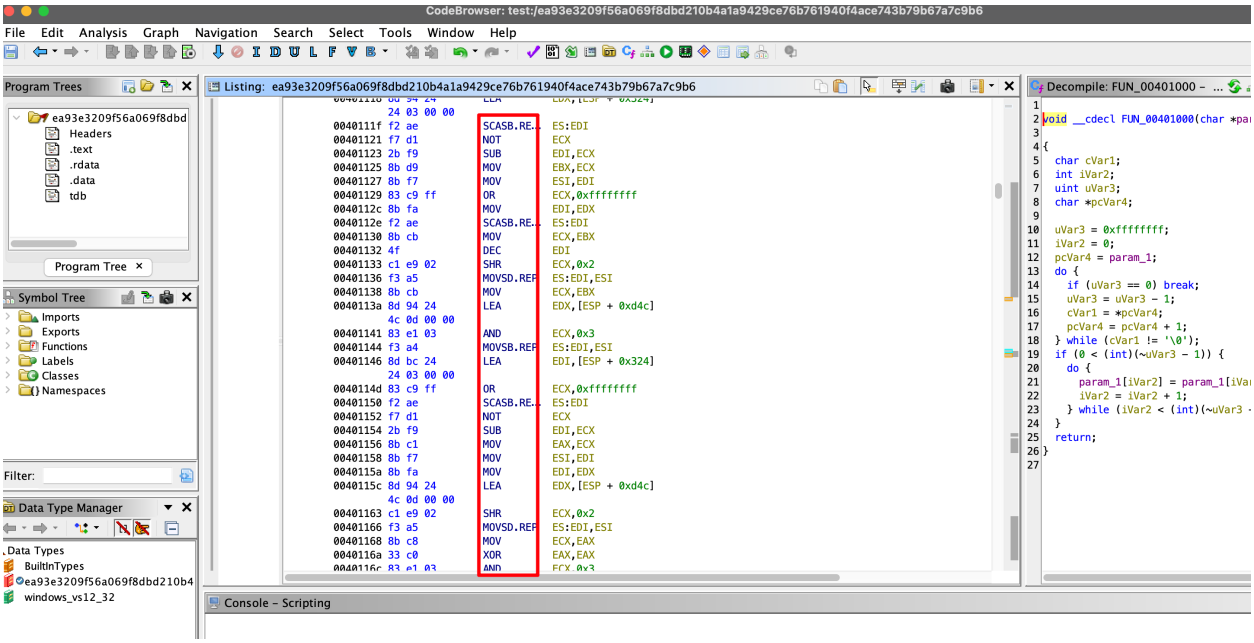
submission3

Find out what is opcode

I open payloads in ghidra, and google it. Opcode is the `mnemonics` instructions.



Write python code for extraction

I use `objdump` command to get Disassembly file, and extract mnemonics from the disassembly .txt file and save to .opcode file.

```
import os
import zipfile
import subprocess
from pathlib import Path
import shutil
import re

def save_disassembly_to_txt(file_path, output_txt_path):
    """Disassemble the binary file and save the output to a .
txt file."""
```

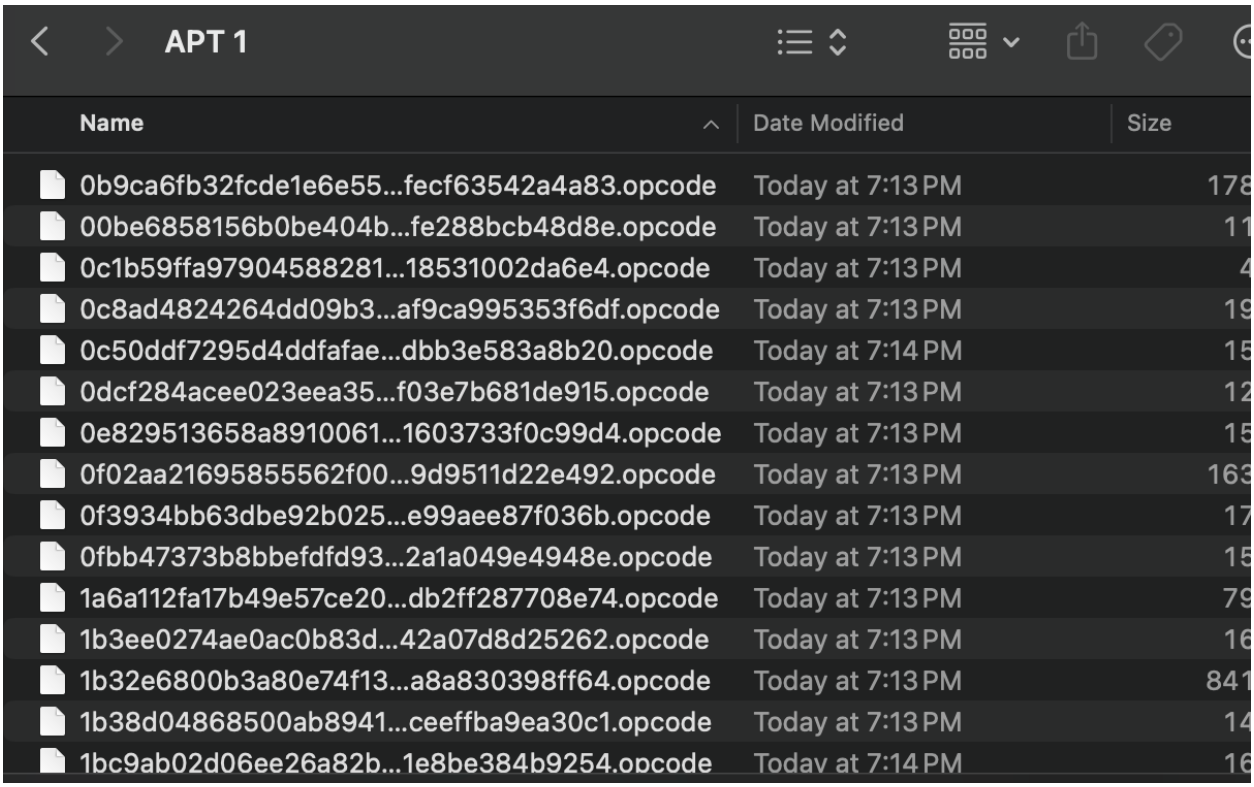
```
    try:
        with open(output_txt_path, 'w') as f:
            subprocess.run(
                ["objdump", "-d", "-M", "intel", str(file_pat
h)],
                stdout=f,
                text=True,
                check=True
            )
        print(f"Disassembly saved to {output_txt_path}")
    except subprocess.CalledProcessError as e:
        print(f"Error running objdump on {file_path}: {e}")

def extract_mnemonics_from_txt(txt_file_path, opcode_output_p
ath):
    """Extract mnemonics from the disassembly .txt file and s
ave to .opcode file."""
    mnemonics = []
    mnemonic_pattern = re.compile(r'^\s*[0-9a-f]+:\s+([0-9a-
f]{2}\s+)+\s+([a-z]+)\b', re.IGNORECASE)

    try:
        with open(txt_file_path, 'r') as f:
            for line in f:
                match = mnemonic_pattern.search(line)
                if match:
                    mnemonic = match.group(2)
                    mnemonics.append(mnemonic)

        # Save mnemonics to .opcode file
        with open(opcode_output_path, 'w') as f:
            f.write("\n".join(mnemonics))
        print(f"Mnemonic sequence saved to {opcode_output_pat
h}")

    except FileNotFoundError:
```

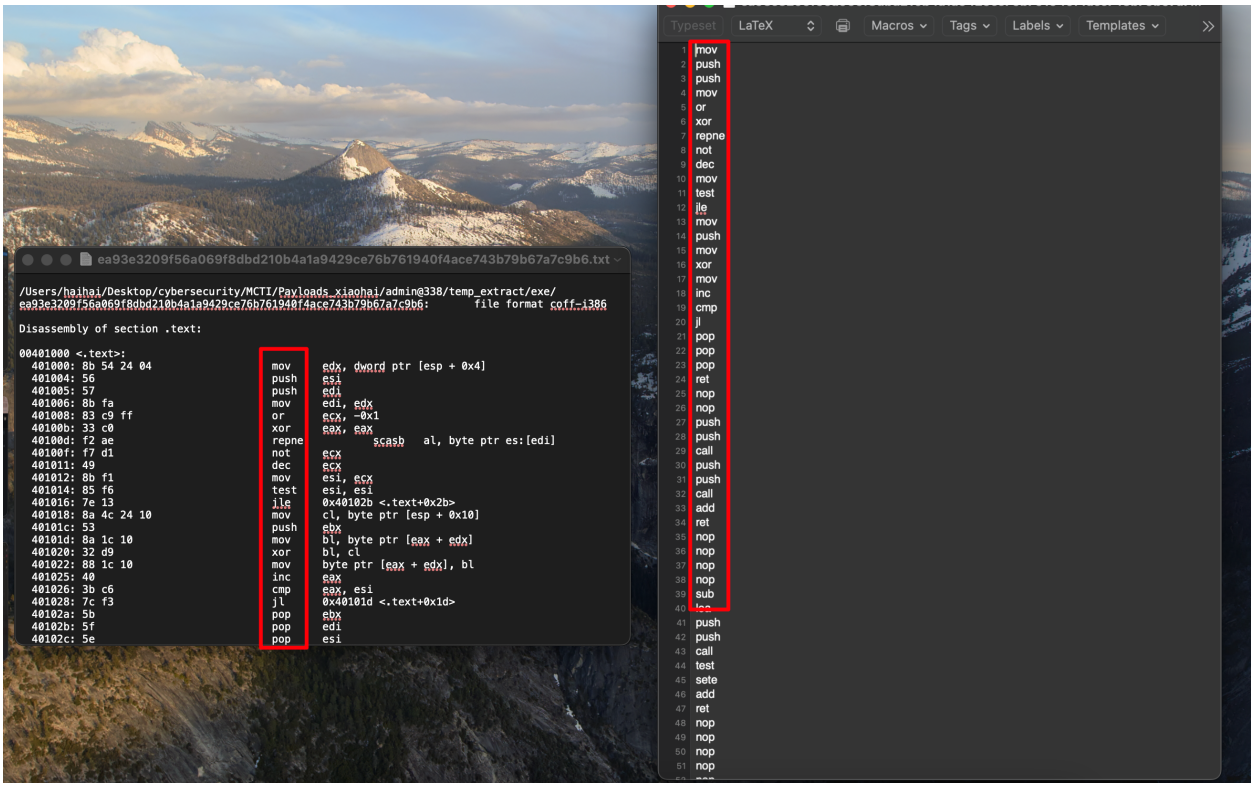
```
        print(f"Error: The file {txt_file_path} was not foun
d.")

def process_zip_file(zip_path, opcode_result_dir):
    """Process the zip file, extract binary files, disassembl
e, and save mnemonics."""
    zip_path = Path(zip_path)
    zip_name = zip_path.stem  # Get the name of the ZIP file
without the extension

    # Create subdirectories for txt files and opcode files
    txt_output_dir = zip_path.parent / "disassembly_txt"
    txt_output_dir.mkdir(exist_ok=True)

    # Create a specific folder for this ZIP's opcode files in
side the opcode_result_dir
    opcode_zip_dir = opcode_result_dir / zip_name
    opcode_zip_dir.mkdir(parents=True, exist_ok=True)

    # Create a temporary directory for extracting the ZIP con
tents
    temp_dir = zip_path.parent / "temp_extract"
    temp_dir.mkdir(exist_ok=True)

    # Extract the ZIP file
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(temp_dir)

    # Look for the 'exe' folder within the extracted contents
    exe_folder = next(temp_dir.rglob("exe"), None)
    if not exe_folder:
        print(f"No 'exe' folder found in {zip_path.stem}")
        return

    # Process each file in the 'exe' folder
    for file_path in exe_folder.iterdir():
```

```
        if file_path.is_file() and not file_path.suffix:
            print(f"Processing file: {file_path.name}")

            # Define paths for the txt and opcode files
            disassembly_txt_path = txt_output_dir / f"{file_p
ath.stem}.txt"
            opcode_file_path = opcode_zip_dir / f"{file_path.
stem}.opcode"

            # Save disassembly to the txt file
            save_disassembly_to_txt(file_path, disassembly_tx
t_path)

            # Extract mnemonics and save to the opcode file
            extract_mnemonics_from_txt(disassembly_txt_path,
opcode_file_path)

    # Clean up the temporary extraction directory
    shutil.rmtree(temp_dir)
    print(f"Temporary files cleaned up for {zip_path.stem}.")

def process_all_folders(root_directory, opcode_result_dir):
    """Process each folder in the root directory, looking for
a single ZIP file in each."""
    root_path = Path(root_directory)
    opcode_result_dir = Path(opcode_result_dir)

    # Ensure the root path and opcode result directory exist
    if not root_path.exists():
        print(f"Error: The root directory '{root_path}' does
not exist.")
        return
    opcode_result_dir.mkdir(parents=True, exist_ok=True)

    # Process each folder in the root directory
    for folder in root_path.iterdir():
```

```
        if folder.is_dir():
            # Find the first ZIP file in the folder
            zip_files = list(folder.glob("*.zip"))
            if zip_files:
                zip_file_path = zip_files[0]
                print(f"\nProcessing folder: {folder.name} wi
th ZIP file: {zip_file_path.name}")
                process_zip_file(zip_file_path, opcode_result
_dir)
            else:
                print(f"No ZIP file found in {folder.name}")

def main():
    root_directory = "/Users/haihai/Desktop/cybersecurity/MCT
I/Payloads_xiaohai"  # Set to your root directory path
    opcode_result_dir = "/Users/haihai/Desktop/cybersecurity/
MCTI/opcode_result"  # Path to save opcode files
    process_all_folders(root_directory, opcode_result_dir)

if __name__ == "__main__":
    main()
```

Get opcode file


APT 1
Name
Date Modified
Size
0b9ca6fb32fcde1e6e55...fecf63542a4a83.opcode
Today at 7:13 PM
178
00be6858156b0be404b...fe288bcb48d8e.opcode
Today at 7:13 PM
11
0c1b59ffa97904588281...18531002da6e4.opcode
Today at 7:13 PM
4
0c8ad4824264dd09b3...af9ca995353f6df.opcode
Today at 7:13 PM
19
0c50ddf7295d4ddfafae...dbb3e583a8b20.opcode
Today at 7:14 PM
15
0dcf284acee023eea35...f03e7b681de915.opcode
Today at 7:13 PM
12
0e829513658a8910061...1603733f0c99d4.opcode
Today at 7:13 PM
15
0f02aa21695855562f00...9d9511d22e492.opcode
Today at 7:13 PM
163
0f3934bb63dbe92b025...e99aee87f036b.opcode
Today at 7:13 PM
17
0fbb47373b8bbefdfd93...2a1a049e4948e.opcode
Today at 7:13 PM
15
1a6a112fa17b49e57ce20...db2ff287708e74.opcode
Today at 7:13 PM
79
1b3ee0274ae0ac0b83d...42a07d8d25262.opcode
Today at 7:13 PM
16
1b32e6800b3a80e74f13...a8a830398ff64.opcode
Today at 7:13 PM
841
1b38d04868500ab8941...ceeffba9ea30c1.opcode
Today at 7:13 PM
14
1bc9ab02d06ee26a82b...1e8be384b9254.opcode
Today at 7:14 PM
16


ea93e3209f56a069f8dbd210b4a1a9429ce76b761940f4ace743b79b67a7c9b6.txt
/Users/haihai/Desktop/cybersecurity/MCTI/Payloads_xiaohai/admin@338/temp_extract/exe/
ea93e3209f56a069f8dbd210b4a1a9429ce76b761940f4ace743b79b67a7c9b6: file format coff-i386
Disassembly of section .text:
00401000 <.text>:
401000: 8b 54 24 04 mov edx, dword ptr [esp + 0x4]
401004: 56 push esi
401005: 57 push edi
401006: 8b fa mov edi, edx
401008: 83 c9 ff or ecx, -0x1
40100b: 33 c0 xor eax, eax
40100d: f2 ae repne scasb al, byte ptr es:[edi]
40100f: f7 d1 not ecx
401011: 49 dec ecx
401012: 8b f1 mov esi, ecx
401014: 85 f6 test esi, esi
401016: 7e 13 jle 0x40102b <.text+0x2b>
401018: 8a 4c 24 10 mov cl, byte ptr [esp + 0x10]
40101c: 53 push ebx
40101d: 8a 1c 10 mov bl, byte ptr [eax + edx]
401020: 32 d9 xor bl, cl
401022: 88 1c 10 mov byte ptr [eax + edx], bl
401025: 40 inc eax
401026: 3b c6 cmp eax, esi
401028: 7c f3 jl 0x40101d <.text+0x1d>
40102a: 5b pop ebx
40102b: 5f pop edi
40102c: 5e pop esi
Typeset
LaTeX
Macros
Tags
Labels
Templates
mov
push
push
mov
or
xor
repne
not
dec
mov
test
jle
mov
push
mov
xor
mov
inc
cmp
jl
pop
pop
pop
ret
nop
nop
push
push
call
push
push
call
add
ret
nop
nop
nop
nop
sub
push
push
call
test
sete
add
ret
nop
nop
nop
nop