MALWARE

Upatre Continued to Evolve with new Anti-Analysis Techniques

9 min read

RELATED PRODUCTS

Advanced WildFire

Cortex XDR

By: Mike Harbison, Brittany Barbehenn

Published: July 13, 2018

Categories: Cybercrime, Malware, Threat Research

Tags: Dot-bit, Downloader, Namecoin, Upatre

Share

First discovered in 2013, Upatre is primarily a downloader tool responsible for delivering additional trojans onto the victim host. It is most well-known for being tied with the Dyre banking trojan, with a peak of over 250,000 Upatre infections per month delivering Dyre back in July 2015. In November 2015 however, an organization thought to be associated with the Dyre operation was raided, and subsequently the usage of Upatre delivering Dyre dropped dramatically, to less than 600 per month by January 2016.

Today, the Upatre downloader tool is effectively no longer in use by criminal organizations. However, one of the many interesting aspects of the Upatre tool had always been its constant adaptive nature where the developers continuously added features and capabilities to the tool to increase its efficacy.

In March 2018, Unit 42 researchers collected a sample of Upatre which was compiled in December 2016 but at the time was largely undetectable by most automated detection systems . Because of this, we analyzed the sample to afford awareness to those interested in this malware and its evolution. This previously undocumented variant features significant code flow obscuration, a pro re nata means of decryption for network communications, and of particular interest, the method in which this variant evades virtual machine detection.

In this post we highlight these techniques identified during our analysis.

Malware Overview

Upatre is a stage-0 malware, which basically means it's a downloader. The malware is used to download and install a payload onto the affected system. The payload is retrieved from hardcoded domain(s) and is typically another piece of malware. Historically, Upatre has acted as a downloader for malware families such as **Dyre**, **GameOver Zeus**, **Kegotip, Locky, and Dridex** to name a few. However, in this case no payload was delivered. Additionally, variants such as this one collect information from the target and transmit the data via an HTTP POST request.

This newly observed variant comes packed with several characteristics and capabilities that stood out to us during analysis. Attributes in the PE header suggest that the malware is written in Visual C++ and several of the PE sections have high entropy classification, which indicates that the binary is packed. The PE resource section also contains images of Google Chrome, so when the binary is placed on the target machine, it appears to be that of the Google Chrome web browser.

One of the key features about this variant that stood out during our analysis is how it detects whether or not it is running within a virtual machine. Although virtual machine detection is anything but new, in this variant, it is handled a bit differently than **other samples** previously analyzed by Unit 42. To, evade detection, the newly observed variant enumerates the running processes on the host, generates a CRC32 hash of the process name, performs an XOR with a hard-coded key of 0x0F27DC411, and finally compares the newly computed value against a list of values stored in an array within the code. We observed the following values:

0x6BA08023	0xDFF859A5	0x9649C9DF	0x91B88065	0xF663B61C
0xC6E1589A	0xC63B2FDF	0xA9D475EF	0xCE9F7AE2	0xCF3B343A

RELATED ARTICLES

Script-Based Malware: A New Attacker Trend on Internet Explorer

Tracking Subaat: Targeted Phishing Attack Leads to Threat Actor's Repository

Upatre: Old Dog, New [Anti-Analysis] Tricks

This version of Upatre will **not** transmit any data via HTTP POST to any of the target domains if one of these values is found. In the event one of the values are found, the malware will sleep for six seconds and then will restart the entire check again. We were unable to determine every corresponding process name from the CRC32 list above, however, we were able to decipher the following process names:

Process Name	CRC32
vmtoolsd.exe	0xD5F11B49
vmacthlp.exe	0x403C2A93
Python.exe	0x209202D5

Other notable functionality of this new version of the Upatre malware includes:

- In-memory loading of code
- Disables the following Windows services:
  - Windows Security Center
  - Internet Connection Sharing
  - Windows Firewall
  - Windows Defender
  - Windows Update
  - Windows Defender Network Inspection Service
- Disables Windows security notification balloons on Windows 7 and up
- Disables Internet Explorer Phishing Filter
- Disables Windows User Access Control Notifications
- Launches a trusted Windows application msiexec.exe and injects code into its memory space using an undocumented technique
- Heavy use of obfuscated and optimized code to thwart code analysis
- Use of non-essential Windows API's for stack pivoting to mask intended API
- Multiple layers of custom encoding used for individual strings decoding. Does not share encoding routine with other encoded values

Network Communication

Another feature of this sample is the use of top level domains (TLD) of. bit. The intended domains are encrypted and only decrypted when the malware is ready to use them. This new sample attempts to resolve two domains, bookreader[.]bit and doghunter[.]bit via the following hardcoded DNS Servers:

- 31.3.135[.]232
- 193.183.98[.]154
- 5.135.183[.]146
- 84.201.32[.]108
- 185.133.72[.]100
- 96.90.175[.]167
- 104.238.186[.]189

DNS resolution for .bit domains use hardcoded DNS servers and is handled via TCP versus traditional UDP. This is because .bit domains are based on Namecoin and aren't regulated by ICANN. Additionally, the hardcoded DNS server IPs we identified in this sample are all associated with OpenNIC Public DNS servers.
According to **OpenNIC**, when using OpenNIC DNS servers .bit domains are resolved through centralized servers that generate a DNS zone from the Namecoin blockchain; therefore, the secure nature of using Namecoin as a decentralized means of DNS is not actually being utilized here.

```
jKß±¯Z].613ŒÐ.|XÝõÿÊý;nU0Mw,M»«1/2:Û?/æ.$)¶ì_X.p.q.8.Ÿj€¬.æàï..B·âožM.êÅb¥8aÕù©s$ÏmhT£»¤/wwÆ6
y=€ƒÖG™Úu yÍÆ2".® !BùìâÃ.=¦6<)°
5M´yótI\
```


Note, the HTTP POST does not contain any User-Agent strings.

At this time, we don't fully understand the encryption method; however, we know that the data sent in the POST request is encrypted using a custom encryption algorithm. Below is an example of data captured prior to encryption:


```
00000000   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000010   00 00 00 00 60 00 00 00  00 00 00 00 01 00 00 00   ....`...........
00000020   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000030   11 27 00 00 00 00 00 00  20 00 00 00 20 00 00 00   .'...... ... ...
00000040   57 49 4E 2D 52 49 38 38  4C 38 56 4D 45 38 4D 5F   WIN-RI88L8VME8M_
00000050   45 35 33 32 36 34 38 41  37 36 31 46 34 43 35 41   E532648A761F4C5A
00000060   AB AB AB AB AB AB AB AB  00 00 00 00 00 00 00 00   ««««««««........
00000070   8B BB 9F 39 09 63 00 00  30 28 31 00 08 21 31 00   ‹»Ÿ9.c..0(1..!1.
00000080   EE FE EE FE EE FE EE FE  EE FE EE FE EE FE EE FE   îþîþîþîþîþîþîþîþ
00000090   EE FE EE FE EE FE EE FE  EE FE EE FE EE FE EE FE   îþîþîþîþîþîþîþîþ
000000A0   9F BB 9C 2E 00 63 00 18  70 1B 31 00 60 21 31 00   Ÿ»œ..c..p.1.`!1.
000000B0   78 1B 31 00 68 21 31 00  70 21 31 00 80 CF 2B 00   x.1.h!1.p!1.€Ï+.
000000C0   00 00 1D 74 8D 12 1D 74  00 70 00 00 3C 00 3E 00   ...t...t.p..<.>.
000000D0   38 20 31 00 14 00 16 00  60 20 31 00 04 40 0C 00   8 1.....` 1..@..
000000E0   01 00 00 00 B0 A6 BA 77  BC 38 28 00 43 DB 5B 4A   ....°¦ºw1/48(.CÛ[J
000000F0   00 00 00 00 00 00 00 00  F8 1F 31 00 F8 1F 31 00   ........ø.1.ø.1.
00000100   00 20 31 00 00 20 31 00  38 23 31 00 90 20 31 00   . 1.. 1.8#1.. 1.
00000110   54 05 B4 77 00 00 D9 3F  D5 8A 48 B8 82 D1 D3 01   T.´w..Ù?ÕŠH¸‚ÑÓ.
00000120   AB AB AB AB AB AB AB AB  00 00 00 00 00 00 00 00   ««««««««........
00000130   86 BB 9C 37 14 63 00 1A  43 00 3A 00 5C 00 57 00   †»œ7.c..C.:.\.W.
00000140   69 00 6E 00 64 00 6F 00  77 00 73 00 5C 00 73 00   i.n.d.o.w.s.\.s.
00000150   79 00 73 00 74 00 65 00  6D 00 33 00 32 00 5C 00   y.s.t.e.m.3.2.\.
00000160   57 00 49 00 4E 00 4E 00  53 00 49 00 2E 00 44 00   W.I.N.N.S.I...D.
```


Obscuring Code Flow

This version of Upatre contains significantly obfuscated code to increase the difficulty of analysis. Figure 1 below shows an example API call disassembled in IDA Pro.

Upatre_1

Figure 1-IDA Disassembly of API call

For conventional naming, the function at address 0x00137ED6 has been renamed to the Windows API **RegQueryValueEx**_0. According to MSDN this function takes six parameters, the frame pointer is ESP based and the stack frame would resemble the following:

Upatre_2

Figure 2-Inside Func_RegQueryvalueEx_0

In the above figure, Func_RegQueryValueEx_0 is EBP based and performs the following:

- Saves the current stack pointer in EBP
- The stack pointer is adjusted 268 bytes (thwarting stack frame analysis)
- Pushes a pointer, which points to the REGKEY string

After the call into sub_140CBE the stack would resemble the following:

Upatre_3

Figure 3--Inside Sub_140CBE

Function Sub_140CBE does the following:

- Pushes 0x13 on the stack
- Calls another function, which ends up jumping into the Windows API GetSystemMetrics

0x13 is the SM_CSURSOR index used by GetSystemMetrics, which returns the width of a cursor in pixels. Retrieving this value has **no** bearing on the program as the return value is not used.
How the stack looks after the call to func_GetSytemMetrics

Upatre_4

Figure 4--Inside Func_GetSystemMetrics

Some interesting observations about this function:

- The JMP instruction is used versus the CALL instruction as JMP doesn't affect the stack.
- The two PUSH instructions are junk values and only used to pivot the stack, so the correct return address is on the stack during the return.

Here is how the stack looks prior to the jump:

Upatre_5

Return address **0x001414FD** is the address that is used to open and query the hosts registry, and this is the target address after executing the above instructions. The return code flow is as follows:

1. The two junk data values pushed on the stack are cleared during the executing of the GetSystemMetrics API.
2. The stack pointer is incremented past 0x13
3. Address 0x00140CC5 has a retn instruction
4. Address 0x001414FD is now on the top of the stack and the section within the malware that handles Windows registry enumeration is called (RegQueryValueEx).

This stack pivot is performed entirely to make static analysis of the file more difficult, but the end result is still that the API function executes, and the malware accomplishes its task.

Persistence Technique

To establish persistence, this new version of Upatre creates the following registry key:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run\

- String value-->x$msbuild where x$ is a random alpha character. *Note: the name of the binary depends on the executable that is running. The stub program grabs the Windows name of the EXE and prepends it with a random value.*
- Data -->C:\ProgramData\MSBuild\x$exe

we were never able to capture the ultimate payload for this new Upatre variant. However, open source analysis on this variant identified another sample configured with the same dot-bit domains. The sample, 94a8b4b22dab4171edde5b1bafbf2f17dbe3c3c4c01335c36ba3b6e5d3635b83, was compiled six days after our Upatre sample and delivered the Chthonic banking trojan via RIG exploit kit.

Although the delivery mechanism was not observed during our analysis, Upatre typically arrives via an email link/attachment or through a compromised website.

Defending Against this Threat

The Upatre malware is constantly changing and is capable of downloading many different malware families, some, destructive. Using threat detection and prevention solutions such as the Palo Alto Networks next-generation security platform are highly recommended as part of a proactive cyber security strategy. WildFire and Traps both detect the samples described in this report as malicious.

Not all dot-bit domains are malicious, but organizations should take steps to ensure they can control access to all potentially malicious domains. Blocking outbound access to DNS servers and re-routing DNS requests to internally controlled DNS servers can help protect a network from malware using dot-bit domains provided by the Namecoin network.

Palo Alto Networks customers remain protected from Upatre and can identify this threat using the **Upatre** tag in AutoFocus.

Indicators of compromise associated with this analysis include:

Upatre

SHA256: 8ac7909730269d62efaf898d1a5e87251aadccf4349cd95564ad6a3634ba4ef4

Cthonic

SHA256: 94a8b4b22dab4171edde5b1bafbf2f17dbe3c3c4c01335c36ba3b6e5d3635b83

C2s

Domain: doghunter[.]bit
Domain: bookreader[.]bit
IP Address: 31.3.135[.]232
IP Address: 193.183.98[.]154
IP Address: 5.135.183[.]146
IP Address: 84.201.32[.]108
IP Address: 185.133.72[.]100
IP Address: 96.90.175[.]167
IP Address: 104.238.186[.]189

Updated on 7/13/2018 to clarify that the Upatre sample discussed was compiled in 2016 but is newly discovered in 2018 and to more clearly identify samples with their hashes.

Back to top

TAGS

Dot-bit | Downloader | Namecoin | Upatre

Threat Research Center

Next: Threat Brief: Why You Need to Be Careful of Links in Email

Related Resources

THREAT RESEARCH

THREAT RESEARCH

THREAT RESEARCH

THREAT ACTOR GROUPS