CIS 6530 SUBMISSION 3

Technical Report on Malware OpCodes Extraction

REPORT

GROUP 6
XIAOHAI WANG
SAMUEL TRACZ
YU LENG

Introduction

This report documents the process of extracting Operational Codes (OpCodes) from malware samples to prepare data for training and testing AI agents. We utilized code-reversing tools like Ghidra as the primary method, supplemented by objdump, to perform in-depth analysis on executable files. Platform involves linux and windows virtual machine.

Tools and Environment

- **Ghidra**: A reverse-engineering framework used to disassemble and extract OpCodes from executables. We automated the process using Ghidra's Headless Mode for batch processing.
- **objdump**: A secondary tool used for OpCode extraction, providing additional verification and flexibility.
- **UPX**: Used to unpack packed executables, which is a common technique malware uses to hide code.

Methodology

Due to the malware sample may packed by APT group (author) to prevent the Dismantling engineering to extract the operation codes. We use the Ghidra and UPX scan and unpack all all possible samples.

```
root@kali:~/Desktop/APTOPCODETEST/APTEXE/APT28# chmod +x unpack.sh
root@kali:~/Desktop/APTOPCODETEST/APTEXE/APT28# ./unpack.sh
```

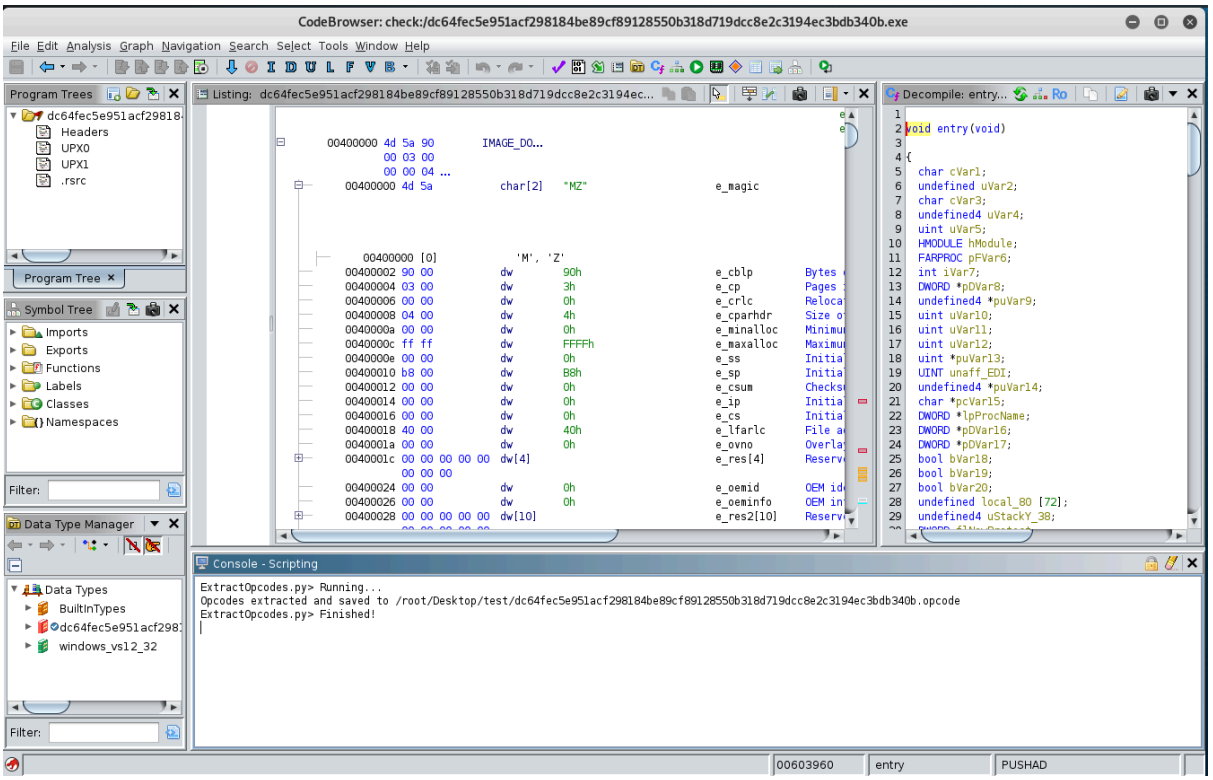
Sample results:

```
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
   3666944 <-   1510400   41.19%    win32/pe     fcf03bf5ef4babce577dd1348339134
4e957fd2c855624c9f0573880b8cba62e.exe

Unpacked 1 file.
.//fcf03bf5ef4babce577dd13483391344e957fd2c855624c9f0573880b8cba62e.exe unpacked
 successfully.
Repacking .//fcf03bf5ef4babce577dd13483391344e957fd2c855624c9f0573880b8cba62e.ex
e...
```

Using Ghidra and run the Ghidra script and to extract the filtered OpCode.

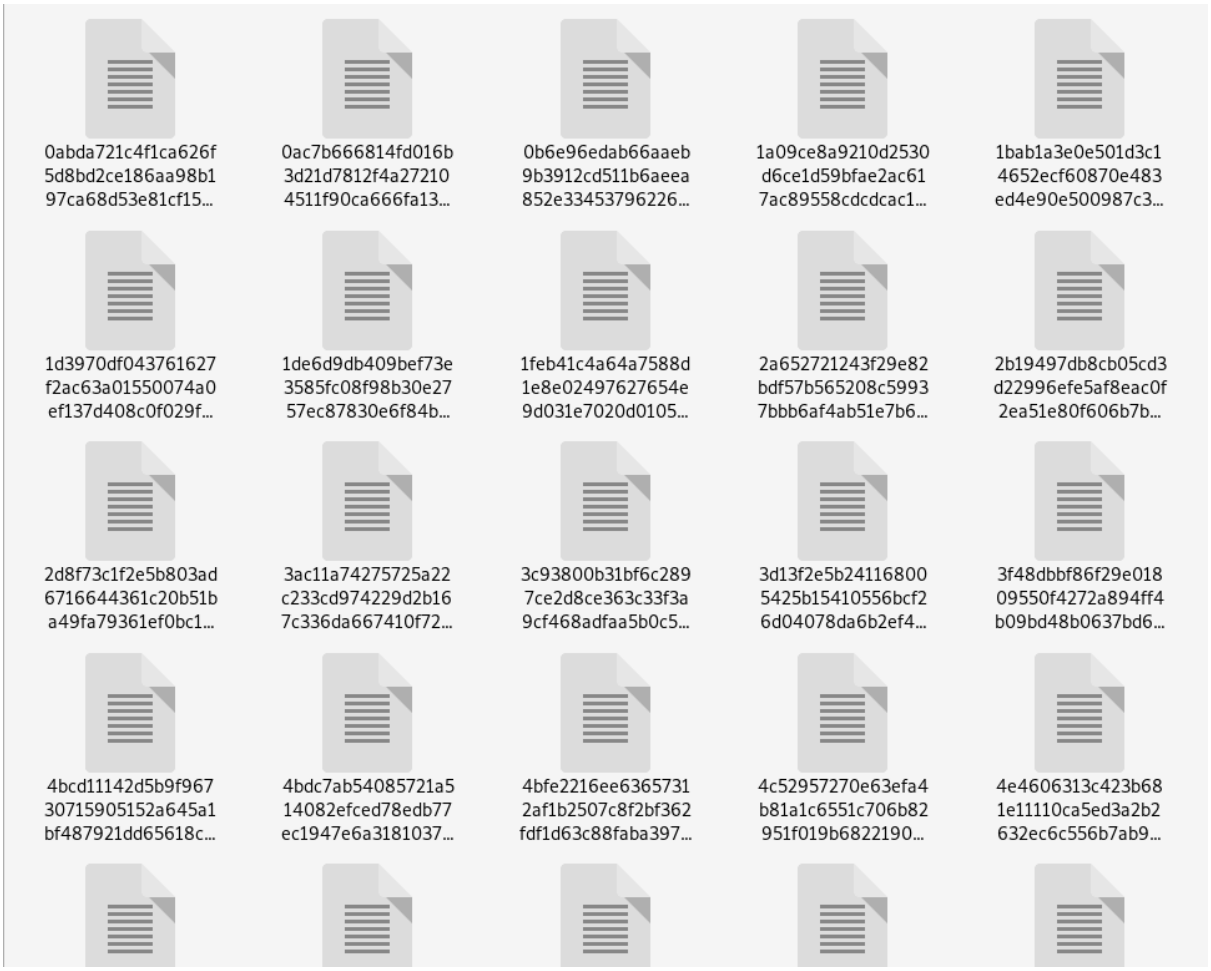



```
ExtractOpcodes.py> Running...
Opcodes extracted and saved to /root/Desktop/test/dc64fec5e951acf298184be89cf89128550b318d719dcc8e2c3194ec3bdb340b.opcode
ExtractOpcodes.py> Finished!
```

Using objdump script ‘testDESMF.sh’ and ‘testDESM’’ to extract all possible OpCodes involves mnemonic and hashes.

```
root@kali:~/Desktop/APTOPCODETEST/APTEXE/APT28# chmod +x testDESMF.sh
root@kali:~/Desktop/APTOPCODETEST/APTEXE/APT28# ./testDESMF.sh
Extracting and simplifying opcodes for 018a3fbea5a8a5c0d2680428ae48ba865c4c88cb8
09e6875208368f5d016a51b...
Opcode extraction and simplification complete for 018a3fbea5a8a5c0d2680428ae48ba
865c4c88cb809e6875208368f5d016a51b
Extracting and simplifying opcodes for 0320298eea0206b71d12f3a69730bbbec9768c5c3
23dfe131047f7ba4f4a8868...
Opcode extraction and simplification complete for 0320298eea0206b71d12f3a69730bb
```

Outputs samples (including opcode mnemonic and hex):



Conclusion

Using Ghidra as the primary tool allows for accurate, structured opcode extraction. objdump serves as a quick verification tool, useful for analysis tasks. By combining these methods, we were able to extract a reliable set of opcode mnemonics from the samples for further analysis. Collection and useful scripts were uploaded to the GitHun team repositories and ready for further process.