PrivateLoader. Analyzing the Malware Encryption and Decryption

By khr0x

Published: 2023-07-19 · Archived: 2026-04-05 12:53:14 UTC

In this article, we delve into the inner workings of PrivateLoader, a notorious malicious loader family. We will explore the encryption and decryption processes utilized by this malware, particularly focusing on its ability to protect itself using VMProtect, as well as its decryption of loaded libraries. Let’s dive in!


PrivateLoader

PrivateLoader analysis introduction

PrivateLoader is a malicious loader family, written in C++ and first discovered in early 2021.

It is known for distributing a wide range of malware, from simple information stealers to complex rootkits and spyware, utilizing payloads.

The distribution of this type of malware is managed by the Pay-Per-Install (PPI) service, a popular tool within the cybercriminal ecosystem that generates revenue by adding payloads to malware.

- The code itself involves the decryption of loaded libraries.
- At present, there are two versions of PrivateLoader available: one protected by VMProtect, and a regular version.
- Every day, between 2 and 4 samples of this malware are uploaded.

Static Analysis of the Source File

SHA256: 27c1ed01c767f504642801a7e7a7de8d87dbc87dee88fbc5f6adb99f069afde4

Using the **Detect It Easy** utility, we can see that the analyzed executable file is compiled in C++. There is no information about the packer, which could mean it was not possible to identify it.


PrivateLoader's sample data
Fig. 1 – PrivateLoader’s sample data

The next step is to search for unencrypted strings using the **strings** command:

```
strings --encoding=l loader.exe
```


Interesting strings detected in the executable file
Fig. 2 – Interesting strings detected in the executable file

Analyzing the discovered strings allows us to identify several interesting elements:

- A user-agent, which is likely used to masquerade as a legitimate browser application
- URL addresses for determining the current IP and geolocation

PrivateLoader dynamic analysis with ANY.RUN

We analyzed the sample in ANY.RUN interactive malware sandbox.

Here’s a link to the task:
https://app.any.run/tasks/3e359dc7-934b-4ae1-89bf-ad33e346ed60

The process tree generated by the executable file appears as follows:


PrivateLoader's process tree
Fig 3. – PrivateLoader’s process tree

Analyzing the process tree leads to the following conclusions:

1. The main PrivateLoader process creates a child process named “FhuC750omh76YtB1xgR7diEy.exe”, whose executable file is located in the user’s “Pictures” directory (T1564 – Hide Artifacts):

C:\Users\admin**Pictures**\Minor Policy

2. The created child process is added to the startup using Task Scheduler (T1053.005 – Scheduled Task/Job: Scheduled Task):

schtasks /create /f /RU “admin” /tr “”C:\Program Files (x86)\ClipManagerP0\ClipManager_Svc.exe”” /tn “LOLPA4DESKHR” /sc HOURLY /rl HIGHEST

The executable file of the child process was downloaded from the Internet (T1105 – Ingress Tool Transfer). We will not go into the detailed analysis of it.


PrivateLoader downloaded payload
Fig 4. – PrivateLoader downloaded payload

Analyzing the HTTP requests, we can observe connections and data exchanges with the C2 server (T1071.001 – Application Layer Protocol):


PrivateLoader C2 addresses
Fig. 5 – C2 addresses

The content sent (as well as received) in POST requests consists of BASE64-encoded strings (T1132.001 – Data Encoding: Standard Encoding). Decoding these strings does not yield any readable results:

data=-
kSYhy9HPjD5Jhn9y6Evty4XFfJ3JgIwrSzln5bGnLfKDmbXix2ebDEXy6Ty3Bb8Hz2GB8w0Y2SL2JeBSZ4G80iHAkSS7JJyeiPwZOpWJON(
ljR9hkvX_TJhqr1nNqQpYUB2lQ9i7NmmHeL_QSx8hUka_C3jOxi02ml5FyDDruXM_IWwPXvAGxtT8TV-
i9wLtfd0mF1O369GUAEeI45sF1pKeyDfssmqE=

Moving forward to the indicators, we can see that the malware steals user credentials from browsers (T1552.001 Credentials In Files):


PrivateLoader Stealing data
Fig. 6 – Stealing data

Technical Analysis of PrivateLoader

For the technical analysis, the following tasks were set:

1. Locate the C2 server within the code
2. Identify the encryption algorithms for the C2 server and, if possible, for strings as well.
3. Automate the decryption of the C2 server and strings

The analysis of the executable file revealed that string encryption is done using the XOR algorithm (T1027 – Obfuscated Files or Information). Initially, the data and key are loaded into the stack, and then decrypted using the SIMD instruction “PXOR” and the “XMM” register. The result of the XOR operation is also stored in the stack.

The three stages of C2 server decryption are shown below.

1. Loading encrypted data into the stack:


PrivateLoader data
Fig. 7 – Data

2. Loading the encryption key into the stack:


Key PrivateLoader
Fig. 8 – Key

3. Decrypting the C2 server using the “PXOR” instruction and saving the results in the stack:


Decrypting PrivateLoader

Fig. 9 – Decrypting

During the analysis process, it was also found that the method similar to C2 decryption is used to decrypt the following:

- Used API functions (T1027.007 – Obfuscated Files or Information: Dynamic API Resolution)
- Payloads
- URLs and more

Some of the analyzed samples are protected by VMProtect. The search for string decryption is complicated by the fact that the decryption data is located in one function, while the XOR and key are in another. Moreover, the key is always the same.

Fig 10. – Decript VMprotect sample

Example of automating C2 server decryption of PrivateLoader

To automate the extraction of data and configuration, we can use the Triton framework. It will emulate code blocks that contain all the necessary encrypted information.

You can find an example of a script for emulating a specific block in our GitHub repository. The output of the script will be the decrypted C2 server.

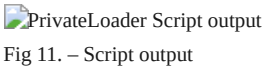
PrivateLoader Script output

Fig 11. – Script output

Therefore, by emulating all the code blocks that contain encrypted data, we can obtain a set of strings with the necessary information, including the C2 server.

Extracting the PrivateLoader configuration

In our service, you can view the configuration, which is extracted automatically:

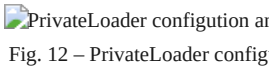
PrivateLoader configution and strings

Fig. 12 – PrivateLoader configution and strings

The decrypted data includes C2 addresses and strings. The strings contain information such as: used libraries and their functions, registry keys, paths to crypto wallets and browsers, etc.

Conclusion

In this article, we discussed encryption in PrivateLoader.

Its main feature is the XOR of all strings it interacts with (C2, URLs, DLLs). Also, some samples are protected by VMprotect, which makes the code a bit more complex due to the use of many functions.

If you'd like to read more content like this, read our LimeRAT Malware Analysis. Or check out our deep dive into the encryption and decryption process of XLoader/FormBook.

MITRE (ARMATTACK)

Tactics	Techniques	Description
TA0007: Software discovery	T1518: Software Discovery	Searches for installed software in the system in the “Uninstall” key
	T1082: System Information Discovery	Collects system data
TA0011: Command and Control	T1071.001: Application Layer Protocol	Sending collected data to the control server
	T1105 Ingress Tool Transfer	requests binary from the Internet
	T1132.001 – Data Encoding: Standard Encoding	encode data with BASE64
TA0006: Credential Access	T1552.001: Credentials In Files	Stealing of personal data – login data
TA0005: Defense Evasion	T1564 Hide Artifacts	attempt to hide artifacts in user folder
	T1027.007 – Obfuscated Files or Information: Dynamic API Resolution	obfuscate then dynamically resolve API functions called by their malware
	T1027 – Obfuscated Files or Information	attempt to make an executable or file difficult to discover or analyze by encrypting XOR
TA0002: Execution	T1053.005 – Scheduled Task/Job: Scheduled Task	abuse the Windows Task Scheduler to create file in statup

IOCs

Title	Description
Name	27c1ed01c767f504642801a7e7a7de8d87dbc87dee88fbc5f6adb99f069afde4 exe
MD5	6cc7d9664c1a89c58549e57b5959bb38
SHA1	85b665c501b9ab38710050e9a5c1b6d2e96acccc
SHA256	27c1ed01c767f504642801a7e7a7de8d87dbc87dee88fbc5f6adb99f069afde4

Extracted URLs

- http://23[.]254[.]227[.]214/api/tracemap[.]php
- http://23[.]254[.]227[.]205/api/tracemap[.]php
- http://23[.]254[.]227[.]202/api/tracemap[.]php

- http://208[.]67[.]104[.]60/api/tracemap[.]php
- http://208[.]67[.]104[.]60/api/firegate[.]php
- http://163[.]123[.]143[.]4/download/YT_Client[.]exe

Dropped executable file

Title	Description
Name	C:\Users\admin\AppData\Local\Microsoft\Windows\INetCache\IE\AH8CR9J5\YT_Client[1].exe
SHA256	041f891934add72852c8fda245c95da959d7f98cc580383d198e42f2de039634

DNS requests

- iplogger.org
- ipinfo.io
- Iplis.ru

Connections (IP)

- “23[.]254.227.214”
- “23[.]254.227.202”
- “23[.]254.227.205”
- “208[.]67.104.60”

MORE SAMPLES FOR YOUR RESEARCH

https://app.any.run/tasks/ff1872a6-6c1f-4f79-89da-995b9bd56152/

https://app.any.run/tasks/6a8f93eb-be36-41bc-bf7f-534938a7e3a2/

https://app.any.run/tasks/cc2cb367-82e9-4705-9767-8c12f7a67a21/

https://app.any.run/tasks/c32312d8-4026-4a81-84e5-3d90ab2e309a/

https://app.any.run/tasks/235754fa-6aa3-49dd-bbc4-1a7f9361f455/

khr0x

I'm 21 years old and I work as a malware analyst for more than a year. I like finding out what kind of malware got on my computer. In my spare time I do sports and play video games.

I'm 21 years old and I work as a malware analyst for more than a year. I like finding out what kind of malware got on my computer. In my spare time I do sports and play video games.
