may implant cloud or container images with malicious code to establish persistence after gaining access to an environment. Amazon Web Services (AWS) Amazon Machine

Images (AMIs), Google Cloud Platform (GCP) Images, and Azure Images as well as popular container runtimes such as Docker can be implanted or backdoored. Unlike Upload Malware, this technique focuses on adversaries implanting an image in a registry within a victim's environment. Depending on how the infrastructure is provisioned, this could provide persistent access if the infrastructure provisioning tool is instructed to always use the latest image. T1556 Modify Authentication Process Adversaries may modify authentication mechanisms and processes to access user credentials or enable otherwise unwarranted access to accounts. The authentication process is handled by mechanisms, such as the Local Security Authentication Server (LSASS) process and the Security Accounts Manager (SAM) on Windows, pluggable authentication modules (PAM) on Unix-based systems, and authorization plugins on MacOS systems, responsible for gathering, storing, and validating credentials. By modifying an authentication process, an adversary may be able to authenticate to a service or system without using Valid Accounts. .001 Domain Controller Authentication Adversaries may patch the authentication process on a domain controller to bypass the typical authentication mechanisms and enable access to accounts. .002 Password Filter DLL Adversaries may register malicious password filter dynamic link libraries (DLLs) into the authentication process to acquire user credentials as they are validated. .003 Pluggable Authentication Modules Adversaries may modify pluggable authentication modules (PAM) to access user credentials or enable otherwise unwarranted access to accounts. PAM is a modular system of configuration files, libraries, and executable files which guide authentication for many services. The most common authentication module is `pam_unix.so`, which retrieves, sets, and verifies account authentication information in `/etc/passwd` and `/etc/shadow`. .004 Network Device Authentication Adversaries may use Patch System Image to hard code a password in the operating system, thus bypassing of native authentication mechanisms for local accounts on network devices. .005 Reversible Encryption An adversary may abuse Active Directory authentication encryption properties to gain access to credentials on Windows systems. The `AllowReversiblePasswordEncryption` property specifies whether reversible password encryption for an account is enabled or disabled. By default this property is disabled (instead storing user credentials as the output of one-way hashing functions) and should not be enabled unless legacy or other software require it. .006 Multi-Factor Authentication Adversaries may disable or modify multi-factor authentication (MFA) mechanisms to enable persistent access to compromised accounts. .007 Hybrid Identity Adversaries may patch, modify, or otherwise backdoor cloud authentication processes that are tied to on-premises user identities in order to bypass typical authentication mechanisms, access credentials, and enable persistent access to accounts. .008 Network Provider DLL Adversaries may register malicious network provider dynamic link libraries (DLLs) to capture cleartext user credentials during the authentication process. Network provider DLLs allow Windows to interface with specific network protocols and can also support add-on credential management functions. During the logon process, Winlogon (the interactive logon module) sends credentials to the local `mpnotify.exe` process via RPC. The `mpnotify.exe` process then shares the credentials in cleartext with registered credential managers when notifying that a logon event is happening. .009 Conditional Access Policies Adversaries may disable or modify conditional access policies to enable persistent access to compromised accounts. Conditional access policies are additional verifications used by identity providers and identity and access management systems to determine whether a user should be granted access to a resource. T1112 Modify Registry Adversaries may interact with the Windows Registry as part of a variety of other techniques to aid in defense evasion, persistence, and execution. T1137 Office Application Startup Adversaries may leverage Microsoft Office-based applications for persistence between startups. Microsoft Office is a fairly common application suite on Windows-based operating systems within an enterprise network. There are multiple mechanisms that can be used with Office for persistence when an Office-based application is started; this can

include the use of Office Template Macros and add-ins. .001 Office Template Macros Adversaries may abuse Microsoft Office templates to obtain persistence on a compromised system. Microsoft Office contains templates that are part of common Office applications and are used to customize styles. The base templates within the application are used each time an application starts. .002 Office Test Adversaries may abuse the Microsoft Office "Office Test" Registry key to obtain persistence on a compromised system. An Office Test Registry location exists that allows a user to specify an arbitrary DLL that will be executed every time an Office application is started. This Registry key is thought to be used by Microsoft to load DLLs for testing and debugging purposes while developing Office applications. This Registry key is not created by default during an Office installation. .003 Outlook Forms Adversaries may abuse Microsoft Outlook forms to obtain persistence on a compromised system. Outlook forms are used as templates for presentation and functionality in Outlook messages. Custom Outlook forms can be created that will execute code when a specifically crafted email is sent by an adversary utilizing the same custom Outlook form. .004 Outlook Home Page Adversaries may abuse Microsoft Outlook's Home Page feature to obtain persistence on a compromised system. Outlook Home Page is a legacy feature used to customize the presentation of Outlook folders. This feature allows for an internal or external URL to be loaded and presented whenever a folder is opened. A malicious HTML page can be crafted that will execute code when loaded by Outlook Home Page. .005 Outlook Rules Adversaries may abuse Microsoft Outlook rules to obtain persistence on a compromised system. Outlook rules allow a user to define automated behavior to manage email messages. A benign rule might, for example, automatically move an email to a particular folder in Outlook if it contains specific words from a specific sender. Malicious Outlook rules can be created that can trigger code execution when an adversary sends a specifically crafted email to that user. .006 Add-ins Adversaries may abuse Microsoft Office add-ins to obtain persistence on a compromised system. Office add-ins can be used to add functionality to Office programs. There are different types of add-ins that can be used by the various Office products; including Word/Excel add-in Libraries (WLL/XLL), VBA add-ins, Office Component Object Model (COM) add-ins, automation add-ins, VBA Editor (VBE), Visual Studio Tools for Office (VSTO) add-ins, and Outlook add-ins. T1653 Power Settings Adversaries may impair a system's ability to hibernate, reboot, or shut down in order to extend access to infected machines. When a computer enters a dormant state, some or all software and hardware may cease to operate which can disrupt malicious activity. T1542 Pre-OS Boot Adversaries may abuse Pre-OS Boot mechanisms as a way to establish persistence on a system. During the booting process of a computer, firmware and various startup services are loaded before the operating system. These programs control flow of execution before the operating system takes control. .001 System Firmware Adversaries may modify system firmware to persist on systems.The BIOS (Basic Input/Output System) and The Unified Extensible Firmware Interface (UEFI) or Extensible Firmware Interface (EFI) are examples of system firmware that operate as the software interface between the operating system and hardware of a computer. .002 Component Firmware Adversaries may modify component firmware to persist on systems. Some adversaries may employ sophisticated means to compromise computer components and install malicious firmware that will execute adversary code outside of the operating system and main system firmware or BIOS. This technique may be similar to System Firmware but conducted upon other system components/devices that may not have the same capability or level of integrity checking. .003 Bootkit Adversaries may use bootkits to persist on systems. A bootkit is a malware variant that modifies the boot sectors of a hard drive, allowing malicious code to execute before a computer's operating system has loaded. Bootkits reside at a layer below the operating system and may make it difficult to perform full remediation unless an organization suspects one was used and can act accordingly. .004 ROMMONkit Adversaries may abuse the ROM Monitor (ROMMON) by loading an unauthorized firmware with adversary code to provide

persistent access and manipulate device behavior that is difficult to detect. .005 TFTP Boot Adversaries may abuse netbooting to load an unauthorized network device operating system from a Trivial File Transfer Protocol (TFTP) server. TFTP boot (netbooting) is commonly used by network administrators to load configuration-controlled network device images from a centralized management server. Netbooting is one option in the boot sequence and can be used to centralize, manage, and control device images. T1053 Scheduled Task/Job Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the remote system. .002 At Adversaries may abuse the at utility to perform task scheduling for initial or recurring execution of malicious code. The at utility exists as an executable within Windows, Linux, and macOS for scheduling tasks at a specified time and date. Although deprecated in favor of Scheduled Task's schtasks in Windows environments, using at requires that the Task Scheduler service be running, and the user to be logged on as a member of the local Administrators group. In addition to explicitly running the `at` command, adversaries may also schedule a task with at by directly leveraging the Windows Management Instrumentation `Win32_ScheduledJob` WMI class. .003 Cron Adversaries may abuse the `cron` utility to perform task scheduling for initial or recurring execution of malicious code. The `cron` utility is a time-based job scheduler for Unix-like operating systems. The `crontab` file contains the schedule of cron entries to be run and the specified times for execution. Any `crontab` files are stored in operating system-specific file paths. .005 Scheduled Task Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The schtasks utility can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library and Windows Management Instrumentation (WMI) to create a scheduled task. Adversaries may also utilize the Powershell Cmdlet `Invoke-CimMethod`, which leverages WMI class `PS_ScheduledTask` to create a scheduled task via an XML path. .006 Systemd Timers Adversaries may abuse systemd timers to perform task scheduling for initial or recurring execution of malicious code. Systemd timers are unit files with file extension `.timer` that control services. Timers can be set to run on a calendar event or after a time span relative to a starting point. They can be used as an alternative to Cron in Linux environments. Systemd timers may be activated remotely via the `systemctl` command line utility, which operates over SSH. .007 Container Orchestration Job Adversaries may abuse task scheduling functionality provided by container orchestration tools such as Kubernetes to schedule deployment of containers configured to execute malicious code. Container orchestration jobs run these automated tasks at a specific date and time, similar to cron jobs on a Linux system. Deployments of this type can also be configured to maintain a quantity of containers over time, automating the process of maintaining persistence within a cluster. T1505 Server Software Component Adversaries may abuse legitimate extensible development features of servers to establish persistent access to systems. Enterprise server applications may include features that allow developers to write and install software or scripts to extend the functionality of the main application. Adversaries may install malicious components to extend and abuse server applications. .001 SQL Stored Procedures Adversaries may abuse SQL stored procedures to establish persistent access to systems. SQL Stored Procedures are code that can be saved and reused so that database users do not waste time rewriting frequently used SQL queries. Stored procedures can be invoked via SQL statements to the database using the procedure name

or via defined events (e.g. when a SQL server application is started/restarted). .002 Transport Agent Adversaries may abuse Microsoft transport agents to establish persistent access to systems. Microsoft Exchange transport agents can operate on email messages passing through the transport pipeline to perform various tasks such as filtering spam, filtering malicious attachments, journaling, or adding a corporate signature to the end of all outgoing emails. Transport agents can be written by application developers and then compiled to .NET assemblies that are subsequently registered with the Exchange server. Transport agents will be invoked during a specified stage of email processing and carry out developer defined tasks. .003 Web Shell Adversaries may backdoor web servers with web shells to establish persistent access to systems. A Web shell is a Web script that is placed on an openly accessible Web server to allow an adversary to access the Web server as a gateway into a network. A Web shell may provide a set of functions to execute or a command-line interface on the system that hosts the Web server. .004 IIS Components Adversaries may install malicious components that run on Internet Information Services (IIS) web servers to establish persistence. IIS provides several mechanisms to extend the functionality of the web servers. For example, Internet Server Application Programming Interface (ISAPI) extensions and filters can be installed to examine and/or modify incoming and outgoing IIS web requests. Extensions and filters are deployed as DLL files that export three functions: `Get{Extension/Filter}Version`, `Http{Extension/Filter}Proc`, and (optionally) `Terminate{Extension/Filter}`. IIS modules may also be installed to extend IIS web servers. .005 Terminal Services DLL Adversaries may abuse components of Terminal Services to enable persistent access to systems. Microsoft Terminal Services, renamed to Remote Desktop Services in some Windows Server OSs as of 2022, enable remote terminal connections to hosts. Terminal Services allows servers to transmit a full, interactive, graphical user interface to clients via RDP. .006 vSphere Installation Bundles Adversaries may abuse vSphere Installation Bundles (VIBs) to establish persistent access to ESXi hypervisors. VIBs are collections of files used for software distribution and virtual system management in VMware environments. Since ESXi uses an in-memory filesystem where changes made to most files are stored in RAM rather than in persistent storage, these modifications are lost after a reboot. However, VIBs can be used to create startup tasks, apply custom firewall rules, or deploy binaries that persist across reboots. Typically, administrators use VIBs for updates and system maintenance. T1176 Software Extensions Adversaries may abuse software extensions to establish persistent access to victim systems. Software extensions are modular components that enhance or customize the functionality of software applications, including web browsers, Integrated Development Environments (IDEs), and other platforms. Extensions are typically installed via official marketplaces, app stores, or manually loaded by users, and they often inherit the permissions and access levels of the host application. .001 Browser Extensions Adversaries may abuse internet browser extensions to establish persistent access to victim systems. Browser extensions or plugins are small programs that can add functionality to and customize aspects of internet browsers. They can be installed directly via a local file or custom URL or through a browser's app store - an official online platform where users can browse, install, and manage extensions for a specific web browser. Extensions generally inherit the web browser's permissions previously granted. .002 IDE Extensions Adversaries may abuse an integrated development environment (IDE) extension to establish persistent access to victim systems. IDEs such as Visual Studio Code, IntelliJ IDEA, and Eclipse support extensions - software components that add features like code linting, auto-completion, task automation, or integration with tools like Git and Docker. A malicious extension can be installed through an extension marketplace (i.e., Compromise Software Dependencies and Development Tools) or side-loaded directly into the IDE. T1205 Traffic Signaling Adversaries may use traffic signaling to hide open ports or other malicious functionality used for persistence or command and control. Traffic signaling involves the use of a magic value or

sequence that must be sent to a system to trigger a special response, such as opening a closed port or executing a malicious task. This may take the form of sending a series of packets with certain characteristics before a port will be opened that the adversary can use for command and control. Usually this series of packets consists of attempted connections to a predefined sequence of closed ports (i.e. Port Knocking), but can involve unusual flags, specific strings, or other unique characteristics. After the sequence is completed, opening a port may be accomplished by the host-based firewall, but could also be implemented by custom software. .001 Port Knocking Adversaries may use port knocking to hide open ports used for persistence or command and control. To enable a port, an adversary sends a series of attempted connections to a predefined sequence of closed ports. After the sequence is completed, opening a port is often accomplished by the host based firewall, but could also be implemented by custom software. .002 Socket Filters Adversaries may attach filters to a network socket to monitor then activate backdoors used for persistence or command and control. With elevated permissions, adversaries can use features such as the `libpcap` library to open sockets and install filters to allow or disallow certain types of data to come through the socket. The filter may apply to all traffic passing through the specified network interface (or every interface if not specified). When the network interface receives a packet matching the filter criteria, additional actions can be triggered on the host, such as activation of a reverse shell. T1078 Valid Accounts Adversaries may obtain and abuse credentials of existing accounts as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Compromised credentials may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access, network devices, and remote desktop. Compromised credentials may also grant an adversary increased privilege to specific systems or access to restricted areas of the network. Adversaries may choose not to use malware or tools in conjunction with the legitimate access those credentials provide to make it harder to detect their presence. .001 Default Accounts Adversaries may obtain and abuse credentials of a default account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Default accounts are those that are built-into an OS, such as the Guest or Administrator accounts on Windows systems. Default accounts also include default factory/provider set accounts on other types of systems, software, or devices, including the root user account in AWS, the root user account in ESXi, and the default service account in Kubernetes. .002 Domain Accounts Adversaries may obtain and abuse credentials of a domain account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover users, administrators, and services. .003 Local Accounts Adversaries may obtain and abuse credentials of a local account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service. .004 Cloud Accounts Valid accounts in cloud environments may allow adversaries to perform actions to achieve Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application. Cloud Accounts can exist solely in the cloud; alternatively, they may be hybrid-joined between on-premises systems and the cloud through syncing or federation with other identity sources such as Windows Active Directory.
