**Malware Analysis of**

**Dropper.DownloadFromURL.exe**

**Executive Summary: Malware Analysis of Dropper.DownloadFromURL.exe**

This summary outlines the static analysis of the Dropper.DownloadFromURL.exe malware using the Cutter tool. The analysis reveals crucial details about the malware's architecture, behavior, and potential impact. Key findings include the identification of download functionality from a URL, insight into the program's logical flow, and API call decompilation. The summary emphasizes the importance of caution when handling the malware and suggests further dynamic analysis to uncover additional functionalities.

Navigate to this file path

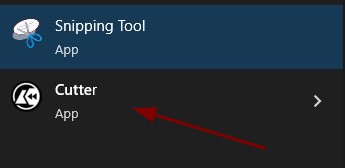
**PMAT-labs/labs/2-1.**

**AdvancedStaticAnalysis/Dropper.DownloadFromURL.exe.malz/Dropper.DownloadFr omURL.exe.7z**

Drag the malware over to Desktop. Don't arm it just yet.

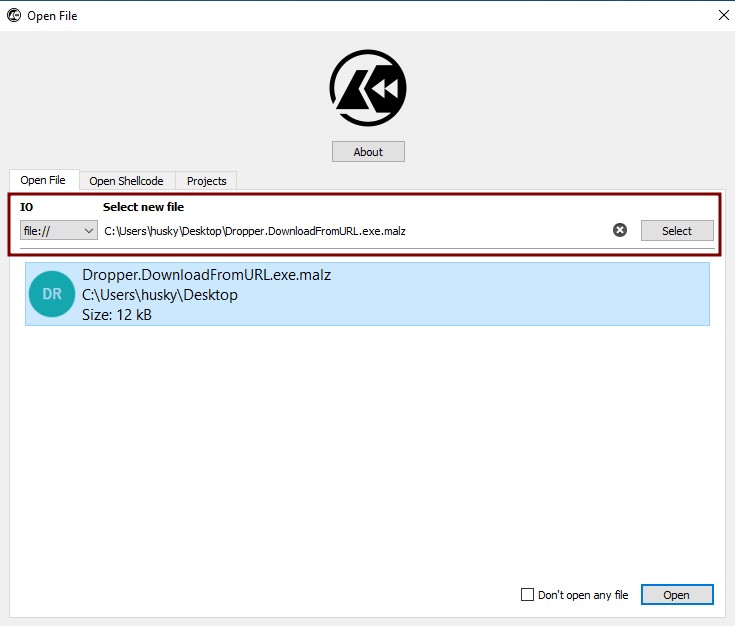


Open up the start menu and search for cutter.

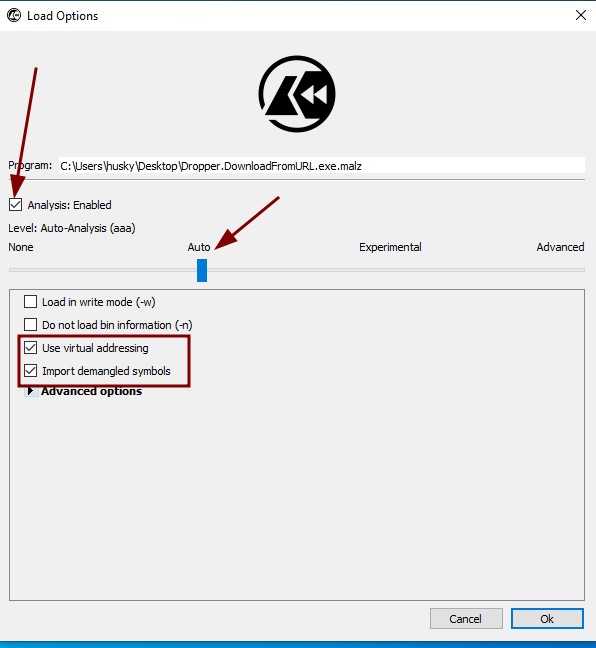


Run it. For IO, ensure it is set to file. Click select file, and find

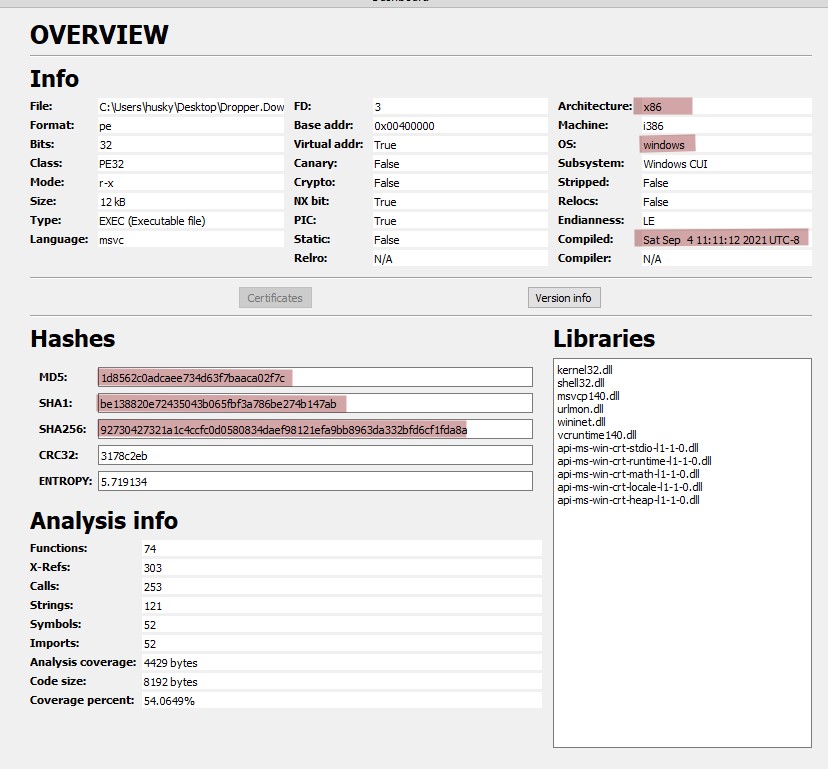
**Dropper.DownloadFromURL.exe.malz**and click Open.



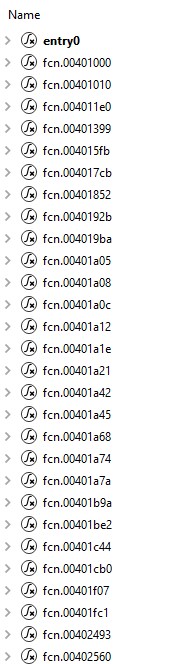
Ensure the correct settings are enabled. Click Ok.



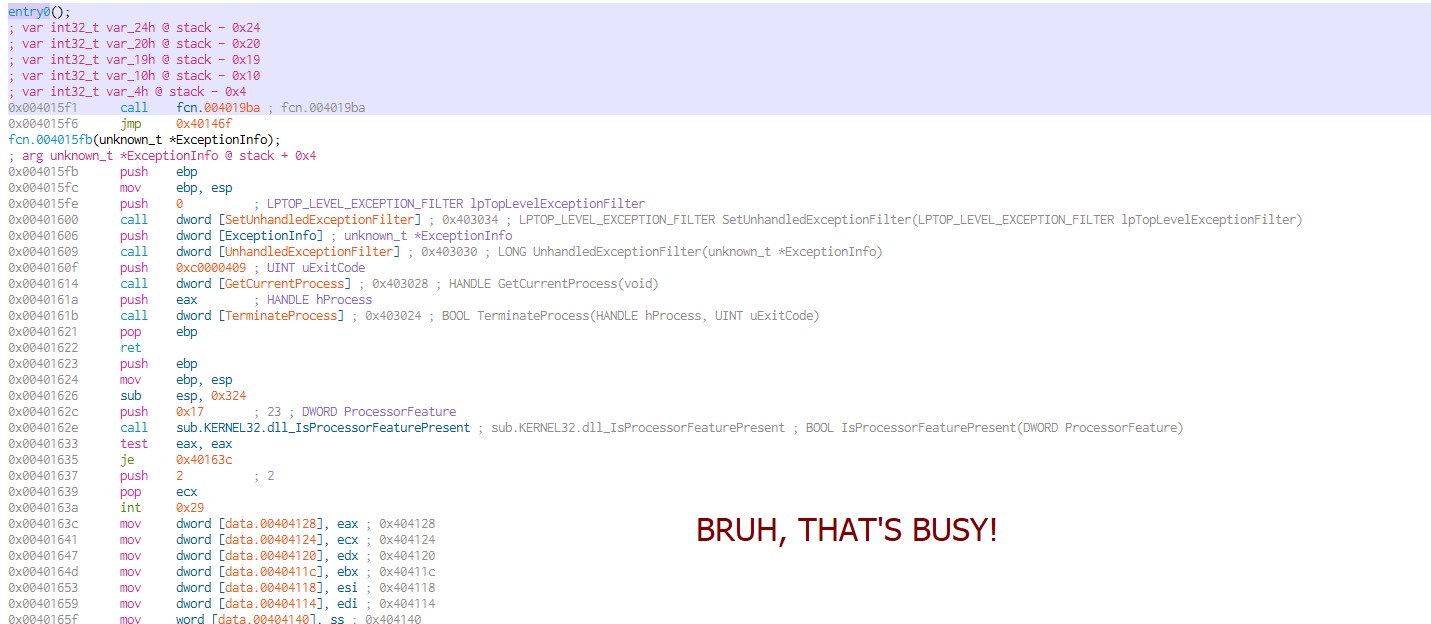
Welcome to the Cutter Dashboard.



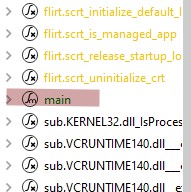
Some cool details we can see is the Architecture, OS, compiled date, MD5, SHA1, and SHA256 hashes. If we look to the left, we can see a called functions list within that respective program.



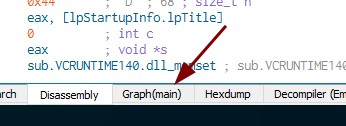
Let's click on entry.



What we are observing is assembly language. When you read and write in assembly, there is nothing to hide, and there is no abstraction to be done. Essentially, this is what is going on when a program is called. Let's find "main" and double click on it.

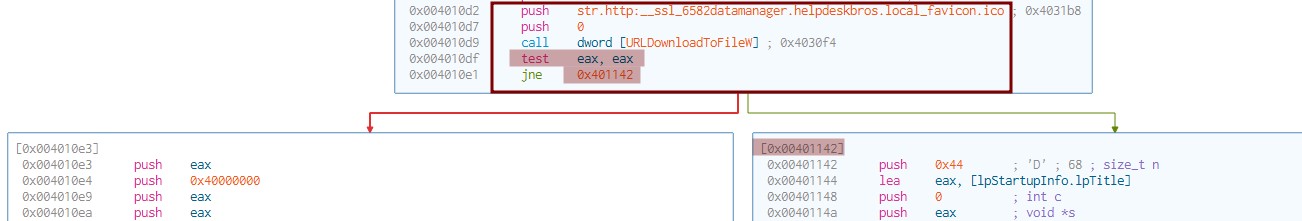


Click on graph at the bottom.

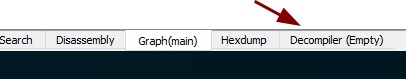




Pretty neat, right? This graphs out the programs logical flow. We can observe the download from URL function. We can also deduce that if it returns as valid, it will execute the rest of the program. If not, it would delete itself from disk.



Let's click on decompiler at the bottom.





So what are we looking at? Essentially, the decompiler will take assembly information within this binary, and attempt to recreate the source code as accurately as possible. We can see the API call and where the call is going.

