Execution, Tactic TA0002 - Enterprise

Archived: 2026-04-05 13:18:38 UTC

T1651 Cloud Administration Command Adversaries may abuse cloud management services to execute commands within virtual machines. Resources such as AWS Systems Manager, Azure RunCommand, and Runbooks allow users to remotely run scripts in virtual machines by leveraging installed virtual machine agents. T1059 Command and Scripting Interpreter Adversaries may abuse command and script interpreters to execute commands, scripts, or binaries. These interfaces and languages provide ways of interacting with computer systems and are a common feature across many different platforms. Most systems come with some built-in command-line interface and scripting capabilities, for example, macOS and Linux distributions include some flavor of Unix Shell while Windows installations include the Windows Command Shell and PowerShell. .001 PowerShell Adversaries may abuse PowerShell commands and scripts for execution. PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the `Start-Process` cmdlet which can be used to run an executable and the `Invoke-Command` cmdlet which runs a command locally or on a remote computer (though administrator permissions are required to use PowerShell to connect to remote systems). .002 AppleScript Adversaries may abuse AppleScript for execution. AppleScript is a macOS scripting language designed to control applications and parts of the OS via inter-application messages called AppleEvents. These AppleEvent messages can be sent independently or easily scripted with AppleScript. These events can locate open windows, send keystrokes, and interact with almost any open application locally or remotely. .003 Windows Command Shell Adversaries may abuse the Windows command shell for execution. The Windows command shell (cmd) is the primary command prompt on Windows systems. The Windows command prompt can be used to control almost any aspect of a system, with various permission levels required for different subsets of commands. The command prompt can be invoked remotely via Remote Services such as SSH. .004 Unix Shell Adversaries may abuse Unix shell commands and scripts for execution. Unix shells are the primary command prompt on Linux, macOS, and ESXi systems, though many variations of the Unix shell exist (e.g. sh, ash, bash, zsh, etc.) depending on the specific OS or distribution. Unix shells can control every aspect of a system, with certain commands requiring elevated privileges. .005 Visual Basic Adversaries may abuse Visual Basic (VB) for execution. VB is a programming language created by Microsoft with interoperability with many Windows technologies such as Component Object Model and the Native API through the Windows API. Although tagged as legacy with no planned future evolutions, VB is integrated and supported in the .NET Framework and cross-platform .NET Core. .006 Python Adversaries may abuse Python commands and scripts for execution. Python is a very popular scripting/programming language, with capabilities to perform many functions. Python can be executed interactively from the command-line (via the `python.exe` interpreter) or via scripts (.py) that can be written and distributed to different systems. Python code can also be compiled into binary executables. .007 JavaScript Adversaries may abuse various implementations of JavaScript for execution. JavaScript (JS) is a platform-independent scripting language (compiled just-in-time at runtime) commonly associated with scripts in webpages, though JS can be executed in runtime environments outside the browser. .008 Network Device CLI Adversaries may abuse scripting or built-in command line interpreters (CLI) on network devices to execute malicious command and payloads. The CLI is the primary means through which users and administrators interact

with the device in order to view system information, modify device operations, or perform diagnostic and administrative functions. CLIs typically contain various permission levels required for different commands. .009 Cloud API Adversaries may abuse cloud APIs to execute malicious commands. APIs available in cloud environments provide various functionalities and are a feature-rich method for programmatic access to nearly all aspects of a tenant. These APIs may be utilized through various methods such as command line interpreters (CLIs), in-browser Cloud Shells, PowerShell modules like Azure for PowerShell, or software developer kits (SDKs) available for languages such as Python. .010 AutoHotKey & AutoIT Adversaries may execute commands and perform malicious tasks using AutoIT and AutoHotKey automation scripts. AutoIT and AutoHotkey (AHK) are scripting languages that enable users to automate Windows tasks. These automation scripts can be used to perform a wide variety of actions, such as clicking on buttons, entering text, and opening and closing programs. .011 Lua Adversaries may abuse Lua commands and scripts for execution. Lua is a cross-platform scripting and programming language primarily designed for embedded use in applications. Lua can be executed on the command-line (through the stand-alone lua interpreter), via scripts (`.lua`), or from Lua-embedded programs (through the `struct lua_State`). .012 Hypervisor CLI Adversaries may abuse hypervisor command line interpreters (CLIs) to execute malicious commands. Hypervisor CLIs typically enable a wide variety of functionality for managing both the hypervisor itself and the guest virtual machines it hosts. .013 Container CLI/API Adversaries may abuse built-in CLI tools or API calls to execute malicious commands in containerized environments. T1609 Container Administration Command Adversaries may abuse a container administration service to execute commands within a container. A container administration service such as the Docker daemon, the Kubernetes API server, or the kubelet may allow remote management of containers within an environment. T1610 Deploy Container Adversaries may deploy a container into an environment to facilitate execution or evade defenses. In some cases, adversaries may deploy a new container to execute processes associated with a particular image or deployment, such as processes that execute or download malware. In others, an adversary may deploy a new container configured without network rules, user limitations, etc. to bypass existing defenses within the environment. In Kubernetes environments, an adversary may attempt to deploy a privileged or vulnerable container into a specific node in order to Escape to Host and access other containers running on the node. T1675 ESXi Administration Command Adversaries may abuse ESXi administration services to execute commands on guest machines hosted within an ESXi virtual environment. Persistent background services on ESXi-hosted VMs, such as the VMware Tools Daemon Service, allow for remote management from the ESXi server. The tools daemon service runs as `vmtoolsd.exe` on Windows guest operating systems, `vmware-tools-daemon` on macOS, and `vmtoolsd` on Linux. T1203 Exploitation for Client Execution Adversaries may exploit software vulnerabilities in client applications to execute code. Vulnerabilities can exist in software due to unsecure coding practices that can lead to unanticipated behavior. Adversaries can take advantage of certain vulnerabilities through targeted exploitation for the purpose of arbitrary code execution. Oftentimes the most valuable exploits to an offensive toolkit are those that can be used to obtain code execution on a remote system because they can be used to gain access to that system. Users will expect to see files related to the applications they commonly used to do work, so they are a useful target for exploit research and development because of their high utility. T1674 Input Injection Adversaries may simulate keystrokes on a victim's computer by various means to perform any type of action on behalf of the user, such as launching the command interpreter using keyboard shortcuts, typing an inline script to be executed, or interacting directly with a GUI-based application. These actions can be preprogrammed into adversary tooling or executed through physical devices such as Human Interface Devices (HIDs). T1559 Inter-Process Communication Adversaries may abuse inter-process communication (IPC) mechanisms for local

code or command execution. IPC is typically used by processes to share data, communicate with each other, or synchronize execution. IPC is also commonly used to avoid situations such as deadlocks, which occurs when processes are stuck in a cyclic waiting pattern. .001 Component Object Model Adversaries may use the Windows Component Object Model (COM) for local code execution. COM is an inter-process communication (IPC) component of the native Windows application programming interface (API) that enables interaction between software objects, or executable code that implements one or more interfaces. Through COM, a client object can call methods of server objects, which are typically binary Dynamic Link Libraries (DLL) or executables (EXE). Remote COM execution is facilitated by Remote Services such as Distributed Component Object Model (DCOM). .002 Dynamic Data Exchange Adversaries may use Windows Dynamic Data Exchange (DDE) to execute arbitrary commands. DDE is a client-server protocol for one-time and/or continuous inter-process communication (IPC) between applications. Once a link is established, applications can autonomously exchange transactions consisting of strings, warm data links (notifications when a data item changes), hot data links (duplications of changes to a data item), and requests for command execution. .003 XPC Services Adversaries can provide malicious content to an XPC service daemon for local code execution. macOS uses XPC services for basic inter-process communication between various processes, such as between the XPC Service daemon and third-party application privileged helper tools. Applications can send messages to the XPC Service daemon, which runs as root, using the low-level XPC Service `C API` or the high level `NSXPCConnection API` in order to handle tasks that require elevated privileges (such as network connections). Applications are responsible for providing the protocol definition which serves as a blueprint of the XPC services. Developers typically use XPC Services to provide applications stability and privilege separation between the application client and the daemon. T1106 Native API Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes. These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations. T1677 Poisoned Pipeline Execution Adversaries may manipulate continuous integration / continuous development (CI/CD) processes by injecting malicious code into the build process. There are several mechanisms for poisoning pipelines: 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. T1648 Serverless Execution Adversaries may abuse serverless computing, integration, and automation services to execute arbitrary code in cloud environments. Many cloud providers offer a variety of serverless resources, including compute engines, application integration services, and web servers. T1129 Shared Modules Adversaries may execute malicious payloads via loading shared modules. Shared modules are executable files that are loaded into processes to provide access to reusable code, such as specific custom functions or invoking OS API functions (i.e., Native API). T1072 Software Deployment Tools Adversaries may gain access to and use centralized software suites installed within an enterprise to execute commands and move laterally through the network. Configuration management and software deployment applications may be used in an enterprise network or cloud environment for routine administration purposes. These systems may also be integrated into CI/CD pipelines. Examples of such solutions include: SCCM, HBSS, Altiris, AWS Systems Manager, Microsoft Intune, Azure Arc, and GCP Deployment Manager. T1569 System Services Adversaries may abuse system services or daemons to execute commands or programs. Adversaries can execute malicious content by interacting with or creating services either locally or remotely. Many services are set to run at boot, which can aid in achieving persistence (Create or Modify System Process), but adversaries can also abuse services for one-time or temporary execution. .001 Launchctl Adversaries may abuse launchctl to execute commands or programs. Launchctl interfaces with launchd, the service management framework for macOS. Launchctl supports taking subcommands on the command-line, interactively, or even redirected from standard input. .002 Service Execution Adversaries may abuse the Windows service control manager to execute malicious commands or payloads. The Windows service control manager (`services.exe`) is an interface to manage and manipulate services. The service control manager is accessible to users via GUI components as well as system utilities such as `sc.exe` and Net. .003 Systemctl Adversaries may abuse systemctl to execute commands or programs. Systemctl is the primary interface for systemd, the Linux init system and service manager. Typically invoked from a shell, Systemctl can also be integrated into scripts or applications. T1204 User Execution An adversary may rely upon specific actions by a user in order to gain execution. Users may be subjected to social engineering to get them to execute malicious code by, for example, opening a malicious document file or link. These user actions will typically be observed as follow-on behavior from forms of Phishing. .001 Malicious Link An adversary may rely upon a user clicking a malicious link in order to gain execution. Users may be subjected to social engineering to get them to click on a

link that will lead to code execution. This user action will typically be observed as follow-on behavior from Spearphishing Link. Clicking on a link may also lead to other execution techniques such as exploitation of a browser or application vulnerability via Exploitation for Client Execution. Links may also lead users to download files that require execution via Malicious File. .002 Malicious File An adversary may rely upon a user opening a malicious file in order to gain execution. Users may be subjected to social engineering to get them to open a file that will lead to code execution. This user action will typically be observed as follow-on behavior from Spearphishing Attachment. Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, .cpl, .reg, and .iso. .003 Malicious Image Adversaries may rely on a user running a malicious image to facilitate execution. 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 backdoored. Backdoored images may be uploaded to a public repository via Upload Malware, and users may then download and deploy an instance or container from the image without realizing the image is malicious, thus bypassing techniques that specifically achieve Initial Access. This can lead to the execution of malicious code, such as code that executes cryptocurrency mining, in the instance or container. .004 Malicious Copy and Paste An adversary may rely upon a user copying and pasting code in order to gain execution. Users may be subjected to social engineering to get them to copy and paste code directly into a Command and Scripting Interpreter. One such strategy is "ClickFix," in which adversaries present users with seemingly helpful solutions—such as prompts to fix errors or complete CAPTCHAs—that instead instruct the user to copy and paste malicious code. .005 Malicious Library Adversaries may rely on a user installing a malicious library to facilitate execution. Threat actors may Upload Malware to package managers such as NPM and PyPi, as well as to public code repositories such as GitHub. User may install libraries without realizing they are malicious, thus bypassing techniques that specifically achieve Initial Access. This can lead to the execution of malicious code, such as code that establishes persistence, steals data, or mines cryptocurrency. T1047 Windows Management Instrumentation Adversaries may abuse Windows Management Instrumentation (WMI) to execute malicious commands and payloads. WMI is designed for programmers and is the infrastructure for management data and operations on Windows systems. WMI is an administration feature that provides a uniform environment to access Windows system components.
