

# Wreaking Havoc - CSCG2023

**Category:** Forensics

**Difficulty:** Medium

**Author:** Kolja

"One of our admins fell for a phishing site and downloaded malware. The attacker wreaked havoc on our system, but you might find out what was exfiltrated using the attached network capture."

## Recon

Given to us is a singular PCAPNG file ([wreaking-havoc.pcapng](#)).  
As such, our first order of business is looking at this file in [Wireshark](#).

There aren't too many packages captured in this file, but filtering them out by usefulness still provides a good start. The description mentions that an admin fell for a phishing site and that we have to find the data, which a piece of malware exfiltrated.

Luckily for us, the phishing website seems to have been using HTTP without TLS, so filtering with http actually yields us the website traffic without encryption.

We can now follow the traffic of the very first HTTP GET packet and download its result to see how the index page looks like.

file:///home/sw1tchbl4d3/DataTransfer/hacking/2023/cscg/forensics/havoc/re

h m d o v

**KeePass**  
Password Safe

**Home**

- Home & News
- Forums
- Feature List
- Screenshots

**Getting KeePass**

- Downloads
- Translations
- Plugins / Ext.

**Information / WWW**

- Help
- FAQ
- Security
- Awards
- Links

**Support KeePass**

- Donate

**Getting KeePass - Downloads**

Here you can download KeePass:

**KeePass 2.53**

**Installer for Windows (2.53):**

**Download Now**  
KeePass-2.53-Setup.exe

Download the EXE file above, run it and follow the steps of the installation program. You need local installation rights (use the Portable version on the right, if you don't have local installation rights).

**Portable (2.53):**

**Download Now**  
KeePass-2.53-Portable.zip

Download the ZIP package above and extract it. KeePass runs without any additional application directory.

**Supported operating systems:** Windows 7 / 8 / 10 / 11 (each 32-bit and 64-bit), [Mono](#) (Linux, MacOS, BSD, ...).

**KeePass 1.41**

**Installer for Windows (1.41):**

**Download Now**  
KeePass-1.41-Setup.exe

Download the EXE file above, run it and follow the steps of the installation program. You need local installation rights (use the Portable version on the right, if you don't have local installation rights).

**Portable (1.41):**

**Download Now**  
KeePass-1.41-Portable.zip

Download the ZIP package above and extract it. KeePass runs without any additional application directory.

**Supported operating systems:** Windows 7 / 8 / 10 / 11 (each 32-bit and 64-bit), Wine.

Unsure which edition (1.x or 2.x) to choose? See the [Edition Comparison Table](#). See also the [Development Status](#).

Other downloads and links:

**Contributed/Unofficial KeePass Ports**

- [KeePassDroid](#) (for Android)
- [KeePass2Android](#) (for Android)

It seems to be a replica of the keepass download webpage.

Clicking the download links actually works and yields us the malware. This is because it is embedded into the website's HTML as a base64 string.

Once we have our malware keepass.exe downloaded, let's see what [virustotal](#) has to say about this file.

51

/ 69

Community Score

peexe

overlay

runtime-modules

detect-debug-environment

checks-network-adapters

long-sleeps

direct-cpu-cloc

51 security vendors and no sandboxes flagged this file as malicious

e8aa24b890249f19149c3e7f2289d6e7ebb124508b23efb8167c6adbd964908f

82.00 KB

2023-04-24

Size

7 days ago

keepass.exe

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 6

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to [automate checks](#).

Popular threat label

trojan.havoc/havokiz

Threat categories

trojan

Family label

Security vendors' analysis

AhnLab-V3	Backdoor/Win.Havoc.C5403085	Alibaba	Back
ALYac	Generic.Trojan.Havokiz.Marte.C.9924B27D	Antiy-AVL	Troja
Arcabit	Generic.Trojan.Havokiz.Marte.C.9924B27D	Avast	Win6
AVG	Win64:Evo-gen [Trj]	Avira (no cloud)	HEU
BitDefender	Generic.Trojan.Havokiz.Marte.C.9924B27D	ClamAV	Win.I
Cylance	Unsafe	Cynet	Malic
Cyren	W64/Ulise.EO.genIEldorado	DeepInstinct	MAL
DrWeb	BackDoor.Siggen2.4410	Elastic	Malic

That does look rather malicious, and it seems to have a name befitting the name of the challenge, Havoc.

After a bit of looking around with various search engines on what this malware is and how it behaves, two important links can be found.

For one, the actual [source code](#) of the malware is on Github, neat!

And possibly even better, a [blogpost on zscaler.com](#) about an attack involving just this malware.

In figure 28 of the blogpost, the metadata structure of the packets the malware sends is shown. When trying to find a TCP package that looks like this however, Wireshark will come up with no results.

All relevant traffic happens on port 80 and port 443. Since port 443 will be encrypted, all that is left is plain http traffic.

And that's exactly where we can find these packets. The blogpost explains that a magic value `0xdeadbeef` is at position 4 of the initialization package. In the later HTTP POST request, we can find just this pattern.

0000	50 4f 53 54 20 2f 73 74	61 74 75 73 2e 70 68 70	POST /st atus.php
0010	20 48 54 54 50 2f 31 2e	31 0d 0a 43 61 63 68 65	HTTP/1. 1 ··Cache
0020	2d 43 6f 6e 74 72 6f 6c	3a 20 6e 6f 2d 63 61 63	-Control : no-cac
0030	68 65 0d 0a 43 6f 6e 6e	65 63 74 69 6f 6e 3a 20	he ··Conn ection:
0040	4b 65 65 70 2d 41 6c 69	76 65 0d 0a 50 72 61 67	Keep-Ali ve ··Prag
0050	6d 61 3a 20 6e 6f 2d 63	61 63 68 65 0d 0a 43 6f	ma: no-c ache ··Co
0060	6e 74 65 6e 74 2d 54 79	70 65 3a 20 74 65 78 74	ntent-Ty pe: text
0070	2f 70 6c 61 69 6e 0d 0a	55 73 65 72 2d 41 67 65	/plain ·· User-Age
0080	6e 74 3a 20 4d 6f 7a 69	6c 6c 61 2f 35 2e 30 20	nt: Mozi lla/5.0
0090	28 57 69 6e 64 6f 77 73	20 4e 54 20 36 2e 31 3b	(Windows NT 6.1;
00a0	20 57 4f 57 36 34 29 20	41 70 70 6c 65 57 65 62	WOW64) AppleWeb
00b0	4b 69 74 2f 35 33 37 2e	33 36 20 28 4b 48 54 4d	Kit/537. 36 (KHTM
00c0	4c 2c 20 6c 69 6b 65 20	47 65 63 6b 6f 29 20 43	L, like Gecko) C
00d0	68 72 6f 6d 65 2f 39 36	2e 30 2e 34 36 36 34 2e	hrome/96 .0.4664.
00e0	31 31 30 20 53 61 66 61	72 69 2f 35 33 37 2e 33	110 Safa ri/537.3
00f0	36 0d 0a 43 6f 6e 74 65	6e 74 2d 4c 65 6e 67 74	6 ··Conte nt-Lengt
0100	68 3a 20 32 30 33 0d 0a	48 6f 73 74 3a 20 6b 65	h: 203 ·· Host: ke
0110	65 70 61 73 73 2e 78 79	7a 0d 0a 0d 0a 00 00 00	epass.xy z ···· ····
0120	c7 de ad be ef 5c d9 f7	4c 00 00 00 63 4a ba 74	····\··· L···cJ·t
0130	dc f2 86 fc 2e b4 66 5c	80 40 3e 76 9a 1c a8 00	·····f\ ·@>v····
0140	a4 da 9a 5c bc 36 9a 6c	22 8e 92 6c de 70 9c 2c	···\·6·l "··l·p·,

Marked in blue is the HTTP POST data, and in red the signature we were looking for.

This is package number 1716 in the attached PCAPNG file.

Supposedly, behind this signature there is supposed to be an AES key and IV.

This lines up with us not being able to read any of the data of this or subsequent packages.

We can also observe this in the source code of [Demon.c:129](#).

## Exploitation

This knowledge, when applied to package 1716, means that the key and IV are the following:

Key: 4aba74dcf286fc2eb4665c80403e769a1ca800a4da9a5cbc369a6c228e926cde

IV: 709c2c3a74a6580a722cacfc8c5e26ca

With these, we should be able to decrypt the data the malware sent, finding what was exfiltrated.

The last issue will be finding said data and also finding what type of AES was actually used.

According to figure 29 of the blogpost, the encrypted data should be right behind the IV.

But instead of finding which AES algorithm was used, I've decided to compile against Havoc to use their crypto library directly.

```

4  unsigned char KEY[] = {
5      74, 186, 116, 220, 242, 134, 252, 46,
6      180, 102, 92, 128, 64, 62, 118, 154,
7      28, 168, 0, 164, 218, 154, 92, 188,
8      54, 154, 108, 34, 142, 146, 108, 222
9  };
10
11 unsigned char IV[] = {
12     112, 156, 44, 58, 116, 166, 88, 10,
13     114, 44, 172, 252, 140, 94, 38, 202
14 };
15
16 unsigned char DATA[] = {
17     68, 247, 23, 41, 250, 195, 163, 95,
18     94, 56, 45, 249, 11, 33, 116, 47,
19     150, 160, 37, 19, 121, 170, 30, 51,
20     56, 107, 44, 136, 206, 243, 19, 25,
21     170, 115, 120, 143, 172, 115, 1, 215,
22     41, 173, 104, 22, 211, 170, 24, 193,
23     4, 124, 139, 203, 15, 204, 3, 11,
24     61, 16, 193, 252, 116, 28, 100, 162,
25     62, 44, 233, 42, 48, 47
26 };
27
28 int wmain(int argc, WCHAR* argv[])
29 {
30     AESCTX AesCtx = { 0 };
31
32     AesInit( &AesCtx, KEY, IV );
33     AesXCryptBuffer( &AesCtx, DATA, 70 );
34
35     for (int i = 0; i < 70; i++) {
36         printf("%02X", DATA[i]);
37     }
38
39     printf("\n");
40 }

```

This snippet of code uses the key and IV we found in number form, along with the DATA variable that contains any encrypted data we want to decrypt.

We can now try to run several of the encrypted packages through it.

And with one of the later packages, that being number 1839, the program yields us the flag.

```
sw1tchbl4d3@pts/10 [~/githubs/Havoc/Teamserver/data/implants/Demon/Source/Crypt] %  
x86_64-w64-mingw32-gcc AesCrypt.c test.c -o aes -I../Include/ 2>/dev/null  
sw1tchbl4d3@pts/10 [~/githubs/Havoc/Teamserver/data/implants/Demon/Source/Crypt] %  
wine aes.exe  
0000000A00000001866006C00610067002E007400780074002E0074007800740000000022435343477B3  
16E35336375  
72335F5472346E35703072745F336E6372707431306E7D  
[10] sw1tchbl4d3@pts/10 [~/githubs/Havoc/Teamserver/data/implants/Demon/Source/Cryp  
t] % python3  
Python 3.10.9 (main, Dec 25 2022, 21:29:15) [GCC 12.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import binascii  
>>> binascii.unhexlify("0000000A00000001866006C00610067002E007400780074002E007400780  
07400000000022435343477B316E3533637572335F5472346E35703072745F336E6372707431306E7D")  
b'\x00\x00\x00\n\x00\x00\x00\x18f\x00l\x00a\x00g\x00.\x00t\x00x\x00t\x00.\x00t\x00x  
\x00t\x00\x00\x00"CSCG{1n53cur3_Tr4n5p0rt_3ncrpt10n}'  
>>> □
```

## Mitigations

On the attacker's side: Don't exchange symmetric keys through an unsecured channel.

To make this challenge unsolvable, the attacker could pregenerate a secure, asymmetric key pair, keeping the private key secret, and encrypting all data with a public key embedded into the program.

This could be done with RSA, Elliptic curves and more asymmetric cryptography algorithms.

On the defender's side, things could've been mitigated by thorough teaching of phishing methods. Anyone with a higher role in such a system should be taught how to identify phishing sites to avoid having critical data get stolen.

~sw1tchbl4d3, 01/05/2023 (dd/mm/yyyy)