Rerverse Engineer: Lab 2 Report

Jesus Lopez
Worked in class with Alejandro Rodriguez and Eduardo Menendez
Dr. Saidur Rahman

September 16, 2024

Contents

Chapter 1

IDA

1.1 Introduction

The tool IDA is a powerful dissasembler for reverse engineering and binary analysis. the input of the program is the executable file and transform it into assembly code. For this program there are 2 versions, the IDA Free and IDA PRO, the main difference is that is does not include a decompiler and it can only be used in traditional architecture such as x86 and x64.

1.2 Exercise 1: Basic Disassembly Navigation

1.2.1 Objective:

Get comfortable with loading binaries, viewing dissasembly, and basic navigation using IDA.

1.2.2 Steps Taken:

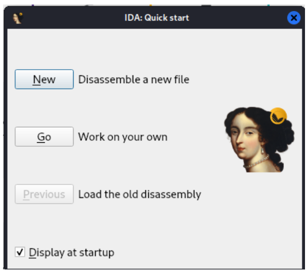
1. We have to download a Windows PE file, it can be from blackboard or from **Bazaar.abuse.h**
2. Load the PE File into IDA Free by selecting **New File** and navigating to the PE file.
3. In the dissasemly view, we have to locate the entry point.
4. We have to Identify the first five assembly instructions listed in the dissasemly window.

1.2.3 Analysis

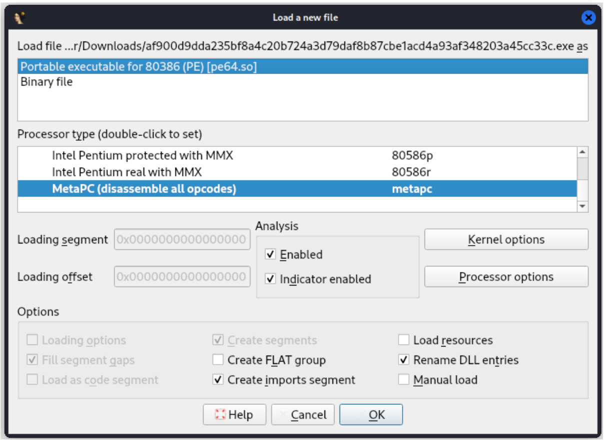
In order to start using IDA, we have to downloaded and give permissions:

```
chmod 777 idafree84_linux.run
./idafree84_linux.run
```

After that we have to go to the folder and run it. It will pop a screen like this:



We are going to be analyzing a Windows PE File. From the browser in out VM we are going to be downloading it, We are going to be using Socks5Systemz malware.



Load file ...r/Downloads/af900d9dda235bf8a4c20b724a3d79daf8b87cbe1acd4a93af348203a45cc33c.exe as

- Portable executable for 80386 (PE) [pe64.so]
- Binary file

Processor type (double-click to set)

Intel Pentium protected with MMX	80586p
Intel Pentium real with MMX	80586r
MetaPC (disassemble all opcodes)	metapc

Loading segment 0x0000000000000000

Loading offset 0x0000000000000000

Analysis

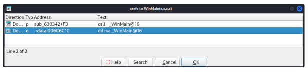
☑ Enabled

☑ Indicator enabled

Options

□ Loading options	☑ Create segments	□ Load resources
☑ Fill segment gaps	□ Create FLAT group	☑ Rename DLL entries
□ Load as code segment	☑ Create imports segment	□ Manual load

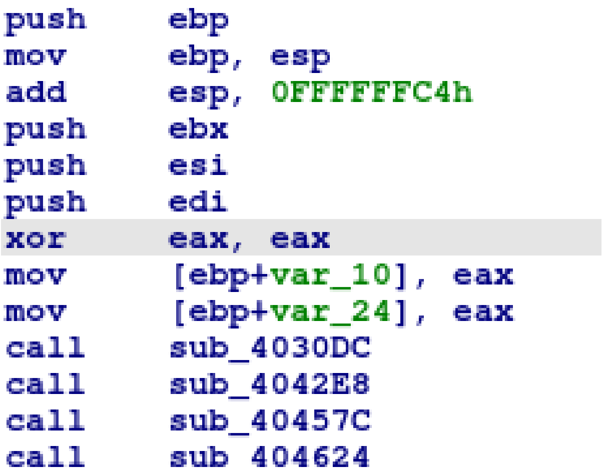
Here we are just going to click OK and continue. I have to locate an entry point. On the exports tab we can see the start:



Direction	Typ	Address	Text
Do...	p	sub_630342+F3	call _WinMain@16
Do...	o	.rdata:006C6C1C	dd rva _WinMain@16

Line 2 of 2

The entry point includes the address where everything starts.



```
push    ebp
mov     ebp, esp
add     esp, 0FFFFFFC4h
push    ebx
push    esi
push    edi
xor     eax, eax
mov     [ebp+var_10], eax
mov     [ebp+var_24], eax
call    sub_4030DC
call    sub_4042E8
call    sub_40457C
call    sub 404624
```

The first instruction EBP that stands for Extended Base Pointer, it will be pushed and it will set EBP to the current stack pointer and adjust it to create space for local variables and the it push EBX.

1.3 Exercise 2: Function Identification and Cross-References

1.3.1 Objective:

Learn to identify key functions in the disassembly and explore function relationships using cross-referencing and graph mode.

1.3.2 Steps Taken:

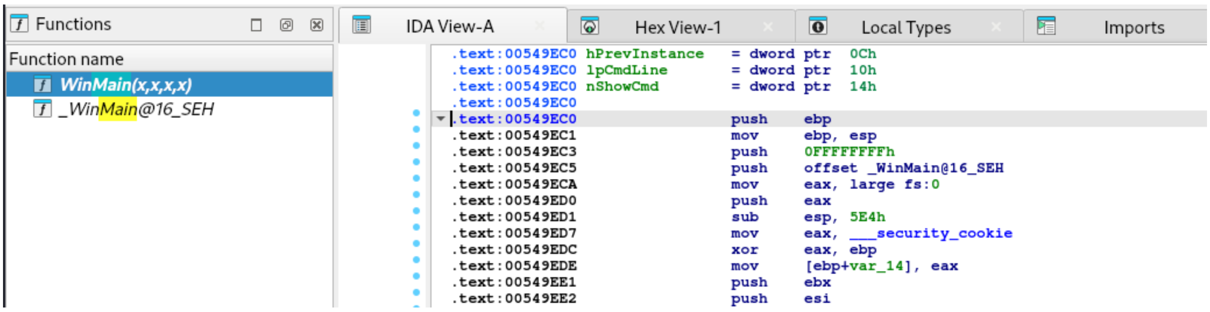
- Using another PE file from https://bazaar.abuse.ch/browse/, load it into IDA Free
- Navigate through the function names in the ”Functions” window (on the left sidebar).
- Identify the main or WinMain function (depending on the binary).

- Cross-reference (X) the main function to see how many times it is called and from which addresses.
- Switch to Graph Mode to visualize the control flow of the main function.
- Take screenshots or notes of the function calls and their relationships.

1.3.3 Screenshots

1.3.4 Analysis

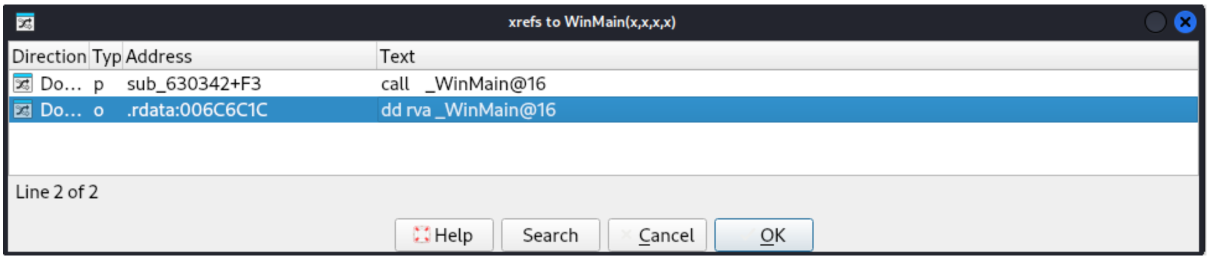
For this portion of the lab, we are going to be using the folder that was provided to us by the professor, and we are going to load the binary.



Function name
WinMain(x,x,x,x)
_WinMain@16_SEH

```
.text:00549EC0 hPrevInstance   = dword ptr  0Ch
.text:00549EC0 lpCmdLine       = dword ptr  10h
.text:00549EC0 nShowCmd        = dword ptr  14h
.text:00549EC0
.text:00549EC0                 push    ebp
.text:00549EC1                 mov     ebp, esp
.text:00549EC3                 push    0FFFFFFFFh
.text:00549EC5                 push    offset _WinMain@16_SEH
.text:00549ECA                 mov     eax, large fs:0
.text:00549ED0                 push    eax
.text:00549ED1                 sub     esp, 5E4h
.text:00549ED7                 mov     eax, ___security_cookie
.text:00549EDC                 xor     eax, ebp
.text:00549EDE                 mov     [ebp+var_14], eax
.text:00549EE1                 push    ebx
.text:00549EE2                 push    esi
```

With ctrl+ F we can find the winmain function. if we double click it we go to the instruction that is on the previous picture. If we right click and then look for the option of cross referencing we get the following:

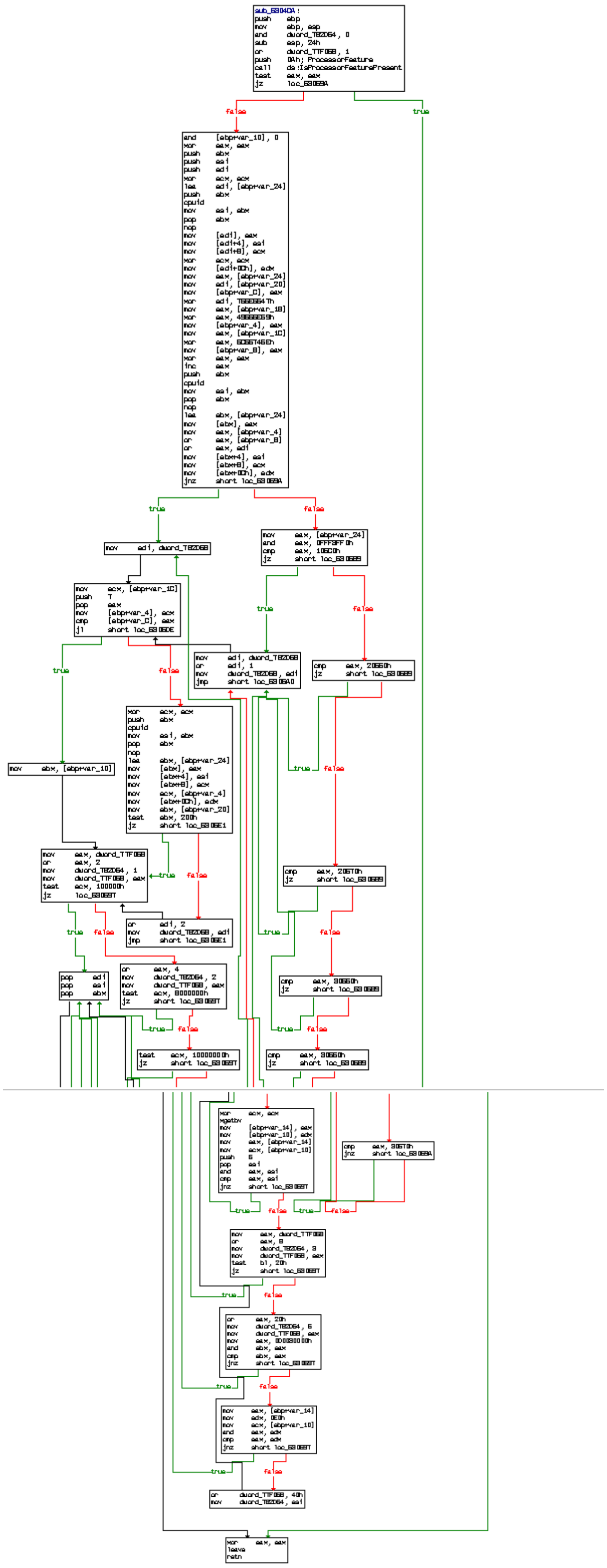


xrefs to WinMain(x,x,x,x)

Direction	Typ	Address	Text
Do...	p	sub_630342+F3	call _WinMain@16
Do...	o	.rdata:006C6C1C	dd rva _WinMain@16

Line 2 of 2

We can notice that it has been cross-referenced. Taking a look at the assembly instructions we can notice ut pushes ebp and then it pushes 0FFFFFFFFFFFFh. The security cookie is secured to protect against buffer overflow.



```
sub_6304DA :
push    ebp
mov     ebp, esp
and     dword_7B2064 , 0
sub     esp, 24h
or      dword_77F068 , 1
push    0Ah; ProcessorFeature
call    ds:IsProcessorFeaturePresent
test    eax, eax
jz      loc_63069A

and     [ebp+var_10], 0
xor     eax, eax
push    ebx
push    esi
push    edi
xor     ecx, ecx
lea     edi, [ebp+var_24]
push    ebx
cpuid
mov     esi, ebx
pop     ebx
nop
mov     [edi], eax
mov     [edi+4], esi
mov     [edi+8], ecx
xor     ecx, ecx
mov     [edi+0Ch], edx
mov     eax, [ebp+var_24]
mov     edi, [ebp+var_20]
mov     [ebp+var_C], eax
xor     edi, 756E6547h
mov     eax, [ebp+var_18]
xor     eax, 49656E69h
mov     [ebp+var_4], eax
mov     eax, [ebp+var_1C]
xor     eax, 6C65746Eh
mov     [ebp+var_8], eax
xor     eax, eax
inc     eax
push    ebx
cpuid
mov     esi, ebx
pop     ebx
nop
lea     ebx, [ebp+var_24]
mov     [ebx], eax
mov     eax, [ebp+var_4]
or      eax, [ebp+var_8]
or      eax, edi
mov     [ebx+4], esi
mov     [ebx+8], ecx
mov     [ebx+0Ch], edx
jnz     short loc_63069A

mov     eax, [ebp+var_24]
and     eax, 0FFF3FF0h
cmp     eax, 106C0h
jz      short loc_630689

mov     edi, dword_7B2068

mov     ecx, [ebp+var_1C]
push    7
pop     eax
mov     [ebp+var_4], ecx
cmp     [ebp+var_C], eax
jl      short loc_6306DE

mov     edi, dword_7B2068
or      edi, 1
mov     dword_7B2068 , edi
jmp     short loc_6306A0

cmp     eax, 20660h
jz      short loc_630689

mov     ebx, [ebp+var_10]

xor     ecx, ecx
push    ebx
cpuid
mov     esi, ebx
pop     ebx
nop
lea     ebx, [ebp+var_24]
mov     [ebx], eax
mov     [ebx+4], esi
mov     [ebx+8], ecx
mov     ecx, [ebp+var_4]
mov     [ebx+0Ch], edx
mov     ebx, [ebp+var_20]
test    ebx, 200h
jz      short loc_6306E1

cmp     eax, 20670h
jz      short loc_630689

mov     eax, dword_77F068
or      eax, 2
mov     dword_7B2064 , 1
mov     dword_77F068 , eax
test    ecx, 100000h
jz      loc_630697

or      edi, 2
mov     dword_7B2068 , edi
jmp     short loc_6306E1

cmp     eax, 30650h
jz      short loc_630689

pop     edi
pop     esi
pop     ebx

or      eax, 4
mov     dword_7B2064 , 2
mov     dword_77F068 , eax
test    ecx, 8000000h
jz      short loc_630697

test    ecx, 10000000h
jz      short loc_630697

cmp     eax, 30660h
jz      short loc_630689

xor     ecx, ecx
xgetbv
mov     [ebp+var_14], eax
mov     [ebp+var_10], edx
mov     eax, [ebp+var_14]
mov     ecx, [ebp+var_10]
push    6
pop     esi
and     eax, esi
cmp     eax, esi
jnz     short loc_630697

cmp     eax, 30670h
jnz     short loc_63069A

mov     eax, dword_77F068
or      eax, 8
mov     dword_7B2064 , 3
mov     dword_77F068 , eax
test    bl, 20h
jz      short loc_630697

or      eax, 20h
mov     dword_7B2064 , 5
mov     dword_77F068 , eax
mov     eax, 0D0030000h
and     ebx, eax
cmp     ebx, eax
jnz     short loc_630697

mov     eax, [ebp+var_14]
mov     edx, 0E0h
mov     ecx, [ebp+var_10]
and     eax, edx
cmp     eax, edx
jnz     short loc_630697

or      dword_77F068 , 40h
mov     dword_7B2064 , esi

xor     eax, eax
leave
retn
```

Graph mode

1.4 Exercise 3: Analyzing Conditional Branches

1.4.1 Objective:

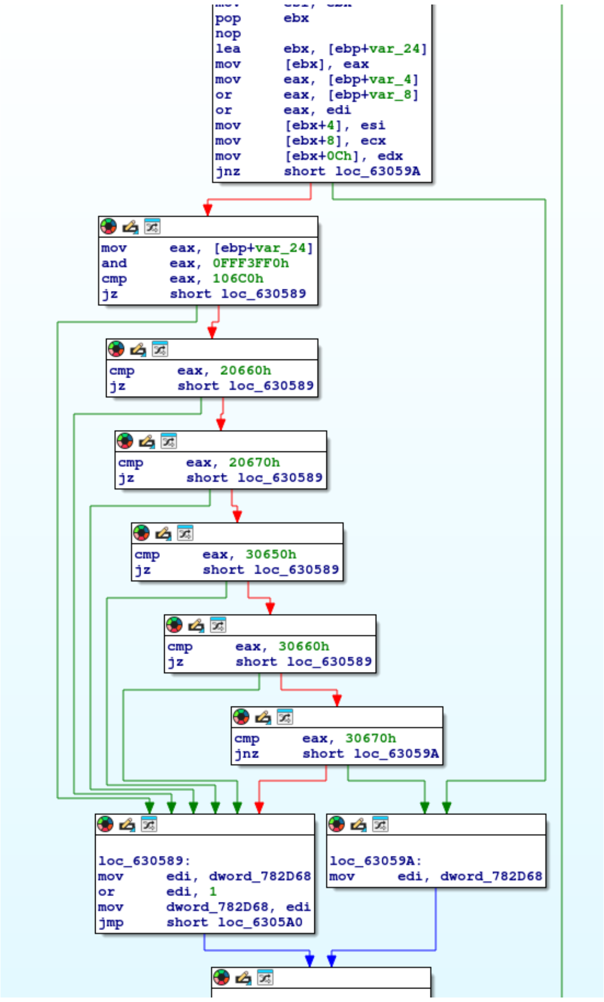
Understand conditional branding in disassembly and how control flows based on condition in the code.

1.4.2 Steps Taken:

- Download a PE file with multiple functions and conditional branches from Link.
- Load the PE file into IDA Free and navigate to a function with a conditional branch (e.g., if-else or switch-case).
- Use the **Graph Mode** (press "Space") to visualize the control flow and observe the branches.
- Identify the instructions that check conditions (e.g., CMP, TEST) and note where the control jumps (JMP, JE, JNE).
- Document how different branches lead to different parts of the code.

1.4.3 Analysis

We are going to be using the same executable from last exercise:



```
pop     ebx
nop
lea     ebx, [ebp+var_24]
mov     [ebx], eax
mov     eax, [ebp+var_4]
or      eax, [ebp+var_8]
or      eax, edi
mov     [ebx+4], esi
mov     [ebx+8], ecx
mov     [ebx+0Ch], edx
jnz     short loc_63059A
```

```
mov     eax, [ebp+var_24]
and     eax, 0FFF3FF0h
cmp     eax, 106C0h
jz      short loc_630589
```

```
cmp     eax, 20660h
jz      short loc_630589
```

```
cmp     eax, 20670h
jz      short loc_630589
```

```
cmp     eax, 30650h
jz      short loc_630589
```

```
cmp     eax, 30660h
jz      short loc_630589
```

```
cmp     eax, 30670h
jnz     short loc_63059A
```

```
loc_630589:
mov     edi, dword_782D68
or      edi, 1
mov     dword_782D68, edi
jmp     short loc_6305A0
```

```
loc_63059A:
mov     edi, dword_782D68
```

Well in the image provided above, we notice that the conditional starts in *jnz* which is jump if not zero. if its not zero, it will *cmp*, compare what we have on *eax* with *106C0h* and then it jumps if zero, then it will have a lot of comparisons. Is like a lot of else if just checking for 6 different comparisons. In the case that it runs into a zero it will jump into short loc_630589.

1.5 Exercise 4: String Analysis in a Binary

1.5.1 Objective:

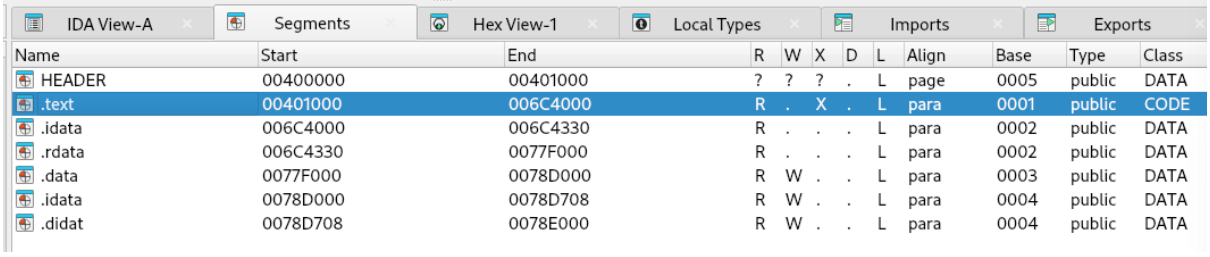
Learn to analyze and trace strings in a binary, understanding how they are referenced in the program's code.

1.5.2 Steps Taken:

- Download a PE file containing multiple strings from Link
- Navigate to the **.rdata** or **.data** section where strings are usually stored.
- Locate and identify the strings in the disassembly.
- Cross-reference where each string is used in the code by pressing the "X" key.
- Document the function(s) that use these strings and explain their role in the program

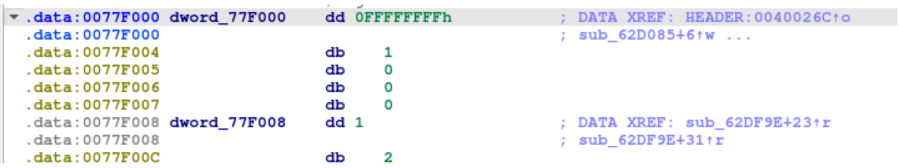
1.5.3 Analysis

We are going to continue working on the same file from last exercise. from the instructions we know that the strings are in *.rdata* or *.data* section, to view this is on **View, then Open Subviews, then Segments**, here we can see the following:



Name	Start	End	R	W	X	D	L	Align	Base	Type	Class
HEADER	00400000	00401000	?	?	?	.	L	page	0005	public	DATA
.text	00401000	006C4000	R	.	X	.	L	para	0001	public	CODE
.idata	006C4000	006C4330	R	.	.	.	L	para	0002	public	DATA
.rdata	006C4330	0077F000	R	.	.	.	L	para	0002	public	DATA
.data	0077F000	0078D000	R	W	.	.	L	para	0003	public	DATA
.idata	0078D000	0078D708	R	W	.	.	L	para	0004	public	DATA
.didat	0078D708	0078E000	R	W	.	.	L	para	0004	public	DATA

If we click on the *.data* we get the following:



```
.data:0077F000 dword_77F000    dd 0FFFFFFFFh          ; DATA XREF: HEADER:0040026C↑o
.data:0077F000                                        ; sub_62D085+6↑w ...
.data:0077F004                 db    1
.data:0077F005                 db    0
.data:0077F006                 db    0
.data:0077F007                 db    0
.data:0077F008 dword_77F008    dd 1                   ; DATA XREF: sub_62DF9E+23↑r
.data:0077F008                                        ; sub_62DF9E+31↑r
.data:0077F00C                 db    2
```

1.6 Exercise 5: Function Hooking in the Disassembly

1.6.1 Objective:

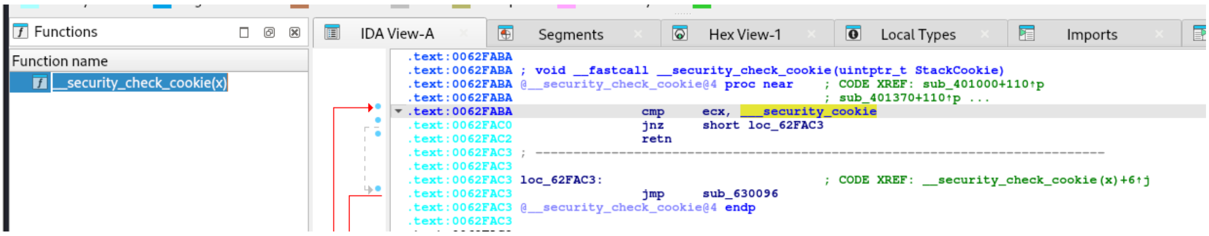
Gain an understanding of function hooking and how control flow can be altered in a binary.

1.6.2 Steps Taken:

- Download a moderately complex PE file from Link.
- Identify a critical function (e.g., network communication, file I/O, or user authentication).
- Use the "Functions" window and cross-referencing (X) to mark all instances where this function is called.
- Hypothesize how you might modify the function's behavior (e.g., injecting a hook that alters the input or output).
- Document the changes in control flow necessary to redirect the function's execution.
- Discuss how such a hook could be used maliciously (e.g., for man-in-the-middle attacks or altering data transmission).

1.6.3 Analysis

We are going to continue using the same program to answer this question. The first thing we are asked is to identify critical functions that are in the program, such as network communication, file I/o or user authentication. One of them should be **_security_check_cookie**.



```
.text:0062FABA
.text:0062FABA ; void __fastcall __security_check_cookie(uintptr_t StackCookie)
.text:0062FABA @__security_check_cookie@4 proc near    ; CODE XREF: sub_401000+110↑p
.text:0062FABA                                         ; sub_401370+110↑p ...
.text:0062FABA                 cmp     ecx, ___security_cookie
.text:0062FAC0                 jnz     short loc_62FAC3
.text:0062FAC2                 retn
.text:0062FAC3 ; ---------------------------------------------------------------------------
.text:0062FAC3
.text:0062FAC3 loc_62FAC3:                             ; CODE XREF: __security_check_cookie(x)+6↑j
.text:0062FAC3                 jmp     sub_630096
.text:0062FAC3 @__security_check_cookie@4 endp
.text:0062FAC3
```

Now it is important to understand how ti works, this is used to protect against stack-based buffer overflows. If we want to modify, there are a couple things we can do, one of them is to change the value of ecx before it does the comparisons just like.

```
mov ecx, [esp]
add ecx, 0x10
mov [esp], ecx
```

This will help us bypass the security checks it has.

1.7 Exercise 6: Identifying Packed Code

1.7.1 Objective:

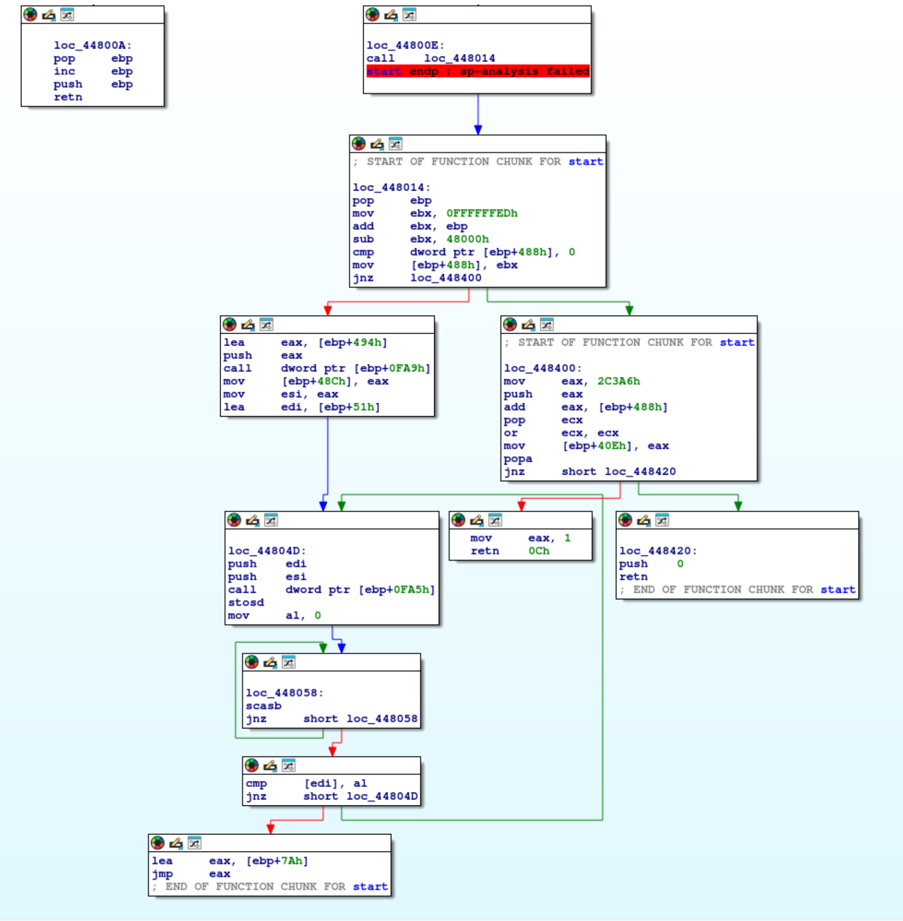
Learn to detect and analyze packed or obfuscated code, and practice unpacking binaries for further analysis.

1.7.2 Steps Taken:

- Obtain a packed PE file from Link.
- Load the packed file into IDA Free and review the disassembly for irregularities (e.g., long loops, uncommon instructions, or opaque jumps).
- Try to unpack the binary using tools such as UPX (https://upx.github.io/) or IDA's own unpacking capabilities (if available).
- Compare the unpacked code with the original packed code, noting the differences and transformations.
- Document how packed code is typically structured to evade detection or analysis and how reverse engineers can overcome these obstacles

1.7.3 Analysis

We are going to analyze a package that wa obtained from the repo listed above. the name is aspack_PAGEANT_32.exe. We loaded it into IDA and we got the this function tree.



```
loc_44800A:
pop     ebp
inc     ebp
push    ebp
retn
```

```
loc_44800E:
call    loc_448014
start endp ; sp-analysis failed
```

```
; START OF FUNCTION CHUNK FOR start

loc_448014:
pop     ebp
mov     ebx, 0FFFFFFEDh
add     ebx, ebp
sub     ebx, 48000h
cmp     dword ptr [ebp+488h], 0
mov     [ebp+488h], ebx
jnz     loc_448400
```

```
lea     eax, [ebp+494h]
push    eax
call    dword ptr [ebp+0FA9h]
mov     [ebp+48Ch], eax
mov     esi, eax
lea     edi, [ebp+51h]
```

```
; START OF FUNCTION CHUNK FOR start

loc_448400:
mov     eax, 2C3A6h
push    eax
add     eax, [ebp+488h]
pop     ecx
or      ecx, ecx
mov     [ebp+40Eh], eax
popa
jnz     short loc_448420
```

```
loc_44804D:
push    edi
push    esi
call    dword ptr [ebp+0FA5h]
stosd
mov     al, 0
```

```
mov     eax, 1
retn    0Ch
```

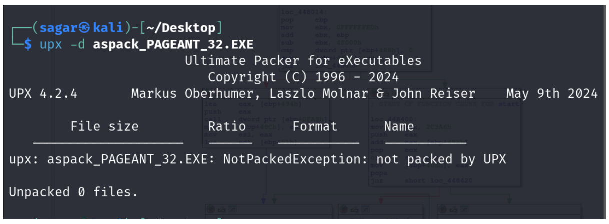
```
loc_448420:
push    0
retn
; END OF FUNCTION CHUNK FOR start
```

```
loc_448058:
scasb
jnz     short loc_448058
```

```
cmp     [edi], al
jnz     short loc_44804D
```

```
lea     eax, [ebp+7Ah]
jmp     eax
; END OF FUNCTION CHUNK FOR start
```

I tried using UPX to unpack it but i got this output:



```
┌──(sagar㉿kali)-[~/Desktop]
└─$ upx -d aspack_PAGEANT_32.EXE
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.4       Markus Oberhumer, Laszlo Molnar & John Reiser    May 9th 2024

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: aspack_PAGEANT_32.EXE: NotPackedException: not packed by UPX

Unpacked 0 files.
```

This means it was not packed by UPX. Usually it has some signs suck as signs of packed code, such as long loops, uncommon instructions, and opaque jumps, to make it more difficult to reverse engineers to reverse the program/service.

Chapter 2

Binary Ninja

2.1 Introduction

Binary Ninja is an interactive decompiler, disassembler, debugger, and binary analysis platform built by reverse engineers. It can disassemble a binary and display the disassembly in linear or graph views.

2.2 Exercise 1: Basic Disassembly and Navigation

2.2.1 Objective:

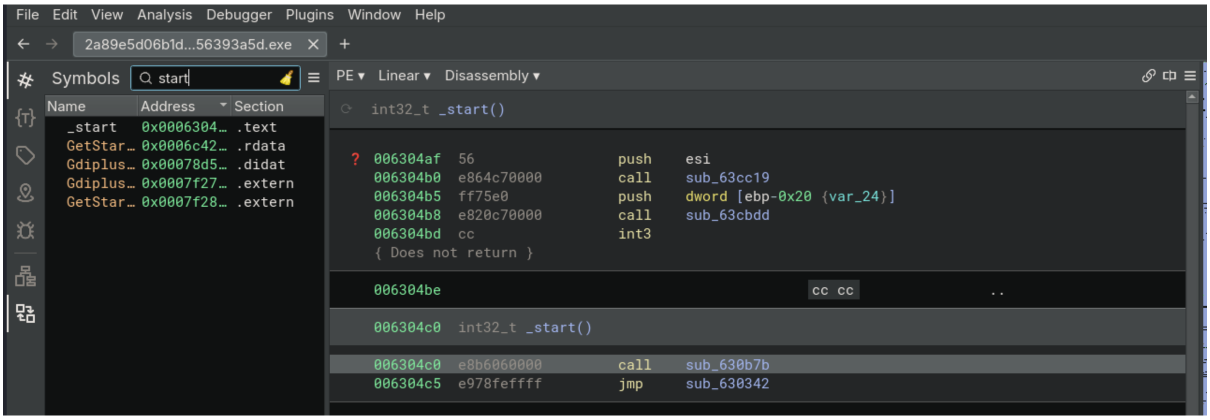
Familiarize yourself with Binary Ninja's interface and basic disassembly features.

2.2.2 Steps Taken:

- Download a PE file from https://bazaar.abuse.ch/browse/.
- Load the PE file into Binary Ninja.
- In the disassembly view, locate the entry point and identify the first five instructions.
- Use the side panel to jump between functions and symbols.
- Document the key functions and initial instructions

2.2.3 Analysis

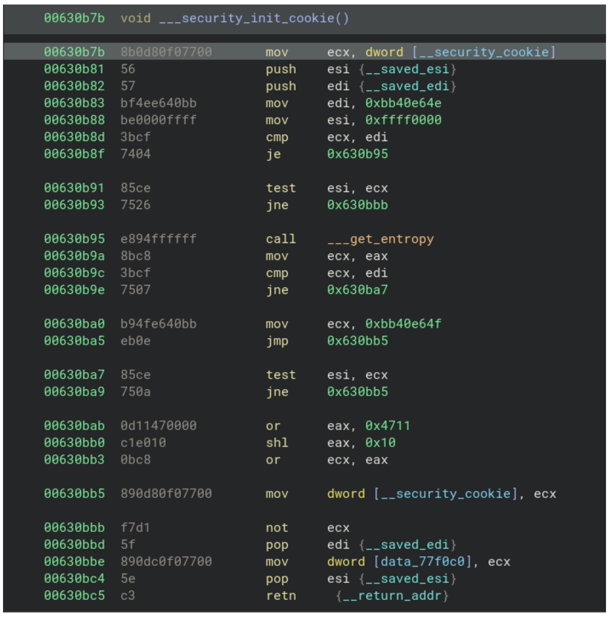
For this exercise we are going to be using the PE executable that we used for IDA. we are going to open it and search for an entry point, thankfully we have a search bar and we only have to search for start:


File Edit View Analysis Debugger Plugins Window Help
2a89e5d06b1d...56393a5d.exe
Symbols
start
PE ▾ Linear ▾ Disassembly ▾

Name	Address	Section
_start	0x0006304…	.text
GetStar…	0x0006c42…	.rdata
Gdiplus…	0x00078d5…	.didat
Gdiplus…	0x0007f27…	.extern
GetStar…	0x0007f28…	.extern

```
int32_t _start()

? 006304af  56                push    esi
  006304b0  e864c70000        call    sub_63cc19
  006304b5  ff75e0            push    dword [ebp-0x20 {var_24}]
  006304b8  e820c70000        call    sub_63cbdd
  006304bd  cc                int3
  { Does not return }

  006304be                                     cc cc          ..

  006304c0  int32_t _start()

  006304c0  e8b6060000        call    sub_630b7b
  006304c5  e978feffff        jmp     sub_630342
```

Here we ca notice that this start function will call __security_init_cookie and then execute a jump into sub_630342. If we double click the call we see the following:



```
00630b7b  void ___security_init_cookie()

00630b7b  8b0d80f07700        mov     ecx, dword [__security_cookie]
00630b81  56                  push    esi {__saved_esi}
00630b82  57                  push    edi {__saved_edi}
00630b83  bf4ee640bb          mov     edi, 0xbb40e64e
00630b88  be0000ffff          mov     esi, 0xffff0000
00630b8d  3bcf                cmp     ecx, edi
00630b8f  7404                je      0x630b95

00630b91  85ce                test    esi, ecx
00630b93  7526                jne     0x630bbb

00630b95  e894ffffff          call    ___get_entropy
00630b9a  8bc8                mov     ecx, eax
00630b9c  3bcf                cmp     ecx, edi
00630b9e  7507                jne     0x630ba7

00630ba0  b94fe640bb          mov     ecx, 0xbb40e64f
00630ba5  eb0e                jmp     0x630bb5

00630ba7  85ce                test    esi, ecx
00630ba9  750a                jne     0x630bb5

00630bab  0d11470000          or      eax, 0x4711
00630bb0  c1e010              shl     eax, 0x10
00630bb3  0bc8                or      ecx, eax

00630bb5  890d80f07700        mov     dword [__security_cookie], ecx

00630bbb  f7d1                not     ecx
00630bbd  5f                  pop     edi {__saved_edi}
00630bbe  890dc0f07700        mov     dword [data_77f0c0], ecx
00630bc4  5e                  pop     esi {__saved_esi}
00630bc5  c3                  retn     {__return_addr}
```

Which as previously mentioned that call will set up security measures by loading it into the ecx and sets some constant values.

2.3 Exercise 2: SCross-Referencing Functions

2.3.1 Objective:

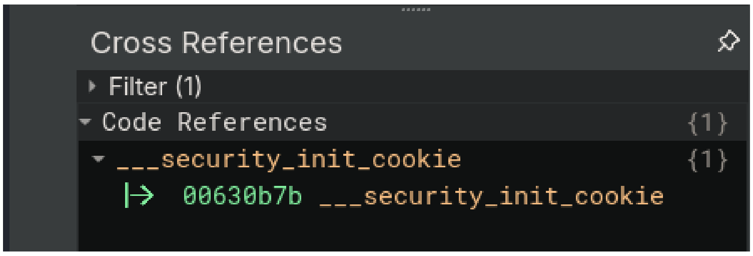
Learn how to cross-reference functions and trace calls in Binary Ninja.

2.3.2 Steps Taken:

- Using another PE file from https://bazaar.abuse.ch/browse/, load it into Binary Ninja.
- Identify a function (such as main) and use cross-referencing to see where it is called.
- Use the cross-references window to trace function calls.
- Record the flow of function calls from the main function

2.3.3 Analysis

In this exercise we want to look for the cross-referencing, like the ones in ida, we can look at them in binary ninja. we are going to look at the start function, luckily instead of doing right click and then following them, we can just click the function and we are going to have a section for cross references.



Cross References

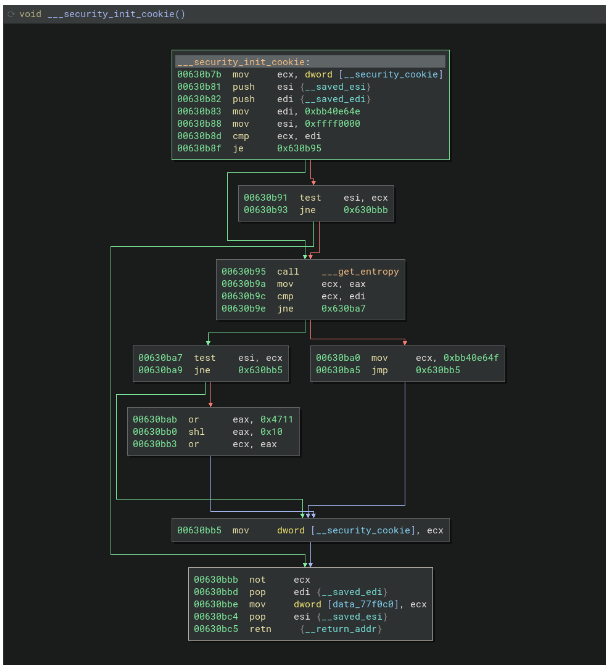
Filter (1)

Code References {1}

___security_init_cookie {1}

00630b7b ___security_init_cookie

The only function call it does is to the security init cookie(), this is the flow graph:



```
void ___security_init_cookie()

___security_init_cookie:
00630b7b  mov     ecx, dword [__security_cookie]
00630b81  push    esi {__saved_esi}
00630b82  push    edi {__saved_edi}
00630b83  mov     edi, 0xbb40e64e
00630b88  mov     esi, 0xffff0000
00630b8d  cmp     ecx, edi
00630b8f  je      0x630b95

00630b91  test    esi, ecx
00630b93  jne     0x630bbb

00630b95  call    ___get_entropy
00630b9a  mov     ecx, eax
00630b9c  cmp     ecx, edi
00630b9e  jne     0x630ba7

00630ba7  test    esi, ecx
00630ba9  jne     0x630bb5

00630ba0  mov     ecx, 0xbb40e64f
00630ba5  jmp     0x630bb5

00630bab  or      eax, 0x4711
00630bb0  shl     eax, 0x10
00630bb3  or      ecx, eax

00630bb5  mov     dword [__security_cookie], ecx

00630bbb  not     ecx
00630bbd  pop     edi {__saved_edi}
00630bbe  mov     dword [data_77f0c0], ecx
00630bc4  pop     esi {__saved_esi}
00630bc5  retn     {__return_addr}
```

In the end it will return to the start so that it can jump into sub_630342

2.4 Exercise 3: Conditional Branching and Control Flow

2.4.1 Objective:

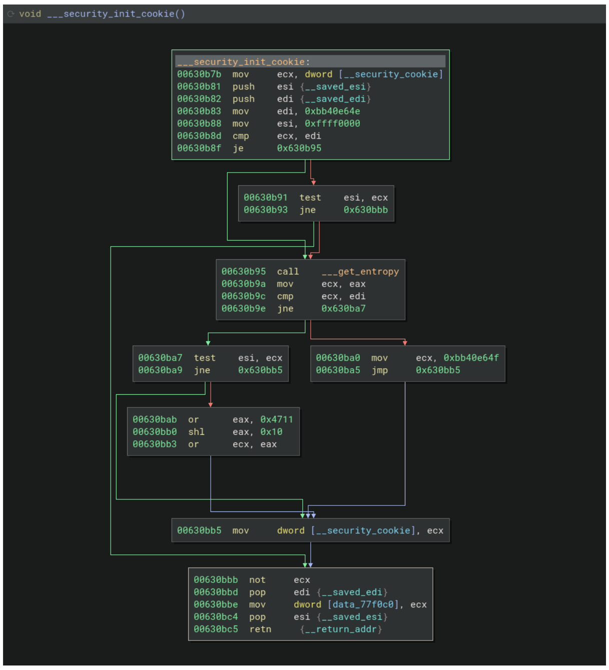
Understand how control flow changes in disassembly when conditional branches are present.

2.4.2 Steps Taken:

- Load a PE file with conditional branches from https://bazaar.abuse.ch/browse/.
- In Binary Ninja, navigate to a function that contains conditional branches (e.g., if-else or switch-case).
- Use the Graph View to visualize how the control flow changes based on conditions.
- Identify the conditions being checked (e.g., CMP, TEST) and document the branches.

2.4.3 Analysis

We are going to be using the same PE file from the last exercise:

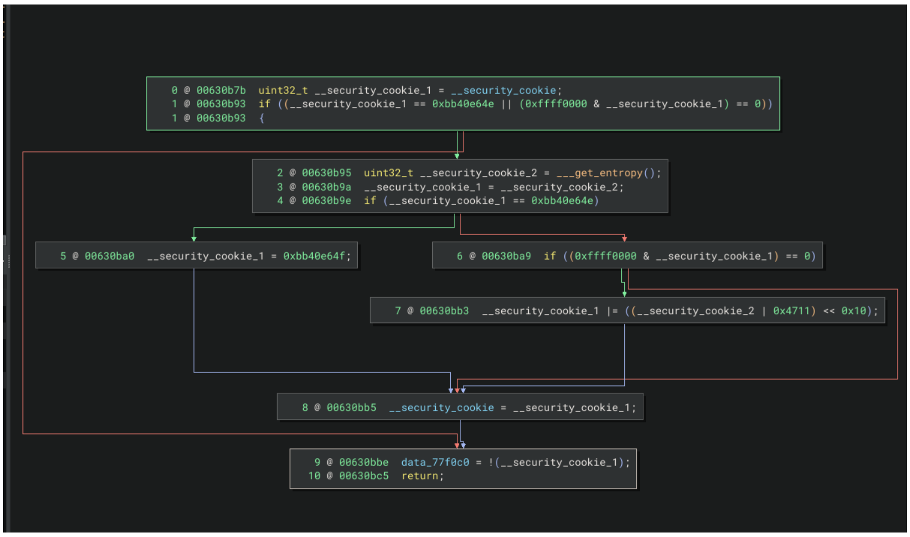


```
void ___security_init_cookie()

___security_init_cookie:
00630b7b  mov     ecx, dword [__security_cookie]
00630b81  push    esi {__saved_esi}
00630b82  push    edi {__saved_edi}
00630b83  mov     edi, 0xbb40e64e
00630b88  mov     esi, 0xffff0000
00630b8d  cmp     ecx, edi
00630b8f  je      0x630b95

00630b91  test    esi, ecx
00630b93  jne     0x630bbb

00630b95  call    ___get_entropy
00630b9a  mov     ecx, eax
00630b9c  cmp     ecx, edi
00630b9e  jne     0x630ba7

00630ba7  test    esi, ecx
00630ba9  jne     0x630bb5

00630ba0  mov     ecx, 0xbb40e64f
00630ba5  jmp     0x630bb5

00630bab  or      eax, 0x4711
00630bb0  shl     eax, 0x10
00630bb3  or      ecx, eax

00630bb5  mov     dword [__security_cookie], ecx

00630bbb  not     ecx
00630bbd  pop     edi {__saved_edi}
00630bbe  mov     dword [data_77f0c0], ecx
00630bc4  pop     esi {__saved_esi}
00630bc5  retn     {__return_addr}
```

This is the tree graph from the security cookie. In order to get knowledge in conditional statements, lets start:

In the picture from above, we can notice that it will compare edi with ecx, this is becasue the assembly instructions are read backawards, it will jump into get entrpy if zero, if not it will perform a test of ecx with esi which it will perform a bitwise AND but it will not modify the bits. then it will jump if not equal almost to the end, if equal it will call for the entropy. after this it will compare again edi with ecx and then jump if not equal can decide to mov 0xbb40e64f to ecx and tehn jump into mov exc to dword. the other possibility is to perform a test and it not equal jump into dword if equal perform and or and jump into dword.

This sounds really complicate it but lets see the C implementation.



```
 0 @ 00630b7b  uint32_t __security_cookie_1 = __security_cookie;
 1 @ 00630b93  if ((__security_cookie_1 == 0xbb40e64e || (0xffff0000 & __security_cookie_1) == 0))
 1 @ 00630b93  {

 2 @ 00630b95  uint32_t __security_cookie_2 = ___get_entropy();
 3 @ 00630b9a  __security_cookie_1 = __security_cookie_2;
 4 @ 00630b9e  if (__security_cookie_1 == 0xbb40e64e)

 5 @ 00630ba0  __security_cookie_1 = 0xbb40e64f;

 6 @ 00630ba9  if ((0xffff0000 & __security_cookie_1) == 0)

 7 @ 00630bb3  __security_cookie_1 |= ((__security_cookie_2 | 0x4711) << 0x10);

 8 @ 00630bb5  __security_cookie = __security_cookie_1;

 9 @ 00630bbe  data_77f0c0 = !(__security_cookie_1);
10 @ 00630bc5  return;
```

2.5 Exercise 4: String Analysis

2.5.1 Objective:

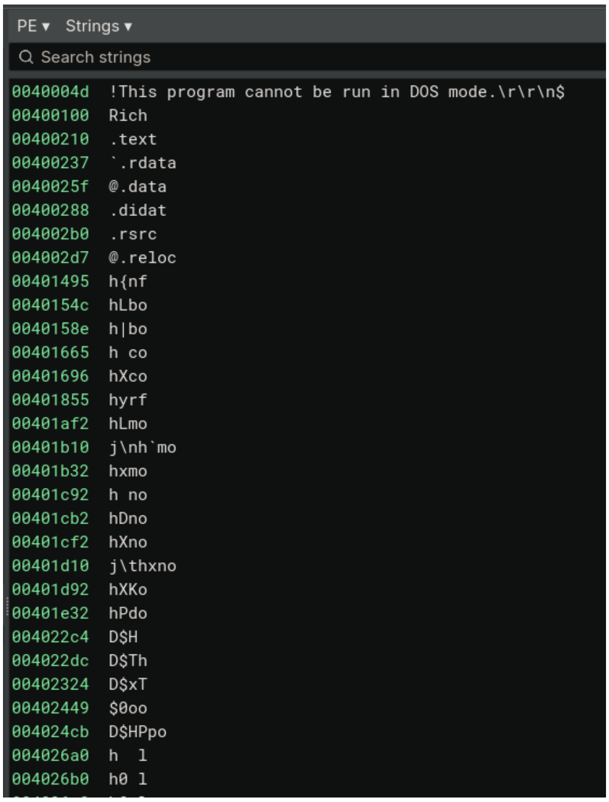
Learn to locate and analyze strings in a binary and understand how they relate to the program's functionality.D

2.5.2 Steps Taken:

- Load a PE file containing multiple strings (you can find samples at link).
- Use Binary Ninjas Strings window to locate all strings in the binary.
- Cross-reference the locations where the strings are used in the disassembly.
- Document the functions that use these strings and explain their purpose in the program.

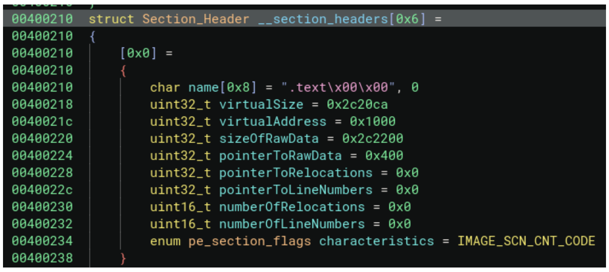
2.5.3 Analysis

In otder to look for strings we only have to change from graph to strings in the binary ninja interface.



```
0040004d  !This program cannot be run in DOS mode.\r\r\n$
00400100  Rich
00400210  .text
00400237  `.rdata
0040025f  @.data
00400288  .didat
004002b0  .rsrc
004002d7  @.reloc
00401495  h{nf
0040154c  hLbo
0040158e  h|bo
00401665  h co
00401696  hXco
00401855  hyrf
00401af2  hLmo
00401b10  j\nh`mo
00401b32  hxmo
00401c92  h no
00401cb2  hDno
00401cf2  hXno
00401d10  j\thxno
00401d92  hXKo
00401e32  hPdo
004022c4  D$H
004022dc  D$Th
00402324  D$xT
00402449  $0oo
004024cb  D$HPpo
004026a0  h  l
004026b0  h0 l
```

This is what we can see. lets say we want to see .data or .rdata just like in IDA, we only have to double click them. For example .rdata will take us here:



```
00400210  struct Section_Header __section_headers[0x6] =
00400210  {
00400210      [0x0] =
00400210      {
00400210          char name[0x8] = ".text\x00\x00", 0
00400218          uint32_t virtualSize = 0x2c20ca
0040021c          uint32_t virtualAddress = 0x1000
00400220          uint32_t sizeOfRawData = 0x2c2200
00400224          uint32_t pointerToRawData = 0x400
00400228          uint32_t pointerToRelocations = 0x0
0040022c          uint32_t pointerToLineNumbers = 0x0
00400230          uint16_t numberOfRelocations = 0x0
00400232          uint16_t numberOfLineNumbers = 0x0
00400234          enum pe_section_flags characteristics = IMAGE_SCN_CNT_CODE
00400238      }
```

2.6 Exercise 5: Function Hooking and Modifications

2.6.1 Objective:

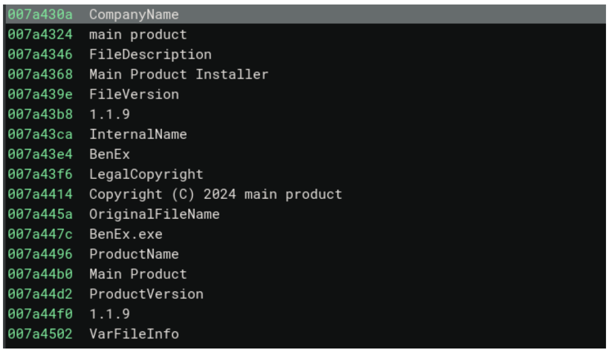
Understand how function hooking works and how to modify program behavior using Binary Ninja

2.6.2 Steps Taken:

- Download a PE file from https://bazaar.abuse.ch/browse/.
- Identify a key function (e.g., one responsible for network communication or file access).
- Hypothesize how you might modify or hook this function in a way that alters its behavior.
- Document possible points where hooks could be inserted and the expected impact on program flow

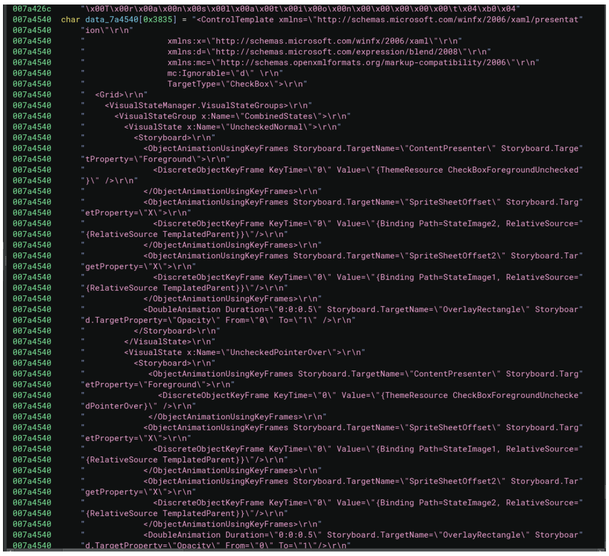
2.6.3 Analysis

Lets say i want to modify this function so that we alter the behaviour, we can go into the strings to find valueable information. for example :



```
007a430a  CompanyName
007a4324  main product
007a4346  FileDescription
007a4368  Main Product Installer
007a439e  FileVersion
007a43b8  1.1.9
007a43ca  InternalName
007a43e4  BenEx
007a43f6  LegalCopyright
007a4414  Copyright (C) 2024 main product
007a445a  OriginalFileName
007a447c  BenEx.exe
007a4496  ProductName
007a44b0  Main Product
007a44d2  ProductVersion
007a44f0  1.1.9
007a4502  VarFileInfo
```

Here we can get a lot of information:



```
007a426c      "\x00T\x00r\x00a\x00n\x00s\x00l\x00a\x00t\x00i\x00o\x00n\x00\x00\x00\x00\x00\t\x04\xb0\x04"
007a4540  char data_7a4540[0x3835] = "<ControlTemplate xmlns=\"http://schemas.microsoft.com/winfx/2006/xaml/presentat"
007a4540      "ion\"\r\n"
007a4540      "                  xmlns:x=\"http://schemas.microsoft.com/winfx/2006/xaml\"\r\n"
007a4540      "                  xmlns:d=\"http://schemas.microsoft.com/expression/blend/2008\"\r\n"
007a4540      "                  xmlns:mc=\"http://schemas.openxmlformats.org/markup-compatibility/2006\"\r\n"
007a4540      "                  mc:Ignorable=\"d\" \r\n"
007a4540      "                  TargetType=\"CheckBox\">\r\n"
007a4540      "  <Grid>\r\n"
007a4540      "    <VisualStateManager.VisualStateGroups>\r\n"
007a4540      "      <VisualStateGroup x:Name=\"CombinedStates\">\r\n"
007a4540      "        <VisualState x:Name=\"UncheckedNormal\">\r\n"
007a4540      "          <Storyboard>\r\n"
007a4540      "            <ObjectAnimationUsingKeyFrames Storyboard.TargetName=\"ContentPresenter\" Storyboard.Targe"
007a4540      "tProperty=\"Foreground\">\r\n"
007a4540      "              <DiscreteObjectKeyFrame KeyTime=\"0\" Value=\"{ThemeResource CheckBoxForegroundUnchecked"
007a4540      "}\" />\r\n"
007a4540      "            </ObjectAnimationUsingKeyFrames>\r\n"
007a4540      "            <ObjectAnimationUsingKeyFrames Storyboard.TargetName=\"SpriteSheetOffset\" Storyboard.Targ"
007a4540      "etProperty=\"X\">\r\n"
007a4540      "              <DiscreteObjectKeyFrame KeyTime=\"0\" Value=\"{Binding Path=StateImage2, RelativeSource="
007a4540      "{RelativeSource TemplatedParent}}\"/>\r\n"
007a4540      "            </ObjectAnimationUsingKeyFrames>\r\n"
007a4540      "            <ObjectAnimationUsingKeyFrames Storyboard.TargetName=\"SpriteSheetOffset2\" Storyboard.Tar"
007a4540      "getProperty=\"X\">\r\n"
007a4540      "              <DiscreteObjectKeyFrame KeyTime=\"0\" Value=\"{Binding Path=StateImage1, RelativeSource="
007a4540      "{RelativeSource TemplatedParent}}\"/>\r\n"
007a4540      "            </ObjectAnimationUsingKeyFrames>\r\n"
007a4540      "            <DoubleAnimation Duration=\"0:0:0.5\" Storyboard.TargetName=\"OverlayRectangle\" Storyboar"
007a4540      "d.TargetProperty=\"Opacity\" From=\"0\" To=\"1\" />\r\n"
007a4540      "          </Storyboard>\r\n"
007a4540      "        </VisualState>\r\n"
007a4540      "        <VisualState x:Name=\"UncheckedPointerOver\">\r\n"
007a4540      "          <Storyboard>\r\n"
007a4540      "             <ObjectAnimationUsingKeyFrames Storyboard.TargetName=\"ContentPresenter\" Storyboard.Targ"
007a4540      "etProperty=\"Foreground\">\r\n"
007a4540      "               <DiscreteObjectKeyFrame KeyTime=\"0\" Value=\"{ThemeResource CheckBoxForegroundUnchecke"
007a4540      "dPointerOver}\" />\r\n"
007a4540      "             </ObjectAnimationUsingKeyFrames>\r\n"
007a4540      "            <ObjectAnimationUsingKeyFrames Storyboard.TargetName=\"SpriteSheetOffset\" Storyboard.Targ"
007a4540      "etProperty=\"X\">\r\n"
007a4540      "              <DiscreteObjectKeyFrame KeyTime=\"0\" Value=\"{Binding Path=StateImage1, RelativeSource="
007a4540      "{RelativeSource TemplatedParent}}\"/>\r\n"
007a4540      "            </ObjectAnimationUsingKeyFrames>\r\n"
007a4540      "            <ObjectAnimationUsingKeyFrames Storyboard.TargetName=\"SpriteSheetOffset2\" Storyboard.Tar"
007a4540      "getProperty=\"X\">\r\n"
007a4540      "              <DiscreteObjectKeyFrame KeyTime=\"0\" Value=\"{Binding Path=StateImage2, RelativeSource="
007a4540      "{RelativeSource TemplatedParent}}\"/>\r\n"
007a4540      "            </ObjectAnimationUsingKeyFrames>\r\n"
007a4540      "            <DoubleAnimation Duration=\"0:0:0.5\" Storyboard.TargetName=\"OverlayRectangle\" Storyboar"
007a4540      "d.TargetProperty=\"Opacity\" From=\"0\" To=\"1\"/>\r\n"
```

The information denoted here contains valuable information that its kinds of related to the keyvault that assure has. there is an xml for a checkbox control template and more things that indicates for who is designed. this is like the magic of the application where is the functionality.

2.7 Exercise 6: Analyzing Packed Code

2.7.1 Objective:

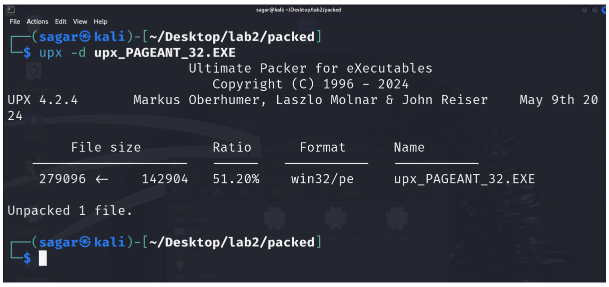
Learn to detect and analyze packed binaries and understand the unpacking process.

2.7.2 Steps Taken:

- Obtain a packed PE file (you can pack your own using UPX or download a sample from LINK).
- Load the packed file into Binary Ninja and analyze the disassembly.
- Look for signs of packing (e.g., repetitive loops, encrypted strings, suspicious jumps).
- Attempt to unpack the file and compare the disassembled code before and after unpacking.

2.7.3 Analysis

First we have to package our own PE file because the one from the repo is down.



```
┌──(sagar㉿kali)-[~/Desktop/lab2/packed]
└─$ upx -d upx_PAGEANT_32.EXE
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.4       Markus Oberhumer, Laszlo Molnar & John Reiser    May 9th 20
24

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    279096 <-    142904   51.20%    win32/pe     upx_PAGEANT_32.EXE

Unpacked 1 file.

┌──(sagar㉿kali)-[~/Desktop/lab2/packed]
└─$ 
```

forgot to add the pictures, but the code is pretty similar.

Chapter 3

Bandit

In order to work with bandit, it was necessary to read the documentation page. Here is the link Documentation Link

3.1 Exercise 1: Running Bandit on a simple python Script

Side note: I used the following repo for the purpose of this question `https://github.com/abhishek305/Calculator-in-python3-tkinter`

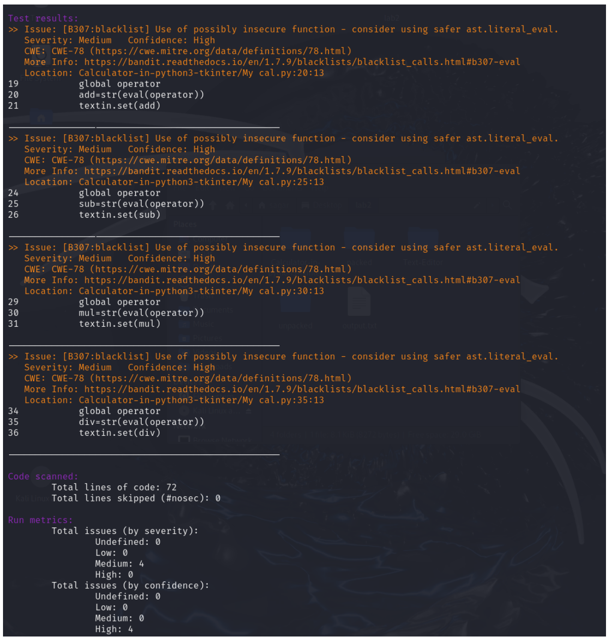
3.1.1 Objective:

Learn how Bandit detects basic vulnerabilities.

3.1.2 Steps Taken:

1. Download a Script of python
2. Run bandit on the script using: **bandit -r**
3. Identify security issues Flagged by bandit and document their severity and confidence levels.

3.1.3 Screenshots



```
Test results:
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer ast.literal_eval.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b307-eval
   Location: Calculator-in-python3-tkinter/My cal.py:20:13
19          global operator
20          add=str(eval(operator))
21          textin.set(add)

--------------------------------------------------
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer ast.literal_eval.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b307-eval
   Location: Calculator-in-python3-tkinter/My cal.py:25:13
24          global operator
25          sub=str(eval(operator))
26          textin.set(sub)

--------------------------------------------------
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer ast.literal_eval.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b307-eval
   Location: Calculator-in-python3-tkinter/My cal.py:30:13
29          global operator
30          mul=str(eval(operator))
31          textin.set(mul)

--------------------------------------------------
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer ast.literal_eval.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b307-eval
   Location: Calculator-in-python3-tkinter/My cal.py:35:13
34          global operator
35          div=str(eval(operator))
36          textin.set(div)

--------------------------------------------------

Code scanned:
        Total lines of code: 72
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 4
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 4
```

3.1.4 Analysis & Conclusion

From this code we notice there are 4 vulnerabilities that are indeed medium and the confidence is high. Now, lets take a deeper look at the results. Well, the problem is that is using insecure functions, like the following code:

```
def equlbut():
  global operator
  add=str(eval(operator))
  textin.set(add)
  operator=''
```

The problem it has is with code injection, if *eval()* is used with untrusted input, a malicious user can pass dangerous code to it, such as **rm -rf** and it is hard to sanitize the input. An example on how it can be exploited is the following:

```
user_input = input("Enter␣a␣Python␣expression:␣")
eval(user_input)
```

a malicious user can enter:

```
__import__('os').system('rm -rf /')
```

A safer alternative is to use **literal_eval**. This is much safer because it limits the input to literals such like strings, numbers, etc. It does not execute arbitrary code.

3.2 Exercise 2: Analyzing Bandit Output

Side note: I used the following repo for the purpose of this question `https://github.com/SchBenedikt/Text-Editor`

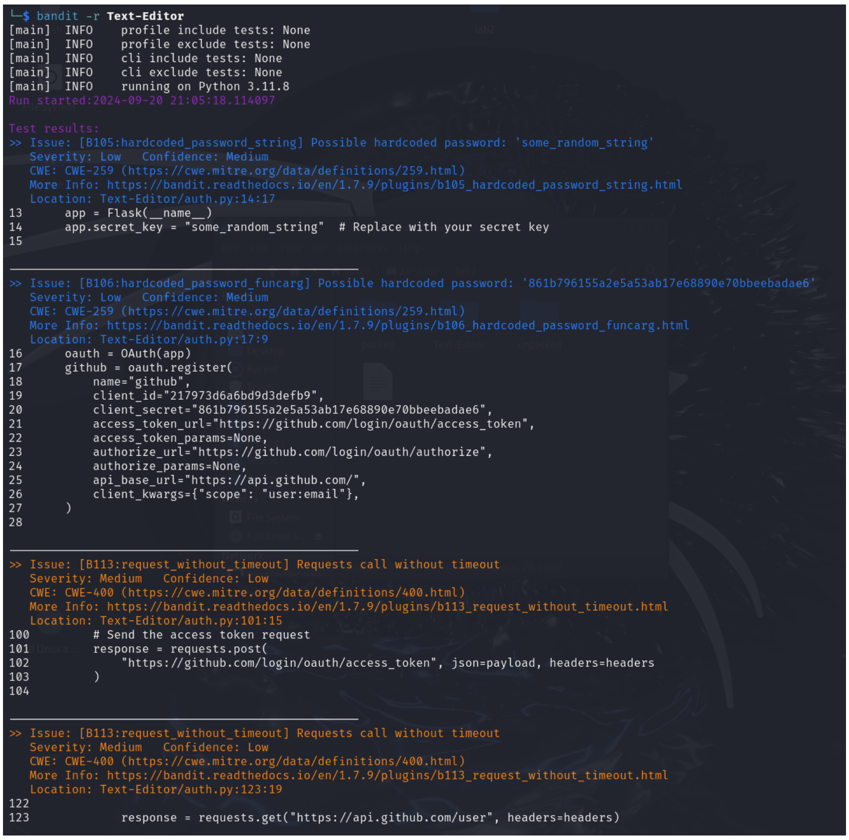
3.2.1 Objective:

Practice interpreting Bandit's results and prioritizing security fixes.

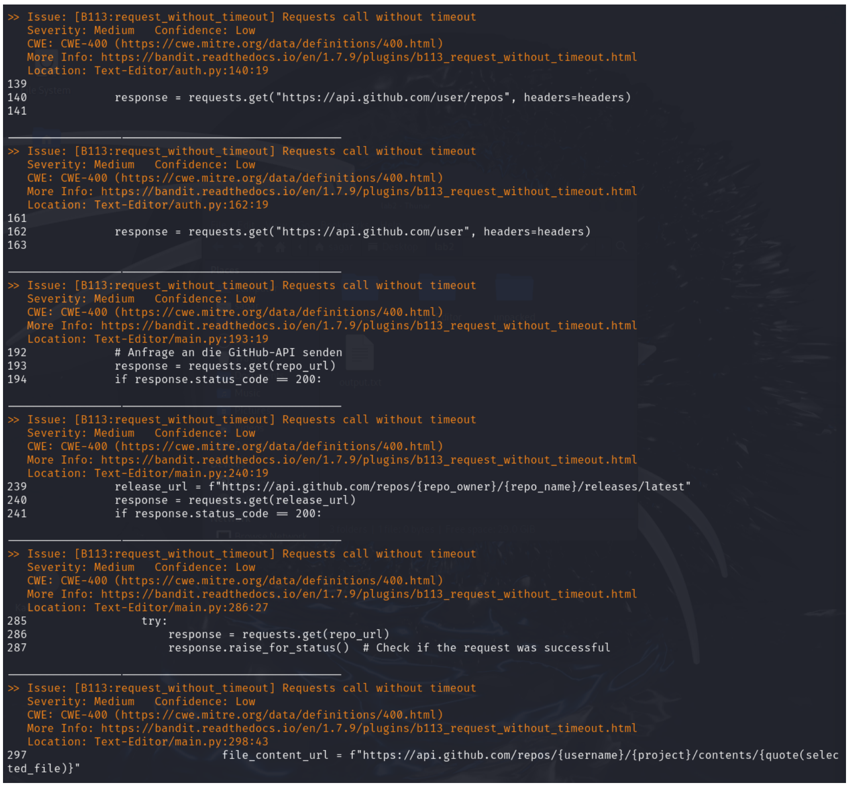
3.2.2 Steps Taken:

1. Download a Script of python
2. Run bandit on the script using: **bandit -r**
3. Review the output for high-severity issues, document them, and suggest mitigation strategies.

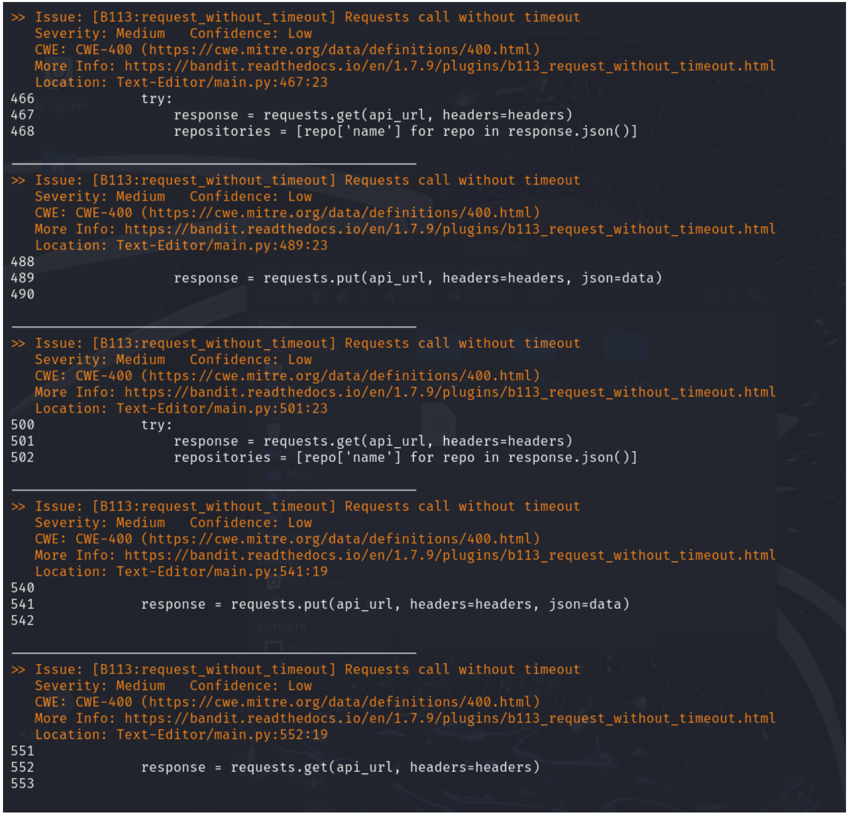
3.2.3 Screenshots



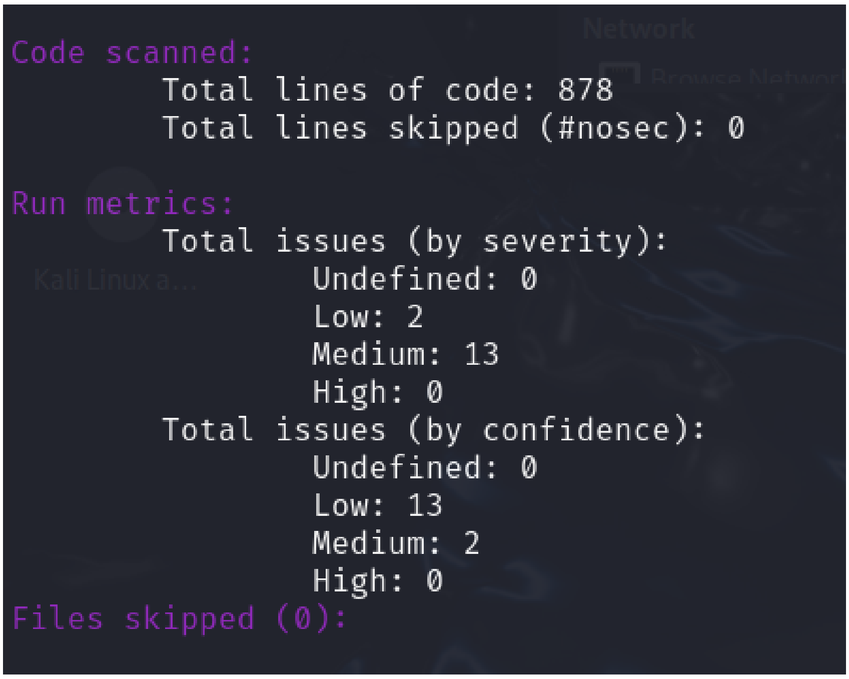
```
└─$ bandit -r Text-Editor
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.8
Run started:2024-09-20 21:05:18.114097

Test results:
>> Issue: [B105:hardcoded_password_string] Possible hardcoded password: 'some_random_string'
   Severity: Low   Confidence: Medium
   CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b105_hardcoded_password_string.html
   Location: Text-Editor/auth.py:14:17
13      app = Flask(__name__)
14      app.secret_key = "some_random_string"  # Replace with your secret key
15

──────────────────────────────────────────────────
>> Issue: [B106:hardcoded_password_funcarg] Possible hardcoded password: '861b796155a2e5a53ab17e68890e70bbeebadae6'
   Severity: Low   Confidence: Medium
   CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b106_hardcoded_password_funcarg.html
   Location: Text-Editor/auth.py:17:9
16      oauth = OAuth(app)
17      github = oauth.register(
18          name="github",
19          client_id="217973d6a6bd9d3defb9",
20          client_secret="861b796155a2e5a53ab17e68890e70bbeebadae6",
21          access_token_url="https://github.com/login/oauth/access_token",
22          access_token_params=None,
23          authorize_url="https://github.com/login/oauth/authorize",
24          authorize_params=None,
25          api_base_url="https://api.github.com/",
26          client_kwargs={"scope": "user:email"},
27      )
28

──────────────────────────────────────────────────
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/auth.py:101:15
100         # Send the access token request
101         response = requests.post(
102             "https://github.com/login/oauth/access_token", json=payload, headers=headers
103         )
104

──────────────────────────────────────────────────
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/auth.py:123:19
122
123             response = requests.get("https://api.github.com/user", headers=headers)
```



```
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/auth.py:140:19
139
140             response = requests.get("https://api.github.com/user/repos", headers=headers)
141

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/auth.py:162:19
161
162             response = requests.get("https://api.github.com/user", headers=headers)
163

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:193:19
192             # Anfrage an die GitHub-API senden
193             response = requests.get(repo_url)
194             if response.status_code == 200:

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:240:19
239             release_url = f"https://api.github.com/repos/{repo_owner}/{repo_name}/releases/latest"
240             response = requests.get(release_url)
241             if response.status_code == 200:

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:286:27
285                 try:
286                     response = requests.get(repo_url)
287                     response.raise_for_status()  # Check if the request was successful

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:298:43
297                             file_content_url = f"https://api.github.com/repos/{username}/{project}/contents/{quote(selec
ted_file)}"
```



```
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:467:23
466             try:
467                 response = requests.get(api_url, headers=headers)
468                 repositories = [repo['name'] for repo in response.json()]

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:489:23
488
489                 response = requests.put(api_url, headers=headers, json=data)
490

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:501:23
500             try:
501                 response = requests.get(api_url, headers=headers)
502                 repositories = [repo['name'] for repo in response.json()]

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:541:19
540
541             response = requests.put(api_url, headers=headers, json=data)
542

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Requests call without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html
   Location: Text-Editor/main.py:552:19
551
552             response = requests.get(api_url, headers=headers)
553
```



```
Code scanned:
        Total lines of code: 878
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 2
                Medium: 13
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 13
                Medium: 2
                High: 0
Files skipped (0):
```

3.1 This is the summary Results

3.2.4 Analysis & Conclusion

We can see from the scan summary that there are a total of 878 lines of code. In the metrics we can see the severity and confidence. The **severity** indicates the potential impact or seriousness of a detected issue if it were to be exploited. The **confidence** refers to the likelihood that the identified issue is a real problem based on the analysis performed. now lets compare the results we got:

Severity

- Low: 2
- Medium: 13

Confidence

- Low: 13
- Medium: 2

We can notice that the program makes some mistakes. The issue is the following:

```
Issue: [B105:hardcoded_password_string]
Possible hardcoded password:'some_random_string'
Severity: Low   Confidence: Medium
```

Of course if take a look that the lines of code :

```
app = Flask(__name__)
app.secret_key = ”some_random_string”  # Replace with your secret key
```

We can notice its not a real vulnerability. it is just a mistake, and the program ranked as confidence medium and severity low. The rest are some of the same vulnerabilities, its doing request but it does not have timeout for connections. Here is an example:

```
Issue: [B113:request_without_timeout] Requests call without timeout
Severity: Medium   Confidence: Low
```

In this case we do not have high severity problems

3.3 Exercise 3: Using Bandit with Custom Configuration

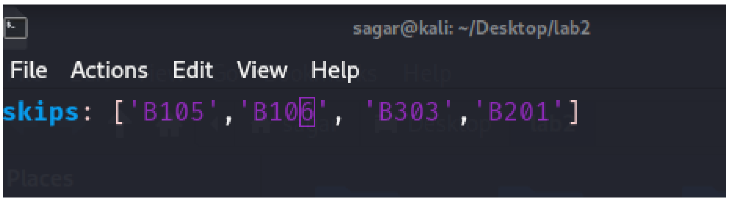
3.3.1 Objective:

Customize Bandit to focus on specific security scenarios.

3.3.2 Steps Taken:

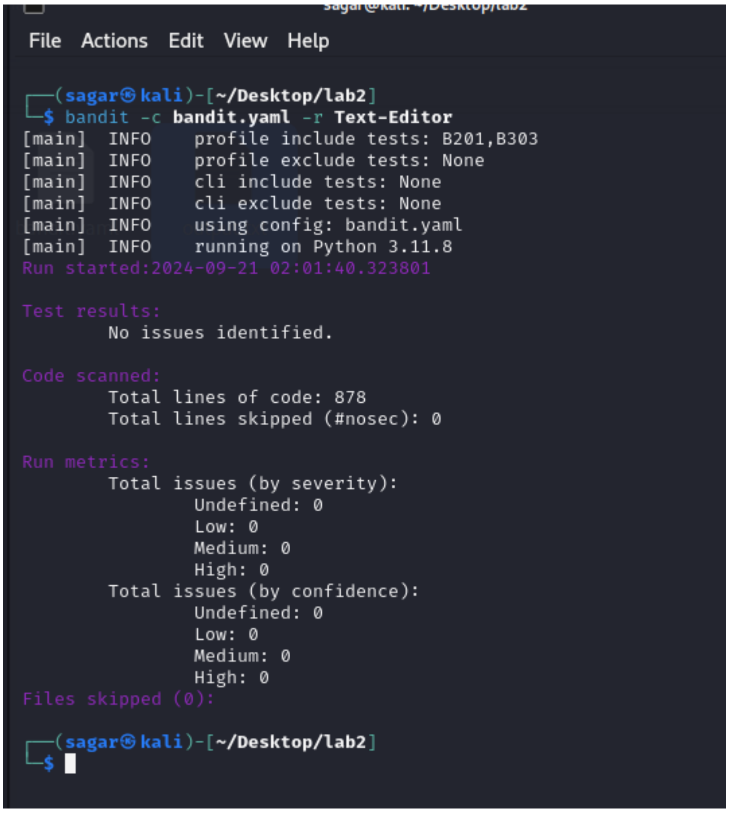
- Create a custom configuration file (.bandit) to ignore certain checks or focus on specific vulnerabilities.
- Example: Exclude low-severity issues or checks for weak cryptography.
- Run Bandit with the custom configuration: bandit-r .-c .bandit
- Document how the custom configuration changes the scan results

3.3.3 Screenshots



```
skips: ['B105','B106', 'B303','B201']
```

3.1 bandit.yaml



```
┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ bandit -c bandit.yaml -r Text-Editor
[main]  INFO    profile include tests: B201,B303
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    using config: bandit.yaml
[main]  INFO    running on Python 3.11.8
Run started:2024-09-21 02:01:40.323801

Test results:
        No issues identified.

Code scanned:
        Total lines of code: 878
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
Files skipped (0):

┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ █
```

3.2 Program execution with custom configuration

3.3.4 Analysis & Conclusion

The picture 3.1 is the whole bandit.yaml it was needed to skip low-severity and exclude checks for weak cryptography. if you want to exclude, you have to utilize skips, with the bandit codes.

The **Bandit Codes**: In this case, we used the codes:

- **B105** : This is used for the Hardcoded password string.
- **B106** : This is used for the Hardcoded password funcarg.
- **B303** : Detects Use of Weak Hashing Algorithms.

So the syntax is the following:

```
skips: ['B105','B106','B303']
```

In conclusion, having a custom configuration for the Bandit tool proves to be immensely beneficial. By tailoring the settings to specific requirements, users can enhance their understanding of the security checks being performed on their programs. This customization allows for a more focused analysis, enabling developers to identify potential vulnerabilities more effectively.

3.4 Exercise 4: Automated Security Testing in CI/CD Pipeline

3.4.1 Objective:

Learn how to automate security testing with Bandit in a CI/CD environment.

3.4.2 Steps Taken:

- Integrate Bandit into a CI/CD pipeline, such as GitHub Actions, Jenkins, or GitLab CI.
- Set up the pipeline to run Bandit automatically on every code push.
- Fail the build if high-severity vulnerabilities are detected.
- Document the pipeline configuration and analyze the results

3.4.3 Analysis & Conclusion

For the exercise 4, I did not completed it because it is quite risky to put my github account in this kali linux VM, the reason is that we are analyzing malware and I do not want to compromise any of my credentials.

I hope this explanation could help.

3.5 Exercise 5: Creating Custom Bandit Plugins

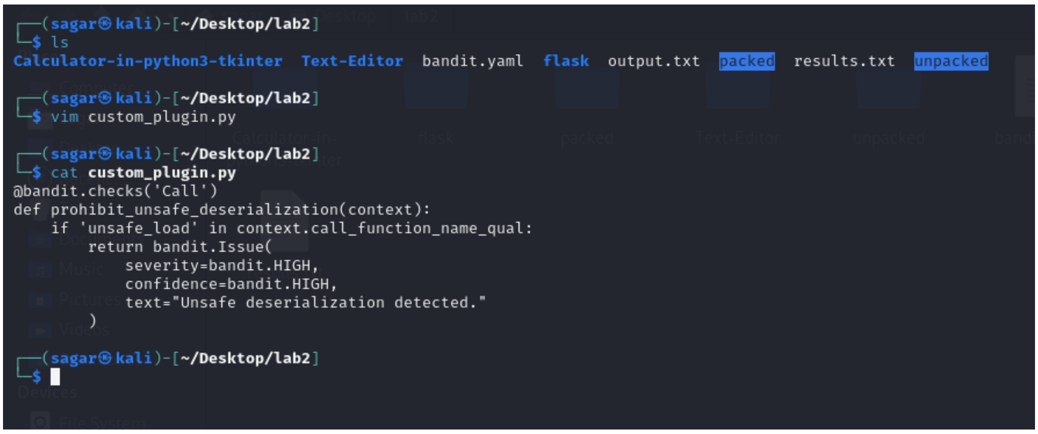
3.5.1 Objective:

Extend Bandit's capabilities by creating custom security checks.

3.5.2 Steps Taken:

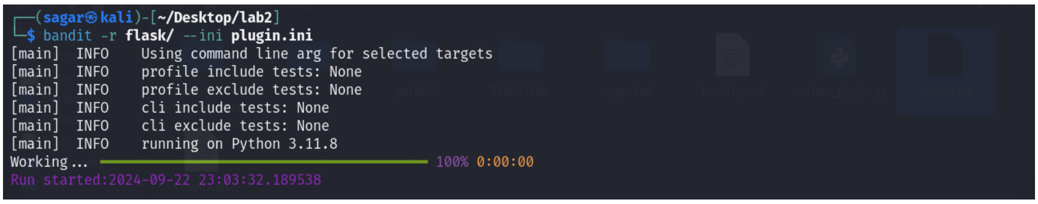
- Develop a custom Bandit plugin to detect a specific security issue in your project.
- write the plugin logic in Python.
- Integrate the plugin with Bandit and run a test scan using: bandit-r .–ini custom-plugin.ini
- Document the development process and how the plugin enhances security checks.

3.5.3 Screenshots



```
┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ ls
Calculator-in-python3-tkinter  Text-Editor  bandit.yaml  flask  output.txt  packed  results.txt  unpacked

┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ vim custom_plugin.py

┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ cat custom_plugin.py
@bandit.checks('Call')
def prohibit_unsafe_deserialization(context):
    if 'unsafe_load' in context.call_function_name_qual:
        return bandit.Issue(
            severity=bandit.HIGH,
            confidence=bandit.HIGH,
            text="Unsafe deserialization detected."
        )

┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ █
```

4.1 default_plugin.py



```
┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ bandit -r flask/ --ini plugin.ini
[main]  INFO    Using command line arg for selected targets
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.8
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
Run started:2024-09-22 23:03:32.189538
```

4.2 Plugin running

3.5.4 Analysis & Conclusion

First we have to get started, I created a custom plugin. In the picture **4.1** we can see the code that will check for hard coded credentials. This is quite helpful because sometimes we want to check we did no left something important an attacker can use.

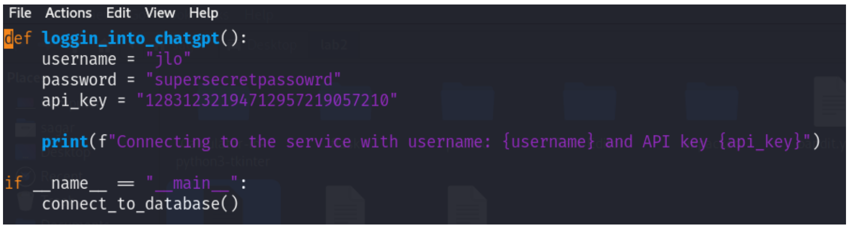
The way to map the plugin is the following:

```
[bandit]
plugin = custom_plugin
```

In order to run it we can do:

```
bandit -r flask/ --ini plugin.ini
```

On picture 4.2 we can see it running with no problems, thankfully in flask we do not have any hardcoded credentials but let us try it in this example:

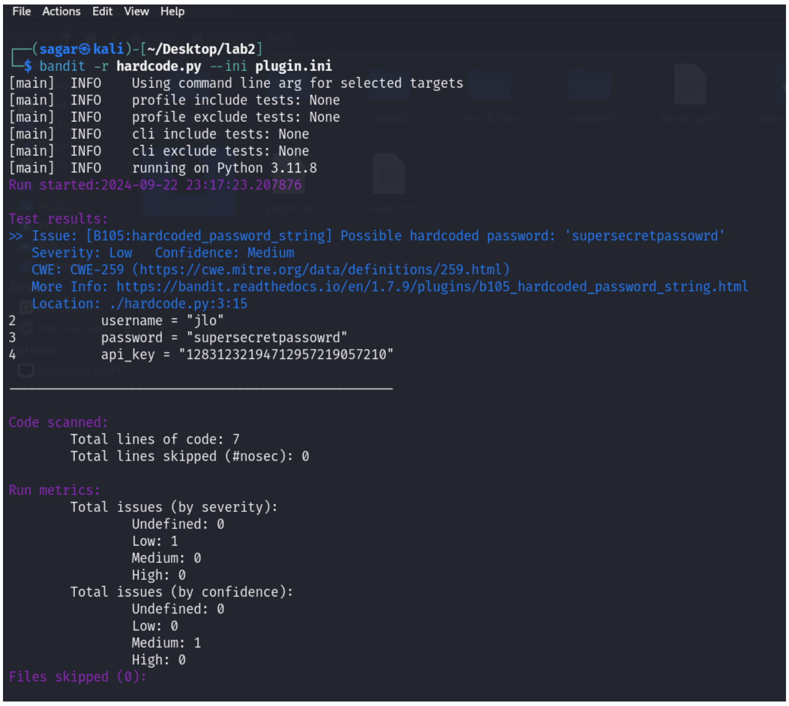


```
def loggin_into_chatgpt():
    username = "jlo"
    password = "supersecretpassowrd"
    api_key = "1283123219471295721905721O"

    print(f"Connecting to the service with username: {username} and API key {api_key}")

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

let's say we are working on a coding tutorial on how to connect to the chatGPT API, and we have thousands of lines. We want to make sure that before we submit the code to Github or any other service that is going to be exposed to the internet, we are safe, we have to run the script of **4.1** and pass it our code, like this:

```
bandit -r hardcode.py --ini plugin.ini
```

This is the output:



```
┌──(sagar㉿kali)-[~/Desktop/lab2]
└─$ bandit -r hardcode.py --ini plugin.ini
[main]  INFO    Using command line arg for selected targets
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.8
Run started:2024-09-22 23:17:23.207876

Test results:
>> Issue: [B105:hardcoded_password_string] Possible hardcoded password: 'supersecretpassowrd'
   Severity: Low   Confidence: Medium
   CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b105_hardcoded_password_string.html
   Location: ./hardcode.py:3:15
2           username = "jlo"
3           password = "supersecretpassowrd"
4           api_key = "12831232194712957219057210"

--------------------------------------------------

Code scanned:
        Total lines of code: 7
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 1
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 1
                High: 0
Files skipped (0):
```

Now we are safe and we can change those lines of codes inmmediately.

3.6 Exercise 6: Scanning a Large Python Project

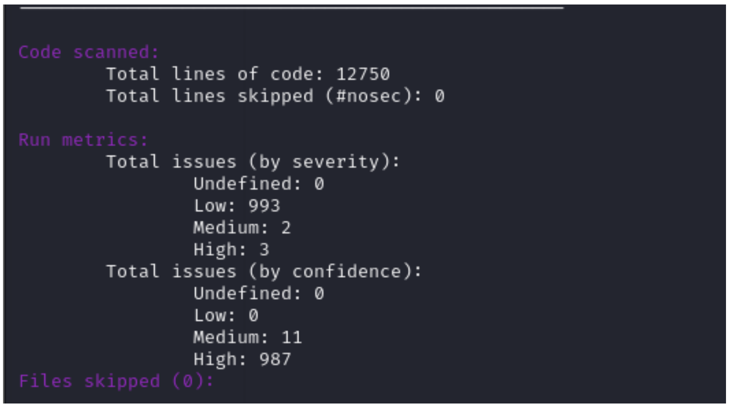
3.6.1 Objective:

Conduct a comprehensive security audit using Bandit on a large-scale project

3.6.2 Steps Taken:

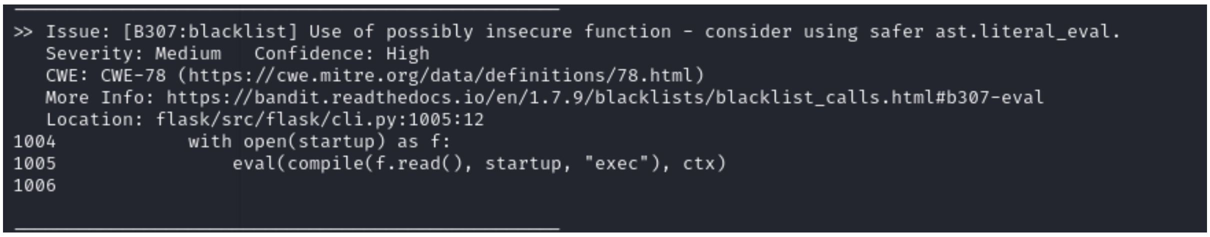
- Choose a large open-source Python project (e.g., Django, Flask) from https://github.com.
- Run Bandit on the project and focus on high-severity vulnerabilities.
- Analyze and document your findings, along with recommendations for fixing them.

3.6.3 Screenshots



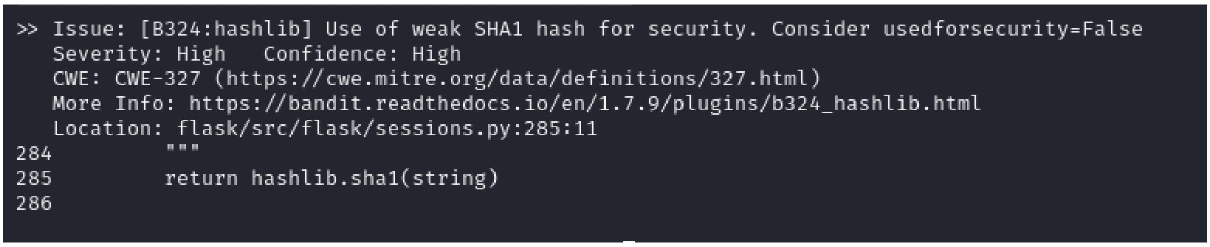
```
--------------------------------------------------

Code scanned:
        Total lines of code: 12750
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 993
                Medium: 2
                High: 3
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 11
                High: 987
Files skipped (0):
```

5.1 Summary analysis



```
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer ast.literal_eval.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b307-eval
   Location: flask/src/flask/cli.py:1005:12
1004            with open(startup) as f:
1005                eval(compile(f.read(), startup, "exec"), ctx)
1006
```

5.2 Medium error



```
>> Issue: [B324:hashlib] Use of weak SHA1 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.9/plugins/b324_hashlib.html
   Location: flask/src/flask/sessions.py:285:11
284         """
285         return hashlib.sha1(string)
286
```

5.3 High error

3.6.4 Analysis & Conclusion

In this exercise, we have a total lines of code of 12750. The scan showed that there are 993 severity Low, 2 medium and 3 high. Now, to further do an inspection I ran the following command, which it will put the results in a text file:

```
bandit -r flask >> results.txt
```

One of the errors we encounter is the one we can see at the image 5.2, which is one we preciously mentioned that replace eval with literal_eval. For the second error, in this case it severity: high. lets analyse this error, the error is the use of weak SHA1, of course we could fix it but i do not have the expertise to do it.

3.7 Conclusion

As a conclusion, I can say that bandit is a powerful tool to incorporate into our daily programming, the interesting thing is that you can incorporate it into your own pipeline when doing commits, and we can test repos that you download before running. I was watching a YouTube video and it had malicious code but it was tabbed out, so it cannot be visible to the user reading the code.

Video of malicious activity python

Using bandit, with the correct setting to look for **os.system** and it will help us avoid getting a compromised git.

Chapter 4

Reflections

I loved this lab, specially all the tools we used to reverse engineer malware, this copuld be used even to reverse engineer common stuff. On the section of bandit i put a video that is one truly help on how bandit works. I believe reverse engineer could be applied in any ambit where we need to understand stuff.

Something funny is that in popular devices such as HDMI tester is common practice they use hot glue to cover all the circuits so that another company cannot recreate their product. some times it is useful to apply some reverse engineer so that we know how things work. referring to viruses/malware is so useful so that we train antivirus in how to detect those common programs and be safe.