PureLocker: New Ransomware-as-a-Service Being Used in Targeted Attacks Against Servers

intezer.com/blog-purelocker-ransomware-being-used-in-targeted-attacks-against-servers/

November 12, 2019

Written by Michael Kajiloti - 12 November 2019

Get Free Account

Join Now

Analysis by Intezer and IBM X-Force points its origins to a Malware-as-a-Service (MaaS) provider utilized by the Cobalt Gang and FIN6 attack groups

This is a mutual research between Intezer and IBM's X-Force IRIS team

We have found a new and undetected ransomware threat that is being used for targeted attacks against production servers of enterprises. Using code reuse analysis, we discovered this threat is closely related to the "**more_eggs"** backdoor malware, which is sold on the dark web by a veteran MaaS provider and has been used by the Cobalt Gang, FIN6, and other threat groups.

While the samples we analyzed are for the Windows platform, we have noticed that the group operating this ransomware is also employing a Linux variant in order to attack the Linux infrastructure of its targets.

We have named this ransomware **PureLocker** because it's written in the PureBasic programming language. As part of our analysis, we have identified the evasion methods and design features that have allowed this ransomware to remain under the radar for several months. Below we present our findings through a technical analysis of the malware samples.

Initial Analysis

The Windows sample we analyzed is a 32-bit DLL, masquerading as a C++ cryptography library called Crypto++:

File Version Information

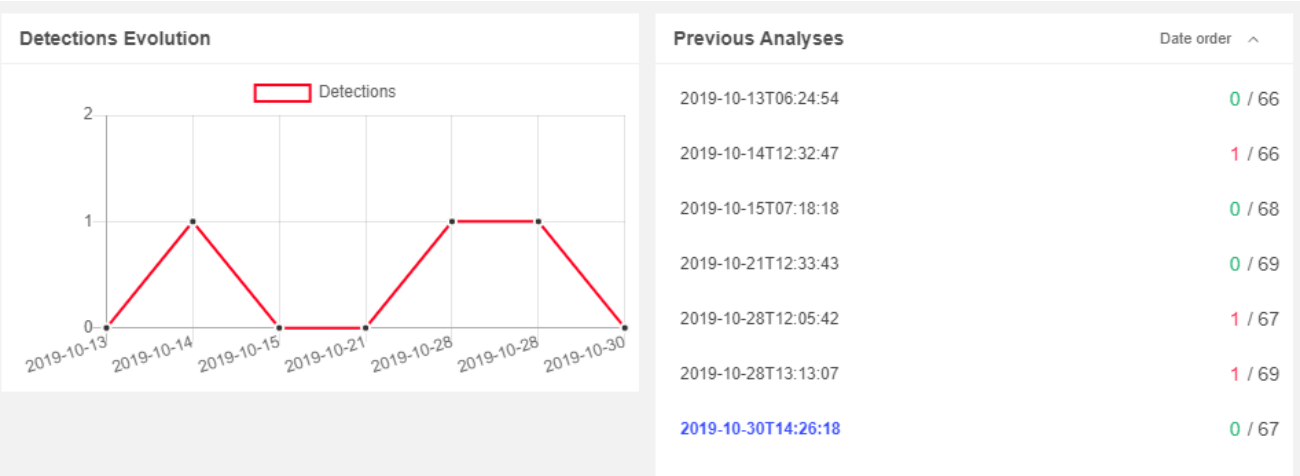
Copyright	Copyright © 1995-2006 by Wei Dai
Product	Crypto++® Library
Description	Crypto++® Library DLL
Original Name	cryptopp.dll
Internal Name	cryptopp
File Version	5, 3, 0, 0
Comments	free crypto library, more information available at www.cryptopp.com

In viewing the exports section we quickly noticed that something was unusual, as the library supposedly contains functions related to music playback.

Exports

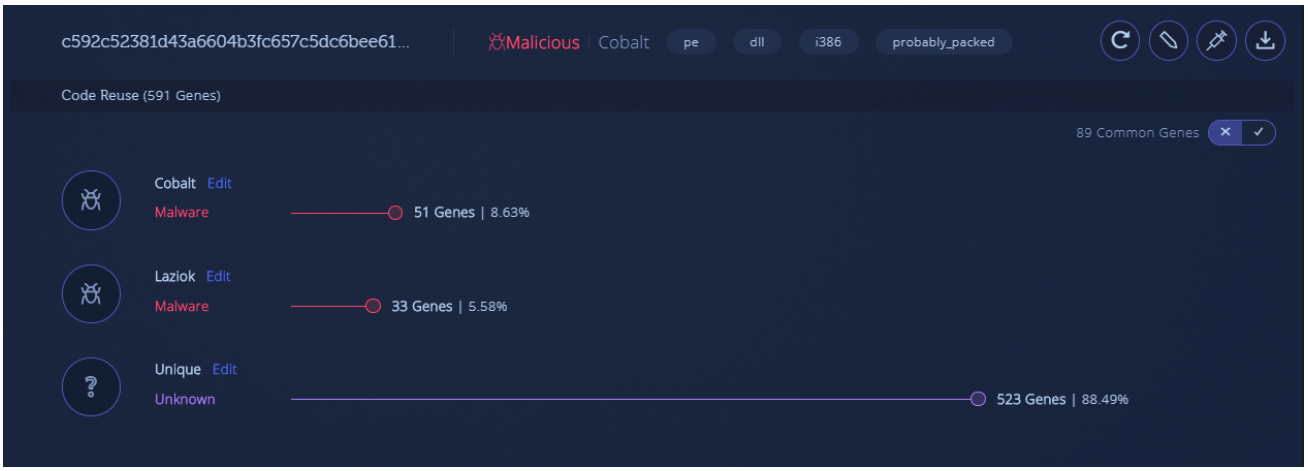
DeleteMusic
DllRegisterServer
FindMusic
MoveMusic
SeekMusic
UploadMusic

When looking at anti-virus vendor scan results in VirusTotal, we observed that the file has been essentially undetected for more than three weeks now, which is quite rare for a malicious file:



Additionally, when we executed this file in several sandbox environments it didn't exhibit any malicious or suspicious behaviors.

However, after genetically analyzing the file in Intezer Analyze we made three key observations:



1. There is no Crypto++ code connection here, meaning the sample is not a Crypto++ library.
2. The file contains reused code from several malware families, mainly from Cobalt Gang binaries. This means the file is malicious and may have relations to Cobalt Gang.
3. The majority of the relevant code in this file is unique, indicating that it's likely a new or highly modified malware.

A Closer Look

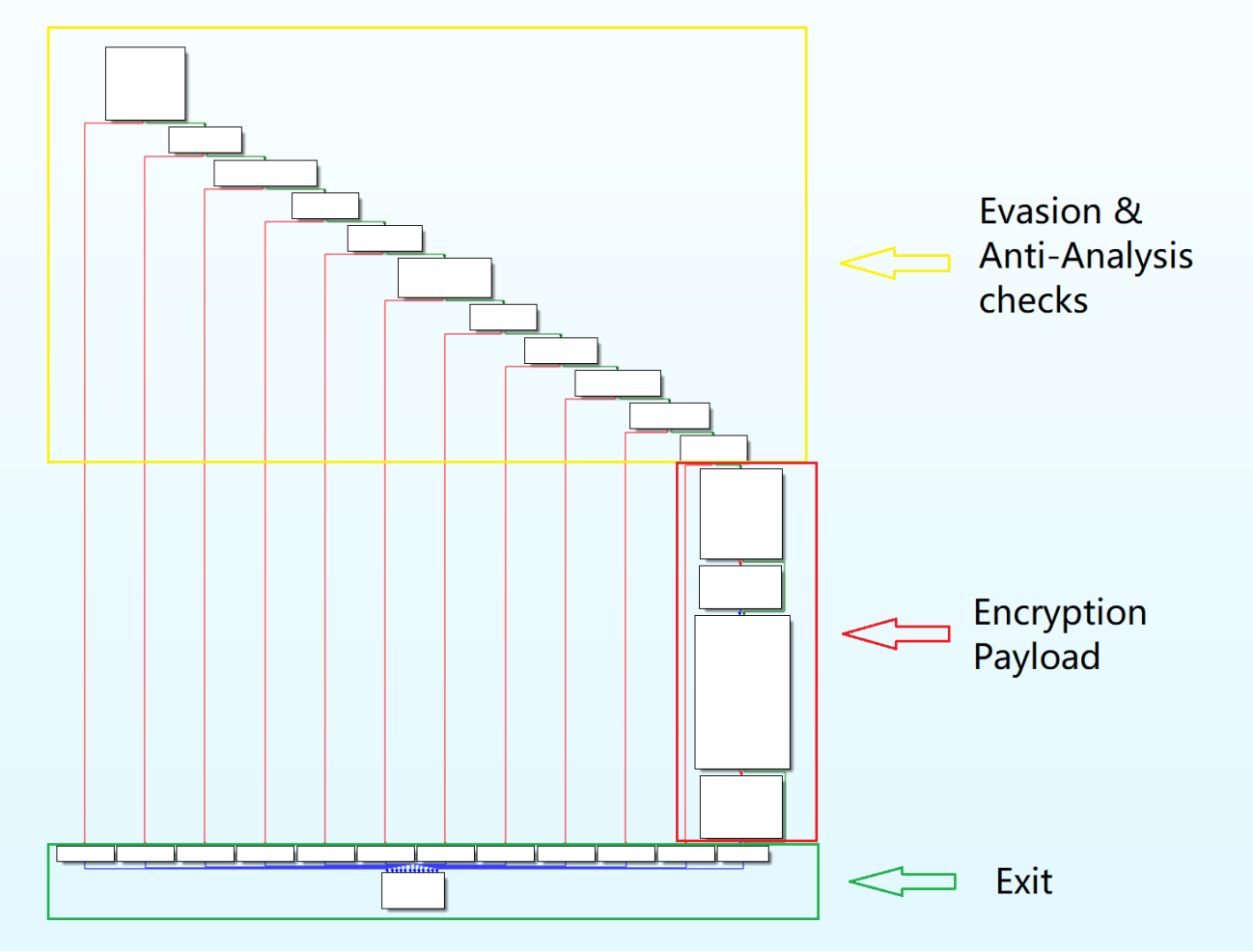
A closer look reveals this file is a ransomware written in the PureBasic programming language, a rather uncommon programming language. This unusual choice poses advantages for the attacker. AV vendors have trouble generating reliable detection signatures for PureBasic binaries. In addition, PureBasic code is portable between Windows, Linux, and OS-X, making targeting different platforms easier.

The malware is designed to be executed as a COM server DLL by regsrv32.exe, which will invoke the DllRegisterServer export, where the malware's code resides. All of the other, music related, exports have no functionality and are included in the ransomware for deception only.

The malware's strings are encoded and stored as Unicode hex strings. Each string is decoded on demand by calling a string decoding function.

```
push    eax ; result
push    strings_decrypt_key ; key
lea     eax, ransom_note_piece_ ; "6123502C33273040175B4A165B2B4766432C363"...
push    eax ; encrypted_string_hex
call    decode_string
pop     eax
lea     edx, asc_10015630 ; "\r\n\r\n"
push    edx ; Str
call    strstr
```

The malware's code begins by checking if it was executed as intended by the attackers, and that it's not being analyzed or debugged. If any of these checks fail, the malware will exit immediately, without deleting itself, likely as an anti-analysis method not to raise suspicion. The main function flow graph is presented below:



In case the malware's payload is executed, the malware will delete itself immediately afterwards.

Part of a Targeted Attack Chain

There are several hints suggesting that this component is part of a targeted and multi-stage attack. The malware begins by checking whether it was executed with the **"/s /i"** arguments, which instructs regsrv32.exe to install the DLL component without raising any dialogues (silent):

```
push    strings_decrypt_key ; key
lea     eax, encrypted_string_arguments ; "1C35136C2C"
push    eax ; encrypted_string_hex
call    decode_string ; "/s /i"
call    tls_get_value
push    [esp+1Ch+exec_commandline] ; String
call    tls_get_value_
add     [esp+20h+arguments_string], edx
call    is_substring
```

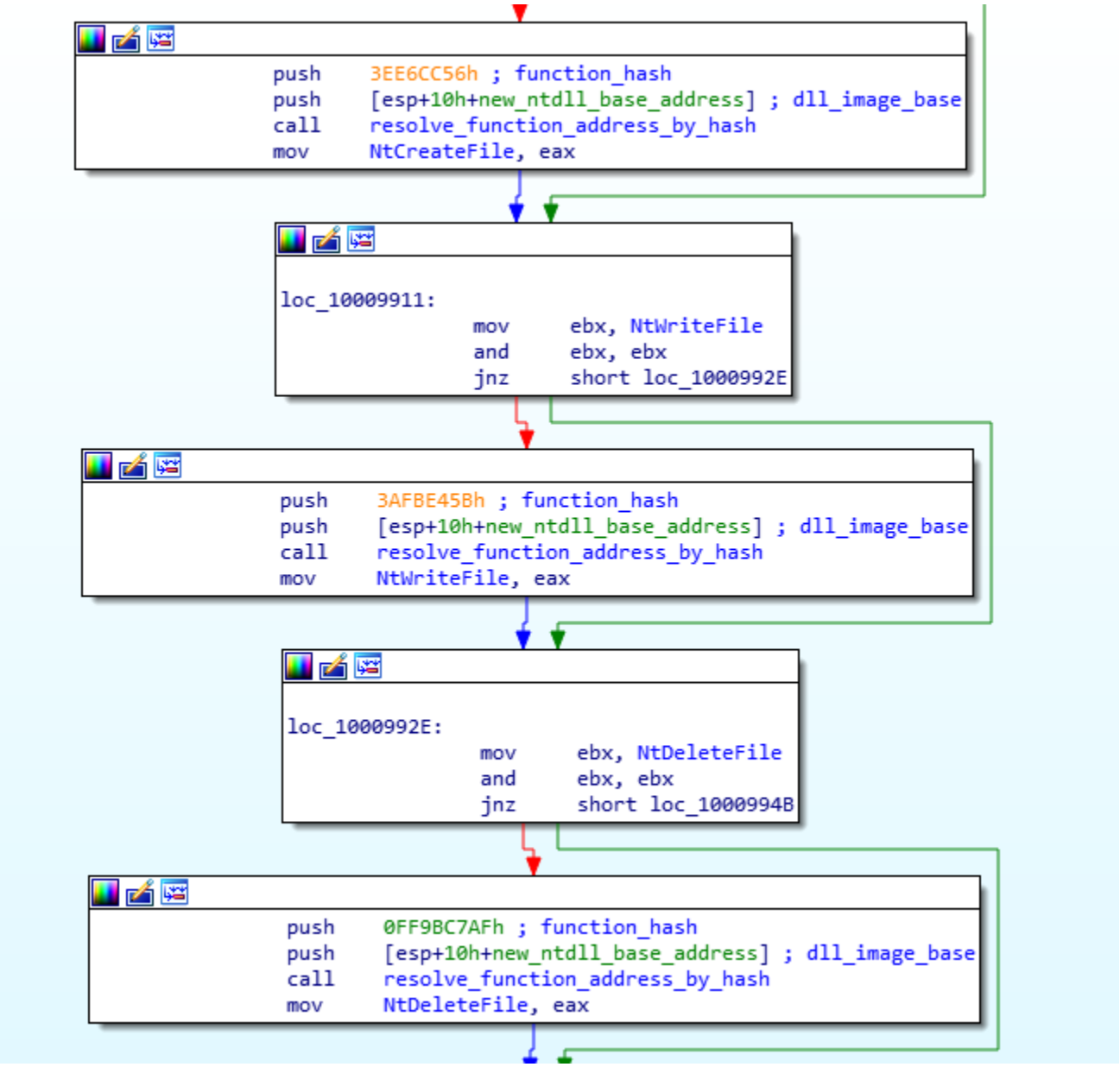
Later on the malware verifies that it's indeed executed by "regsrv32.exe", and that its file extension is either *".dll"* or *".ocx"*. It also verifies that the current year on the machine is 2019, and that it has administrator rights. As mentioned, if any of these checks fail, the malware will exit without performing any malicious activity, likely in an attempt to conceal its functionality.

This type of behavior is not common in ransomware, which typically prefer to infect as many victims as possible in the hopes of gaining as much profit as possible. Additionally, being a DLL file designed to be executed in a very specific manner reveals this ransomware is a later-stage component of a multi-stage attack.

Evasion and Anti-Analysis Techniques

Uncharacteristically for a ransomware, the malware uses an anti-hooking technique by manually loading another copy of "ntdll.dll" and resolving API addresses manually from there. This is an attempt by the malware to evade user-mode hooking of ntdll functions. While it's a known trick, it's rarely used in ransomware.

The imports themselves are stored as 32bit hash values, and the ransomware uses the familiar resolve-by-hash method to obtain the function addresses.

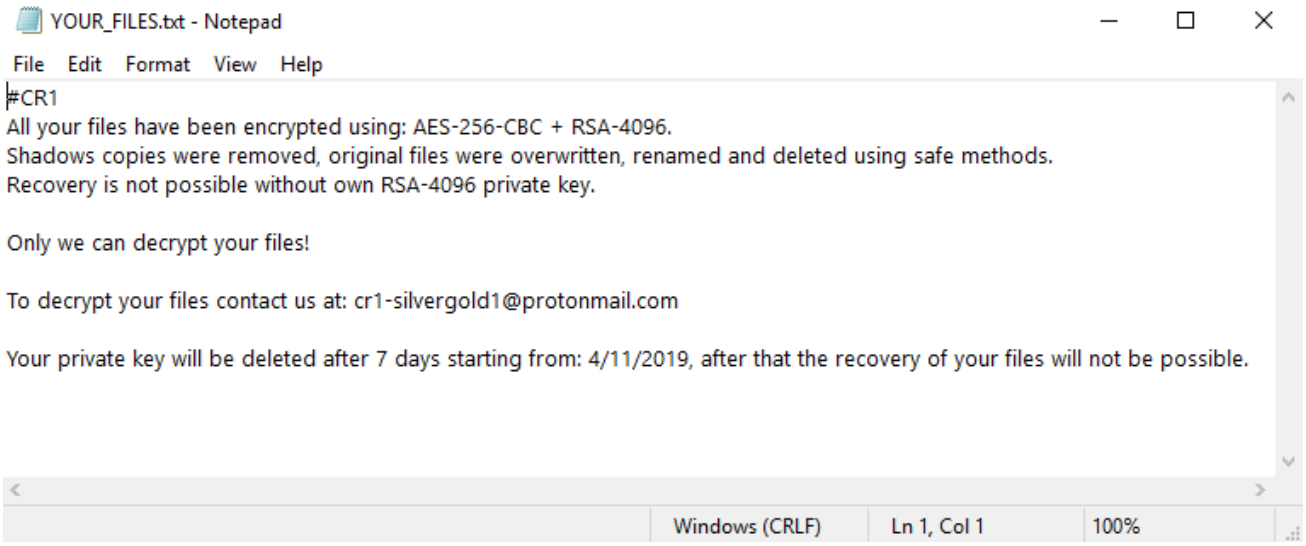


It's also worth noting that the malware uses low level Windows API functions in ntdll.dll for most of its functionality—with some exceptions from kernel32.dll and advapi32.dll—notably for file manipulation. The malware also does not use the Windows Crypto API functions, instead relying on the compiled-in purebasic crypto library for its encryption needs, with the exception of utilizing SystemFunction036 from advapi32.dll (RtlGenRandom) for pseudo-random number generation.

Encryption and Ransom Note

In the event that all anti-analysis and integrity tests performed by the malware are satisfied, it proceeds to encrypt the files on the victim's machine with the standard AES+RSA combination, using a hard-coded RSA key. The ransomware adds the "**.CR1**" extension for each encrypted file. It encrypts mostly data files, skipping encryption for executable files

according to the particular file's extension. The ransomware then secure-deletes the original files in order to prevent recovery. Once the malware has completed the encryption it leaves a ransom note file on the user's desktop named YOUR_FILES.txt.



It's worth noting that the ransom note does not ask for the payment type or for the monetary amount inside of the note itself, instead instructing the victim to contact the attacker via email. The attackers use the anonymous and encrypted Proton email service. Each sample we analyzed contained a different email address, which might be how the attackers can link between different victims and their respective decryption keys (each email corresponds to a specific RSA key pair). This is further evidence that this threat is different from typical forms of ransomware.

Another element to note is the "**CR1**" string, which appears in the attacker email addresses, the encrypted file extension, and ransom note. Since this is a RaaS, we believe this string is most likely the identifier of the group that is operating these specific samples.

Code Connections and Origin

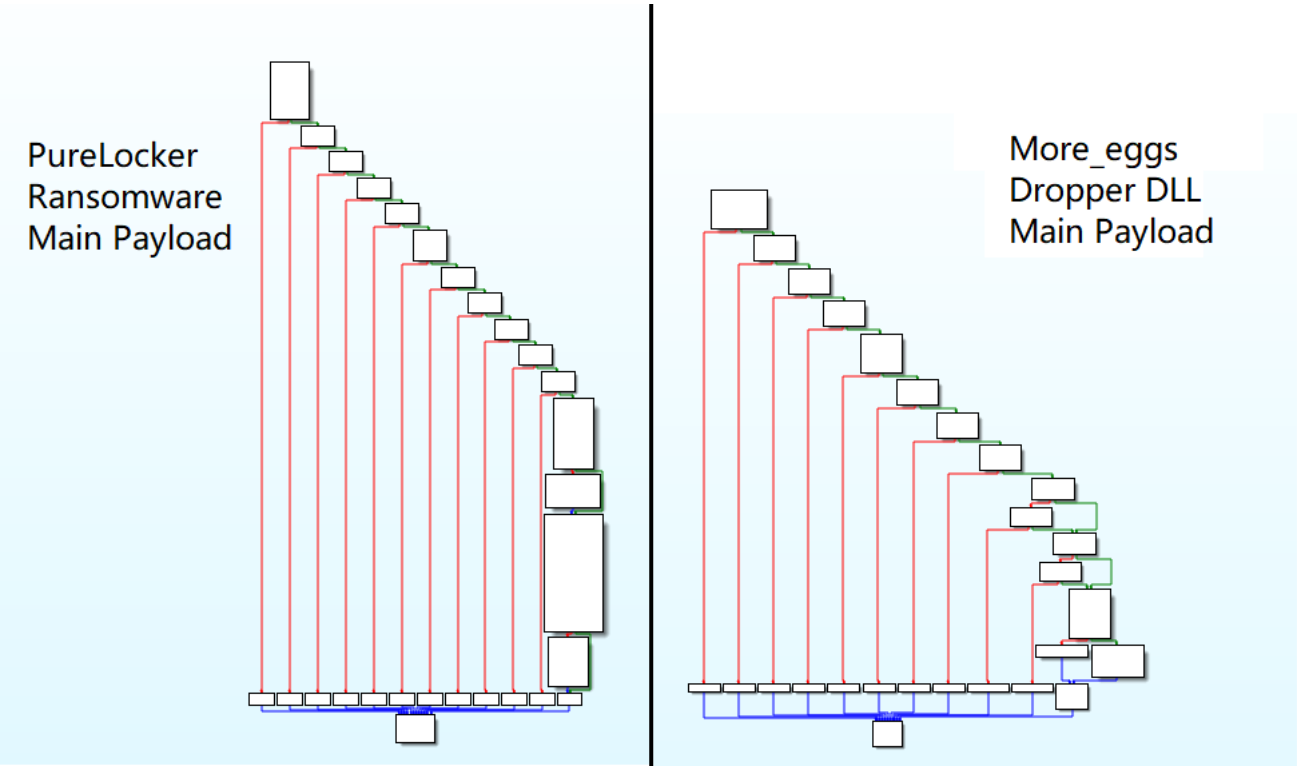
A deeper examination of the genetic analysis results reveals that the code reuse connections to Cobalt Gang are related to a specific component used by the group in its attack chain, described here by Morphisec as the Stage 3 Dropper DLL. More specifically, this component is the loader part of the **"more_eggs"** JScript backdoor, also known as "SpicyOmelette".

Last year, QuoScient uncovered that Cobalt Gang had been buying its malware kits from a malware-as-a-service (MaaS) provider on underground cybercrime forums. QuoScient also observed two additional threat groups using the same MaaS kits in their operations, including the "more_eggs" backdoor.

Most recently, IBM X-Force has uncovered several campaigns by FIN6 (also known as ITG08) where they observed heavy usage of the "more_eggs" malware kit.

A comparison between the PureLocker ransomware samples and recent more_eggs loader samples reveals it's extremely likely that they were created by the same author. The lines of similarity are evident:

- COM Server DLL components written in PureBasic
- Nearly identical pre-payload stage in both functionality and code, with identical evasion and anti-analysis methods
- Identical string encoding and decoding methods



These findings strongly suggest that the MaaS provider of "more_eggs" has added a new malware kit to its offerings, by modifying the "more_eggs" loader's payload from a JScript backdoor to a ransomware.

While we have a good sense regarding the malware's origin, it's unclear at this time whether the **"CR1**" group that's using this ransomware for targeted attacks is a previous customer of the MaaS provider, such as Cobalt Gang and FIN6, or a new one.

Conclusion

PureLocker is a rather unorthodox ransomware. Instead of trying to infect as many victims as possible, it was designed to conceal its intentions and functionalities unless executed in the intended manner. This approach has worked well for the attackers who have managed to successfully use it for targeted attacks, while remaining undetected for several months.

It’s interesting to note that the code of the evasion and anti-analysis functionalities described in this blog is directly copied from the “more_eggs” backdoor loader. Some of these duplicated features have allowed the ransomware to stay undetected by evading automated analysis systems. This provides an example of the importance of code reuse analysis for malware detection and classification. It twists the usage of any previously used code, even code for effective evasion and anti-analysis, into a reliable indicator for detection.

The PureLocker ransomware is now indexed in Intezer’s code genome database and the genetic analysis of its samples can be viewed in Intezer Analyze.

Intezer would like to thank IBM’s X-Force IRIS team for its collaboration on this mutual research. For more information, check out IBM’s X-Force exchange collection related to this threat.

IOCs

1fd15c358e2df47f5dde9ca2102c30d5e26d202642169d3b2491b89c9acc0788
c592c52381d43a6604b3fc657c5dc6bee61aa288bfa37e8fc54140841267338d

Michael Kajiloti
Michael is a security researcher turned product manager, who previously led the Intezer Analyze product lifecycle. He has presented his malware research at conferences such as REcon and the Chaos Communications Congress (CCC).