When adversaries bring their own virtual machine for persistence

susannah.matt@redcanary.com December 9, 2025

Adversaries are constantly seeking new and unconventional methods to achieve their objectives. Earlier in 2025, Red Canary Intelligence uncovered an interesting tactic; following a noisy spam bombing campaign, an adversary introduced their own virtual machine (VM) into a compromised environment and established persistence.

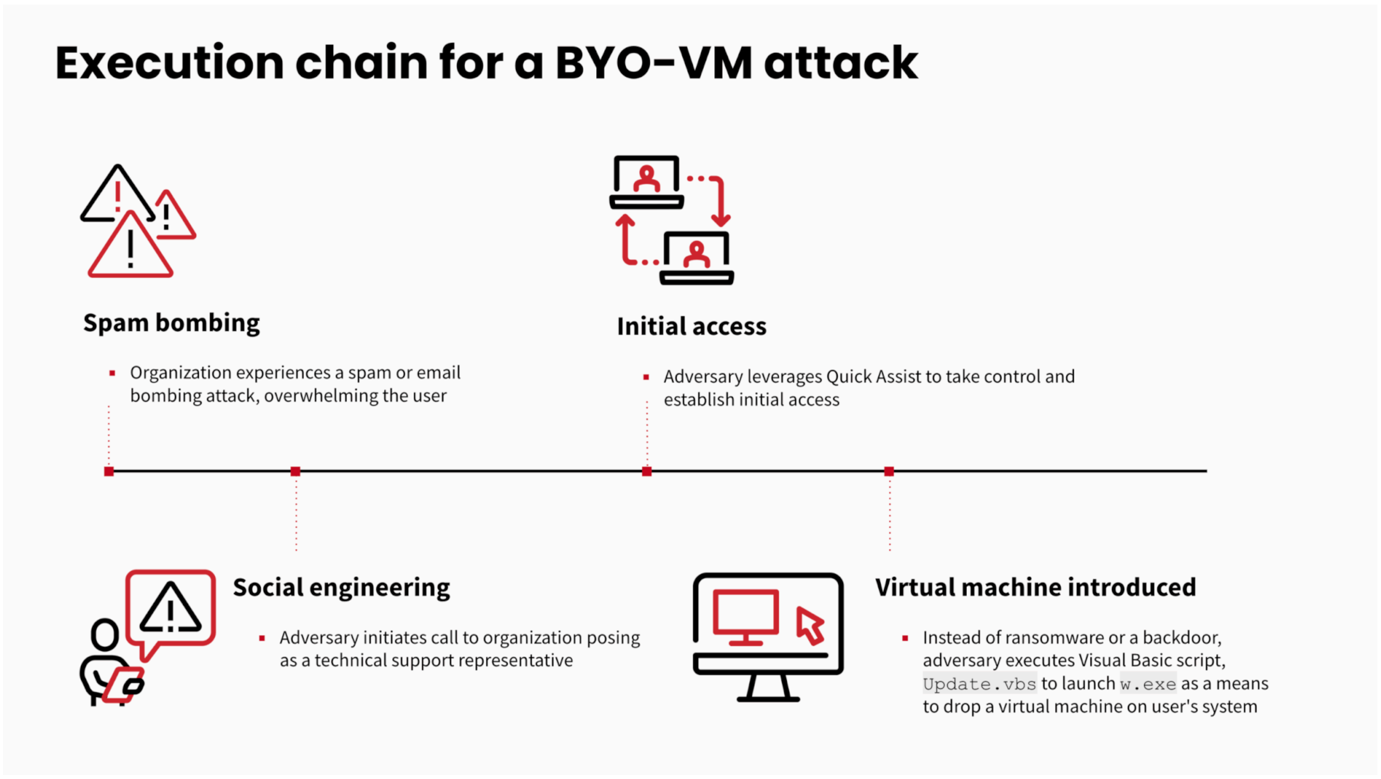
While the email bombing activity initially drew comparison to behavior we've seen leading to Black Basta ransomware infections, it later became clear that the threat actor had a specific set of tooling, specifically the deployment of a custom QEMU VM, which diverged from typical Black Basta tactics.

> This is the first time Red Canary Intelligence has detected an adversary bringing their own QEMU VM into an environment under the guise of a technical support call following a spam bombing attack.

Here we'll unravel the incident, exploring the initial social engineering attack, the adversary's choice of tools, and how we pieced together the intrusion. We'll end by discussing how the strategy represents an evolution in adversary methodology and what it means for defenders.

A familiar smokescreen: Spam bombing and social engineering

The incident began with an organization experiencing a spam or email bombing attack. This technique, wherein a victim's inbox is flooded with thousands of unsolicited emails, has become a popular distraction tactic, and favored by ransomware groups. The goal is to overwhelm the victim, making it difficult to spot legitimate communications and prime them for a follow-up social engineering attempt.


Execution chain for a BYO-VM attack
Spam bombing
Organization experiences a spam or email bombing attack, overwhelming the user
Social engineering
Adversary initiates call to organization posing as a technical support representative
Initial access
Adversary leverages Quick Assist to take control and establish initial access
Virtual machine introduced
Instead of ransomware or a backdoor, adversary executes Visual Basic script, Update.vbs to launch w.exe as a means to drop a virtual machine on user's system

In this scenario, the adversary wasted no time. After flooding the inbox, they initiated a call to the organization posing as a technical support representative. Their offer was simple: help alleviate the deluge. It's a calculated maneuver, leveraging distress to gain trust.

After gaining the user's confidence, the adversary leveraged remote assistance software Quick Assist. This legitimate, built-in Windows application allows a trusted person to take control of another computer remotely. While it can be used for support—like many remote monitoring and management (RMM) tools these days—it can be misused by threat actors to establish initial access and deploy malicious payloads.

A twist: The adversary's virtual machine

Instead of directly dropping ransomware or a standard backdoor, the adversary used the Quick Assist foothold to introduce their own VM. Anytime an adversary leaves remnants of their activity, especially an entire file system, it presents an opportunity for forensic analysis—so that's just what we did.

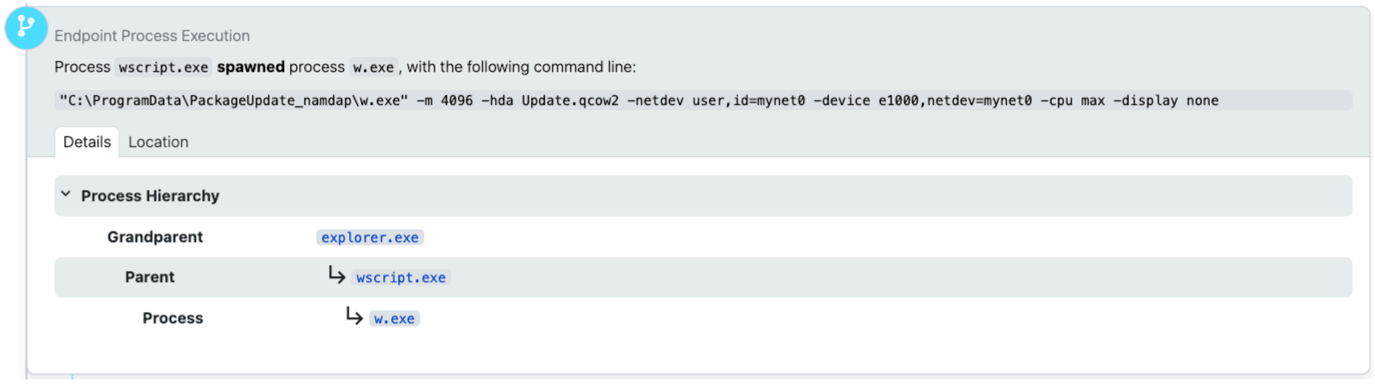
Unpacking the QEMU disk: A forensic deep dive

The adversary's actions began with the execution of a Visual Basic Script (VBScript), Update.vbs. Due to Quick Assist's nature of piggybacking on the user's Explorer session, many of these initial actions appeared to originate from explorer.exe itself, something that might raise red flags if not for the context of the remote session.

The primary function of `Update.vbs` was to launch `w.exe`; this executable was invoked with several specific parameters:

- `-m 4096` to allocate 4096 MB of memory for the VM
- –`hda Update.qcow2` to specify the virtual hard disk image
- networking parameters like `-netdev user, id=mynet0 -device e1000, netde", '0', "false"`

These network settings gave the VM full access to the internet, allowing for command and control (C2) communications, and crucially, permitted it to scan the local network of the compromised organization.


Endpoint Process Execution

Process `wscript.exe` **spawned** process `w.exe`, with the following command line:

```
"C:\ProgramData\PackageUpdate_namdap\w.exe" -m 4096 -hda Update.qcow2 -netdev user,id=mynet0 -device e1000,netdev=mynet0 -cpu max -display none
```

Details Location

Process Hierarchy

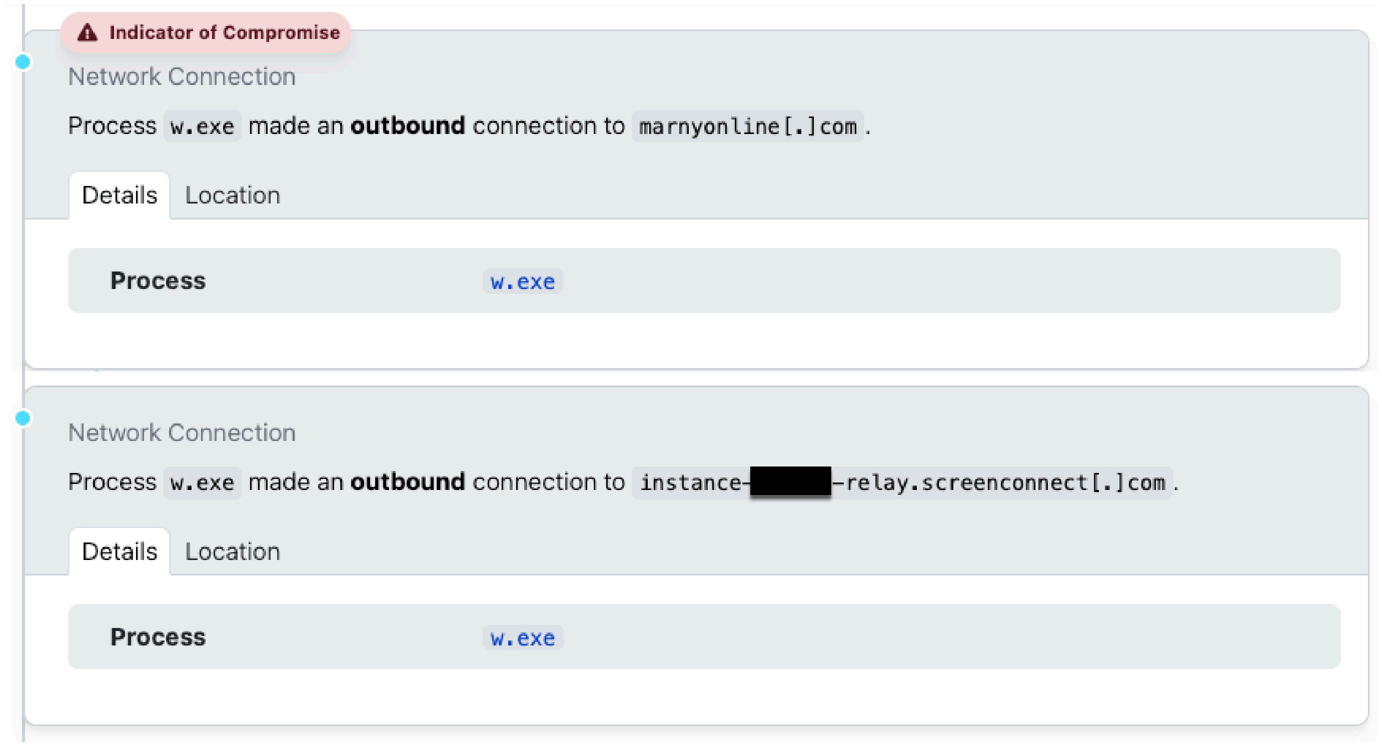
Grandparent	`explorer.exe`
Parent	↳ `wscript.exe`
Process	↳ `w.exe`

A quick look at the metadata for `w.exe` on VirusTotal immediately revealed its true identity: `qemu-system-x86_64.exe`, a component of QEMU, an open source emulator and virtualizer. While legitimate, its presence on a standard user's system—especially one invoked under suspicious circumstances—is highly unusual. Most everyday users, even power users, don't have a need for a full system emulator.

Initial network reconnaissance from within the VM

Once the QEMU VM was up and running, it started exhibiting tell-tale signs of reconnaissance. We observed numerous network connections to various local hosts; this internal scanning is a critical step for adversaries, allowing them to map out the network topography, identify potential targets, and prepare for lateral movement.

Simultaneously, the VM established external network connections. One notable connection was to `marnyonline[.]com`, an external domain that would later prove to be a C2 server. Another connection was made to the legitimate remote support and access software ScreenConnect. Like Quick Assist, the use of remote admin tools for malicious purposes can help adversaries blend in with normal network traffic.


Indicator of Compromise

Network Connection

Process `w.exe` made an **outbound** connection to `marnyonline[.]com`.

Details | Location

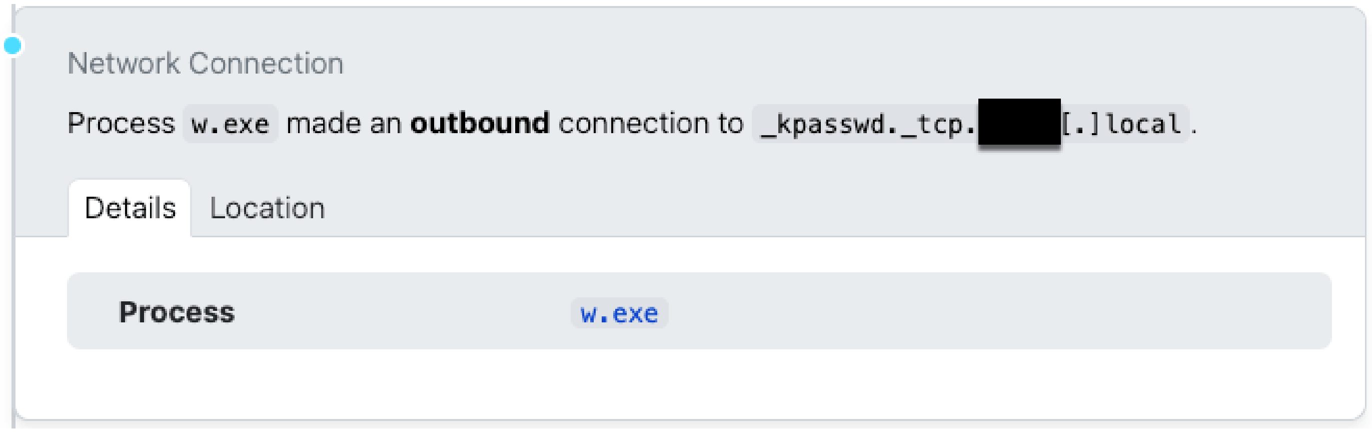
Process `w.exe`

Network Connection

Process `w.exe` made an **outbound** connection to `instance-`[redacted]`-relay.screenconnect[.]com`.

Details | Location

Process `w.exe`


Network Connection

Process `w.exe` made an **outbound** connection to `_kpasswd._tcp.`[redacted]`[.]local`.

Details | Location

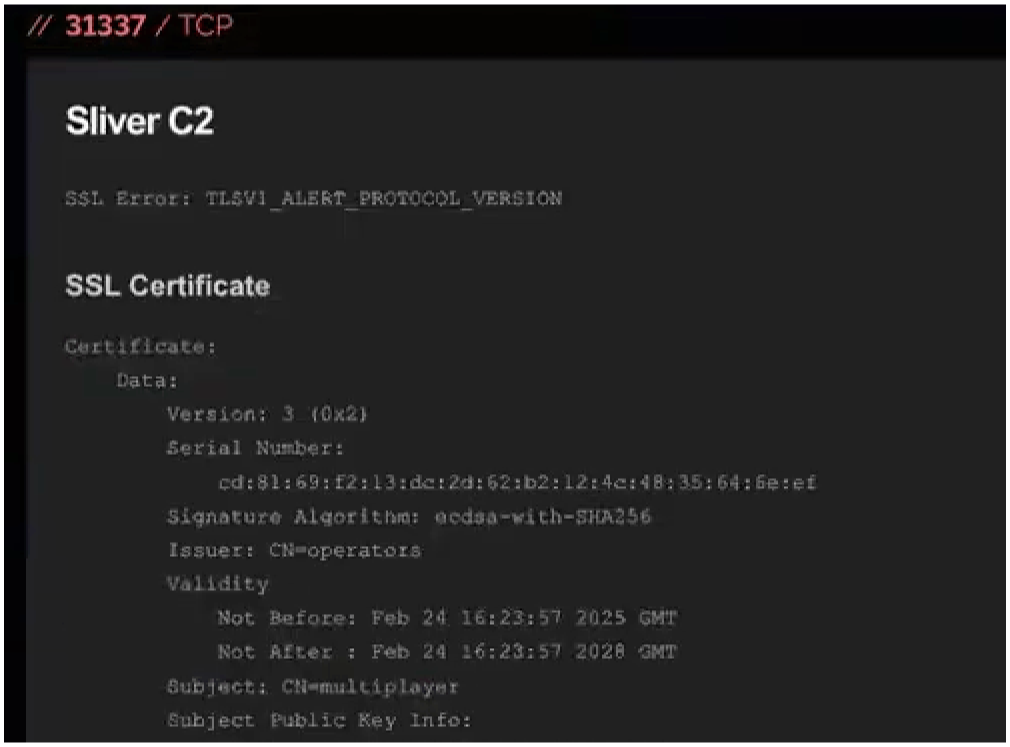
Process `w.exe`

Further network activity included DNS resolution requests for service records (SRV records) for the local DNS domain. These records are fundamental to how services within an organization's network locate each other. For instance, Windows systems use SRV records to find Active Directory domain controllers. Adversaries can query these records to gain significant insights into an organization's domain infrastructure, identifying critical servers and services for potential exploitation. This type of service reconnaissance is a primary method of understanding the target environment.

Sliver C2 and "multi-player" mode

Upon investigation, Red Canary Intelligence identified an IP address (`45[.]61[.]169[.]127`) and discovered, via Shodan, that it was associated with Sliver, an open source command and control (C2) framework.

What made this association particularly straightforward was its configuration in Sliver. When running in “team server” mode—a configuration allowing multiple adversaries to control multiple compromised hosts from a single server—Sliver defaults to setting up a server with an SSL certificate with the subject name CN=multiplayer and issuer O=operators. While likely intended for internal team identification, this can also provide a unique fingerprint for tracking Sliver team servers via Shodan.



// 31337 / TCP

Sliver C2

```
SSL Error: TLSV1_ALERT_PROTOCOL_VERSION
```

SSL Certificate

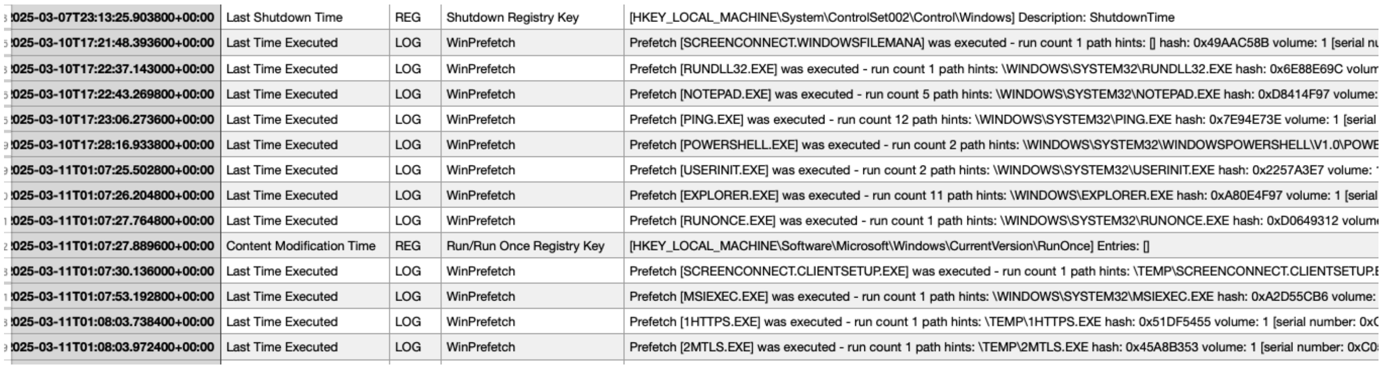
```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            cd:81:69:f2:13:dc:2d:62:b2:12:4c:48:35:64:6e:ef
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: CN=operators
        Validity
            Not Before: Feb 24 16:23:57 2025 GMT
            Not After : Feb 24 16:23:57 2028 GMT
        Subject: CN=multiplayer
        Subject Public Key Info:
```

Reconstructing the adversary’s actions: Insights from prefetch and browser history

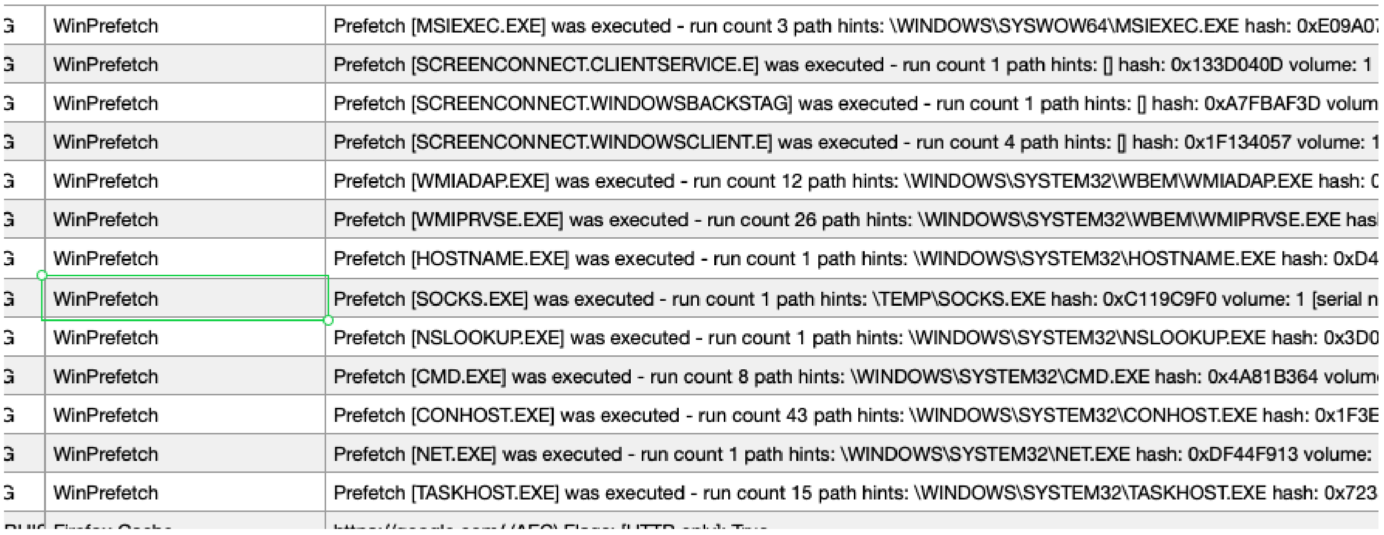
Performing forensic analysis, particularly with a tool like Plaso—a Python-based engine for creating digital timelines—can be a lengthy and arduous process. Analyzing an 8 gigabyte disk image, like the one left by this adversary, can generate a 200 MB spreadsheet timeline, overwhelming even robust tools like Google Sheets. Despite these challenges, such timelines are indispensable for understanding a sequence of events like those in an incident like this.

Our analysis of prefetch and application compatibility cache data provided additional insight. Prefetch files, automatically generated by Windows for performance optimization and stored in `C:\Windows\Prefetch`, record information about application usage. While application prefetching is disabled on Windows Server by default, the adversary’s VM was running Windows 7 Service Pack 1, where prefetch was active. This allowed us to determine which programs the adversary ran, even without command-line arguments. While they likely didn’t anticipate someone capturing their disk, disabling prefetch would have been a basic anti-forensic measure.

We determined the VM’s operating system by examining the file properties of the [NT OS kernel executable](), `ntoskrnl.exe`, which reported a file version of 6.1.7601—identifying the OS as Windows 7 Service Pack 1. It’s plausible the adversary used a pre-built VM template, perhaps one of the older, freely distributed, pre-built Microsoft VMs for testing purposes, and then customized it.



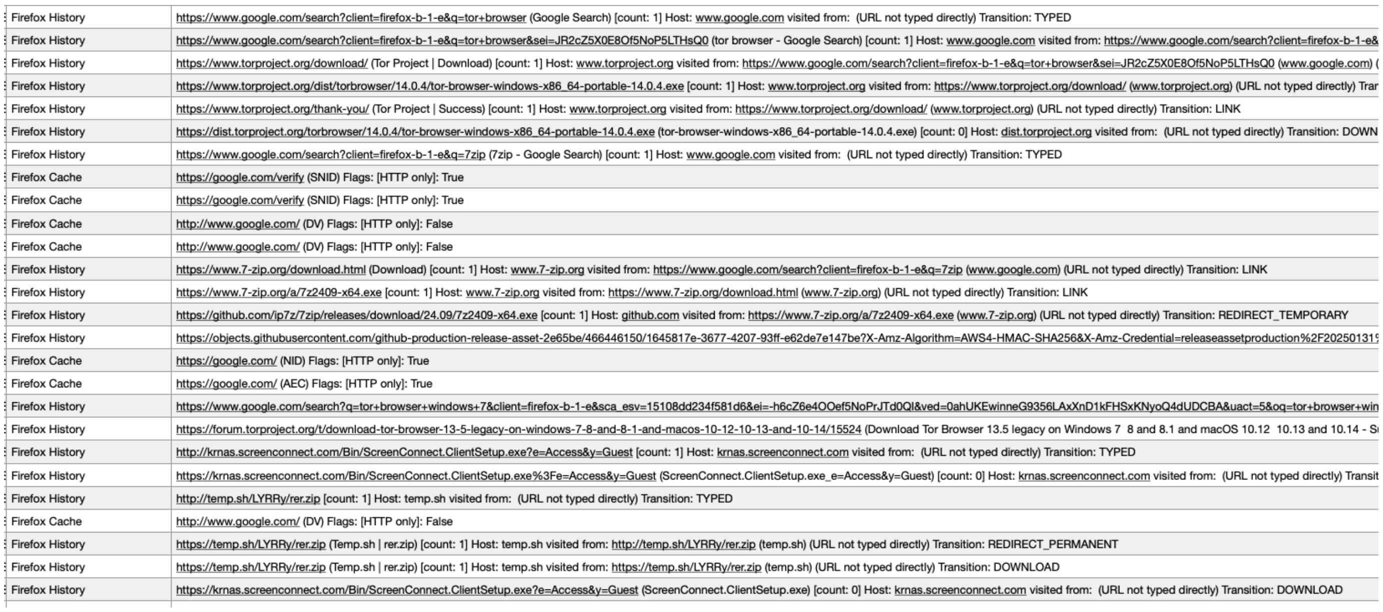
025-03-07T23:13:25.903800+00:00	Last Shutdown Time	REG	Shutdown Registry Key	[HKEY_LOCAL_MACHINE\System\ControlSet002\Control\Windows] Description: ShutdownTime
025-03-10T17:21:48.393600+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [SCREENCONNECT.WINDOWSFILEMANA] was executed - run count 1 path hints: [] hash: 0x49AAC58B volume: 1 [serial nu
025-03-10T17:22:37.143000+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [RUNDLL32.EXE] was executed - run count 1 path hints: \WINDOWS\SYSTEM32\RUNDLL32.EXE hash: 0x6E88E69C volum
025-03-10T17:22:43.269800+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [NOTEPAD.EXE] was executed - run count 5 path hints: \WINDOWS\SYSTEM32\NOTEPAD.EXE hash: 0xD8414F97 volume:
025-03-10T17:23:06.273600+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [PING.EXE] was executed - run count 12 path hints: \WINDOWS\SYSTEM32\PING.EXE hash: 0x7E94E73E volume: 1 [serial
025-03-10T17:28:16.933800+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [POWERSHELL.EXE] was executed - run count 2 path hints: \WINDOWS\SYSTEM32\WINDOWSPOWERSHELL\V1.0\POWE
025-03-11T01:07:25.502800+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [USERINIT.EXE] was executed - run count 2 path hints: \WINDOWS\SYSTEM32\USERINIT.EXE hash: 0x2257A3E7 volume:
025-03-11T01:07:26.204800+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [EXPLORER.EXE] was executed - run count 11 path hints: \WINDOWS\EXPLORER.EXE hash: 0xA80E4F97 volume: 1 [serial
025-03-11T01:07:27.764800+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [RUNONCE.EXE] was executed - run count 1 path hints: \WINDOWS\SYSTEM32\RUNONCE.EXE hash: 0xD0649312 volum
025-03-11T01:07:27.889600+00:00	Content Modification Time	REG	Run/Run Once Registry Key	[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce] Entries: []
025-03-11T01:07:30.136000+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [SCREENCONNECT.CLIENTSETUP.EXE] was executed - run count 1 path hints: \TEMP\SCREENCONNECT.CLIENTSETUP.E
025-03-11T01:07:53.192800+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [MSIEXEC.EXE] was executed - run count 1 path hints: \WINDOWS\SYSTEM32\MSIEXEC.EXE hash: 0xA2D55CB6 volume:
025-03-11T01:08:03.738400+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [1HTTPS.EXE] was executed - run count 1 path hints: \TEMP\1HTTPS.EXE hash: 0x51DF5455 volume: 1 [serial number: 0xC
025-03-11T01:08:03.972400+00:00	Last Time Executed	LOG	WinPrefetch	Prefetch [2MTLS.EXE] was executed - run count 1 path hints: \TEMP\2MTLS.EXE hash: 0x45A8B353 volume: 1 [serial number: 0xC0



G	WinPrefetch	Prefetch [MSIEXEC.EXE] was executed - run count 3 path hints: \WINDOWS\SYSWOW64\MSIEXEC.EXE hash: 0xE09A0
G	WinPrefetch	Prefetch [SCREENCONNECT.CLIENTSERVICE.E] was executed - run count 1 path hints: [] hash: 0x133D040D volume: 1
G	WinPrefetch	Prefetch [SCREENCONNECT.WINDOWSBACKSTAG] was executed - run count 1 path hints: [] hash: 0xA7FBAF3D volum
G	WinPrefetch	Prefetch [SCREENCONNECT.WINDOWSCLIENT.E] was executed - run count 4 path hints: [] hash: 0x1F134057 volume: 1
G	WinPrefetch	Prefetch [WMIADAP.EXE] was executed - run count 12 path hints: \WINDOWS\SYSTEM32\WBEM\WMIADAP.EXE hash: 0
G	WinPrefetch	Prefetch [WMIPRVSE.EXE] was executed - run count 26 path hints: \WINDOWS\SYSTEM32\WBEM\WMIPRVSE.EXE has
G	WinPrefetch	Prefetch [HOSTNAME.EXE] was executed - run count 1 path hints: \WINDOWS\SYSTEM32\HOSTNAME.EXE hash: 0xD4
G	WinPrefetch	Prefetch [SOCKS.EXE] was executed - run count 1 path hints: \TEMP\SOCKS.EXE hash: 0xC119C9F0 volume: 1 [serial n
G	WinPrefetch	Prefetch [NSLOOKUP.EXE] was executed - run count 1 path hints: \WINDOWS\SYSTEM32\NSLOOKUP.EXE hash: 0x3D0
G	WinPrefetch	Prefetch [CMD.EXE] was executed - run count 8 path hints: \WINDOWS\SYSTEM32\CMD.EXE hash: 0x4A81B364 volum
G	WinPrefetch	Prefetch [CONHOST.EXE] was executed - run count 43 path hints: \WINDOWS\SYSTEM32\CONHOST.EXE hash: 0x1F3E
G	WinPrefetch	Prefetch [NET.EXE] was executed - run count 1 path hints: \WINDOWS\SYSTEM32\NET.EXE hash: 0xDF44F913 volume:
G	WinPrefetch	Prefetch [TASKHOST.EXE] was executed - run count 15 path hints: \WINDOWS\SYSTEM32\TASKHOST.EXE hash: 0x723
[illegible]	[illegible]	[illegible]

The prefetch data also revealed a timeline of activity within the VM. The first recorded action was the use of ScreenConnect, followed by a series of program executions: `ping.exe`, `Notepad`, and `powershell.exe`. We also identified two particularly interesting executables in the `TEMP` folder: `1HTTPS.EXE` and `2MTLS.EXE`, along with `SOCKS.EXE`. Further prefetch entries showed execution of `NSLOOKUP.EXE` (possibly corroborating the earlier service record lookups) and `NET.EXE` (potentially for mapping remote shares).

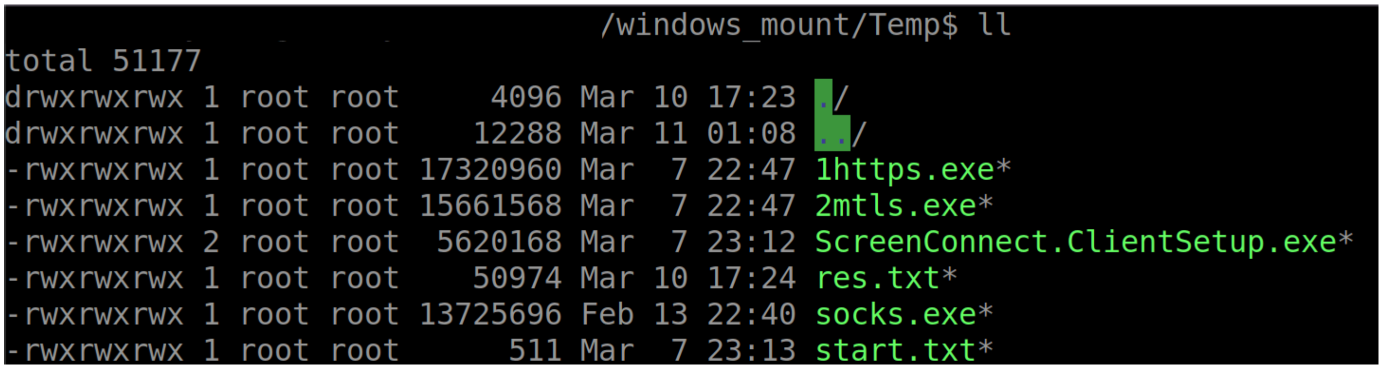
Beyond executables, Plaso can also parse browser databases to reconstruct internet history. Unsurprisingly, the adversary’s first action after what appeared to be a fresh Windows installation was to install a non-Internet Explorer browser: Firefox. The initial Firefox activity included searches for the Tor browser, a download of 7-Zip, a ScreenConnect installer, and an archive named `rer.zip`. The `rer.zip` file was no longer present on the disk, suggesting it was deleted after use. The adversary’s use of Tor aligns with efforts to anonymize their activities, while 7-Zip is a common utility for archiving and exfiltrating data.



Firefox History	https://www.google.com/search?client=firefox-b-1-e&q=tor+browser (Google Search) [count: 1] Host: www.google.com visited from: (URL not typed directly) Transition: TYPED
Firefox History	https://www.google.com/search?client=firefox-b-1-e&q=tor+browser&sei=JR2cZ5X0E8Of5NoP5LTHsQ0 (tor browser - Google Search) [count: 1] Host: www.google.com visited from: https://www.google.com/search?client=firefox-b-1-e&
Firefox History	https://www.torproject.org/download/ (Tor Project | Download) [count: 1] Host: www.torproject.org visited from: https://www.google.com/search?client=firefox-b-1-e&q=tor+browser&sei=JR2cZ5X0E8Of5NoP5LTHsQ0 (www.google.com) (
Firefox History	https://www.torproject.org/dist/torbrowser/14.0.4/tor-browser-windows-x86_64-portable-14.0.4.exe [count: 1] Host: www.torproject.org visited from: https://www.torproject.org/download/ (www.torproject.org) (URL not typed directly) Trar
Firefox History	https://www.torproject.org/thank-you/ (Tor Project | Success) [count: 1] Host: www.torproject.org visited from: https://www.torproject.org/download/ (www.torproject.org) (URL not typed directly) Transition: LINK
Firefox History	https://dist.torproject.org/torbrowser/14.0.4/tor-browser-windows-x86_64-portable-14.0.4.exe (tor-browser-windows-x86_64-portable-14.0.4.exe) [count: 0] Host: dist.torproject.org visited from: (URL not typed directly) Transition: DOWN
Firefox History	https://www.google.com/search?client=firefox-b-1-e&q=7zip (7zip - Google Search) [count: 1] Host: www.google.com visited from: (URL not typed directly) Transition: TYPED
Firefox Cache	https://google.com/verify (SNID) Flags: [HTTP only]: True
Firefox Cache	https://google.com/verify (SNID) Flags: [HTTP only]: True
Firefox Cache	http://www.google.com/ (DV) Flags: [HTTP only]: False
Firefox Cache	http://www.google.com/ (DV) Flags: [HTTP only]: False
Firefox History	https://www.7-zip.org/download.html (Download) [count: 1] Host: www.7-zip.org visited from: https://www.google.com/search?client=firefox-b-1-e&q=7zip (www.google.com) (URL not typed directly) Transition: LINK
Firefox History	https://www.7-zip.org/a/7z2409-x64.exe [count: 1] Host: www.7-zip.org visited from: https://www.7-zip.org/download.html (www.7-zip.org) (URL not typed directly) Transition: LINK
Firefox History	https://github.com/ip7z/7zip/releases/download/24.09/7z2409-x64.exe [count: 1] Host: github.com visited from: https://www.7-zip.org/a/7z2409-x64.exe (www.7-zip.org) (URL not typed directly) Transition: REDIRECT_TEMPORARY
Firefox History	https://objects.githubusercontent.com/github-production-release-asset-2e65be/466446150/1645817e-3677-4207-93ff-e62de7e147be?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250131
Firefox Cache	https://google.com/ (NID) Flags: [HTTP only]: True
Firefox Cache	https://google.com/ (AEC) Flags: [HTTP only]: True
Firefox History	https://www.google.com/search?q=tor+browser+windows+7&client=firefox-b-1-e&sca_esv=15108dd234f581d6&ei=-h6cZ6e4OOef5NoPrJTd0QI&ved=0ahUKEwinneG9356LAxXnD1kFHSxKNyoQ4dUDCBA&uact=5&oq=tor+browser+win
Firefox History	https://forum.torproject.org/t/download-tor-browser-13-5-legacy-on-windows-7-8-and-8-1-and-macos-10-12-10-13-and-10-14/15524 (Download Tor Browser 13.5 legacy on Windows 7 8 and 8.1 and macOS 10.12 10.13 and 10.14 - S
Firefox History	http://krnas.screenconnect.com/Bin/ScreenConnect.ClientSetup.exe?e=Access&y=Guest [count: 1] Host: krnas.screenconnect.com visited from: (URL not typed directly) Transition: TYPED
Firefox History	https://krnas.screenconnect.com/Bin/ScreenConnect.ClientSetup.exe%3Fe=Access&y=Guest (ScreenConnect.ClientSetup.exe_e=Access&y=Guest) [count: 0] Host: krnas.screenconnect.com visited from: (URL not typed directly) Transit
Firefox History	http://temp.sh/LYRRy/rer.zip [count: 1] Host: temp.sh visited from: (URL not typed directly) Transition: TYPED
Firefox Cache	http://www.google.com/ (DV) Flags: [HTTP only]: False
Firefox History	https://temp.sh/LYRRy/rer.zip (Temp.sh | rer.zip) [count: 1] Host: temp.sh visited from: http://temp.sh/LYRRy/rer.zip (temp.sh) (URL not typed directly) Transition: REDIRECT_PERMANENT
Firefox History	https://temp.sh/LYRRy/rer.zip (Temp.sh | rer.zip) [count: 1] Host: temp.sh visited from: https://temp.sh/LYRRy/rer.zip (temp.sh) (URL not typed directly) Transition: DOWNLOAD
Firefox History	https://krnas.screenconnect.com/Bin/ScreenConnect.ClientSetup.exe?e=Access&y=Guest (ScreenConnect.ClientSetup.exe) [count: 0] Host: krnas.screenconnect.com visited from: (URL not typed directly) Transition: DOWNLOAD

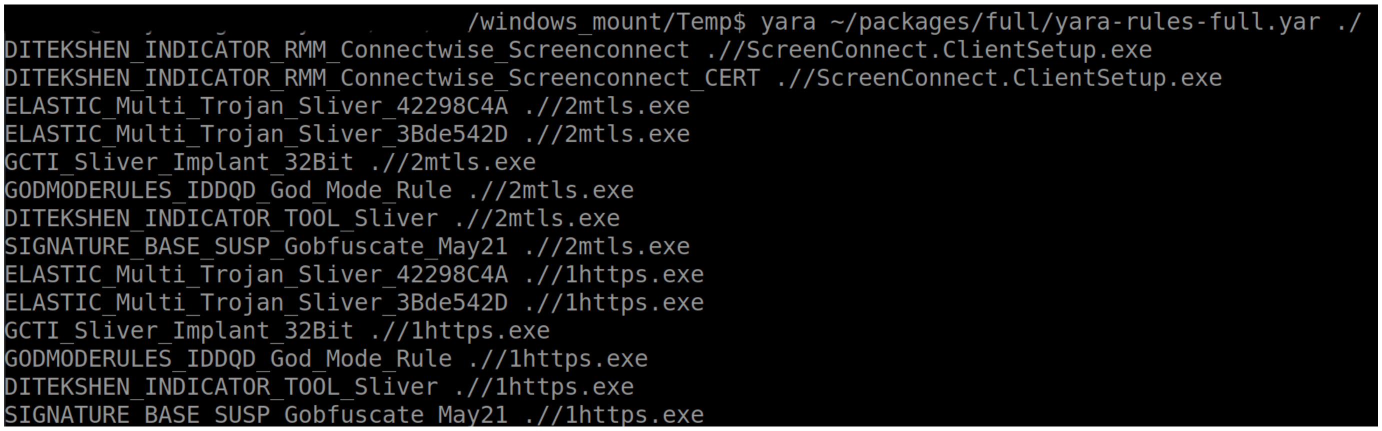
The adversary's toolkit: Sliver, ScreenConnect, and Qdoor

A deeper dive into the `TEMP` folder within the virtual disk yielded a treasure trove of information. Alongside the previously identified `1HTTPS.EXE` and `2MTLS.EXE`, we found `ScreenConnect.ClientSetup.exe`, `res.txt`, and `start.txt`.



```
                                  /windows_mount/Temp$ ll
total 51177
drwxrwxrwx 1 root root     4096 Mar 10 17:23 ./
drwxrwxrwx 1 root root    12288 Mar 11 01:08 ../
-rwxrwxrwx 1 root root 17320960 Mar  7 22:47 1https.exe*
-rwxrwxrwx 1 root root 15661568 Mar  7 22:47 2mtls.exe*
-rwxrwxrwx 2 root root  5620168 Mar  7 23:12 ScreenConnect.ClientSetup.exe*
-rwxrwxrwx 1 root root    50974 Mar 10 17:24 res.txt*
-rwxrwxrwx 1 root root 13725696 Feb 13 22:40 socks.exe*
-rwxrwxrwx 1 root root      511 Mar  7 23:13 start.txt*
```

A YARA scan confirmed `ScreenConnect.ClientSetup.exe` was legitimate ScreenConnect software, while `1HTTPS.EXE` and `2MTLS.EXE` were identified as Sliver implants, likely obfuscated using gobfuscate. `SOCKS.EXE`, however, did not initially trigger any known YARA rules, posing a temporary mystery.



```
/windows_mount/Temp$ yara ~/packages/full/yara-rules-full.yar ./
DITEKSHEN_INDICATOR_RMM_Connectwise_Screenconnect .//ScreenConnect.ClientSetup.exe
DITEKSHEN_INDICATOR_RMM_Connectwise_Screenconnect_CERT .//ScreenConnect.ClientSetup.exe
ELASTIC_Multi_Trojan_Sliver_42298C4A .//2mtls.exe
ELASTIC_Multi_Trojan_Sliver_3Bde542D .//2mtls.exe
GCTI_Sliver_Implant_32Bit .//2mtls.exe
GODMODERULES_IDDQD_God_Mode_Rule .//2mtls.exe
DITEKSHEN_INDICATOR_TOOL_Sliver .//2mtls.exe
SIGNATURE_BASE_SUSP_Gobfuscate_May21 .//2mtls.exe
ELASTIC_Multi_Trojan_Sliver_42298C4A .//1https.exe
ELASTIC_Multi_Trojan_Sliver_3Bde542D .//1https.exe
GCTI_Sliver_Implant_32Bit .//1https.exe
GODMODERULES_IDDQD_God_Mode_Rule .//1https.exe
DITEKSHEN_INDICATOR_TOOL_Sliver .//1https.exe
SIGNATURE_BASE_SUSP_Gobfuscate_May21 .//1https.exe
```

The `res.txt` file was particularly illuminating. It contained 50,974 bytes of text, revealing the adversary's reconnaissance findings. The file documented a ping scan, with each entry representing a single ping instance. This "one ping per instance" approach suggests the adversary was deliberately trying to be stealthier and faster than the default Windows ping behavior, which sends four pings per target. This granular scanning allowed them to map the network discreetly.

The `start.txt` file revealed the adversary's persistence mechanisms. While it might seem odd for an adversary-controlled VM to need persistence, consider its context: this VM is essentially a rogue device on the victim's network. Just like any other remote access tool, the adversary needs to ensure their access remains viable if the VM or the host machine restarts. The persistence configuration in `start.txt` ensured ScreenConnect and the two Sliver processes (`1HTTPS.EXE` and `2MTLS.EXE`) would execute automatically. This redundancy—using both ScreenConnect and two Sliver variants—highlights the adversary's desire for resilient access, anticipating potential network segmentation or firewall rules that might block one access vector but not another. It's a scattershot approach to maintain control.

VMray analysis of `1HTTPS.EXE` in a Windows 7 SP1 isolated environment confirmed it was a Sliver beacon, attempting to communicate with `marnyonline[.]com`. Similarly, `2MTLS.EXE` was confirmed as another Sliver implant, trying to reach `45[.]61[.]169[.]127` over port 8443. Both appeared to be straightforward Sliver implants, designed for beaconing and basic command execution.

To further analyze `SOCKS.EXE`, we used VMray's dynamic analysis to execute it in an isolated network environment. The initial VMray analysis showed `SOCKS.EXE` attempting to connect to `88[.]119[.]167[.]239`, delaying execution, and dynamically resolving API functions but without clear classifications.

A link from LinkedIn

While an exhaustive search for `88[.]119[.]167[.]239` on VirusTotal and other open-source intelligence platforms yielded little, an unusual clue emerged: a LinkedIn Pulse post from ConnectWise, discussing how the BlackSuit ransomware group was leveraging a network tunneling backdoor it dubbed “QDoor.”

Although a YARA rule from the post didn’t immediately match `SOCKS.EXE`, the network address listed in it, `88[.]119[.]167[.]239`, was a direct hit. A subsequent VirusTotal upload of `SOCKS.EXE` by a third party later corroborated this, with its reputation eventually updating to QDoor, suggesting it’s part of the adversary’s arsenal. Looking at the malware’s behavior in VMray, including attempts to gather information from the software framework Qt Project, including `qtlogging.ini` and `qt.conf` was also consistent with known QDoor characteristics as outlined by ConnectWise.

Deleted artifacts and missed opportunities

Part of a thorough forensic investigation involves attempting to recover deleted files. We mounted the QEMU disk image and used The Sleuth Kit with Foremost, a console program that recovers files based on header, footer, and internal data structures, on the unallocated blocks of storage. This process yielded several ZIP files and various image files, mostly remnants of program installations or deletions.

Interestingly, the adversary had placed the Tor browser and WinSCP, a popular free file manager for Windows, onto the image at some point, but these were not found installed on the active file system. This suggests a “prep” phase where these tools were included, followed by a “deployment” phase where they were removed, perhaps to reduce the disk footprint or evade detection.

The presence of volume shadow copies (VSCs) on the VM offered another avenue for data recovery. VSCs store previous versions of files and system states. While we managed to recover a Tor browser installation from a VSC, we were unable to retrieve any browsing history. Tor browser is designed to not store cache or internet history data to disk upon shutdown, which unfortunately meant no historical data was available for analysis, even through VSCs.

The novelty of the technique: A shift in tactics

The deployment of a QEMU VM by an adversary isn’t entirely unprecedented. We’ve seen affiliates of the Ragnar Locker ransomware use stripped-down VirtualBox VMs in the past to evade antivirus detection and establish a stronger foothold within compromised networks. Earlier this year, Sophos saw an adversary using similar tools within a QEMU virtual machine to compromise a domain services account before installing a commercial RMM. Kaspersky Lab has also reported on adversaries using QEMU virtual machines for network tunneling, bypassing security controls.

However, the series of events here marks an interesting deviation: This is the first time Red Canary Intelligence has detected an adversary bringing their own QEMU VM into an environment under the guise of a technical support call following a spam bombing attack. This blend of social engineering, legitimate tool abuse, stealthy persistence and novel payload delivery suggests the adversary behind this incident prioritizes complete control over their operational environment.

Implications for defense

This incident serves as a good reminder that adversaries are continuously adapting their tactics. In this incident, deploying their own virtual machine offers several advantages to an adversary:

- **Isolation:** The VM creates an isolated environment for their tools, making it harder for host-based security solutions to detect and analyze them directly.
- **Portability:** The entire adversary environment can be pre-configured and quickly deployed.
- **Stealth:** Using a legitimate virtualization tool like QEMU and masquerading as technical support helps blend into benign activity.
- **Persistence:** The VM acts as a persistent entry point, potentially circumventing direct endpoint controls.

For defenders, the incident underscores the need for a multi-layered security strategy. Beyond robust email filtering and endpoint detection and response (EDR), organizations must prioritize:

- **Enhanced social engineering training:** Users need to be highly skeptical of unsolicited technical support, especially if it follows unusual activity like a spam bombing.
- **Strict remote access policies:** Implement granular controls over remote assistance tools like Quick Assist, and monitor their usage rigorously.
- **Network segmentation and monitoring**: Limit lateral movement within the network and actively monitor for unusual internal scanning, DNS queries, and external C2 communications, even from seemingly internal hosts.
- **Anomaly detection:** Behavioral analytics that can detect the abnormal execution of virtualization software on endpoints that don't typically use them.
- **Threat intelligence sharing:** Staying informed about novel adversary tactics, such as the unique fingerprint of Sliver team servers.

By understanding the full lifecycle of this attack, from the initial distraction to the forensic reconstruction of the VM's contents, organizations can better prepare to detect and defend against these increasingly sophisticated and innovative threats.

Indicators of compromise

Indicator	Notes
`45[.]61[.]169[.]127`	Sliver C2

Indicator	Notes
`88[.]119[.]167[.]239`	QDoor C2
`sha256:af68dfd0ad3d95ff0869b593289eff4c26f5a6a2793b441010c51da891b58269`	Legitimate QEMU emulator SHA256