A cryptor, a stealer and a banking trojan

By GReAT

Published: 2023-09-28 · Archived: 2026-04-05 14:35:27 UTC

Introduction

As long as cybercriminals want to make money, they'll keep making malware, and as long as they keep making malware, we'll keep analyzing it, publishing reports and providing protection. Last month we covered a wide range of cybercrime topics. For example, we published a private report on a new malware found on underground forums that we call ASMCrypt (related to the DoubleFinger loader). But there's more going on in the cybercrime landscape, so we also published reports on new versions of the Lumma stealer and Zanubis Android banking trojan. This blog post contains excerpts from those reports.

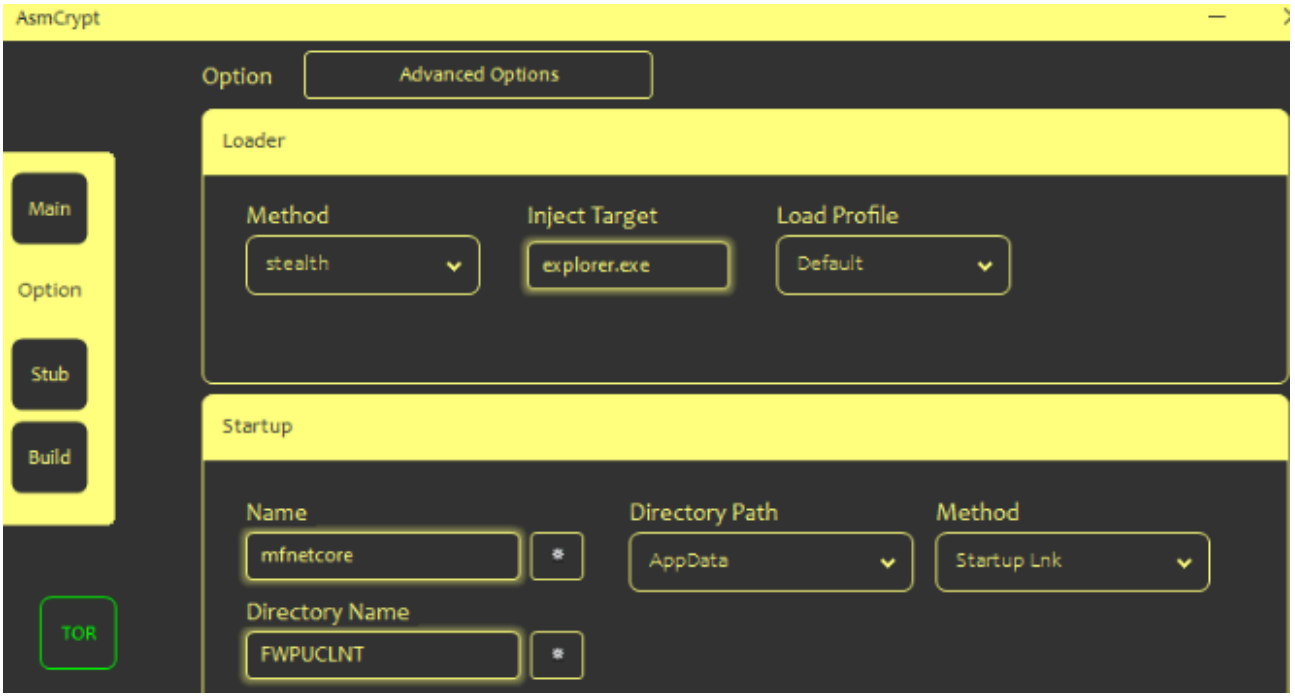
If you want to learn more about our crimeware reporting service, please contact us at crimewareintel@kaspersky.com.

ASMCrypt

As mentioned in our previous blog post, we monitor many underground forums. On one of them we saw an ad, promoting a new cryptor/loader variant called ASMCrypt. The idea behind this type of malware is to load the final payload without the loading process or the payload itself being detected by AV/EDR, etc. This sounds a lot like the DoubleFinger loader we discussed here.

In fact, after careful analysis, we believe with a high degree of confidence that ASMCrypt is an evolved version of DoubleFinger. However, ASMCrypt works slightly differently and is more of a "front" for the actual service that runs on the TOR network.

So how does it work? First the buyer obtains the ASMCrypt binary, which connects to the malware's backend service over the TOR network using hardcoded credentials. If everything is okay, the options menu is shown:



The buyer can choose from the following options:

- Stealth or invisible injection method;
- The process the payload should be injected into;
- Folder name for startup persistence;
- Stub type: either the malware itself masquerading as Apple QuickTime, or a legitimate application that sideloads the malicious DLL.

After selecting all the desired options and pressing the build button, the application creates an encrypted blob hidden inside a .png file. This image must be uploaded to an image hosting site. The malicious DLL (or binary) from the last bullet point above is also created and will be distributed by the cybercriminals.

When the malicious DLL is executed on a victim system, it downloads the .png file, decrypts it, loads it into memory and then executes it.

Lumma

The Arkei stealer, written in C++, first appeared in May 2018 and has been forked/rebranded several times over the last couple of years. It has been known as Vidar, Oski, Mars and now Lumma, which has a 46% overlap with Arkei. Over time, the main functionality of all the variants has remained the same: stealing cached files, configuration files and logs from crypto wallets. It can do this by acting as a browser plugin, but it also supports the standalone Binance application.

But first the infection vector. Lumma is distributed via a spoofed website that mimics a legitimate .docx to .pdf site. When a file is uploaded, it is returned with the double extension .pdf.exe.

Lumma itself first appeared on our radar in August 2022, when we detected new samples. Around the same time, cybersecurity enthusiast Fumik0_ tweeted that Lumma was a “fork/refactor” of Mars. Since then, Lumma has

undergone a number of changes, some of which we will highlight below:

- We found only one sample (MD5 6b4c224c16e852bdc7ed2001597cde9d) that had the functionality to collect the system process list. The same sample also used a different URL to communicate with the C2 (/winsock instead of /socket.php);
- We also found one sample (MD5 844ab1b8a2db0242a20a6f3bbceedf6b) that appears to be a debugging version. When certain code fragments are reached, a notification is sent to the C2. Again, it uses a different URL (/windbg).
- In a more recent sample (MD5 a09daf5791d8fd4b5843cd38ae37cf97), the attackers changed the User-Agent field to “HTTP/1.1”. It is unclear why this was done;
- While all previous samples, including the three mentioned above, downloaded additional libraries from the C2 for 32-bit systems so that specific browser-related files (e.g. passwords and the like) could be parsed, MD5 5aac51312dfd99bf4e88be482f734c79 simply uploads the entire database to the C2;
- MD5 d1f506b59908e3389c83a3a8e8da3276 has a string encryption algorithm. They are now hex encoded and encrypted with an XOR key (first 4 bytes of the string).
- One of the biggest changes we saw involved MD5 c2a9151e0e9f4175e555cf90300b45c9. This sample supports dynamic configuration files retrieved from the C2. The configuration is Base64 encoded and XORed with the first 32 bytes of the configuration file.

```
sub_11E5184();
send_dbg_11E52C4(v4, (int)"ct_start");
if ( sub_11E1061() )
{
  v6 = sub_11E905E(v5);
  send_dbg_11E52C4(v9, (int)"chr_start");
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  collect_chrome_data_11E4D64(v6);
  send_dbg_11E52C4(v10, (int)"moz_start");
  collect_mozilla_data_11E9F46(v6);
  collect_mozilla_data_11E9F46(v6);
  send_dbg_11E52C4(v11, (int)"grb_start");
  grab_wallet_data_11E4F7C(L"%appdata%\\Exodus", L"\\*.json", L"Wallets/Exodus", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\Exodus\\exodus.wallet", L"\\*", L"Wallets/Exodus/exodus_wallet", v6);
  grab_wallet_data_11E4F7C(L"%userprofile%\\Desktop", L"\\*.txt", L"Important Files/Desktop", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\Binance", L"\\app-store.json", L"Wallets/Binance", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\Electrum\\wallets", L"\\*", L"Wallets/Electrum", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\Ethereum", L"\\keystore", L"Wallets/Ethereum", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\Electrum-LTC\\wallets", L"\\*", L"Wallets/Electrum-LTC", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\jaxx\\Local Storage", L"\\*.localstorage", L"Wallets/Jaxx", v6);
  grab_wallet_data_11E4F7C(L"%appdata%\\ElectronCash\\wallets", L"\\default_wallet", L"Wallets/ElectronCash", v6);
  sub_11E10E4(v6);
  if ( v6 )
  {
    sub_11E8BF9(v6);
    sub_11E9018(v6);
    sub_11E835F(v6);
    sub_11E7B32(v6, 1);
    sub_11ECFBF(v6);
    v7 = v12;
  }
  send_dbg_11E52C4(v7, (int)"upl_start");
  send_collected_data_11E5217();
  sub_11F0E72("c:\\ProgramData\\winrarupd.zip");
}
return 0;
```

Code snippet of the “debugging” sample

Zanubis

Zanubis, an Android banking trojan, first appeared around August 2022, targeting financial institution and cryptocurrency exchange users in Peru. Zanubis’s main infection path is through impersonating legitimate Peruvian Android applications and then tricking the user into enabling the Accessibility permissions in order to take full control of the device.

We spotted more recent samples of Zanubis in the wild around April 2023. The malware was disguised as the official Android application for the Peruvian governmental organization SUNAT (Superintendencia Nacional de Aduanas y de Administración Tributaria). We explored the new design and features of the malware, which seemed to have undergone several phases of evolution to reach a new level of sophistication.

Zanubis is obfuscated with the help of Obfuscapk, a popular obfuscator for Android APK files. After the victim grants Accessibility permissions to the malicious app, thus allowing it to run in the background, the malware uses WebView to load a legitimate SUNAT website used for looking up debts. The intention here is to lead the unsuspecting user to believe that the app is part of the SUNAT ecosystem of services.

Communication with the C2 relies on WebSockets and the library called Socket.IO. The latter allows the malware to establish a persistent connection to the C2, which provides failover options (from WebSockets to HTTP and vice versa). Another advantage is that it provides the C2 with a scalable environment where all new infections by Zanubis can receive commands (also called *events*) on a massive scale from the C2 if required. Once the malware starts, the implant calls a function to check the connection to the C2. It establishes two connections to the same C2 server, but they perform different types of actions, and the second connection is established only if requested by the C2.

Intentionally, Zanubis doesn’t count with a pre-populated and hardcoded list of applications to target. In recent years, malware developers have tended to add or remove the names of applications from the target list. To set the targeted applications on the implant, the C2 sends the event *config_packages*. The JSON object sent with the event contains an array specifying the applications that the malware should monitor. The malware parses the list of targeted applications each time an event occurs on the screen, such as an app opening, which the malware detects using the *onAccessibilityEvent* function. Once an application on the list is found running on the device, Zanubis takes one of two actions, depending on its configuration, to steal the victim’s information: logging events/keys, or recording the screen.

Previously, we mentioned initializing the second connection from the infected device, which provides further options for the C2. After Zanubis establishes this new connection, it sends a *VncInit* event to the server to inform it that initialization of the second feature set is complete, and it will send information about screen rendering, such as the display size, every second. We can assume that this is a way for the operators to take control of, or backdoor, the infected phone.

An interesting feature in the second set is the *bloqueoUpdate* event. This is one of the most invasive – and persuasive – actions taken by the malware: it pretends to be an Android update, thus blocking the phone from

being used. As the “update” runs, the phone remains unusable to the point that it can’t be locked or unlocked, as the malware monitors those attempts and blocks them.

Fake update locking the user out of the phone

According to our analysis, the targeted applications are banks and financial entities in Peru. This fact, in conjunction with our telemetry data, leads us to determine that Zanubis targets users in that country specifically. The list of targeted applications contains more than 40 package names. The samples of Zanubis collected to date are capable of infecting any Android phone, but they were all written with Spanish as the system language in mind.

Conclusion

Malware is constantly evolving, as is illustrated by the Lumma stealer, which has multiple variations with varying functionality. Zanubis also aspires to become a fully armed banking trojan that could inflict financial losses and steal the personal data of mobile users. This constant change in malicious code and cybercriminal TTPs is a challenge for defense teams. To protect itself, an organization must learn about new threats as soon as they emerge. Intelligence reports can help you stay on top of the latest malicious tools and attacker TTPs. If you’d like to stay up to date on the latest TTPs being used by criminals, or have questions about our private reports, please contact us at crimewareintel@kaspersky.com.

Indicators of compromise (MD5s)

Lumma

6b4c224c16e852bdc7ed2001597cde9d
844ab1b8a2db0242a20a6f3bbceedf6b
a09daf5791d8fd4b5843cd38ae37cf97
5aac51312dfd99bf4e88be482f734c79

d1f506b59908e3389c83a3a8e8da3276
c2a9151e0e9f4175e555cf90300b45c9

Zanubis

054061a4f0c37b0b353580f644eac554
a518eff78ae5a529dc044ed4bbd3c360
41d72de9df70205289c9ae8f3b4f0bcb
9b00a65f117756134fdb9f6ba4cef61d
8d99c2b7cf55cac1ba0035ae265c1ac5
248b2b76b5fb6e35c2d0a8657e080759
a2c115d38b500c5dfd80d6208368ff55
