Unpacking the spyware disguised as antivirus

blog.malwarebytes.com/threat-analysis/2016/08/unpacking-the-spyware-disguised-as-antivirus/

Malwarebytes Labs

August 25, 2016

Recently we got access to several elements of the espionage toolkit that has been captured attacking Vietnamese institutions. During the operation, the malware was used to dox 400,000 members of Vietnam Airlines.

The payload, distributed disguised as antivirus, is a variant of Korplug RAT (aka PlugX) – a spyware with former associations with Chinese APT groups, and known from targeted attacks at important institutions of various countries.
In this article we will describe the process of extracting the final payload out of it's cover.

Analyzed samples

Set #1:

- 884d46c01c762ad6ddd2759fd921bf71 – McAfee.exe (harmless: reference)
- c52464e9df8b3d08fc612a0f11fe53b2 – McUtil.dll (shellcode loader)

Execution flow:

McAfee.exe -> McUtil.dll -> McUtil.dll.mc -> payload (DLL)

A look at the package

This spyware has an interesting, modular package. As a whole, it tries to pretend to be McAfee antivirus:

ang	2015-06-26 14:54	File	1 KB
McAfee.exe	2013-08-29 08:50	Application	138 KB
McUtil.dll	2013-08-29 08:50	Application extens...	4 KB
McUtil.dll.mc	2013-08-29 08:50	MC File	115 KB
tjuiiarpujhx	2016-05-19 04:47	File	2 KB
vekmfmujufficwveip	2013-08-29 08:50	File	59 KB

If we take a look at the executable, we see that is has been signed by the original certificate:

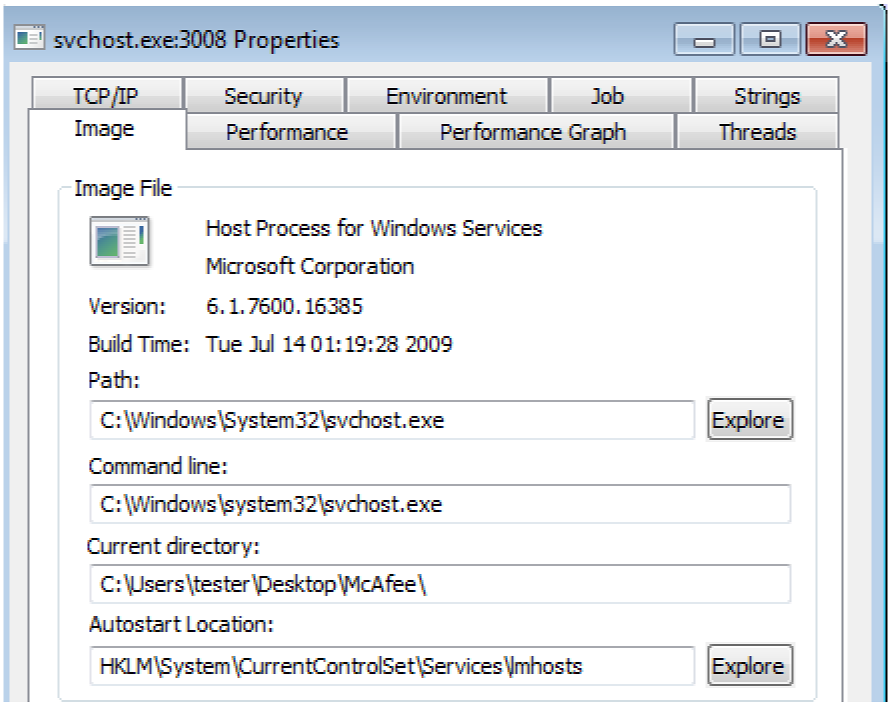
Authenticode signature block and FileVersionInfo properties	
Copyright	Copyright © 2006 McAfee, Inc.
Product	McAfee Oem Module
Original name	mcoemcpy.exe
Internal name	mcoemcpy
File version	2,1,115,0
Description	McAfee OEM Info Copy Files
Signature verification	Signed file, verified signature
Signing date	12:47 AM 6/13/2008
Signers	[+] McAfee [+] VeriSign Class 3 Code Signing 2004 CA [+] VeriSign Class 3 Public Primary CA
Counter signers	[+] VeriSign Time Stamping Services Signer - G2 [+] VeriSign Time Stamping Services CA [+] Thawte Timestamping CA

It is not fake – the executable is a legitimate product. However, it is bundled with the DLL that is not signed – and this it the point that attackers used in order to hijack the execution.

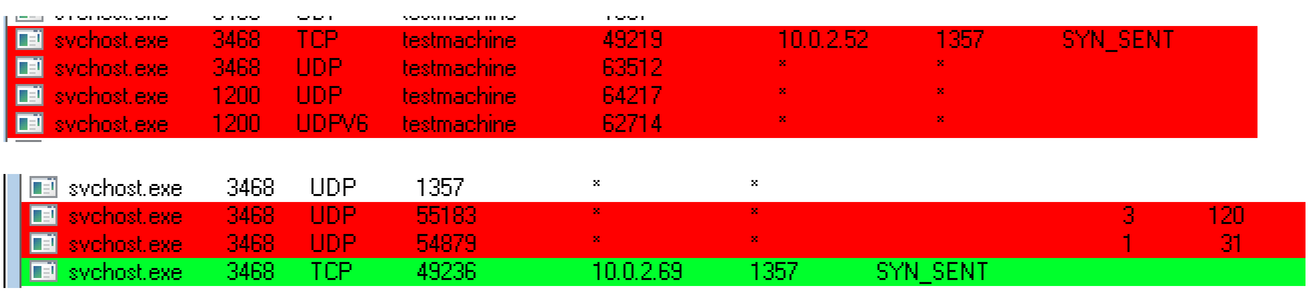
Note that the app used in the attacks is very old (compiled in 2008). The current versions of McAfee Antivirus that we managed to test are no longer vulnerable to this type of abuse.

Behavioral analysis

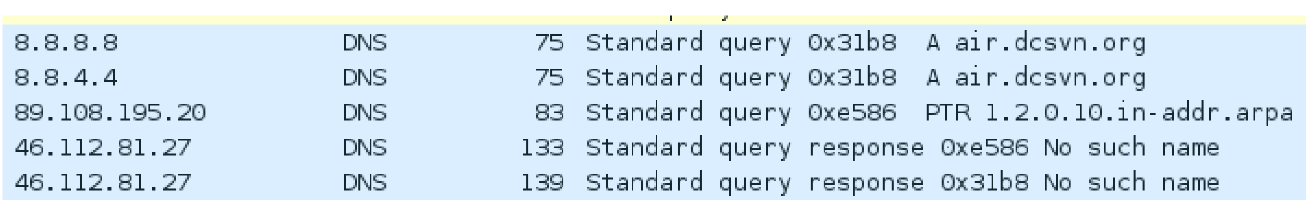
After being deployed, the application runs silently. We can see the main component executing svchost.exe, and then terminating itself. It is caused by the fact that the malicious code has been injected into svchost, and will continue operating from there. Looking at the current directory of svchost.exe we can find that it inherits default directory of the malicious app:



The bot makes reconnaissance in the LAN by scanning for other computers. It enumerates full range of local addresses, from the lowest to the highest:



It also tried to connect with it's C&C (air.dscvn.org), however, at the moment of tests the domain was down:



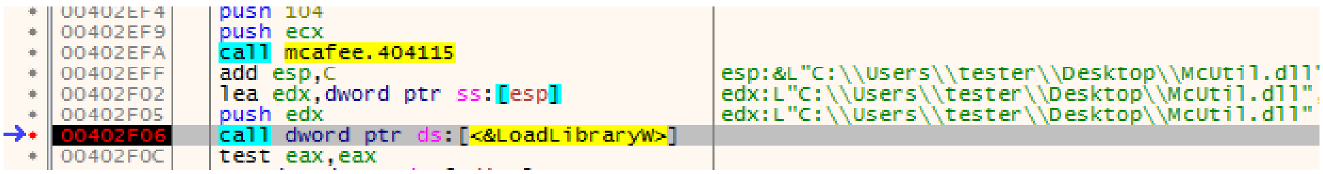
Unpacking

The application have several layers of loaders before it reach the final functionality. The exe file, as well as the DLL are harmless. All the the malicious features lies in the external file, that is a blocks of obfuscated shellcode. Within the shellcode, another DLL is hidden – that is the core spy bot.

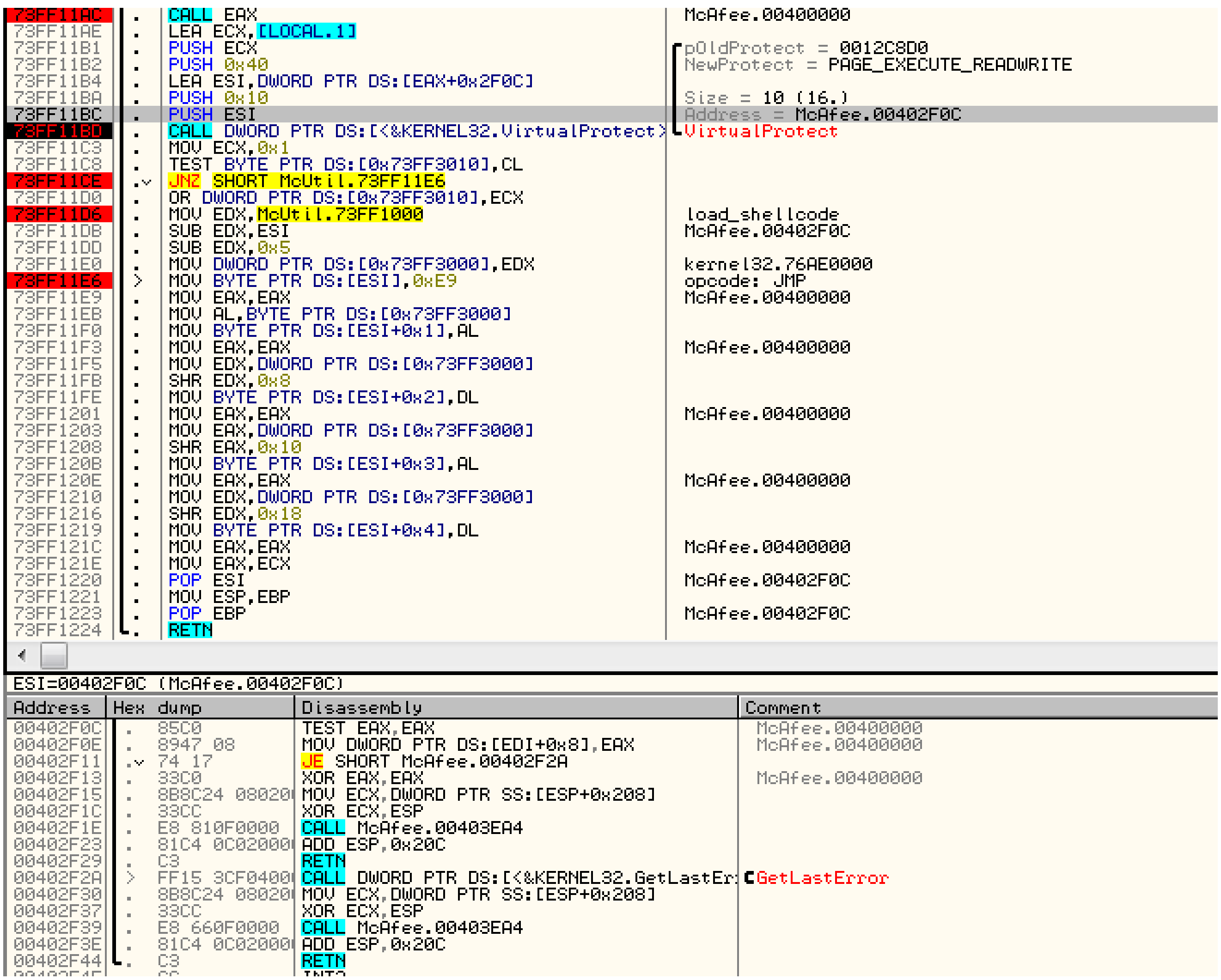
Loading the shellcode

The payload is loaded in an obfuscated way containing some interesting tricks. The authors took great care that it will not be easy to analyze the modules separately.

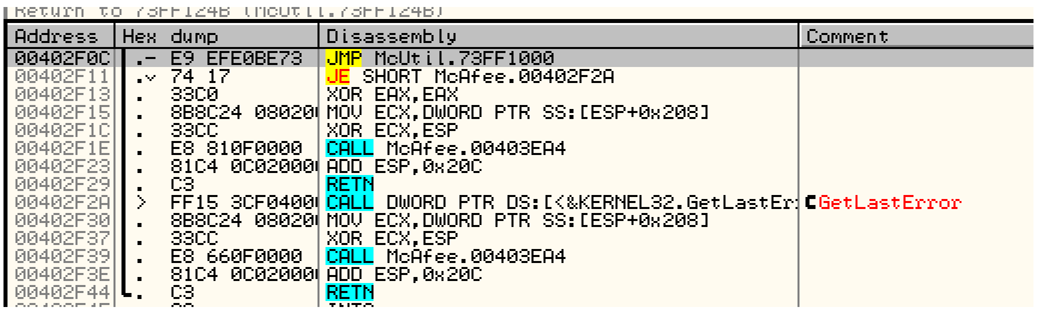
Execution starts from the harmless *McAfee.exe*. Malware utilized the fact that this application loads a library called *McUtil.dll* from the startup directory. It doesn't make any integrity check, so in fact, if we rename any library to the desired name, the executable will just load it:



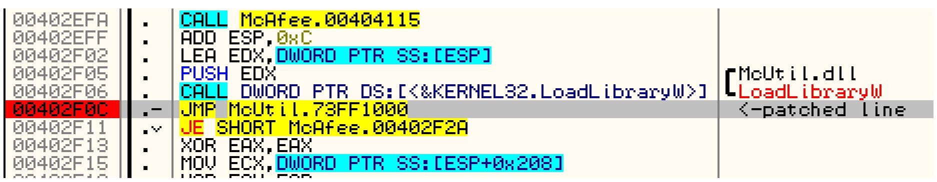
McUtil.dll is supposed to deploy the next file: *McUtil.dll.mc* – however, to make the flow more difficult to follow, it doesn't run it directly. Instead, it patches the caller executable (*McAfee.exe*) and makes it execute the function responsible for reading and loading the next file. Below we can see the fragment of code, that writes the hook into the memory:



That's how the above fragment of caller's code looks after patching. Instead of the first two lines we can see a jump into the *McUtil.dll*:

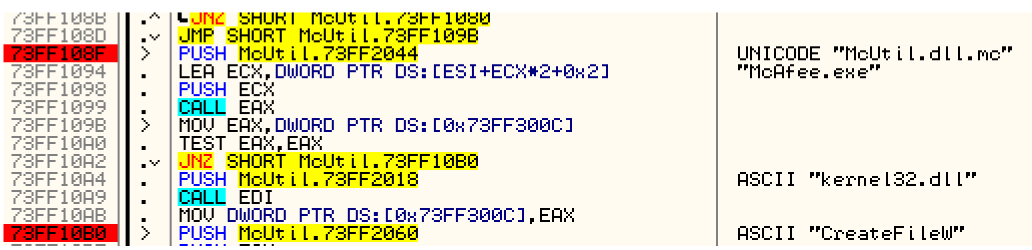


Patching function is in DllMain of the *McUtil.dll* – so, it is called on load. The patched line is just after the call that loaded the library:

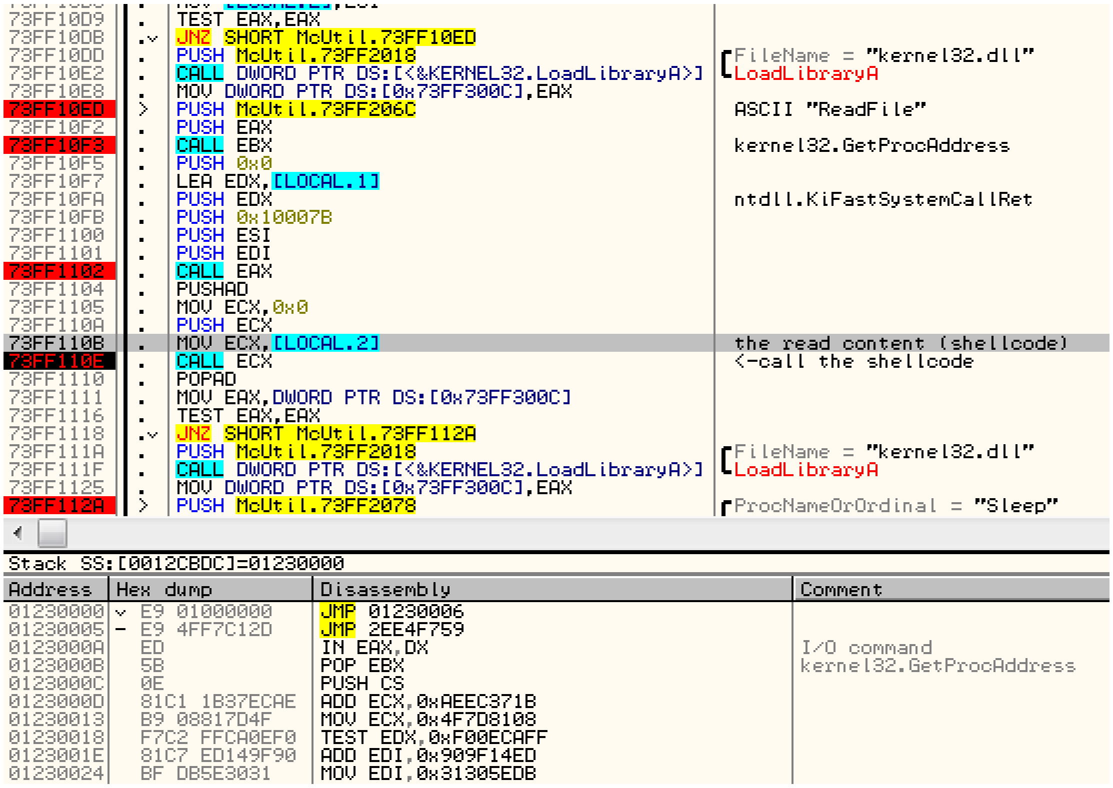


So, the hook will be executed as soon as the loading function returns.

Inside the function called by the hook, the external file is open:

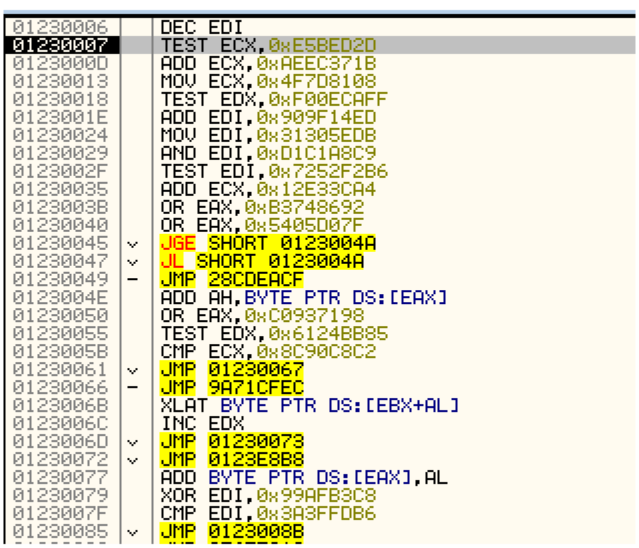


It is read into the memory and then execution is redirected there:

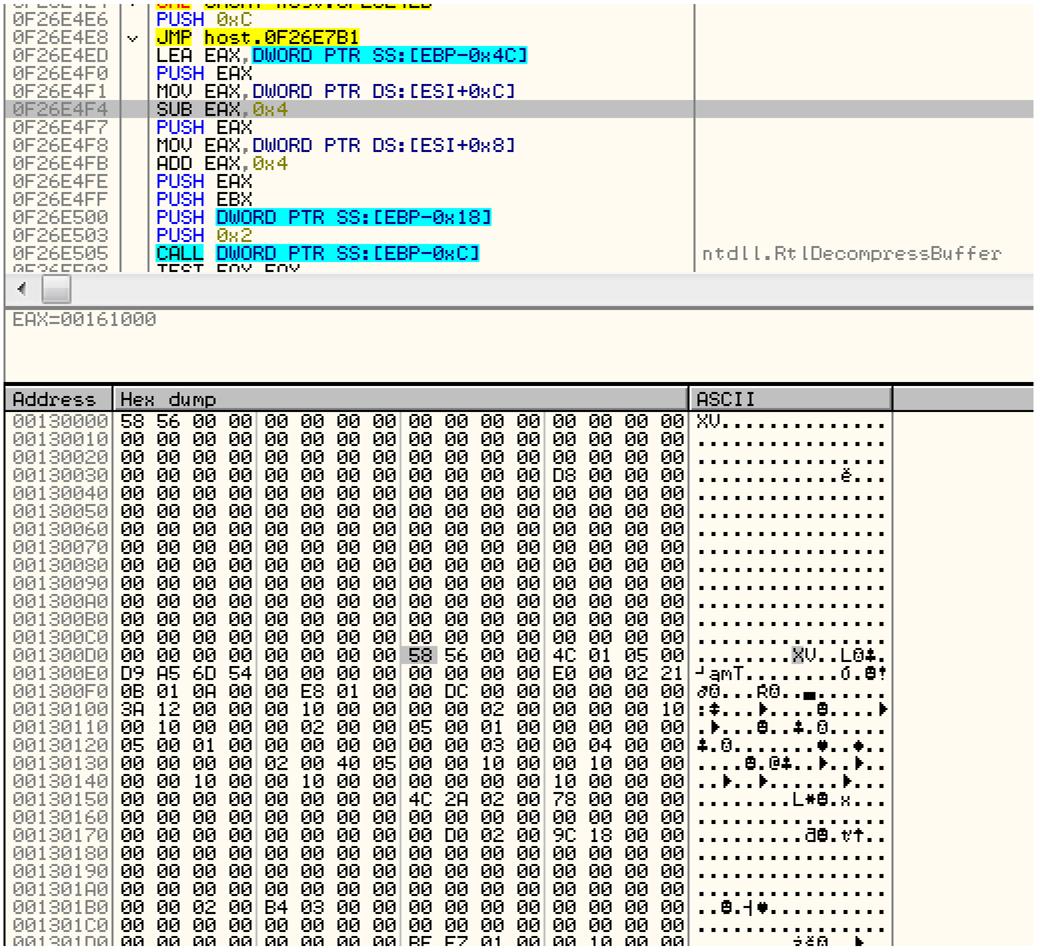


Unpacking the final payload

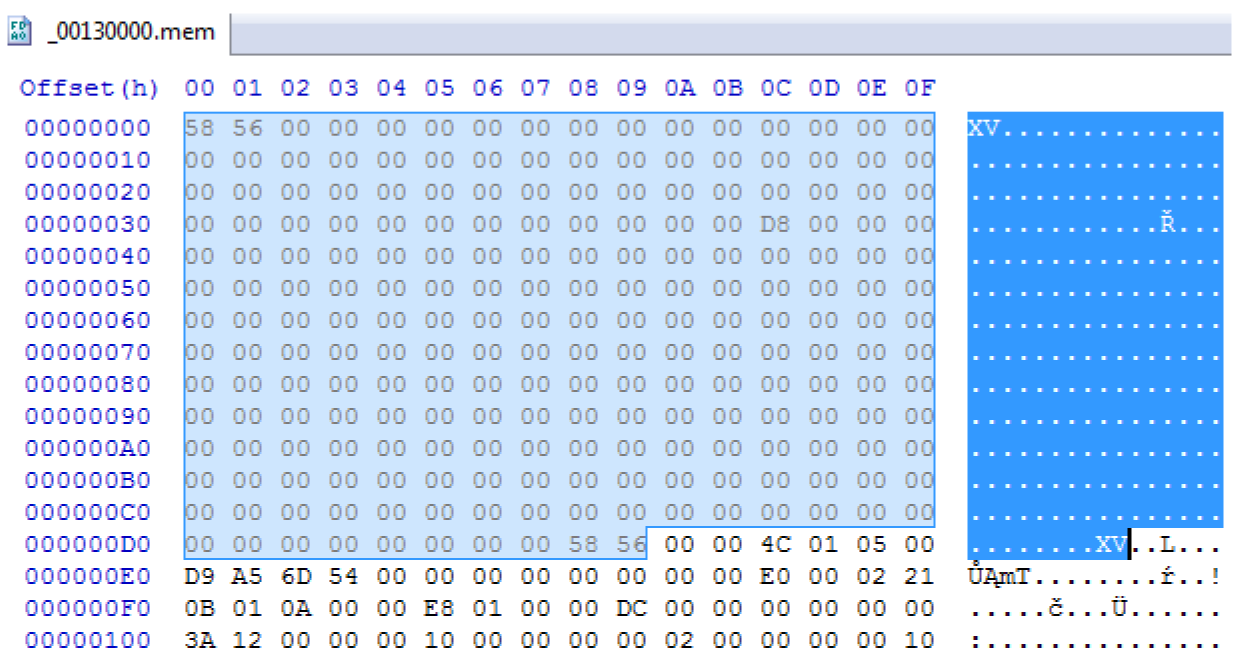
The shellcode is heavily obfuscated:



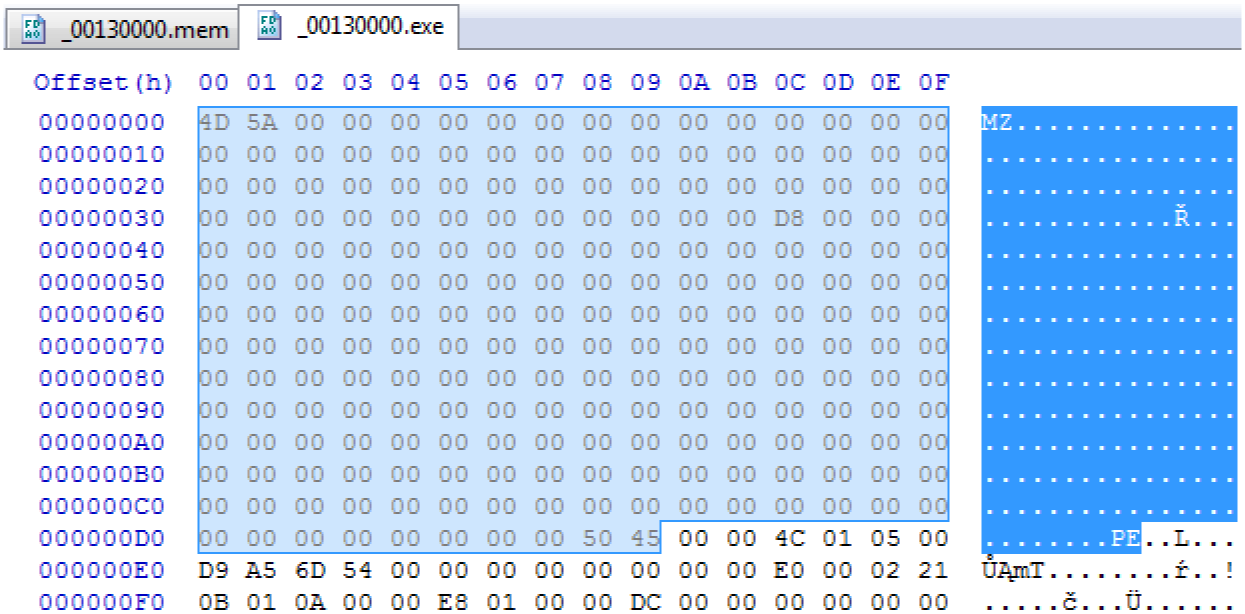
This is not the main stage, but an unpacker and loader of the main spyware. It decompresses the following content into a buffer:



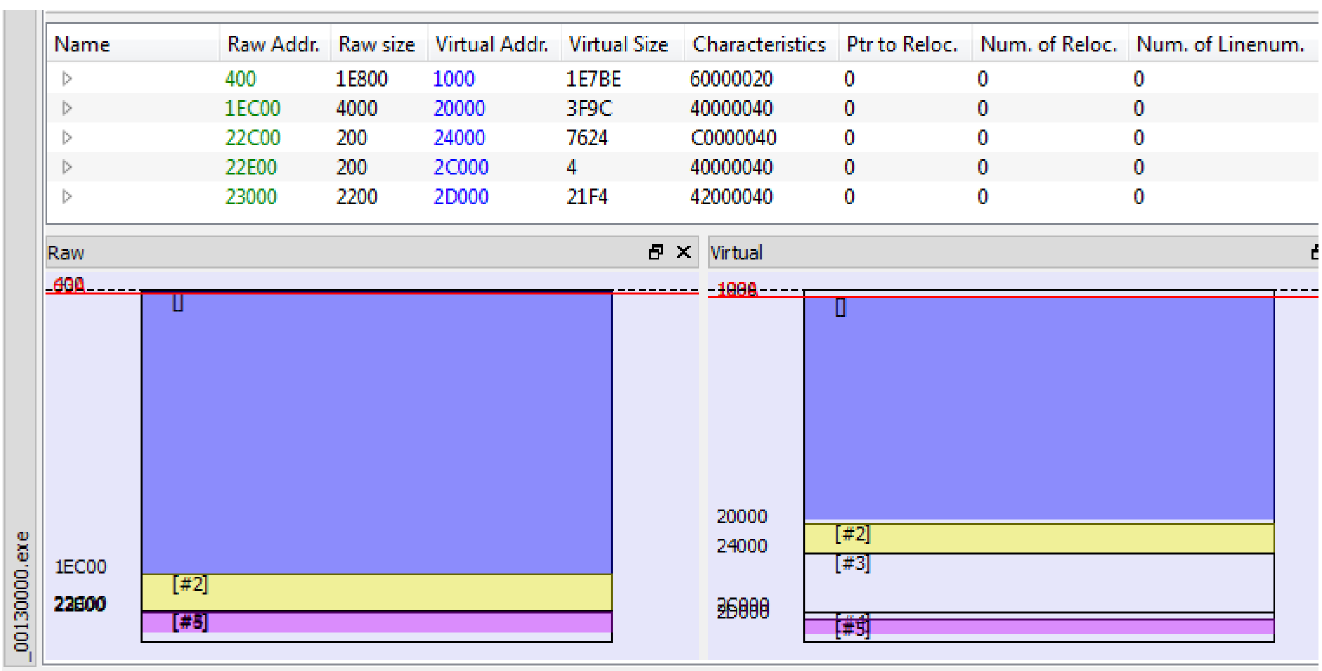
Then it reserves additional memory and starts remapping this content, chunk by chunk. By the way in which it parses it, we can notice similarity with process of remapping raw PE file into a virtual image. And indeed, the unpacked content is a PE file – only the headers are distorted. Delimiters XV were used to substitute the typical "MZ".. "PE" values:



Reconstructing the header is not difficult – we must just substitute back those values by their real meaning:

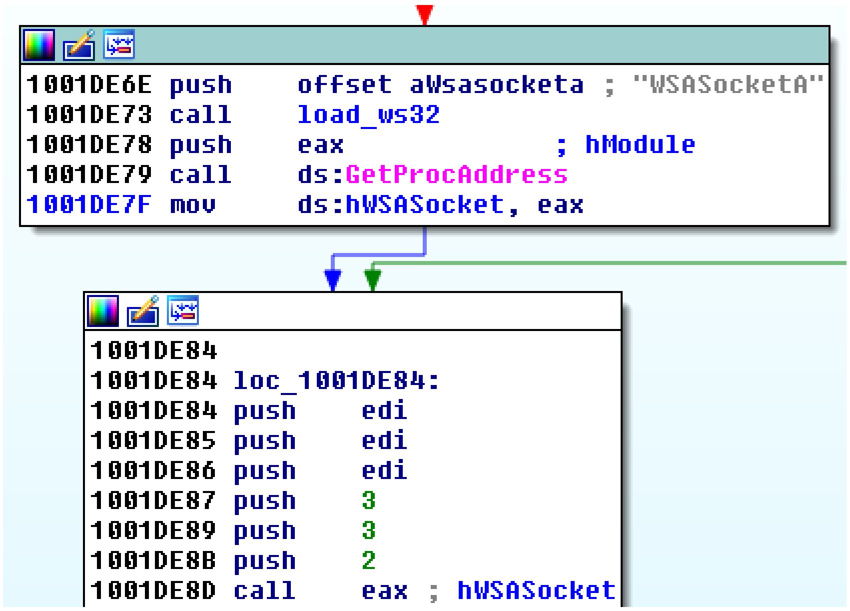


After this small modification, the dumped image can be parsed as a normal PE file (321a2f0abe47977d5c8663bd7a7c7d28). Sections are not named, but all the content is valid:



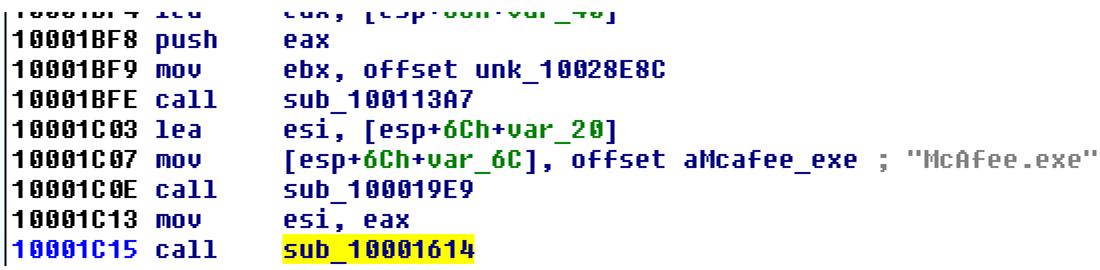
File characteristics describes the payload as a DLL, however, it doesn't have any export table, so we cannot read it's original name.

Looking at the imports loaded by this piece we can suspect that it is the final payload. It loads and uses many functions related to the network communication, i.e:



We can also find the fragment responsible for retrieving the local IP of the current machine and performing LAN scanning that we observed during behavioral analysis.

Authors took care so that the payload will not be run independently. That's why they checks if all the elements are called in the expected order. We can find hardcoded names of the main elements, used for the check:



Conclusion

Malware authors often use fake icons and descriptions in order to disguise as a legitimate product, but this type of attack is going a step forward. Authors used an original McAfee application and hijacked the DLL that it uses, in order to run the malicious code. To make detection more difficult, they tangled elements with each other. None of them can do malicious actions on it's own. That's why, tools that scan each module separately may fail to detect the malicious behavior.

Users are more vigilant about executables – but this time, neither EXE nor DLL file contained the malicious code – they were just used as loaders of the shellcode.

Malwarebytes Anti-Malware detects this threat as 'Trojan.Korplug'.

Appendix

http://e.gov.vn/theo-doi-ngan-chan-ket-noi-va-xoa-cac-tap-tin-chua-ma-doc-a-NewsDetails-37486-14-186.html – info from Vietnamese CERT

http://blog.trendmicro.com/trendlabs-security-intelligence/new-wave-of-plugx-targets-legitimate-apps/ – similar attack from 2013

http://www.welivesecurity.com/2014/11/12/korplug-military-targeted-attacks-afghanistan-tajikistan/ – about the Korplug RAT targeting military of Afganistan and Tajikistan

https://www.blackhat.com/docs/asia-14/materials/Haruyama/Asia-14-Haruyama-I-Know-You-Want-Me-Unplugging-PlugX.pdf – Korplug RAT analysis (presentation from BlackHat)

https://www.f-secure.com/documents/996508/1030745/nanhaishu_whitepaper.pdf – about NanHaiShu APT

This was a guest post written by Hasherezade, an independent researcher and programmer with a strong interest in InfoSec. She loves going in details about malware and sharing threat information with the community. Check her out on Twitter @hasherezade and her personal blog: https://hshrzd.wordpress.com.