May 02 2024

NETMANAGEIT

Intelligence Report

The Darkgate Menace: Leveraging Autohotkey & Attempt to Evade Smartscreen

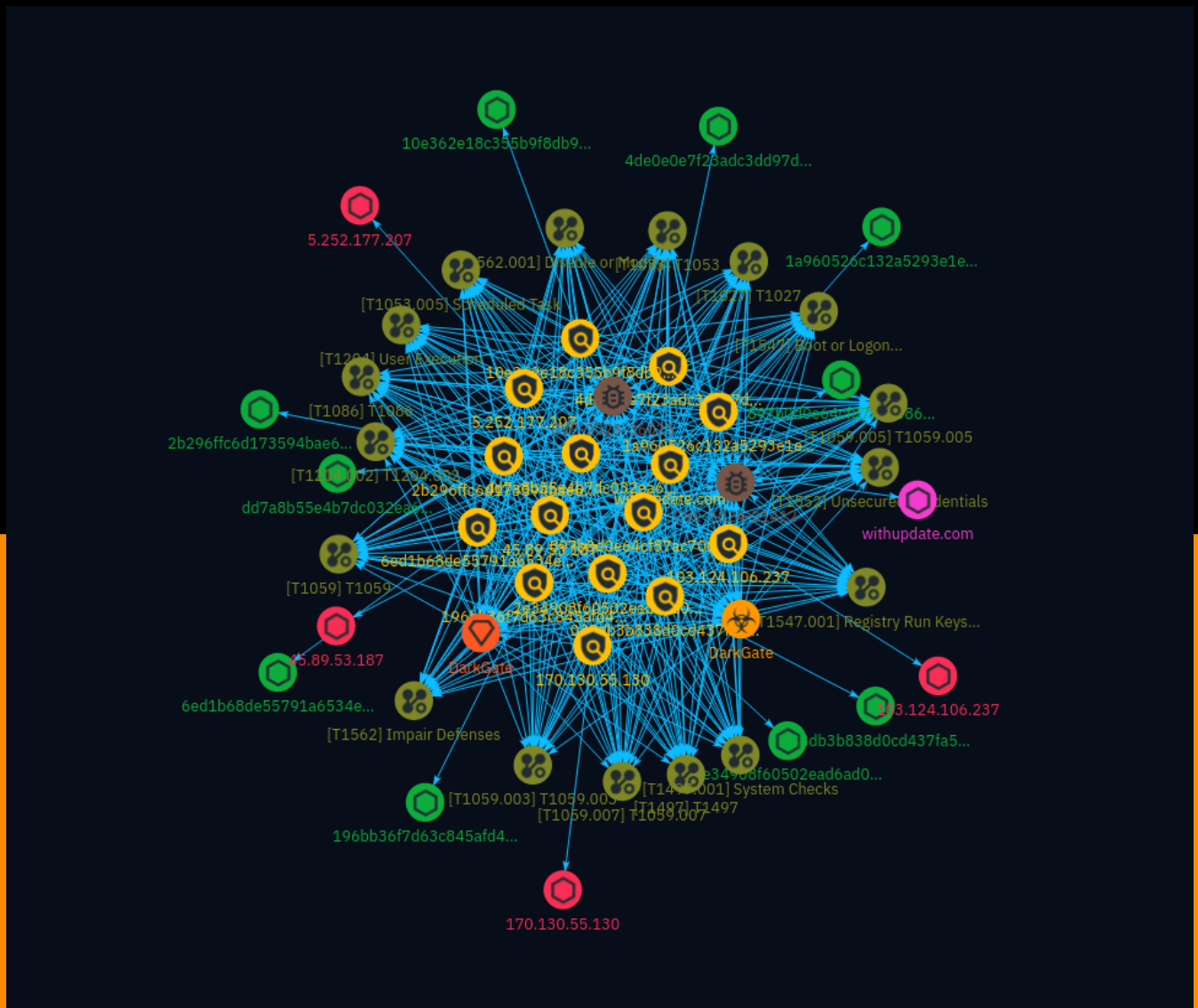

Table of contents

Overview

Entities

Observables

External References

Overview

Description

This report details a novel infection chain associated with DarkGate malware, a Remote Access Trojan (RAT) that exploits the AutoHotkey utility and attempts to bypass Microsoft Defender SmartScreen. The infection begins with an HTML-based entry point or an XLS file, utilizing techniques such as disguising malicious content as legitimate files. The attack chain involves downloading and executing various components, including VBScript, PowerShell scripts, and AutoHotkey scripts, ultimately leading to the execution of the DarkGate payload. The report also highlights the vulnerability CVE-2023-36025 and its exploitation to evade SmartScreen warnings, as well as persistence mechanisms employed by the malware.

Confidence

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

100 / 100

Content

N/A

Indicator

Name

withupdate.com

Pattern Type

stix

Pattern

[domain-name:value = 'withupdate.com']

Name

45.89.53.187

Description

ISP: STARK INDUSTRIES SOLUTIONS LTD **OS:** - ---------- Services: **22:** ``` SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2 Key type: ssh-rsa Key: AAAAB3NzaC1yc2EAAAADAQABAAABAQDFuaopLyw4Wg6jVUbceh9Hcy0IT3Pf5CGAFNK28TCGvK rX Bs0jaDK22WY/ bldmzd4TfIJkkkKm8YeyXRd8C6M0+oxXhK5VM8m61pG4YBnRpS9PAIXAfDFeiTtc 9qoLSOl867RQEZXBRe9dgFzCLMm06BwKFbG92QFqP6gwnmyBS6jylMEP8pI2iCX1ug2P3TKpDzg a yh28REhXvaUIAQ1Pn/ZPwvid+2MjgVSdz1roN952FD/tWjOPO8DDlz/ eKuUFR6qpHVBv532MxC0X G4FR1pFu3qB/ vBPg1oyOT2yzluGNzl24W4NKeU3n+NLWdILvlM0oaWJphS1rIPUn1GjD Fingerprint: 8d:a9:1a:ac: 0b:c8:56:ad:bb:33:d8:64:a6:25:d8:74 Kex Algorithms: curve25519-sha256 curve25519-sha256@libssh.org ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-sha2-nistp521 diffie-

hellman-group-exchange-sha256 diffie-hellman-group16-sha512 diffie-hellman-group18-sha512 diffie-hellman-group14-sha256 diffie-hellman-group14-sha1 Server Host Key Algorithms: rsa-sha2-512 rsa-sha2-256 ssh-rsa ecdsa-sha2-nistp256 ssh-ed25519 Encryption Algorithms: chacha20-poly1305@openssh.com aes128-ctr aes192-ctr aes256-ctr aes128-gcm@openssh.com aes256-gcm@openssh.com MAC Algorithms: umac-64-etm@openssh.com umac-128-etm@openssh.com hmac-sha2-256-etm@openssh.com hmac-sha2-512-etm@openssh.com hmac-sha1-etm@openssh.com umac-64@openssh.com umac-128@openssh.com hmac-sha2-256 hmac-sha2-512 hmac-sha1 Compression Algorithms: none zlib@openssh.com ``` ---------- **80:** ``` HTTP/1.1 200 OK Date: Mon, 29 Apr 2024 12:46:32 GMT Server: Apache/2.4.38 (Debian) Vary: Accept-Encoding Content-Length: 745 Content-Type: text/html;charset=UTF-8 ``` ---------- **123:** ``` NTP protocolversion: 3 stratum: 3 leap: 0 precision: -24 rootdelay: 0.00932312011719 rootdisp: 0.0400390625 refid: 396206248 reftime: 392290742.75 poll: 3 ``` ---------- **137:** ``` NetBIOS Response: Server Name: VM2338426 MAC Address: 00:00:00:00:00:00 Names: VM2338426 <0x0> VM2338426 <0x3> VM2338426 <0x20> \x01\x02__MSBROWSE__\x02 <0x1> WORKGROUP <0x0> WORKGROUP <0x1d> WORKGROUP <0x1e> ``` ---------- **445:** ``` SMB Status: Authentication: disabled SMB Version: 1 OS: Windows 6.1 Software: Samba 4.9.5-Debian Capabilities: dfs, extended-security, infolevel-passthru, large-files, large-readx, large-writex, level2-oplocks, lock-and-read, nt-find, nt-smb, nt-status, raw-mode, rpc-remote-api, unicode, unix Shares Name Type Comments ---------- ---------- ---------- s Disk IPC$ IPC IPC Service (Samba 4.9.5-Debian) ``` ----------

Pattern Type

stix

Pattern

[ipv4-addr:value = '45.89.53.187']

Name

dd7a8b55e4b7dc032ea6d6aed6153bec9b5b68b45369e877bb66ba21acc81455

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'dd7a8b55e4b7dc032ea6d6aed6153bec9b5b68b45369e877bb66ba21acc81455']

Name

897b0d0e64cf87ac7086241c86f757f3c94d6826f949a1f0fec9c40892c0cecb

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '897b0d0e64cf87ac7086241c86f757f3c94d6826f949a1f0fec9c40892c0cecb']

Name

6ed1b68de55791a6534ea96e721ff6a5662f2aefff471929d23638f854a80031

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '6ed1b68de55791a6534ea96e721ff6a5662f2aefff471929d23638f854a80031']

Name

4de0e0e7f23adc3dd97d498540bd8283004aa131a59ae319019ade9ddef41795

Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '4de0e0e7f23adc3dd97d498540bd8283004aa131a59ae319019ade9ddef41795']
Name
2e34908f60502ead6ad08af1554c305b88741d09e36b2c24d85fd9bac4a11d2f
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '2e34908f60502ead6ad08af1554c305b88741d09e36b2c24d85fd9bac4a11d2f']
Name
2b296ffc6d173594bae63d37e2831ba21a59ce385b87503710dc9ca439ed7833
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '2b296ffc6d173594bae63d37e2831ba21a59ce385b87503710dc9ca439ed7833']

Name

1a960526c132a5293e1e02b49f43df1383bf37a0bbadd7ba7c106375c418dad4

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '1a960526c132a5293e1e02b49f43df1383bf37a0bbadd7ba7c106375c418dad4']

Name

196bb36f7d63c845afd40c5c17ce061e320d110f28ebe8c7c998b9e6b3fe1005

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '196bb36f7d63c845afd40c5c17ce061e320d110f28ebe8c7c998b9e6b3fe1005']

Name

10e362e18c355b9f8db9a0dbbc75cf04649606ef96743c759f03508b514ad34e

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '10e362e18c355b9f8db9a0dbbc75cf04649606ef96743c759f03508b514ad34e']

Name

038db3b838d0cd437fa530c001c9913a1320d1d7ac0fd3b35d974a806735c907

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '038db3b838d0cd437fa530c001c9913a1320d1d7ac0fd3b35d974a806735c907']

Name

5.252.177.207

Description

ISP: MivoCloud SRL **OS:** Linux ---------- Services: **22:** ``` SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u2 Key type: ecdsa-sha2-nistp256 Key: AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNKs7B28wSeKIyIJp0VY80dZ33RJ6Gdvska39FbkoQMLXSgVKy1Rx9fbou1gLLg62Rq0JlooWYN0JtuL0CyQMdQ= Fingerprint: 17:bb:a7:a1:88:25:64:bb:ab:9c:2c:f1:b1:f7:34:b1 Kex Algorithms: curve25519-sha256@libssh.org ecdh-sha2-nistp521 ecdh-sha2-nistp384 ecdh-sha2-nistp256 diffie-hellman-group-exchange-sha256 kex-strict-s-v00@openssh.com Server Host Key Algorithms: rsa-sha2-512 rsa-sha2-256 ecdsa-sha2-nistp256 ssh-ed25519 Encryption Algorithms: chacha20-poly1305@openssh.com aes256-gcm@openssh.com aes128-gcm@openssh.com aes256-ctr aes192-ctr aes128-ctr MAC Algorithms: hmac-sha2-512-etm@openssh.com hmac-sha2-256-etm@openssh.com umac-128-etm@openssh.com hmac-sha2-512 hmac-sha2-256 umac-128@openssh.com Compression Algorithms: none zlib@openssh.com ``` ---------- **80:** ``` HTTP/1.1 200 OK Connection: close Content-Type: text/html; charset=utf-8 Content-Length: 39 Date: Sun, 21 Apr 2024 13:20:35 GMT ``` ----------

Pattern Type

stix

Pattern

[ipv4-addr:value = '5.252.177.207']

Name

103.124.106.237

Pattern Type

stix

Pattern

[ipv4-addr:value = '103.124.106.237']

Name

170.130.55.130

Description

ISP: Eonix Corporation **OS:** - ---------- Services: **22:** ``` SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u3 Key type: ssh-rsa Key: AAAAB3NzaC1yc2EAAAADAQABAAABgQC2t0N9sfbnScj+ZtoHff9bAgLBlW0CmbKuXe1rbhKEXmCK jxdLaZw1A2DPT/t3I7PTn/IeRkf5QPma03pyTMDJ6rTZOKlS0bkezj3SLuq3PSaWXJAvGK+hdMXo k4vQLu3PMdQpxH7P1dNeapIikNHhED7w4VJR2ZhUP/WjvxJ9/LnsEo9vUpvu5d5J3gO3EiC3urpR wWoqsHMlMs6mYz1WHblvBhq8KHIdOwTcCsy5TH1b/ Zf+BPqD4kPlFtwK79NXnB3SRgOcxjn0gkU5 WvmLNiUxqgDZ32G9GOu524cs6hIcIXJkoQYGBT7k7XY2w7uHD6VSJigADR4r5wdRwZejHXvz4QuK DBjCunrK9M/ PheTyD8xtbSWPwGrpnXeCSp0FNsYmA3AVRkwgSwbydi2GOrZbFcvUB5Bl5MvU0UIq u6evvkSD4niAXQpKXu9VaTd7VPdICZs0NH/pzTwYG1jkKDFkkKR3WhhODXudFTiXPQ00y6y9rQzv

0jNWpBG3LqE= Fingerprint: 71:5f:b0:55:54:30:73:ed:1e:f2:5e:53:56:15:68:0b Kex Algorithms: curve25519-sha256 curve25519-sha256@libssh.org ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-sha2-nistp521 diffie-hellman-group-exchange-sha256 diffie-hellman-group16-sha512 diffie-hellman-group18-sha512 diffie-hellman-group14-sha256 kex-strict-s-v00@openssh.com Server Host Key Algorithms: rsa-sha2-512 rsa-sha2-256 ssh-rsa ecdsa-sha2-nistp256 ssh-ed25519 Encryption Algorithms: chacha20-poly1305@openssh.com aes128-ctr aes192-ctr aes256-ctr aes128-gcm@openssh.com aes256-gcm@openssh.com MAC Algorithms: umac-64-etm@openssh.com umac-128-etm@openssh.com hmac-sha2-256-etm@openssh.com hmac-sha2-512-etm@openssh.com hmac-sha1-etm@openssh.com umac-64@openssh.com umac-128@openssh.com hmac-sha2-256 hmac-sha2-512 hmac-sha1 Compression Algorithms: none zlib@openssh.com ``` ---------- **80:** ``` HTTP/1.1 200 OK Date: Tue, 27 Feb 2024 02:20:55 GMT Server: Apache/2.4.52 (Ubuntu) Last-Modified: Thu, 18 Jan 2024 14:02:16 GMT ETag: "29af-60f38cfa7d44c" Accept-Ranges: bytes Content-Length: 10671 Vary: Accept-Encoding Content-Type: text/html ``` ---------- **445:** ``` SMB Status: Authentication: disabled SMB Version: 2 Capabilities: raw-mode Shares Name Type Comments ---------- share Disk IPC$ IPC IPC Service (Samba 4.13.13-Debian) ``` ----------

Pattern Type

stix

Pattern

[ipv4-addr:value = '170.130.55.130']

Vulnerability

Name
CVE-2024-21412
Description
Microsoft Windows Internet Shortcut Files contains an unspecified vulnerability that allows for a security feature bypass.
Name
CVE-2023-36025
Description
Microsoft Windows SmartScreen contains a security feature bypass vulnerability that could allow an attacker to bypass Windows Defender SmartScreen checks and their associated prompts.

Attack-Pattern

Name
T1086
ID
T1086
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

T1059.003

ID

T1059.003

Description

Adversaries may abuse the Windows command shell for execution. The Windows command shell ([cmd](https://attack.mitre.org/software/S0106)) 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](https://attack.mitre.org/techniques/T1021) such as [SSH](https://attack.mitre.org/techniques/T1021/004).(Citation: SSH in Windows) Batch files (ex: .bat or .cmd) also provide the shell with a list of sequential commands to run, as well as normal scripting operations such as conditionals and loops. Common uses of batch files include long or repetitive tasks, or the need to run the same set of commands on multiple systems. Adversaries may leverage [cmd](https://attack.mitre.org/software/S0106) to execute various commands and payloads. Common uses include [cmd](https://attack.mitre.org/software/S0106) to execute a single command, or abusing [cmd](https://attack.mitre.org/software/S0106) interactively with input and output forwarded over a command and control channel.

Name

Disable or Modify Tools

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

Registry Run Keys / Startup Folder

ID

T1547.001

Description

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. (Citation: Microsoft Run Key) These programs will be executed under the context of the user and will have the account's associated permissions level. The following run keys are created by default on Windows systems: * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run` * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce` * `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run` * `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce` Run keys may exist under multiple hives.(Citation: Microsoft Wow6432Node 2018)(Citation: Malwarebytes Wow6432Node 2016) The `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx` is also available but is not created by default on Windows Vista and newer. Registry run key entries can reference programs directly or list them as a dependency.(Citation: Microsoft Run Key) For example, it is possible to load a DLL at logon using a "Depend" key with RunOnceEx: `reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v 1 /d "C:\temp\evil[.]dll"` (Citation: Oddvar Moe RunOnceEx Mar 2018) Placing a program within a startup folder will also cause that program to execute when a user logs in. There is a startup folder location for individual user accounts as well as a system-wide startup folder that will be checked regardless of which user account logs in. The startup folder path for the current user is `C:\Users\[Username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`. The startup folder path for all users is `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp`. The following Registry keys can be used to set startup folder items for persistence: * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders` * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders` * `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell

Folders` * `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders` The following Registry keys can control automatic startup of services during boot: * `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce` * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce` * `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices` * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices` Using policy settings to specify startup programs creates corresponding values in either of two Registry keys: * `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run` * `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run` Programs listed in the load value of the registry key `HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows` run automatically for the currently logged-on user. By default, the multistring `BootExecute` value of the registry key `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager` is set to `autocheck autochk *`. This value causes Windows, at startup, to check the file-system integrity of the hard disks if the system has been shut down abnormally. Adversaries can add other programs or processes to this registry value which will automatically launch at boot. Adversaries can use these configuration locations to execute malware, such as remote access tools, to maintain persistence through system reboots. Adversaries may also use [Masquerading](https://attack.mitre.org/techniques/T1036) to make the Registry entries look as if they are associated with legitimate programs.

Name

T1059

ID

T1059

Description

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](https://attack.mitre.org/

techniques/T1059/004) while Windows installations include the [Windows Command Shell](https://attack.mitre.org/techniques/T1059/003) and [PowerShell](https://attack.mitre.org/techniques/T1059/001). There are also cross-platform interpreters such as [Python](https://attack.mitre.org/techniques/T1059/006), as well as those commonly associated with client applications such as [JavaScript](https://attack.mitre.org/techniques/T1059/007) and [Visual Basic](https://attack.mitre.org/techniques/T1059/005). Adversaries may abuse these technologies in various ways as a means of executing arbitrary commands. Commands and scripts can be embedded in [Initial Access](https://attack.mitre.org/tactics/TA0001) payloads delivered to victims as lure documents or as secondary payloads downloaded from an existing C2. Adversaries may also execute commands through interactive terminals/shells, as well as utilize various [Remote Services](https://attack.mitre.org/techniques/T1021) in order to achieve remote Execution. (Citation: Powershell Remote Commands)(Citation: Cisco IOS Software Integrity Assurance - Command History)(Citation: Remote Shell Execution in Python)

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

User Execution

ID

T1204

Description

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](https://attack.mitre.org/techniques/T1566). While [User Execution](https://attack.mitre.org/techniques/T1204) frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after [Internal Spearphishing](https://attack.mitre.org/techniques/T1534). Adversaries may also deceive users into performing actions such as enabling [Remote Access Software](https://attack.mitre.org/techniques/T1219), allowing direct control of the system to the adversary; running malicious JavaScript in their browser, allowing adversaries to [Steal Web Session Cookie](https://attack.mitre.org/techniques/T1539)s; or downloading and executing malware for [User Execution](https://attack.mitre.org/techniques/T1204).(Citation: Talos Roblox Scam 2023)(Citation: Krebs Discord Bookmarks 2023) For example, tech support scams can be facilitated through [Phishing](https://attack.mitre.org/techniques/T1566), vishing, or various forms of user interaction. Adversaries can use a combination of these methods, such as spoofing and promoting toll-free numbers or call centers that are used to direct victims to malicious websites, to deliver and execute payloads containing malware or [Remote Access Software](https://attack.mitre.org/techniques/T1219).(Citation: Telephone Attack Delivery)

Name

Unsecured Credentials

ID

T1552

Description

Adversaries may search compromised systems to find and obtain insecurely stored credentials. These credentials can be stored and/or misplaced in many locations on a system, including plaintext files (e.g. [Bash History](https://attack.mitre.org/techniques/

T1552/003)), operating system or application-specific repositories (e.g. [Credentials in Registry](https://attack.mitre.org/techniques/T1552/002)), or other specialized files/artifacts (e.g. [Private Keys](https://attack.mitre.org/techniques/T1552/004)).(Citation: Brining MimiKatz to Unix)

Name

Impair Defenses

ID

T1562

Description

Adversaries may maliciously modify components of a victim environment in order to hinder or disable defensive mechanisms. This not only involves impairing preventative defenses, such as firewalls and anti-virus, but also detection capabilities that defenders can use to audit activity and identify malicious behavior. This may also span both native defenses as well as supplemental capabilities installed by users and administrators. Adversaries may also impair routine operations that contribute to defensive hygiene, such as blocking users from logging out of a computer or stopping it from being shut down. These restrictions can further enable malicious operations as well as the continued propagation of incidents.(Citation: Emotet shutdown) Adversaries could also target event aggregation and analysis mechanisms, or otherwise disrupt these procedures by altering other system components.

Name

T1053

ID

T1053

Description

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.(Citation: TechNet Task Scheduler Security) Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to [System Binary Proxy Execution](https://attack.mitre.org/techniques/T1218), adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process. (Citation: ProofPoint Serpent)

Name

System Checks

ID

T1497.001

Description

Adversaries may employ various system checks 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) Specific checks will vary based on the target and/or adversary, but may involve behaviors such as [Windows Management Instrumentation](https://attack.mitre.org/techniques/T1047), [PowerShell](https://attack.mitre.org/techniques/T1059/001), [System Information Discovery](https://attack.mitre.org/techniques/T1082), and [Query Registry](https://attack.mitre.org/techniques/T1012) to obtain system information and search for VME artifacts. Adversaries may search for VME artifacts in memory, processes, file system, hardware, and/or the Registry. Adversaries may use scripting to automate these checks into one script and then have the program exit if it

determines the system to be a virtual environment. Checks could include generic system properties such as host/domain name and samples of network traffic. Adversaries may also check the network adapters addresses, CPU core count, and available memory/drive size. Once executed, malware may also use [File and Directory Discovery](https://attack.mitre.org/techniques/T1083) to check if it was saved in a folder or file with unexpected or even analysis-related naming artifacts such as `malware`, `sample`, or `hash`. Other common checks may enumerate services running that are unique to these applications, installed programs on the system, manufacturer/product fields for strings relating to virtual machine applications, and VME-specific hardware/processor instructions.(Citation: McAfee Virtual Jan 2017) In applications like VMWare, adversaries can also use a special I/O port to send commands and receive output. Hardware checks, such as the presence of the fan, temperature, and audio devices, could also be used to gather evidence that can be indicative a virtual environment. Adversaries may also query for specific readings from these devices.(Citation: Unit 42 OilRig Sept 2018)

Name

T1204.002

ID

T1204.002

Description

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](https://attack.mitre.org/techniques/T1566/001). Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl. Adversaries may employ various forms of [Masquerading](https://attack.mitre.org/techniques/T1036) and [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027) to increase the likelihood that a user will open and successfully execute a malicious file. These methods may include using a familiar naming convention and/or password protecting the file and supplying instructions to a user on how to open it.(Citation: Password Protected Word Docs) While [Malicious File](https://attack.mitre.org/techniques/T1204/002) frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it.

This activity may also be seen shortly after [Internal Spearphishing](https://attack.mitre.org/techniques/T1534).

Name

Boot or Logon Autostart Execution

ID

T1547

Description

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.(Citation: Microsoft Run Key)(Citation: MSDN Authentication Packages)(Citation: Microsoft TimeProvider)(Citation: Cylance Reg Persistence Sept 2013)(Citation: Linux Kernel Programming) 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. Since some boot or logon autostart programs run with higher privileges, an adversary may leverage these to elevate privileges.

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).

Name

Scheduled Task

ID

T1053.005

Description

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](https://attack.mitre.org/software/S0111) 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 to create a scheduled task. The deprecated [at](https://attack.mitre.org/software/S0110) utility could also be abused by adversaries (ex:

[At](https://attack.mitre.org/techniques/T1053/002)), though `at.exe` can not access tasks created with `schtasks` or the Control Panel. An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM). Similar to [System Binary Proxy Execution](https://attack.mitre.org/techniques/T1218), adversaries have also abused the Windows Task Scheduler to potentially mask one-time execution under signed/trusted system processes.(Citation: ProofPoint Serpent) Adversaries may also create "hidden" scheduled tasks (i.e. [Hide Artifacts](https://attack.mitre.org/techniques/T1564)) that may not be visible to defender tools and manual queries used to enumerate tasks. Specifically, an adversary may hide a task from `schtasks /query` and the Task Scheduler by deleting the associated Security Descriptor (SD) registry value (where deletion of this value must be completed using SYSTEM permissions).(Citation: SigmaHQ)(Citation: Tarrask scheduled task) Adversaries may also employ alternate methods to hide tasks, such as altering the metadata (e.g., `Index` value) within associated registry keys.(Citation: Defending Against Scheduled Task Attacks in Windows Environments)

Intrusion-Set

Name
DarkGate

Malware

Name
DarkGate

Description
[DarkGate](https://attack.mitre.org/software/S1111) first emerged in 2018 and has evolved into an initial access and data gathering tool associated with various criminal cyber operations. Written in Delphi and named "DarkGate" by its author, [DarkGate](https://attack.mitre.org/software/S1111) is associated with credential theft, cryptomining, cryptotheft, and pre-ransomware actions.(Citation: Ensilo Darkgate 2018) DarkGate use increased significantly starting in 2022 and is under active development by its author, who provides it as a Malware-as-a-Service offering.(Citation: Trellix Darkgate 2023)

Domain-Name

Value
withupdate.com

IPv4-Addr

Value
45.89.53.187
5.252.177.207
103.124.106.237
170.130.55.130

StixFile

Value
dd7a8b55e4b7dc032ea6d6aed6153bec9b5b68b45369e877bb66ba21acc81455
897b0d0e64cf87ac7086241c86f757f3c94d6826f949a1f0fec9c40892c0cecb
6ed1b68de55791a6534ea96e721ff6a5662f2aefff471929d23638f854a80031
4de0e0e7f23adc3dd97d498540bd8283004aa131a59ae319019ade9ddef41795
2e34908f60502ead6ad08af1554c305b88741d09e36b2c24d85fd9bac4a11d2f
2b296ffc6d173594bae63d37e2831ba21a59ce385b87503710dc9ca439ed7833
1a960526c132a5293e1e02b49f43df1383bf37a0bbadd7ba7c106375c418dad4
196bb36f7d63c845afd40c5c17ce061e320d110f28ebe8c7c998b9e6b3fe1005
10e362e18c355b9f8db9a0dbbc75cf04649606ef96743c759f03508b514ad34e
038db3b838d0cd437fa530c001c9913a1320d1d7ac0fd3b35d974a806735c907

External References

- https://www.mcafee.com/blogs/other-blogs/mcafee-labs/the-darkgate-menace-leveraging-autohotkey-attempt-to-evade-smartscreen/
- https://otx.alienvault.com/pulse/6630fc8aef3381ff6fa4c401