Malware-Analysis-Reports/README.md at master · Finch4/Malware-Analysis-Reports · GitHub

github.com/Finch4/Malware-Analysis-Reports/blob/main/13e0f258cfbe3aece8a7e6d29ceb5697/README.md

Finch4

Finch4/**Malware-Analysis-Reports**

My Malware Analysis Reports

1 Contributor | 0 Issues | 11 Stars | 1 Fork

Malware Analysis Report N°2

(Analysis of BitRat will be soon written, this is the analysis of the dropper)

Date: 21/01/2021

Author/s: Finch

Summary of the analysis

Key observation

The sample try to compromise the analysis by looking as a benign executable. All the malicious actions are based on the resources of the executable.

Indicators of compromise

Drop `ywlpCPZYAwl.exe` in `%appdata%/Roaming`

Drop `tmpC51F.tmp` in `%appdata%/Roaming`

`RegSvcs.exe` running lonely and always.

Delivery method

The delivery is made by email

Email example(Thanks to abuse_ch):

Malspam distributing BitRAT:

HELO: mxout.fullmarket-4.vautronserver.de

Sending IP: 151.252.48.227

From: Accounts Payable - Rinaldi finance@chalet-almhuette.at

Reply-To: z0ais@newpacifis.com

Subject: Re: Re: Re: Payment processed (Overdues)

Attachment: Payment_receipt.img (contains "Payment Confirmation Paper - Customer Copy_pdf.exe")

BitRAT C2: 195.206.105.10:3988

Sample identification

File name, type, size

File name: Payment Confirmation Paper - Customer Copy_pdf.exe

Type: .NET

Size: 4.39MB (4,607,488 bytes)

Size on disk: 4.39MB (4,608,000 bytes)

TargetFramework: .NETFramework,Version=v4.0

Assembly title: Lerlibro INC

Assembly company:

Assembly product: Lerlibro INC

File hashes

[Executable]

MD5: 13e0f258cfbe3aece8a7e6d29ceb5697

SHA1: 5890091bacde4d9d62ed76d32dfaefcaa5b988a4

SHA256: 76e5467f267a4bca00af800094c3a92f6bd51de54737f07c533d091e2f219b40

SHA384:
076302660926159d03b133500395d172ee7269491b173ef1aee002c179c0b612af161be3980191a569d9157e30627084

SSDEEP: 98304:1Unj6PEASk4gI/UqE2mCAc1XdZ2aRmPCBvfq:1U+PEZkFIMX2mbcrFBC

human-hash: lemon-zebra-robert-paris

Anti-virus indentifiers

BitRat

Dependencies

Supported OS

.NET [The target is Windows]

Required libraries

.NET

Configuration file/s

None

Script/s and Executable/s

The loaded executables are obfuscated with various techniques.

Network traffic

[Taken from JoeSandbox at the moment]

bita.plumfixa.com

149.154.167.220:443

myexternalip.com

216.239.32.21 216.239.32.21

Sample identification

Infection capabilities

Drop BitRat, Rat

Self-preservation capacity

Tries to detect virtual machine. Scheduled Task. Various encryption. [See images for everything, unfortunately for a few images I didn't take some return values] Process injection to `RegScvs.exe` **[C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegSvcs.exe] MD5**: 2867A3817C9245F7CF518524DFD18F28

Data leakage abilities

[Analysis of dropper]

Remote attacker interactions / C&C

[Taken from JoeSandbox at the moment]

api.telegram.org

`https://api.telegram.org/bot1529818621:AAH0oTC63FmKKTVji9OyVqkhJRsIx42aBDU/getUpdates?offset=-56EBB3`

`https://api.telegram.org/bot1529818621:AAH0oTC63FmKKTVji9OyVqkhJRsIx42aBDU/getUpdates?offset=-5BkIDB`

`https://api.telegram.org/bot1529818621:AAH0oTC63FmKKTVji9OyVqkhJRsIx42aBDU/getUpdates?offset=-5DU/ge`

`https://api.telegram.org/bot1529818621:AAH0oTC63FmKKTVji9OyVqkhJRsIx42aBDU/getUpdates?offset=-5FmKKT`

`https://api.telegram.org/bot1529818621:AAH0oTC63FmKKTVji9OyVqkhJRsIx42aBDU/getUpdates?offset=-5LMEM`

`https://api.telegram.org/bot1529818621:AAH0oTC63FmKKTVji9OyVqkhJRsIx42aBDU/getUpdates?offset=-5TVjiC`

Observations

Behavioral analysis

Executables dropping and decrypting. Invoke. Load Assembly. Windows api imports. Drop `\AppData\Local\Temp\tmpC51F.tmp` [Persistence] , `\AppData\Roaming\ywlpCPZYAwl.exe` [To call a method from the first sample that import ResumeThread]. Vm detection by:

> See screenshots section

Process Injection [**Target**: RegScvs.exe]

Code analysis

The code at first look seems a normal and benign executable. Looking at the resources we find two suspicious samples:

- An image named: "StringComparison"
- A string named: "UrlIdentityPermission" [Gzip compressed]

The method with the role of initializing the malicious behaviour is:

> `object WeakReference()` contained in the class `ProgIdRedirectionEntry()` contained in the namespace `Lerlibro_INC` The payload is first decoded using the function ArgCount [GZip decompress]

Dynamic code analysis

After decoded the assembly will be loaded by the Assembly.Load() method, `GetType("System.Activator").GetMethods()[2].Invoke()` will be called, so the malware will try to invoke a method and passing as parameters `ConcurrentSet.ReadOnlyList` , `ConcurrentSet.PropagationFlags` and `Lerlibro_INC` -> where `ConcurrentSet.ReadOnlyList` is `StringComparison` and `ConcurrentSet.PropagationFlags` is `kZYcCfRI` we will see the the method `tR` accept three strings parameters and will perform the extraction of the payload from `StringComparison` and load it. [See screenshots for more dynamic code analysis]

Memory analysis

None

Supporting figures

Logs

Strings

Function listings

- `KjL()` is the first function with the role of detecting the virtual machine.
- `flag11` will contain the result of `KjL()`.
- `org.a6()` the function will call CompareString to compare your username, with the attempt to detect the virtual machine.
- `or.VD()` the function will call Contains to compare the sample name with the attempt to detect an analysis environment.
- `dP.djP()` another function to detect virtual machine.

Screenshots

[Not everything is in order]

[Missing arguments]

[Missing arguments]

[The screenshot is called trick for the `!=`, it will return `true` if is not detected]

[See `vmdetectionsnippet.cpp` to understand]

[Autoit detection]

[See `vmdetectionsnippet2.cpp` to understand]

[if `flag11 == true`, vm detected]

[`case 0 == vmdetected`]

[Video about the function `nmI`, incoming]

Additional comments

The dropper is similar to others seen used for others differents malware.