May 10 2024

NETMANAGEIT

Intelligence Report

Tracking the Surge in Non-PE Cyber Threats

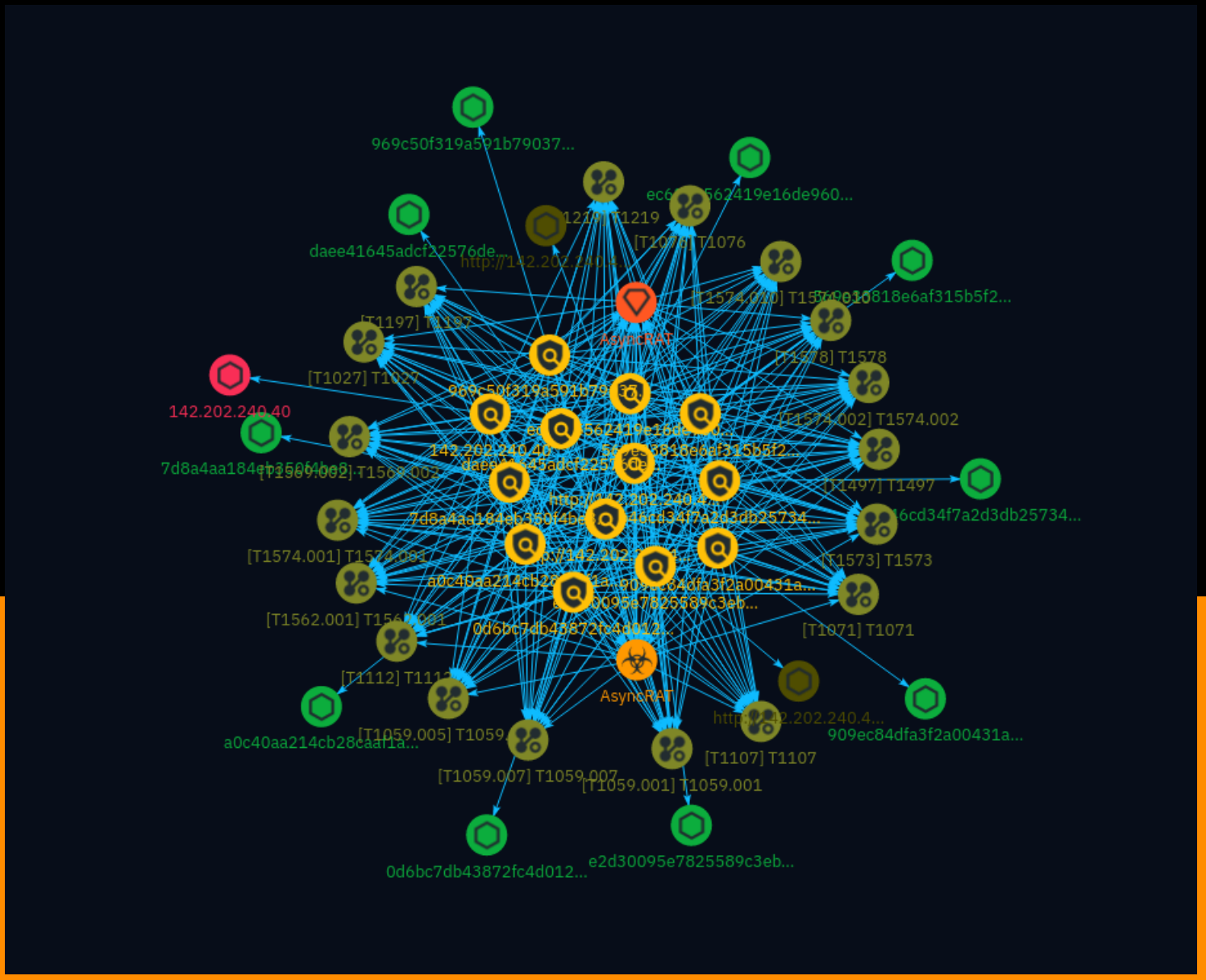

Table of contents

Overview

Entities

Observables

External References

Overview

Description

This intelligence report details a sophisticated infection chain that culminates in the deployment of AsyncRAT, a potent malware designed to breach computer systems and steal confidential data. The meticulous analysis unravels the intricate sequence, commencing with a spam email containing a malicious HTML attachment that triggers the download of various file types, including Windows Script Files, Visual Basic Scripts, and PowerShell scripts. These files work in tandem to bypass security mechanisms, establish persistence, and ultimately lead to the injection of the AsyncRAT payload into the aspnet_compiler.exe process.

Confidence

This value represents the confidence in the correctness of the data contained within this report.

100 / 100

Content

N/A

Indicator

Name

http://142.202.240.40:222/r.jpg

Description

- **Unsafe:** False - **Server:** N/A - **Domain Rank:** 0 - **DNS Valid:** False - **Parking:** False - **Spamming:** False - **Malware:** False - **Phishing:** False - **Suspicious:** True - **Adult:** False - **Category:** N/A - **Domain Age:** {'human': 'N/A', 'timestamp': None, 'iso': None} - **IPQS: Domain:** 142.202.240.40 - **IPQS: IP Address:** N/A

Pattern Type

stix

Pattern

[url:value = 'http://142.202.240.40:222/r.jpg']

Name

http://142.202.240.40:222/1.txt

Description

- **Unsafe:** False - **Server:** N/A - **Domain Rank:** 0 - **DNS Valid:** False - **Parking:** False - **Spamming:** False - **Malware:** False - **Phishing:** False - **Suspicious:** True - **Adult:** False - **Category:** N/A - **Domain Age:** {'human': 'N/

A', 'timestamp': None, 'iso': None} - **IPQS: Domain:** 142.202.240.40 - **IPQS: IP Address:** N/A

Pattern Type

stix

Pattern

[url:value = 'http://142.202.240.40:222/1.txt']

Name

ec6805562419e16de9609e2a210464d58801c8b8be964f876cf062e4ab52681a

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'ec6805562419e16de9609e2a210464d58801c8b8be964f876cf062e4ab52681a']

Name

e2d30095e7825589c3ebd198f31e4c24e213d9f43fc3bb1ab2cf06b70c6eac1d

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'e2d30095e7825589c3ebd198f31e4c24e213d9f43fc3bb1ab2cf06b70c6eac1d']

Name

daee41645adcf22576def12cb42576a07ed5f181a71d3f241c2c14271aad308b

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'daee41645adcf22576def12cb42576a07ed5f181a71d3f241c2c14271aad308b']

Name

b46cd34f7a2d3db257343501fe47bdab67e796700f150b8c51a28bb30650c28f

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'b46cd34f7a2d3db257343501fe47bdab67e796700f150b8c51a28bb30650c28f']

Name

a0c40aa214cb28caaf1a2f5db136bb079780f05cba50e84bbaeed101f0de7fb3

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'a0c40aa214cb28caaf1a2f5db136bb079780f05cba50e84bbaeed101f0de7fb3']

Name

969c50f319a591b79037ca50cda55a1bcf2c4284e6ea090a68210039034211db

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '969c50f319a591b79037ca50cda55a1bcf2c4284e6ea090a68210039034211db']

Name

909ec84dfa3f2a00431a20d4b8a241f2959cac2ea402692fd46f4b7dbf247e90

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '909ec84dfa3f2a00431a20d4b8a241f2959cac2ea402692fd46f4b7dbf247e90']

Name

7d8a4aa184eb350f4be8706afb0d7527fca40c4667ab0491217b9e1e9d0f9c81

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '7d8a4aa184eb350f4be8706afb0d7527fca40c4667ab0491217b9e1e9d0f9c81']

Name

569e33818e6af315b5f290442f9e27dc6c56a25259d9c9866b2ffb4176d07103

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '569e33818e6af315b5f290442f9e27dc6c56a25259d9c9866b2ffb4176d07103']

Name

0d6bc7db43872fc4d012124447d3d050b123200b720d305324ec7631f739d98d

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '0d6bc7db43872fc4d012124447d3d050b123200b720d305324ec7631f739d98d']

Name

142.202.240.40

Description

ISP: 1GSERVERS, LLC **OS:** Windows (build 10.0.17763) ---------- Services: **445:** ``` SMB Status: Authentication: enabled SMB Version: 2 Capabilities: raw-mode ``` ---------- **3389:** ``` Remote Desktop Protocol \x03\x00\x00\x13\x0e\xd0\x00\x00\x124\x00\x02\x1f\x08\x00\x02\x00\x00\x00 Remote Desktop Protocol NTLM Info: OS: Windows 10 (version 1809)/Windows Server 2019 (version 1809) OS Build: 10.0.17763 Target Name: WIN-9DVAMKIGFGL NetBIOS Domain Name: WIN-9DVAMKIGFGL NetBIOS Computer Name: WIN-9DVAMKIGFGL DNS Domain Name: WIN-9DVAMKIGFGL FQDN: WIN-9DVAMKIGFGL ; Administrator SES ``` ---------- **5357:** ``` HTTP/1.1 503 Service Unavailable Content-Type: text/html; charset=us-ascii Server: Microsoft-HTTPAPI/2.0 Date: Sun, 28 Apr 2024 04:30:32 GMT Connection: close Content-Length: 326 ``` ---------- **5985:** ``` HTTP/1.1 404 Not Found Content-Type: text/html; charset=us-ascii Server: Microsoft-HTTPAPI/2.0 Date: Wed, 24 Apr 2024 09:32:12 GMT Connection: close Content-Length: 315 WinRM NTLM Info: OS: Windows Server 2019 (version 1809) OS Build: 10.0.17763 Target Name: WIN-9DVAMKIGFGL NetBIOS Domain Name: WIN-9DVAMKIGFGL NetBIOS Computer Name: WIN-9DVAMKIGFGL DNS Domain Name: WIN-9DVAMKIGFGL FQDN: WIN-9DVAMKIGFGL ``` ----------

Pattern Type

stix

Pattern

[ipv4-addr:value = '142.202.240.40']

Attack-Pattern

Name
T1076
ID
T1076
Name
T1107
ID
T1107
Name
T1569.002
ID
T1569.002
Description
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.(Citation: Microsoft Service Control Manager) The service control manager is accessible to users via GUI components as well as system utilities such as `sc.exe` and [Net](https://attack.mitre.org/software/S0039). [PsExec](https://attack.mitre.org/software/S0029) can also be used to execute commands or payloads via a temporary Windows service created through the service control manager API.(Citation: Russinovich Sysinternals) Tools such as [PsExec](https://attack.mitre.org/software/S0029) and `sc.exe` can accept remote servers as arguments and may be used to conduct remote execution. Adversaries may leverage these mechanisms to execute malicious content. This can be done by either executing a new or modified service. This technique is the execution used in conjunction with [Windows Service](https://attack.mitre.org/techniques/T1543/003) during service persistence or privilege escalation.

Name

T1574.002

ID

T1574.002

Description

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to [DLL Search Order Hijacking](https://attack.mitre.org/techniques/T1574/001), side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s). Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.(Citation: FireEye DLL Side-Loading)

Name

T1059.005

ID

T1059.005

Description

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](https://attack.mitre.org/techniques/T1559/001) and the [Native API](https://attack.mitre.org/techniques/T1106) 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.(Citation: VB .NET Mar 2020)(Citation: VB Microsoft) Derivative languages based on VB have also been created, such as Visual Basic for Applications (VBA) and VBScript. VBA is an event-driven programming language built into Microsoft Office, as well as several third-party applications.(Citation: Microsoft VBA) (Citation: Wikipedia VBA) VBA enables documents to contain macros used to automate the execution of tasks and other functionality on the host. VBScript is a default scripting language on Windows hosts and can also be used in place of [JavaScript](https://attack.mitre.org/techniques/T1059/007) on HTML Application (HTA) webpages served to Internet Explorer (though most modern browsers do not come with VBScript support). (Citation: Microsoft VBScript) Adversaries may use VB payloads to execute malicious commands. Common malicious usage includes automating execution of behaviors with VBScript or embedding VBA content into [Spearphishing Attachment](https://attack.mitre.org/techniques/T1566/001) payloads (which may also involve [Mark-of-the-Web Bypass](https://attack.mitre.org/techniques/T1553/005) to enable execution).(Citation: Default VBS macros Blocking)

Name

T1197

ID

T1197

Description

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](https://attack.mitre.org/techniques/T1559/001) (COM).(Citation: Microsoft COM) (Citation: Microsoft BITS) 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. The interface to create and manage BITS jobs is accessible through [PowerShell](https://attack.mitre.org/techniques/T1059/001) and the [BITSAdmin](https://attack.mitre.org/software/S0190) tool. (Citation: Microsoft BITS)(Citation: Microsoft BITSAdmin) Adversaries may abuse BITS to download (e.g. [Ingress Tool Transfer](https://attack.mitre.org/techniques/T1105)), execute, and even clean up after running malicious code (e.g. [Indicator Removal](https://attack.mitre.org/techniques/T1070)). BITS tasks are self-contained in the BITS job database, without new files or registry modifications, and often permitted by host firewalls.(Citation: CTU BITS Malware June 2016)(Citation: Mondok Windows PiggyBack BITS May 2007)(Citation: Symantec BITS May 2007) BITS enabled execution may also enable persistence by creating long-standing jobs (the default maximum lifetime is 90 days and extendable) or invoking an arbitrary program when a job completes or errors (including after system reboots). (Citation: PaloAlto UBoatRAT Nov 2017)(Citation: CTU BITS Malware June 2016) BITS upload functionalities can also be used to perform [Exfiltration Over Alternative Protocol](https://attack.mitre.org/techniques/T1048).(Citation: CTU BITS Malware June 2016)

Name

T1573

ID

T1573

Description

Adversaries may employ an encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Despite the use of a secure algorithm, these implementations may be vulnerable to reverse engineering if secret keys are encoded and/or generated within malware samples/configuration files.

Name

T1562.001

ID

T1562.001

Description

Adversaries may modify and/or disable security tools to avoid possible detection of their malware/tools and activities. This may take many forms, such as killing security software processes or services, modifying / deleting Registry keys or configuration files so that tools do not operate properly, or other methods to interfere with security tools scanning or reporting information. Adversaries may also disable updates to prevent the latest security patches from reaching tools on victim systems.(Citation: SCADAfence_ransomware) Adversaries may also tamper with artifacts deployed and utilized by security tools. Security tools may make dynamic changes to system components in order to maintain visibility into specific events. For example, security products may load their own modules and/or modify those loaded by processes to facilitate data collection. Similar to [Indicator Blocking](https://attack.mitre.org/techniques/T1562/006), adversaries may unhook or otherwise modify these features added by tools (especially those that exist in userland or are otherwise potentially accessible to adversaries) to avoid detection.(Citation: OutFlank System Calls)(Citation: MDSec System Calls) Adversaries may also focus on specific applications such as Sysmon. For example, the “Start” and “Enable” values in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\EventLog-Microsoft-Windows-Sysmon-Operational` may be modified to tamper with and potentially disable Sysmon logging.(Citation: disable_win_evt_logging) On network devices, adversaries may attempt to skip digital signature verification checks by altering startup configuration files and effectively disabling firmware verification that typically occurs at boot.(Citation: Fortinet Zero-Day and Custom Malware Used by Suspected Chinese Actor in Espionage Operation)(Citation: Analysis of FG-IR-22-369) In cloud environments, tools disabled by adversaries may include cloud monitoring agents that report back to services such as AWS CloudWatch or Google Cloud Monitor. Furthermore, although defensive tools may have anti-tampering mechanisms, adversaries may abuse tools such as legitimate rootkit removal kits to impair and/or disable these tools.(Citation: chasing_avaddon_ransomware)(Citation: dharma_ransomware)(Citation: demystifying_ryuk)(Citation: doppelpaymer_crowdstrike) For example, adversaries have used tools such as GMER to find and shut down hidden processes and antivirus software on infected systems.(Citation: demystifying_ryuk) Additionally, adversaries may exploit legitimate drivers from anti-virus software to gain access to kernel space (i.e. [Exploitation for Privilege Escalation](https://attack.mitre.org/techniques/T1068)), which may lead to bypassing anti-tampering features.(Citation: avoslocker_ransomware)

Name

T1574.010

ID

T1574.010

Description

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. Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

Name

T1059.001

ID

T1059.001

Description

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.(Citation: TechNet PowerShell) 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). PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk. A number of PowerShell-based offensive testing tools are available, including [Empire](https://attack.mitre.org/software/S0363), [PowerSploit](https://attack.mitre.org/software/S0194),

[PoshC2](https://attack.mitre.org/software/S0378), and PSAttack.(Citation: Github PSAttack) PowerShell commands/scripts can also be executed without directly invoking the `powershell.exe` binary through interfaces to PowerShell's underlying `System.Management.Automation` assembly DLL exposed through the .NET framework and Windows Common Language Interface (CLI).(Citation: Sixdub PowerPick Jan 2016)(Citation: SilentBreak Offensive PS Dec 2015)(Citation: Microsoft PSfromCsharp APR 2014)

Name

T1027

ID

T1027

Description

Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behavior that can be used across different platforms and the network to evade defenses. Payloads may be compressed, archived, or encrypted in order to avoid detection. These payloads may be used during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open and [Deobfuscate/Decode Files or Information](https://attack.mitre.org/techniques/T1140) for [User Execution](https://attack.mitre.org/techniques/T1204). The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the adversary. (Citation: Volexity PowerDuke November 2016) Adversaries may also use compressed or archived scripts, such as JavaScript. Portions of files can also be encoded to hide the plain-text strings that would otherwise help defenders with discovery. (Citation: Linux/Cdorked.A We Live Security Analysis) Payloads may also be split into separate, seemingly benign files that only reveal malicious functionality when reassembled. (Citation: Carbon Black Obfuscation Sept 2016) Adversaries may also abuse [Command Obfuscation](https://attack.mitre.org/techniques/T1027/010) to obscure commands executed from payloads or directly via [Command and Scripting Interpreter](https://attack.mitre.org/techniques/T1059). Environment variables, aliases, characters, and other platform/language specific semantics can be used to evade signature based detections and application control mechanisms. (Citation: FireEye Obfuscation June 2017) (Citation: FireEye Revoke-Obfuscation July 2017)(Citation: PaloAlto EncodedCommand March 2017)

Name

T1497

ID

T1497

Description

Adversaries may employ various means to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from [Virtualization/Sandbox Evasion](https://attack.mitre.org/techniques/T1497) during automated discovery to shape follow-on behaviors.(Citation: Deloitte Environment Awareness) Adversaries may use several methods to accomplish [Virtualization/Sandbox Evasion](https://attack.mitre.org/techniques/T1497) such as checking for security monitoring tools (e.g., Sysinternals, Wireshark, etc.) or other system artifacts associated with analysis or virtualization. Adversaries may also check for legitimate user activity to help determine if it is in an analysis environment. Additional methods include use of sleep timers or loops within malware code to avoid operating within a temporary sandbox. (Citation: Unit 42 Pirpi July 2015)

Name

T1112

ID

T1112

Description

Adversaries may interact with the Windows Registry to hide configuration information within Registry keys, remove information as part of cleaning up, or as part of other techniques to aid in persistence and execution. Access to specific areas of the Registry depends on account permissions, some requiring administrator-level access. The built-in

Windows command-line utility [Reg](https://attack.mitre.org/software/S0075) may be used for local or remote Registry modification. (Citation: Microsoft Reg) Other tools may also be used, such as a remote access tool, which may contain functionality to interact with the Registry through the Windows API. Registry modifications may also include actions to hide keys, such as prepending key names with a null character, which will cause an error and/or be ignored when read via [Reg](https://attack.mitre.org/software/S0075) or other utilities using the Win32 API. (Citation: Microsoft Reghide NOV 2006) Adversaries may abuse these pseudo-hidden keys to conceal payloads/commands used to maintain persistence. (Citation: TrendMicro POWELIKS AUG 2014) (Citation: SpectorOps Hiding Reg Jul 2017) The Registry of a remote system may be modified to aid in execution of files as part of lateral movement. It requires the remote Registry service to be running on the target system. (Citation: Microsoft Remote) Often [Valid Accounts](https://attack.mitre.org/techniques/T1078) are required, along with access to the remote system's [SMB/Windows Admin Shares](https://attack.mitre.org/techniques/T1021/002) for RPC communication.

Name

T1219

ID

T1219

Description

An adversary may use legitimate desktop support and remote access software to establish an interactive command and control channel to target systems within networks. These services, such as `VNC`, `Team Viewer`, `AnyDesk`, `ScreenConnect`, `LogMein`, `AmmyyAdmin`, and other remote monitoring and management (RMM) tools, are commonly used as legitimate technical support software and may be allowed by application control within a target environment.(Citation: Symantec Living off the Land) (Citation: CrowdStrike 2015 Global Threat Report)(Citation: CrySyS Blog TeamSpy) Remote access software may be installed and used post-compromise as an alternate communications channel for redundant access or as a way to establish an interactive remote desktop session with the target system. They may also be used as a component of malware to establish a reverse connection or back-connect to a service or adversary-controlled system. Adversaries may similarly abuse response features included in EDR and other defensive tools that enable remote access. Installation of many remote access software may also include persistence (e.g., the software's installation routine creates a [Windows Service](https://attack.mitre.org/techniques/T1543/003)). Remote access modules/features may also exist as part of otherwise existing software (e.g., Google

Chrome's Remote Desktop).(Citation: Google Chrome Remote Desktop)(Citation: Chrome Remote Desktop)

Name

T1071

ID

T1071

Description

Adversaries may communicate using OSI application layer protocols to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server. Adversaries may utilize many different protocols, including those used for web browsing, transferring files, electronic mail, or DNS. For connections that occur internally within an enclave (such as those between a proxy or pivot node and other nodes), commonly used protocols are SMB, SSH, or RDP.(Citation: Mandiant APT29 Eye Spy Email Nov 22)

Name

T1574.001

ID

T1574.001

Description

Adversaries may execute their own malicious payloads by hijacking the search order used to load DLLs. Windows systems use a common method to look for required DLLs to load into a program. (Citation: Microsoft Dynamic Link Library Search Order)(Citation: FireEye Hijacking July 2010) Hijacking DLL loads may be for the purpose of establishing persistence as well as elevating privileges and/or evading restrictions on file execution. There are many ways an adversary can hijack DLL loads. Adversaries may plant trojan dynamic-link

library files (DLLs) in a directory that will be searched before the location of a legitimate library that will be requested by a program, causing Windows to load their malicious library when it is called for by the victim program. Adversaries may also perform DLL preloading, also called binary planting attacks, (Citation: OWASP Binary Planting) by placing a malicious DLL with the same name as an ambiguously specified DLL in a location that Windows searches before the legitimate DLL. Often this location is the current working directory of the program.(Citation: FireEye fxsst June 2011) Remote DLL preloading attacks occur when a program sets its current directory to a remote location such as a Web share before loading a DLL. (Citation: Microsoft Security Advisory 2269637) Phantom DLL hijacking is a specific type of DLL search order hijacking where adversaries target references to non-existent DLL files.(Citation: Adversaries Hijack DLLs) They may be able to load their own malicious DLL by planting it with the correct name in the location of the missing module. Adversaries may also directly modify the search order via DLL redirection, which after being enabled (in the Registry and creation of a redirection file) may cause a program to load a different DLL.(Citation: Microsoft Dynamic-Link Library Redirection) (Citation: Microsoft Manifests)(Citation: FireEye DLL Search Order Hijacking) If a search order-vulnerable program is configured to run at a higher privilege level, then the adversary-controlled DLL that is loaded will also be executed at the higher level. In this case, the technique could be used for privilege escalation from user to administrator or SYSTEM or from administrator to SYSTEM, depending on the program. Programs that fall victim to path hijacking may appear to behave normally because malicious DLLs may be configured to also load the legitimate DLLs they were meant to replace.

Name

T1578

ID

T1578

Description

An adversary may attempt to modify a cloud account's compute service infrastructure to evade defenses. A modification to the compute service infrastructure can include the creation, deletion, or modification of one or more components such as compute instances, virtual machines, and snapshots. Permissions gained from the modification of infrastructure components may bypass restrictions that prevent access to existing infrastructure. Modifying infrastructure components may also allow an adversary to evade detection and remove evidence of their presence.(Citation: Mandiant M-Trends 2020)

Name

T1059.007

ID

T1059.007

Description

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.(Citation: NodeJS) JScript is the Microsoft implementation of the same scripting standard. JScript is interpreted via the Windows Script engine and thus integrated with many components of Windows such as the [Component Object Model](https://attack.mitre.org/techniques/T1559/001) and Internet Explorer HTML Application (HTA) pages.(Citation: JScrip May 2018)(Citation: Microsoft JScript 2007)(Citation: Microsoft Windows Scripts) JavaScript for Automation (JXA) is a macOS scripting language based on JavaScript, included as part of Apple's Open Scripting Architecture (OSA), that was introduced in OSX 10.10. Apple's OSA provides scripting capabilities to control applications, interface with the operating system, and bridge access into the rest of Apple's internal APIs. As of OSX 10.10, OSA only supports two languages, JXA and [AppleScript](https://attack.mitre.org/techniques/T1059/002). Scripts can be executed via the command line utility `osascript`, they can be compiled into applications or script files via `osacompile`, and they can be compiled and executed in memory of other programs by leveraging the OSAKit Framework.(Citation: Apple About Mac Scripting 2016) (Citation: SpecterOps JXA 2020)(Citation: SentinelOne macOS Red Team)(Citation: Red Canary Silver Sparrow Feb2021)(Citation: MDSec macOS JXA and VSCode) Adversaries may abuse various implementations of JavaScript to execute various behaviors. Common uses include hosting malicious scripts on websites as part of a [Drive-by Compromise](https://attack.mitre.org/techniques/T1189) or downloading and executing these script files as secondary payloads. Since these payloads are text-based, it is also very common for adversaries to obfuscate their content as part of [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027).

Intrusion-Set

Name
AsyncRAT

Malware

Name
AsyncRAT

Url

Value
http://142.202.240.40:222/r.jpg
http://142.202.240.40:222/1.txt

StixFile

Value
ec6805562419e16de9609e2a210464d58801c8b8be964f876cf062e4ab52681a
e2d30095e7825589c3ebd198f31e4c24e213d9f43fc3bb1ab2cf06b70c6eac1d
daee41645adcf22576def12cb42576a07ed5f181a71d3f241c2c14271aad308b
b46cd34f7a2d3db257343501fe47bdab67e796700f150b8c51a28bb30650c28f
a0c40aa214cb28caaf1a2f5db136bb079780f05cba50e84bbaeed101f0de7fb3
969c50f319a591b79037ca50cda55a1bcf2c4284e6ea090a68210039034211db
909ec84dfa3f2a00431a20d4b8a241f2959cac2ea402692fd46f4b7dbf247e90
7d8a4aa184eb350f4be8706afb0d7527fca40c4667ab0491217b9e1e9d0f9c81
569e33818e6af315b5f290442f9e27dc6c56a25259d9c9866b2ffb4176d07103
0d6bc7db43872fc4d012124447d3d050b123200b720d305324ec7631f739d98d

IPv4-Addr

Value
142.202.240.40

External References

- https://www.mcafee.com/blogs/other-blogs/mcafee-labs/from-spam-to-asyncrat-tracking-the-surge-in-non-pe-cyber-threats/
- https://otx.alienvault.com/pulse/663ce5f0acaf07b1b1283e34