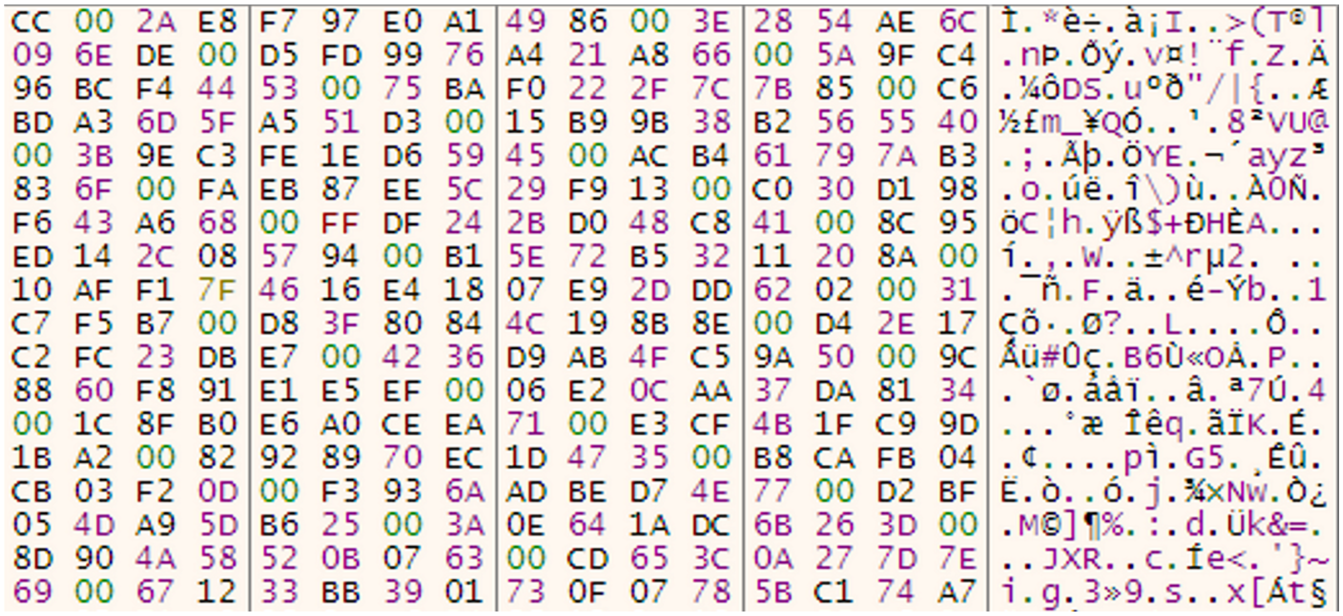
Keyhole Analysis. By: Joshua Platt, Jonathan McCay and… | by Jason Reaves | Walmart Global Tech Blog | Jan, 2024

medium.com/walmartglobaltech/keyhole-analysis-60302922aa03

Jason Reaves January 16, 2024



```
CC 00 2A E8 F7 97 E0 A1 49 86 00 3E 28 54 AE 6C Ì.*è÷.à¡I..>(T®l
09 6E DE 00 D5 FD 99 76 A4 21 A8 66 00 5A 9F C4 .nÞ.Õý.v¤!¨f.Z.Ä
96 BC F4 44 53 00 75 BA F0 22 2F 7C 7B 85 00 C6 .¼ôDS.uºð"/|{..Æ
BD A3 6D 5F A5 51 D3 00 15 B9 9B 38 B2 56 55 40 ½£m_¥QÓ..¹.8²VU@
00 3B 9E C3 FE 1E D6 59 45 00 AC B4 61 79 7A B3 .;.Ãþ.ÖYE.¬´ayz³
83 6F 00 FA EB 87 EE 5C 29 F9 13 00 C0 30 D1 98 .o.úë.î\)ù..À0Ñ.
F6 43 A6 68 00 FF DF 24 2B D0 48 C8 41 00 8C 95 öC¦h.ÿß$+ÐHÈA...
ED 14 2C 08 57 94 00 B1 5E 72 B5 32 11 20 8A 00 í.,.W..±^rµ2. ..
10 AF F1 7F 46 16 E4 18 07 E9 2D DD 62 02 00 31 .¯ñ.F.ä..é-Ýb..1
C7 F5 B7 00 D8 3F 80 84 4C 19 8B 8E 00 D4 2E 17 Çõ·.Ø?..L....Ô..
C2 FC 23 DB E7 00 42 36 D9 AB 4F C5 9A 50 00 9C Âü#Ûç.B6Ù«OÅ.P..
88 60 F8 91 E1 E5 EF 00 06 E2 0C AA 37 DA 81 34 .`ø.áåï..â.ª7Ú.4
00 1C 8F B0 E6 A0 CE EA 71 00 E3 CF 4B 1F C9 9D ...°æ Îêq.ãÏK.É.
1B A2 00 82 92 89 70 EC 1D 47 35 00 B8 CA FB 04 .¢...pì.G5.¸Êû.
CB 03 F2 0D 00 F3 93 6A AD BE D7 4E 77 00 D2 BF Ë.ò..ó.j.¾×Nw.Ò¿
05 4D A9 5D B6 25 00 3A 0E 64 1A DC 6B 26 3D 00 .M©]¶%.:.d.Ük&=.
8D 90 4A 58 52 0B 07 63 00 CD 65 3C 0A 27 7D 7E ..JXR..c.Íe<.'}~
69 00 67 12 33 BB 39 01 73 0F 07 78 5B C1 74 A7 i.g.3»9.s..x[Át§
```

Keyhole Analysis

--

By: Joshua Platt, Jonathan McCay and Jason Reaves

Keyhole is a multi-functional VNC/Backconnect component used extensively by IcedID/Anubis. While the malware contains functionality that has been previously reported on as typical VNC and HDESK capabilities, a general lack of technical information appears to exist around some of the expanded functionality currently present. In fact, the functionality we mapped out for the main Keyhole component rivals that of IcedID itself:

- Collect system information
- VNC
- HDESK
- Socks/Backconnect
- Console command detonation via cmd.exe or powershell
- Multiple methods of injecting explorer.exe
- LDAP queries
- File retrieval from infected systems
- Hijacking browser profiles
- Deleting browser profiles
- Lower security of running browsers via command line manipulation
- Checking for webcam existence
- Taking pictures with webcam
- Turning on microphone in registry for apps
- Enumerating servers and shares in network

Portions of this functionality are spread out over multiple open-source reports but significant analysis appears to be missing on several noteworthy improvements.

Technical Overview

Initial loader piece:

The initial component involves a loader that can handle both PE files or shellcode versions of the main component; they also can utilize a custom PE loader with mangled headers. Ultimately the loader is responsible for decoding and loading the core module and executing it properly.

Decoding Core Module:

The core module is encoded using a 256 byte array of swap values, (key). After locating the initial blob containing the key values, a parsing algorithm is implemented to create the array.

Unparsed Key Blob:

An integer is given to decide how many bytes to pull from the unparsed blob. This integer will double in size until the value exceeds 256. Adding 1 to the amount of times the integer could double before exceeding 256 will be the number of bytes pulled.

Finding an integer to decide how many bytes to pull that is not 0 will result in different behavior. The first time this happens, the appropriate amount of bytes will be added followed by the addition of a null byte. All bytes until the next 0 will be grabbed, and the process repeats until the key len is 256.

Python: Parse Key

Parsed Key / Encoded Core Module

To decode the core module, each byte in the encoded module will be used as an index value. The byte in the key at that index will replace the encoded byte.

Python: Decode Core Module

Subset of Strings in Decoded Core Module

```
user32.dllbad pathnamepath not foundDefaultTOP WNDRtlExitUserProcessMS Shell Dlgalready
existsexplorer.exemovetestauth-err-runasinvokermkdirdiskbusy__compat_layerinvalid namehcddiraction not
foundAllowCREATEAD not foundValueHIDESettingsWINMM.DLL2500disk not foundcombase.dllMOVEdivice not
readedmdiclientf%0.8XUser32.dllLucida Consolentdll.dllmemory allocshell32.dllaccess deniedopenerror{%0.8X-
%0.4X-%0.4X-%0.4X-
%0.4X%0.8X}abcedfikmnopsutwSelectObjectGetCurrentObjectGetObjectACreateCompatibleDCDeleteDCCreateDIBSectionCre
 Definition Audio
```

Main component:

The main component of Keyhole can contain encoded strings, the really interesting part is that the algorithm used is the same as a sample I analyzed in June of 2017:

1cd6f992fbeee0a66e1c329e15db71fe891ae0e845867d6d30df867babe5bed6

Old IcedID string decoding routine:

Keyhole string decoding routine:

Decoding the strings in Keyhole took only a small edit in my old IcedID IDA script:

```
def gen_key(k):
 return(((k << 0x1d) | (k >> 3)) & 0xffffffff)



for addr in XrefsTo(0x322630, flags=0):
 addr = addr.frm
 addr = idc.PrevHead(addr)
 while GetMnem(addr) != "push":
  addr = idc.PrevHead(addr)
 print(hex(addr))
 data_addr = GetOperandValue(addr,0)

  xork_init = Dword(data_addr) data_addr += 4 length_delta = Word(data_addr) data_addr += 2 length =
(xork_init ^ length_delta) & 0xffff out = "" xork = xork_init for i in range(length):  xork = gen_key(xork)
xork += i  out += chr((Byte(data_addr) ^ (xork & 0xFF)) & 0xFF)  data_addr += 1 if out[-2:] == '\x00\x00':
print(out.decode('utf16'))  MakeRptCmt(addr, out.decode('utf16').encode('ascii')) else:  print(out)
MakeRptCmt(addr, out)
```

The main component of Keyhole also expects a parameter to be passed, this address will be the shellcode blob from the previous loader component.

First a DWORD value is pulled out, which will eventually be used as the initial seed value for the XOR loop:

Next every byte after the DWORD is pulled out and run through a XOR loop where the initial seed value is manipulated every iteration, we can also see below that the blob being decoded is 0x14e bytes in length:

The decoded blob turns out to be the config for the main Keyhole component, so the loader component passes the config in itself off to the main component as a parameter. After decoding we have a few values that will be used for C2 communications:

```
['', '', '', '', '\xbb\xa2\xbd\x9e\x1f\x8b\x08\x08', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
<..snip..>'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'hdesk', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '91.238.]50.101', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '\x90\x1f’]
```

A few values we can see, especially the C2 address but also the C2 traffic marker of ‘\x1f\x8b\x08\x08’ and the port 0x1f90 which is 8080.

Functionality

As mentioned the malware will retrieve some basic information about the infected system:

It can also enumerate servers and shares in the network and leverages LDAP queries in the process:

File retrievals are limited in size to roughly 33MB:

Browsers

For Chrome the module can copy the User Data to a new folder in temp:

Along with the parameter of ‘C\\’ we can see below that the folder structure will be built under temp:

In this case with a flag value of 0 we end up having the profile copied to:

```
%TEMP%\D\C\
```

Browser are manipulated in a similar way that old banking trojans used to do it, the module contains code for hooking NtCreateUserProcess which will then look for browsers to be spawned:

This lets the core module manipulate the command lines of the browsers, however it does more depending on which browser is found to be executing.

Chrome

```
- Copies profile data to  - %TEMP%\D\C\- Modifies command line  - --user-data-dir=”  - " --ash-force-desktop
--disable-3d-apis --disable-accelerated-layers --disable-accelerated-plugins --disable-audio --disable-gpu --
disable-d3d11  --disable-d3d12 --disable-accelerated-2d-canvas --disable-deadline-scheduling --disable-ui-
deadline-scheduling --aura-no-shadows  --no-zygote --do-not-de-elevate --mute-audio --disable-renderer-
accessibility --disable-direct-compositing  --disable-gpu-compositing --disable-renderer-backgrounding --in-
process-gpu
```

FireFox

```
- Reads default profile path from profiles.ini file- Copies default profile to  - %TEMP%\D\F\- Edits prefs.js
- user_pref("browser.preferences.defaultPerformanceSettings.enabled", false);  -
user_pref("layers.acceleration.disabled", true);- Modifies command line  - --no-remote -no-deelevate -profile
“
```

Edge

```
- Sets registry keys  - SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers   - C:\Program
Files (x86)\Microsoft\Edge\Application\msedge.exe=~ WIN8RTM- Copies profile data to  - %TEMP%\D\E\- Modifies
command line  - --user-data-dir=”  - " --ash-force-desktop --disable-3d-apis --disable-accelerated-layers --
disable-accelerated-plugins --disable-audio --disable-gpu --disable-d3d11  --disable-d3d12 --disable-
accelerated-2d-canvas --disable-deadline-scheduling --disable-ui-deadline-scheduling --aura-no-shadows  --no-
zygote --do-not-de-elevate --mute-audio --disable-renderer-accessibility --disable-direct-compositing  --
disable-gpu-compositing --disable-renderer-backgrounding --in-process-gpu
```

Internet Explorer

```
- Modifies registry  - Software\Microsoft\Internet Explorer\Main   - NoProtectedModeBanner=1   -
TabProcGrowth=0  - Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3   - 2500=3- Modifies
command line  - -nomerge -noframemerging -nohome
```

Keyhole can also perform injection into newly started explorer processes in a variety of ways:

For BackConnect commands:

For console command line detonation a string is expected for either powershell or cmd detonation:

Also above it can be seen for commands related to the microphone, these are primarily related to manipulating registry values:

After the modifications the explorer will be restarted.

- SOFTWARE\Microsoft\Windows\CurrentVersion\CapabilityAccessManager\ConsentStore\microphone\NonPackaged
- SOFTWARE\Microsoft\Windows\CurrentVersion\CapabilityAccessManager\ConsentStore\microphone
- SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\StuckRects3
- cmd.exe /c taskkill /f /im explorer.exe && start explorer.exe

YARA

```
rule keyhole_32
{
strings:
$config_decode = {694d0cfd43030081c1c39e2600894d0c}
condition:
all of them
}
```

```
rule keyhole_loader{strings:$64exe = {5390 9cbb be0c c453 9d5b 5290}$64dll = {7809 7407 7305 eb03 3941}$32exe
= {770c 569c c1e6 0ac1 ee09 f7d6}$32dll = {5781 f7fc 46d8 5083 c728 bf0b}condition:any of them}
```

IOCS

Sample hash

```
74aa61cc1157529fb98b757fb879616ffc2b54e4d4ff08c9b9d5b6dcec868c2a
```

Command Line

```
powershell.exe  -c "[Console]::OutputEncoding = [Console]::InputEncoding =
[System.Text.Encoding]::GetEncoding('utf-8'); cd c:\; powershell"
```

```
cmd.exe /K chcp 65001 && c: && cd c:\
```

```
cmd.exe /c taskkill /f /im explorer.exe && start explorer.exe
```

```
cmd.exe /c start /wait explorer.exe /factory, {75dff2b7-6936-4c06-a8bb-676a7b00b24b}
```

```
explorer.exe shell:mycomputerfolder
```

```
explorer.exe  shell:mycomputerfolder
```

```
Browsers with params:
--user-data-dir="
```

```
" --ash-force-desktop --disable-3d-apis --disable-accelerated-layers --disable-accelerated-plugins --disable-
audio --disable-gpu --disable-d3d11  --disable-d3d12 --disable-accelerated-2d-canvas --disable-deadline-
scheduling --disable-ui-deadline-scheduling --aura-no-shadows  --no-zygote --do-not-de-elevate --mute-audio -
-disable-renderer-accessibility --disable-direct-compositing  --disable-gpu-compositing --disable-renderer-
backgrounding --in-process-gpu
```

```
--no-remote -no-deelevate -profile
```

File activity:

```
%TEMP%\D\C\%TEMP%\D\F\%TEMP%\D\E\
```

References