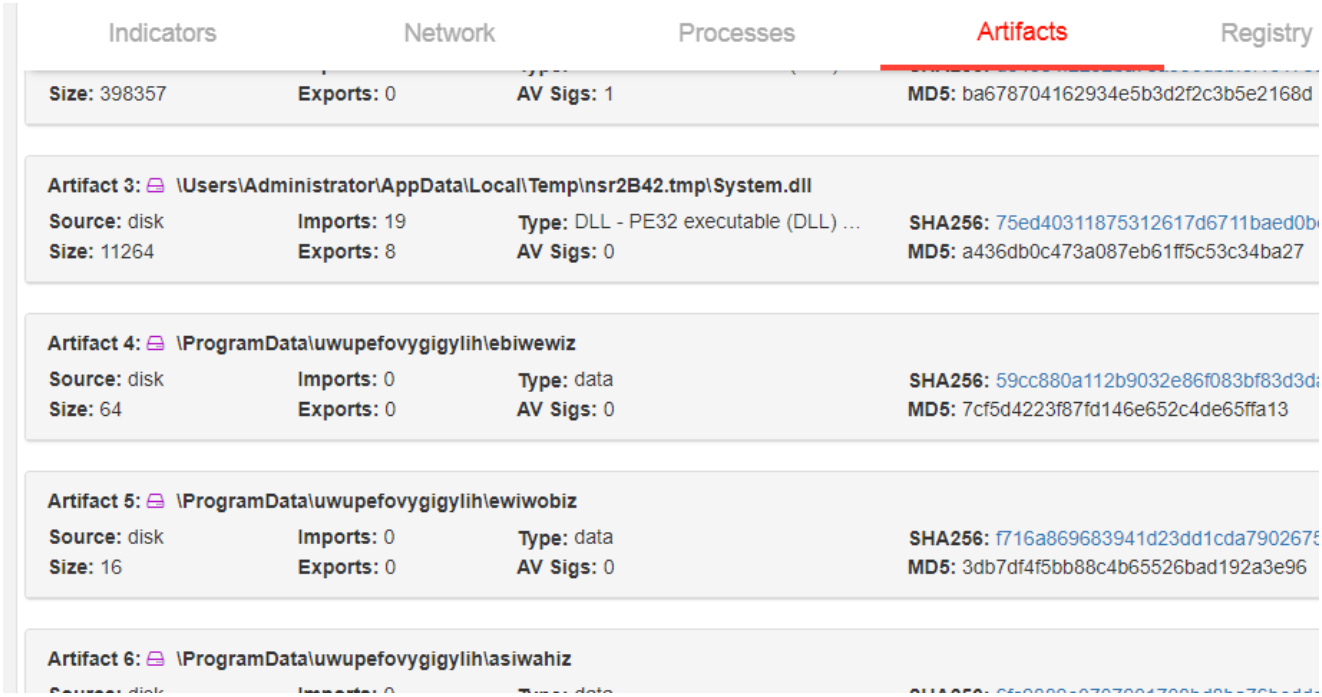
Taking the FIRST look at Crypt0l0cker

blog.talosintelligence.com/2017/08/first-look-crypt0l0cker.html



This post is authored by Matthew Molyett.

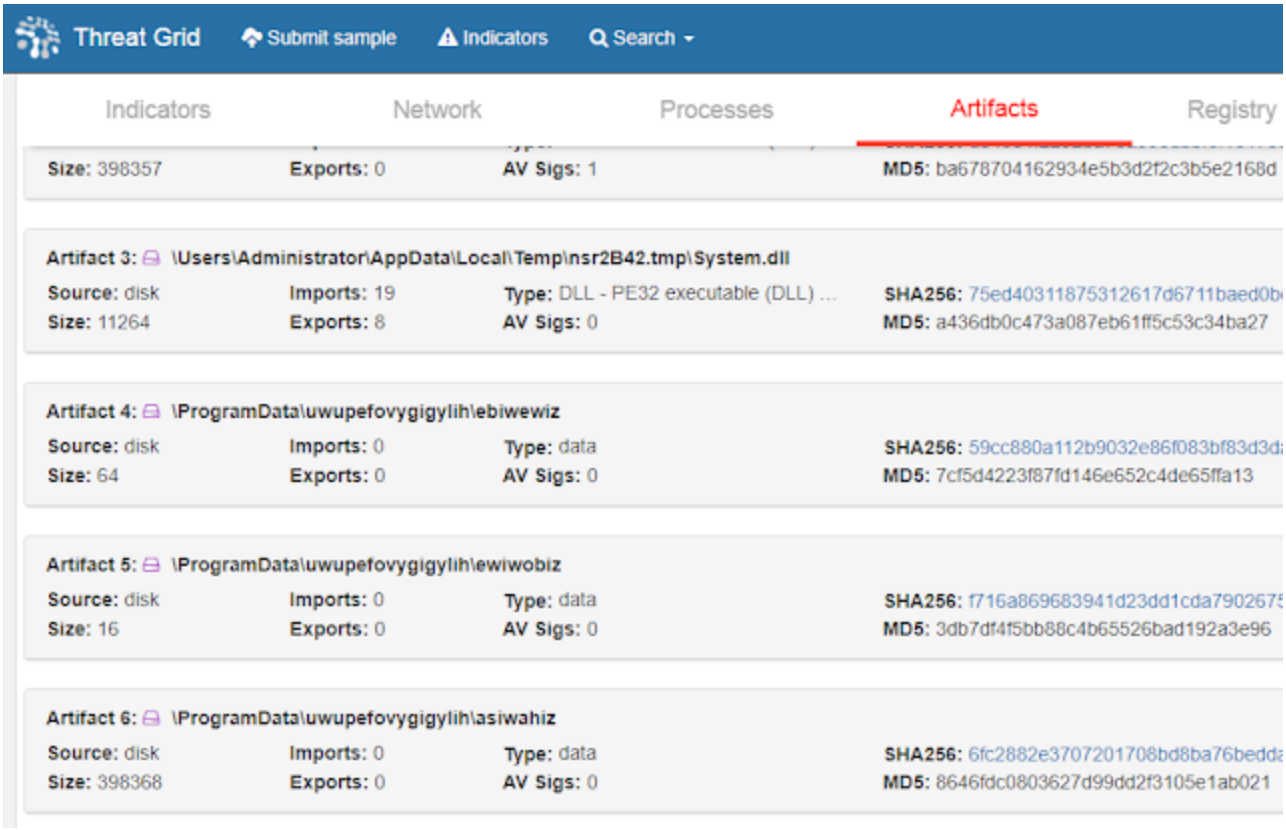
Executive Summary

In March, Talos reported on the details of Crypt0l0cker based on an extensive analysis I carried out on the sample binaries. Binaries -- plural -- because, as noted in the original blog, the Crypt0l0cker payload leveraged numerous executable files which shared the same codebase. Those executables had nearly identical functions in each, but identifying all of those functions repeatedly is tedious and draws time away from improving the analysis. Enter FIRST, the Function Identification and Recovery Signature Tool released by Talos in December 2016.

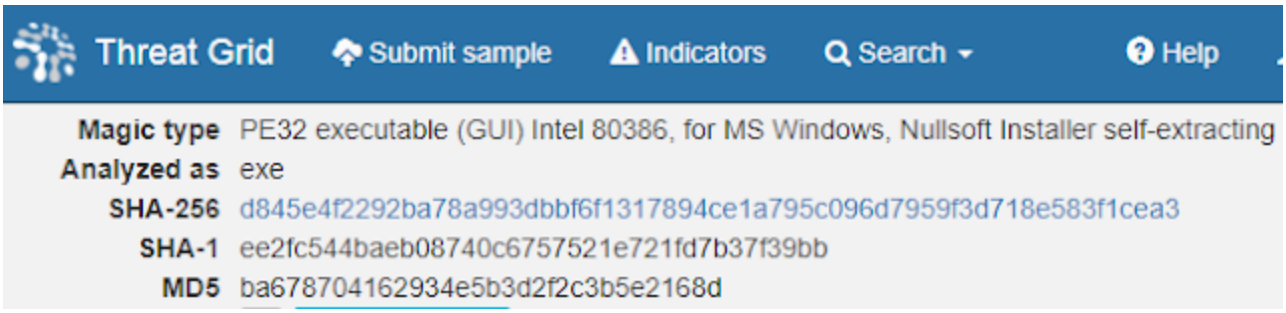
FIRST allowed me to port my analysis from the unpacking dll to the payload file instantly. Once I was satisfied my analysis across both files, I was then handed a suspected previous version of the sample. FIRST was able to identify similar code across the versions and partially port the analysis back to the older file. When the next version of Crypt0l0cker comes out, I will be able to get a jump on my analysis by using FIRST to port that work forward to the similar code. You can use it to port my work to your sample as well. I will demonstrate doing just that with a Crypt0l0cker sample which appeared on VirusTotal in April 2017, more than a month after the Talos blog about it. There has been no targeted analysis of this file to provide background for this post.

Locating the Sample

Procuring a malware sample of a known family without analyzing it can feel like a heavy challenge to overcome. Thankfully, Talos can leverage Threat Grid sandbox reports of suspected malware samples that we receive. Such reports can be scanned for family IOCs. Per our previous analysis into Crypt0l0cker, the infection status of that version is stored in a file named ewiwobiz. By searching Cisco Threat Grid telemetry for files which created ewiwobiz, I identified a file which was probably a Crypt0l0cker executable.



With a report to investigate, I needed to procure the actual sample. My sandbox report shows that the suspected Crypt0l0cker file is nearly 400 kb and likely a Nullsoft Installer file, which is a common packager. Static file information gives me the file hash which arms me with the ability to continue my investigation on VirusTotal.



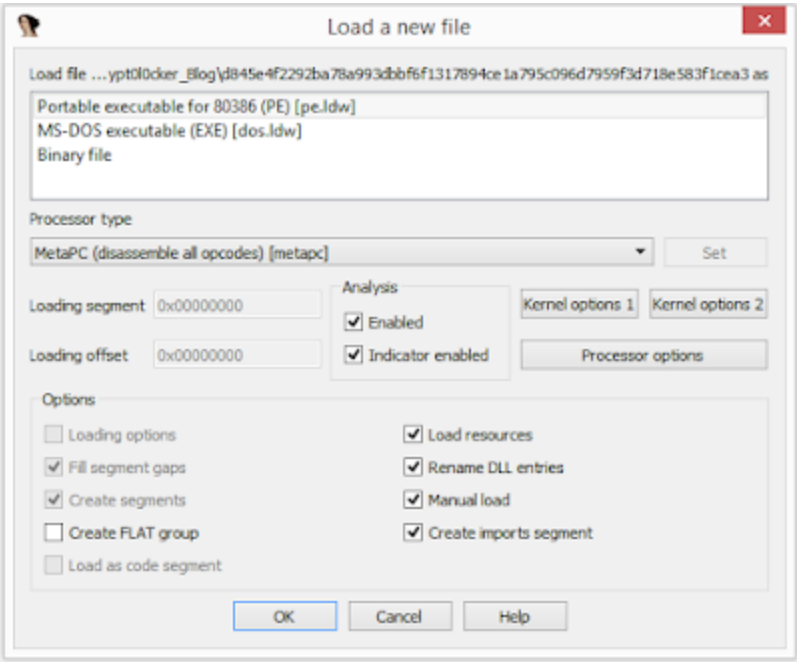
While the sample is clearly malicious, my VirusTotal inspection does not suggest that the

sample belongs to any known family. No detections refer to Crypt0l0cker, TorrentLocker, a listed alias in the original Talos blog, nor Teerac.

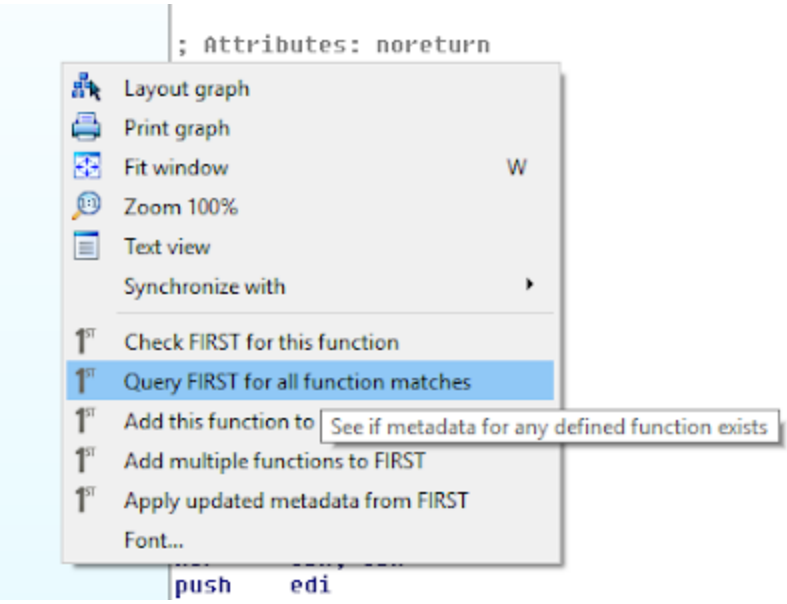
Emsisoft	Trojan.GenericKD.4899437 (B)	4.0.0.834	20170425
Endgame	malicious (high confidence)	0.4.1	20170419
ESET-NOD32	a variant of Win32/Injector.DNXZ	15308	20170425
F-Secure	Trojan.GenericKD.4899437	11.0.19100.45	20170425
Fortinet	W32/Injector.DNXP!tr	5.4.233.0	20170425
GData	Trojan.GenericKD.4899437	A:25.12050B:25.9393	20170425
Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20170425
McAfee	Artemis!BA6787041629	6.0.6.653	20170425
McAfee-GW-Edition	BehavesLike.Win32.Ransom.fc	v2015	20170425

With a file sample in hand, and no static indication that I have located Crypt0l0cker, I move onto FIRST to discover how similar it is to known files.

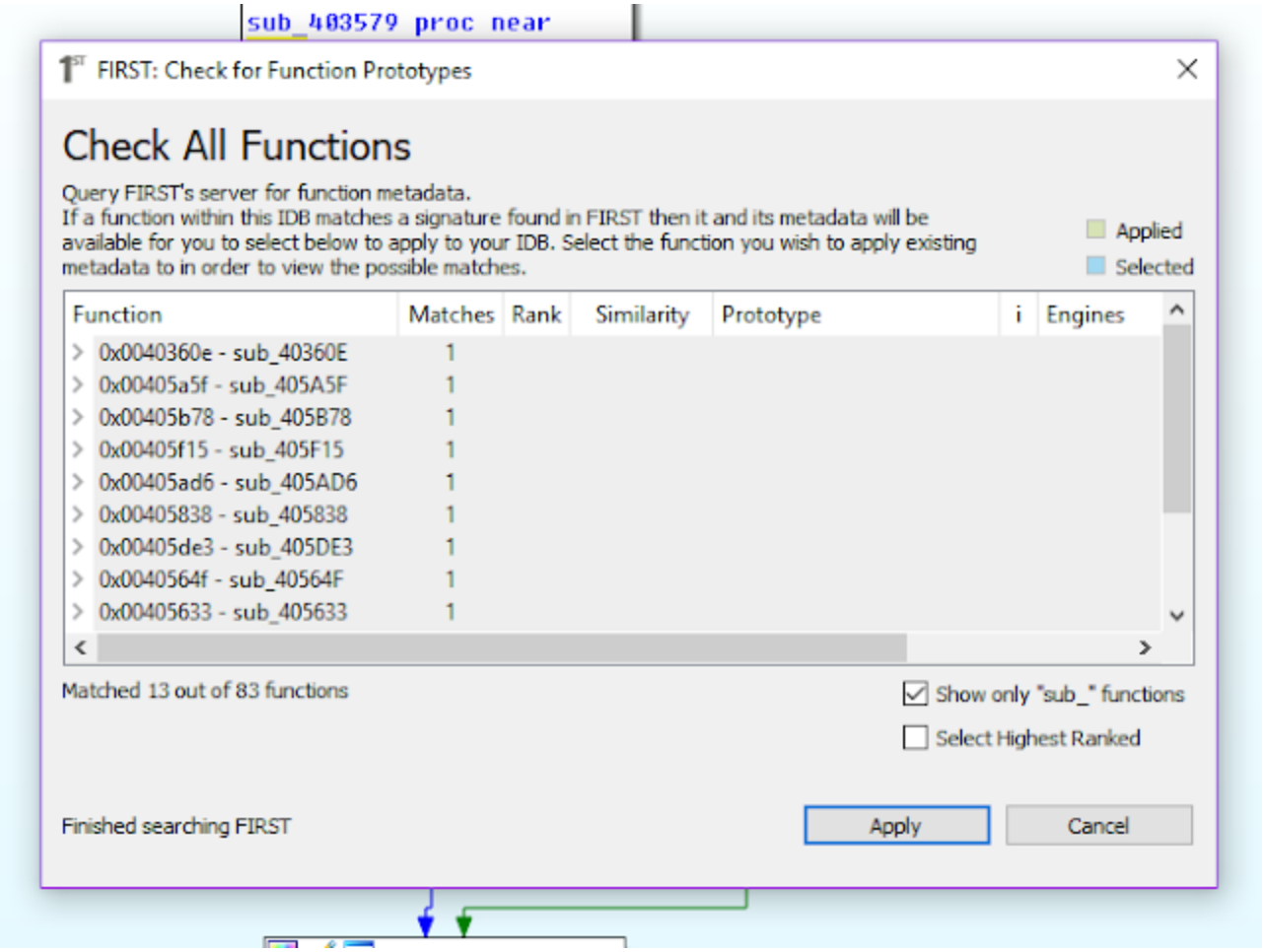
Exploring the Sample



As the FIRST client code is an IDA Pro plugin, my first step was opening the file in my local IDA copy and allow auto-analysis. Upon completion, the start function was displayed in front of me at the graph view. I opened up the graph view context menu and requested FIRST lookups for all of the discovered functions.

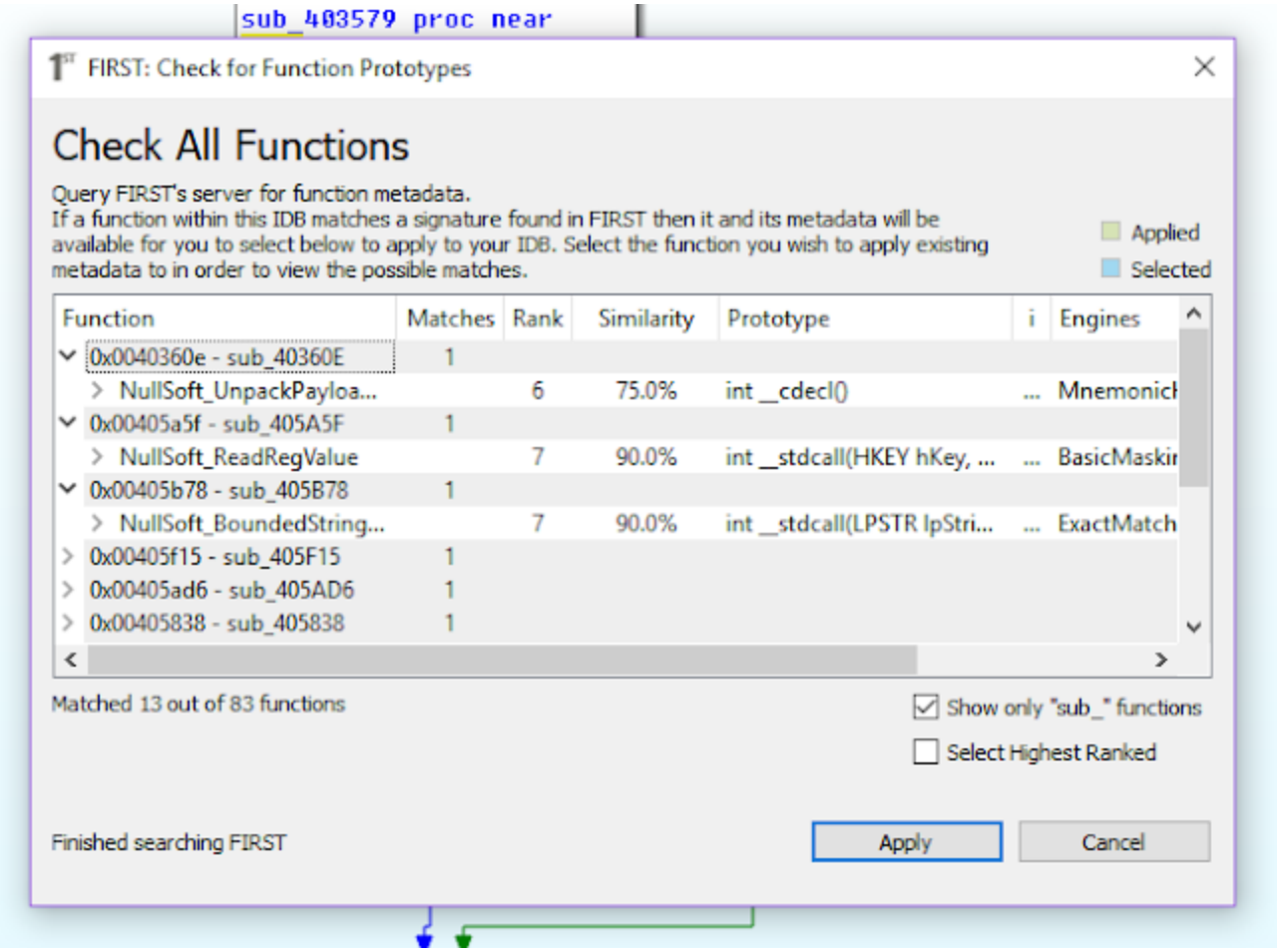


After a minute, the FIRST display shows that 13 of the functions have been previously identified and uploaded.

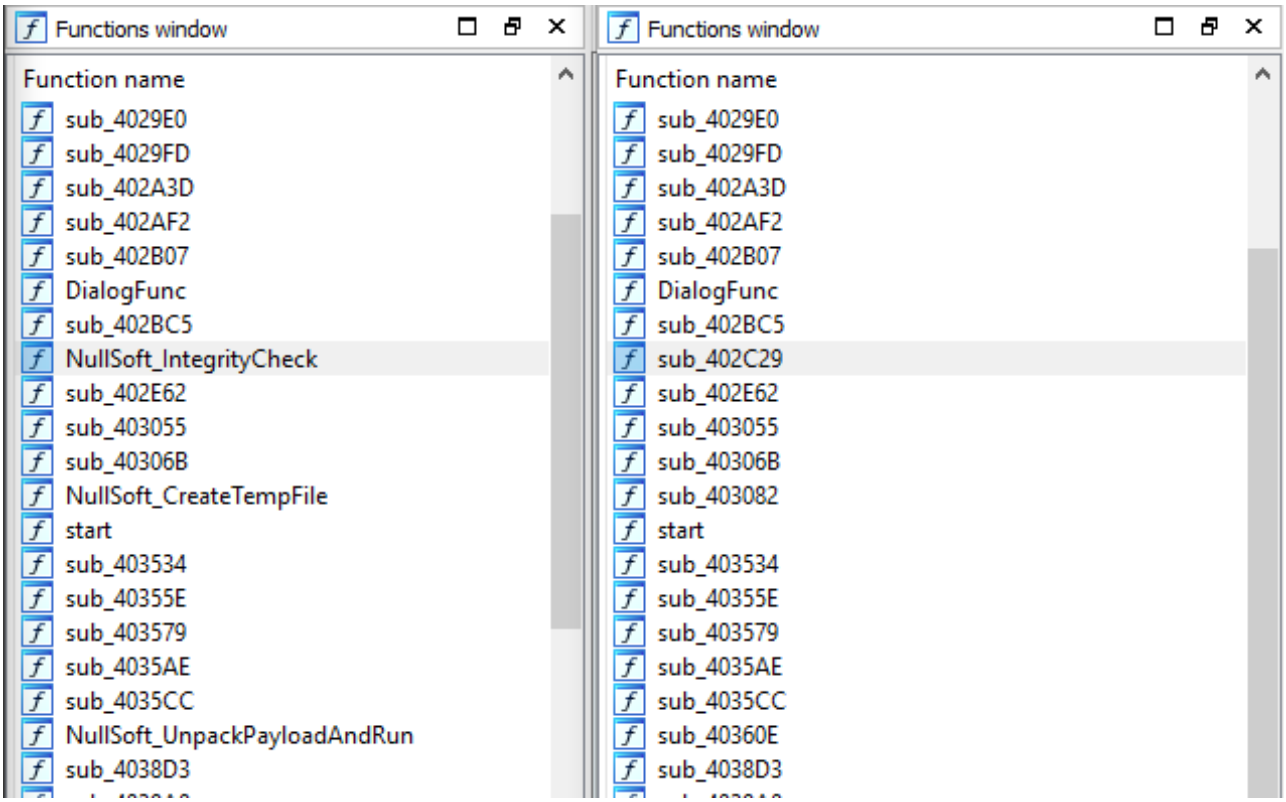


Expanding the matched functions displays the metadata associated with that function, including a proposed name and function prototype. Notice that the files detected in this installer have been named to draw attention to the fact that these functions are known to be

in NullSoft Installers. I had previously marked up a different NullSoft Installer before and uploaded significant functions from it to assure that I would not do so again. In general, a malware analyst is wasting any time spent inspecting such a file. Identifying when a packer is in use and moving along to the true payload is a much better use of time.



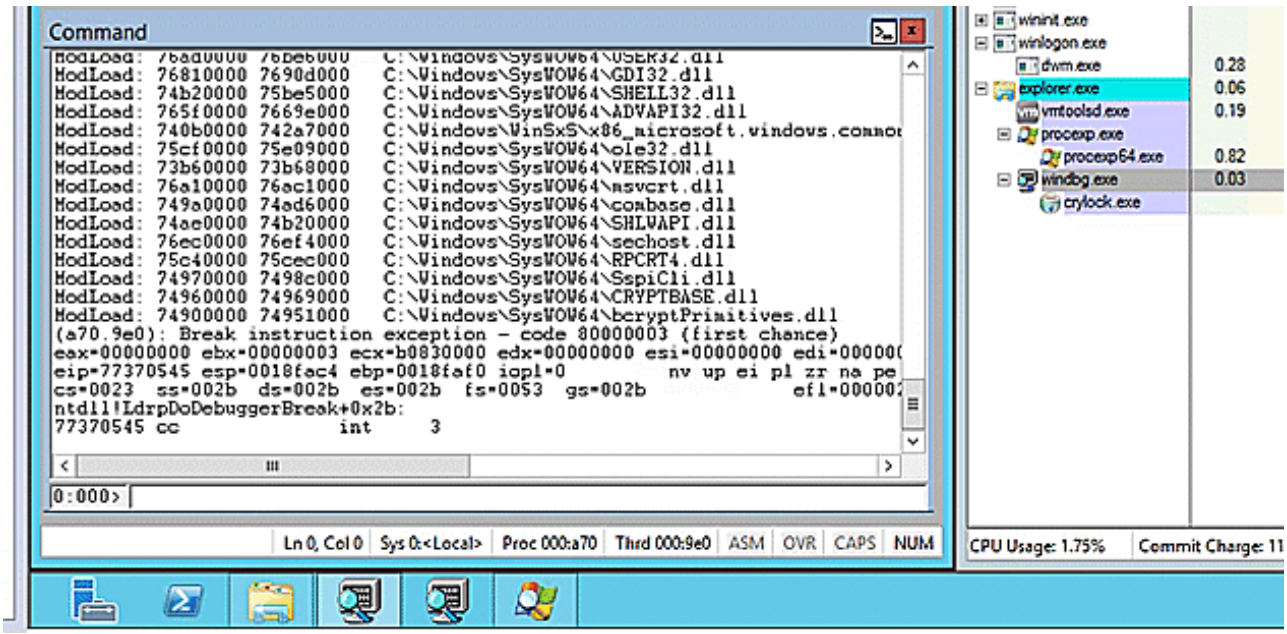
Check the Select Highest Ranked checkbox and click Apply. The function names get applied across the database and we can see clearly that the sandbox analysis was correct. This file is a packer and we need to extract the original.



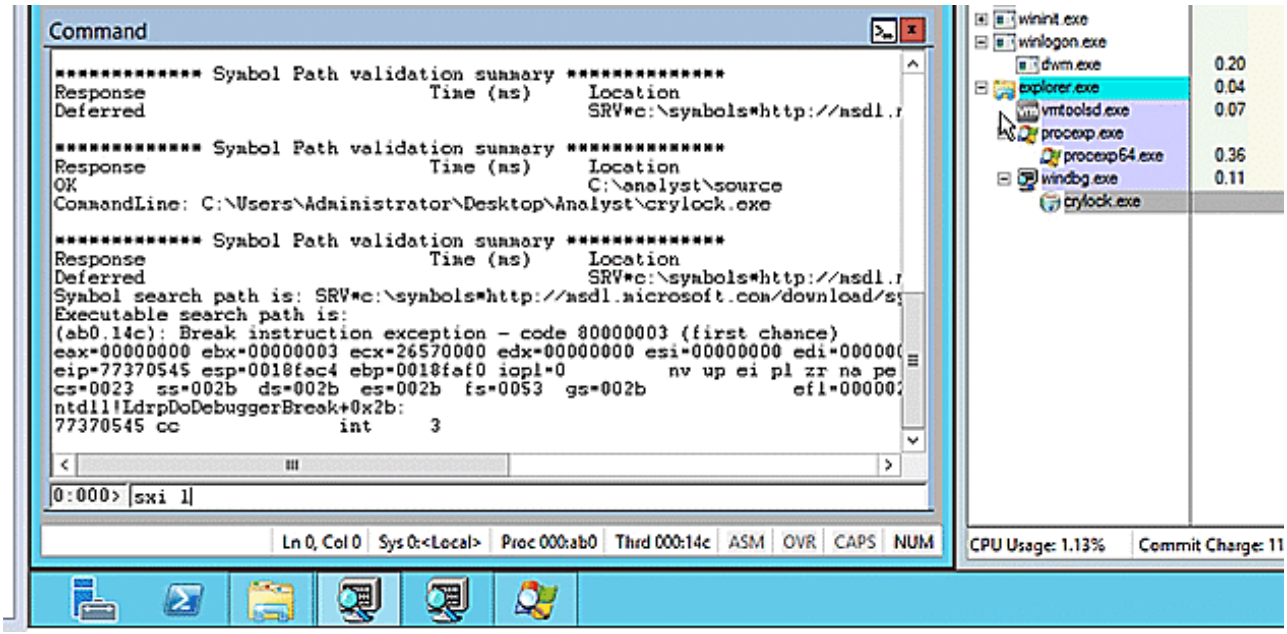
Unpacking the Sample

I admit that at this point I cheated to perform the unpack. From previous extraction of Crypt0l0cker files protected with NullSoft I already knew that the install script consisted of consuming multiple encrypted blobs, internally decrypt the payload, and run it via Process Hollowing. As such, allowing it to run debugged and breaking on WriteProcessMemory should present the payload buffer to me.

There was a complication though, because the install script loaded and unloaded System.dll many times. The ModLoad notification caused the debugger to consume the majority of the process cycles, effectively causing a denial of service against the debugger. I allowed this system to run for over an hour without getting beyond this delay.

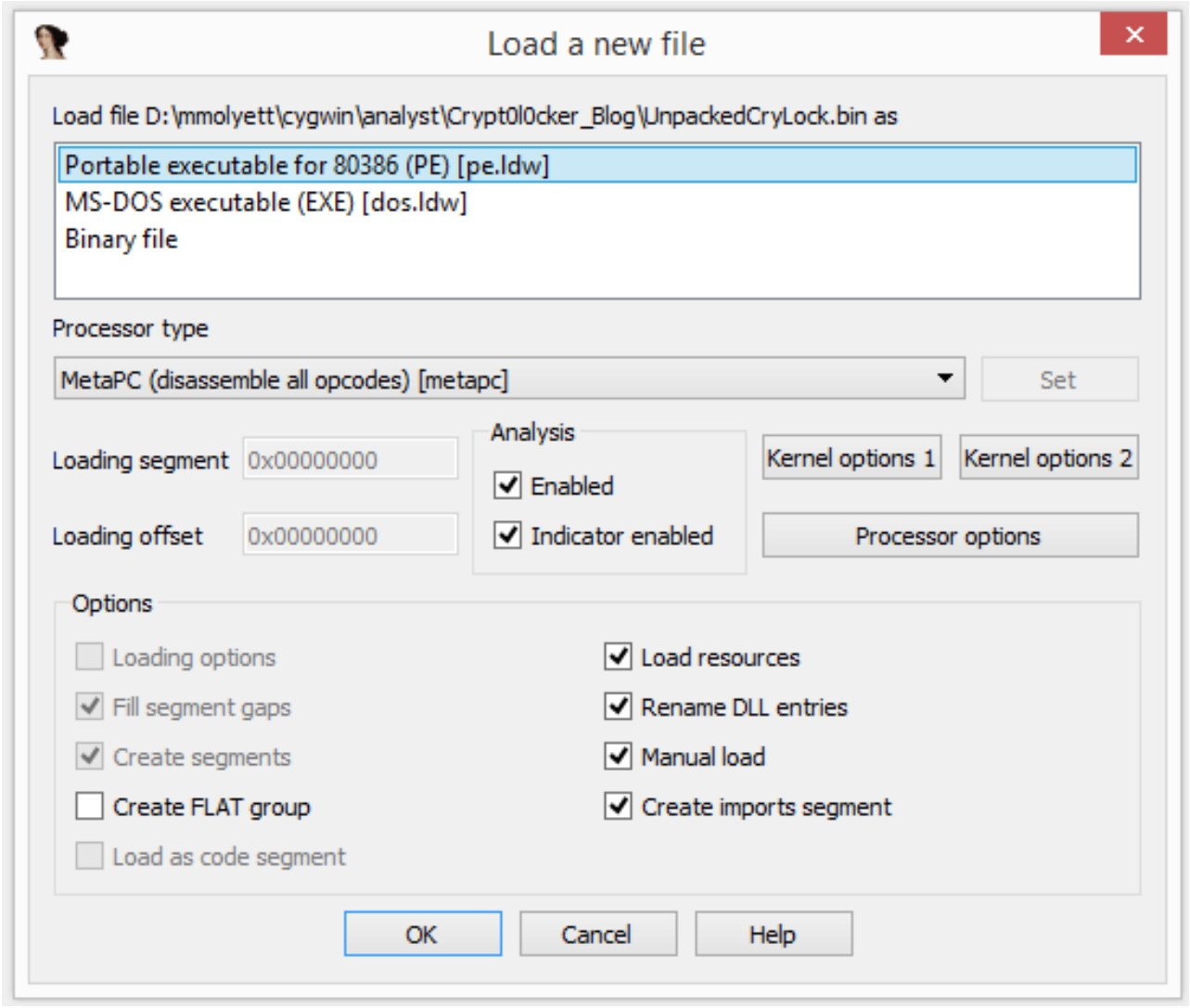


By disabling the ModLoad notification via `sxi ld` I could get my debugger to allow the System.dll file to be loaded without triggering the significant extra processing. Crypt0l0cker then was able to spike up to 99% of the CPU use to, rather than the debugger holding on to 80%.

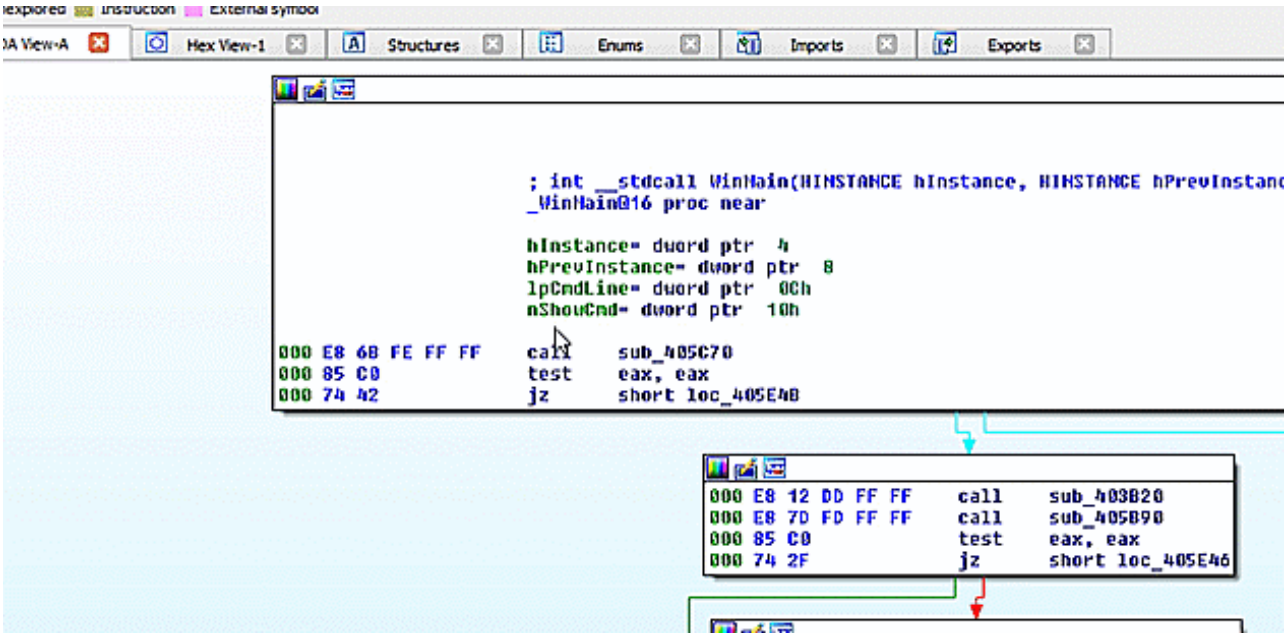


I dumped out the PE image file and prepared to continue with FIRST.

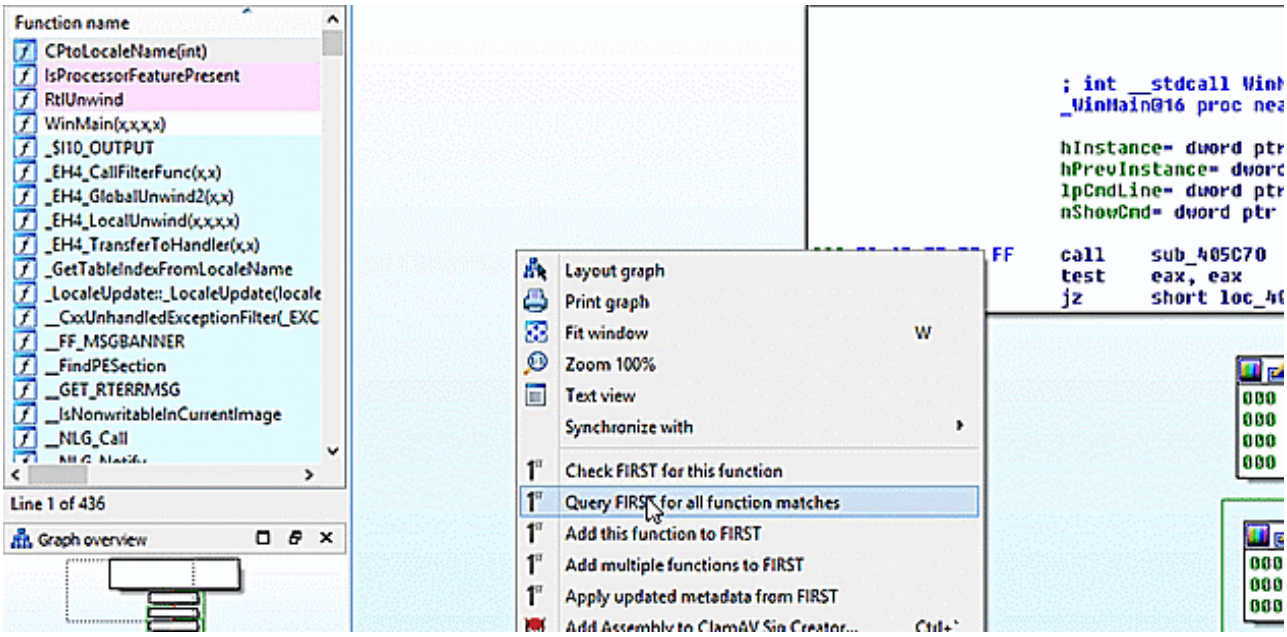
Exploring the Real File



Again, the first step of using FIRST was opening the extracted file in IDA Pro. This file was built as a Windows GUI file on top of the Visual Studio C runtime. Thus, the runtime was identified during auto-analysis and I was left with a graph view displaying the _WinMain@16 function. Using the FIRST command from the context menu, I checked for the existence of metadata for just that one function. It was discovered as Crytp0l0cker_WinMain@16. Looking pretty likely that this is, in fact, Crytp0l0cker.



With confidence that FIRST will be useful, since it had a result for _WinMain@16, I began the search for the full file. At 436 functions this is not a quick lookup, so go get a fresh cup of coffee and let FIRST work. Since this file uses a known runtime, the runtime files are also known to FIRST. You can filter those functions with the Show only "sub_" functions checkbox.



After FIRST completes 78 new function markups are applied out of 295 total known functions. With 78 Crytp0l0cker_* functions identified, you can now dig in on the shoulder of days of professional malware analysis.

Conclusion

FIRST provides the ability to share your work from one file to a similar file, whether that other file is a previous or future version or even an additional step in the module execution. When opening up a new file, FIRST can provide hints as to whether the file is interesting or just needs to be unpacked. When finally extracting a new, embedded binary, FIRST can migrate your notes from the current file to the shared code in the new file. Use FIRST to save your notes, share your discoveries, and speed up your next analysis.

IOC

File Hash

d845e4f2292ba78a993dbbf6f1317894ce1a795c096d7959f3d718e583f1cea3