pyELFer: PYTHON TOOL FOR ANALYZING AND EXPLOITING ELF BINARIES

Ajay S K, Abishek R, Ajay Kumar R
Department of CSE, Panimalar Engineering College, Chennai

ABSTRACT:

Reverse Engineering of a Binary is that the process that hackers use for deciding a program's components and functionalities and sometimes even get access to the program's partial ASCII text file by decompiling it employing a decompiler so as to seek out vulnerabilities within the program. this idea also can be utilized in capturing the flag contest reverse engineering challenges where a participant has got to find the flag secretly hid inside the binary. Our project "pyELFer" may be a python program that reverses an ELF binary (ELF-Executable and Linkable Format) and provides information about the binary and mainly optimized for CTF's challenges where the binary itself find the flag for the challenge and even looks out whether the binary is susceptible to any overflow attacks.

Keywords: ELF, CTF, CyberChef, RSATool

I. INTRODUCTION

Computer security represents a challenge to education because of its interdisciplinary nature. Topics in computer security are drawn from areas ranging from theoretical aspects of computing to applied aspects of knowledge technology management. On concept for this measure has emerged because the 'capture the flag' competition. Attack-oriented CTF competitions attempts to distill the essence of the various aspects of professional computer security work into one short exercise that's objectively measurable. The areas that CTF competitions operate and use are the vulnerability exposure, exploit creation, toolkit creation, and operational tradecraft.

II. CONCEPTS USED IN CTF

a) Cryptography

Cryptography is that the tactic of protecting information which involves encrypting or decrypting a touch of data. Tools are CyberChef, FeatherDuster, Hash Extender, and padding-oracle-attacker, PkCrack, RSACTFTool, RSATool, XORTool, Cryptii, Keyboard Shift, and much of more.

b) Steganography (Stego)

Steganography is that the technique tasked with finding information hidden within the files or images. Tools used are StegCracker, Steghide, Openstego, Stegsolve, Online stego tool, and lots of more.

c) Binary Exploitation/pwn

It's basically exploiting a file and exploiting a server to hunt out the flag.

d) Reverse Engineering

Reverse Engineering in a CTF is typically the process of taking a compiled (machine code, bytecode) program and converting it back into a more human readable format.

e) Web

Web Tools used are Commix, Hackbar, BurpSuite, Raccoon, SQLMap, DirBuster, gobuster, nikto, wpscan, CloudFlare Bypass, Edit This Cookie, File or Directory(robots.txt, /.git/, /admin/), and lots of more.

vi) Forensics

Forensics challenges can include file format analysis, steganography, memory dump analysis, or network packet capture analysis.The challenge to look at and process a hidden piece of data out of static data files (as against executable programs or remote servers) might be considered a Forensics challenge.

III. MODES OF OPERATION

We have two modes of operation –

1. REVERSE ENGINEERING MODE
2. BINARY EXPLOITATION MODE

At the starting of the application the user is prompted for entering the CTF ELF executable binary. After that these two modes will be displayed and ask the user for the mode of use at the beginning. The user after being prompted for the mode chooses one and goes into that mode.

1. REVERSE ENGINEERING MODE:

In Reverse Engineering mode, the following modules are present:

a) ltrace

Prints out the library calls the binary makes to the libc present in the local system with their arguments to the functions.

b) strace

Prints out the system calls made by the binary.

c) Critical String Functions

It detects the presence of critical string functions used to validate the flag and display its arguments being used.

d) Input comparison and Modified input comparison

Used to print the comparison characters that are being compared with the user input string and also detects the comparison characters even if the strings are modified or masked with byte operations.

e) ANGR Simulator

A simulation engine that tries a bruteforce to get the flag or keygen using various methods.

2. BINARY EXPLOITATION MODE:

In Binary Exploitation mode, the following modules are present:

a) Ret2win

An exploitation method where the hidden function has the shell(super user shell maybe) present in them or the flag displaying function. This technique is also called code execution redirection.

b) Injection of Shell Code in .bss

If the NX bit is disabled, the stack and .bss is an executable memory space where injection of shellcode is done and the code is redirected to our injected shellcode to get code execution or shell.

c) Ret2libc

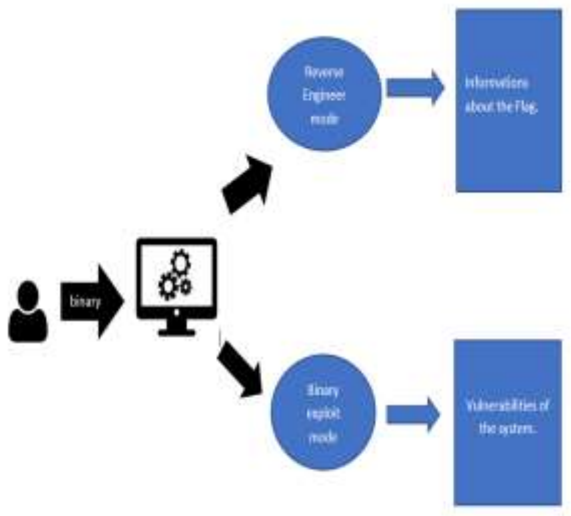
If our binary has NX bit enabled. The stack and .bss section is not executable where no injected instructions can be executed.By using a memory leak and some ROP gadgets we can get to system function and get shell.

d) Ret2csu→Ret2libc

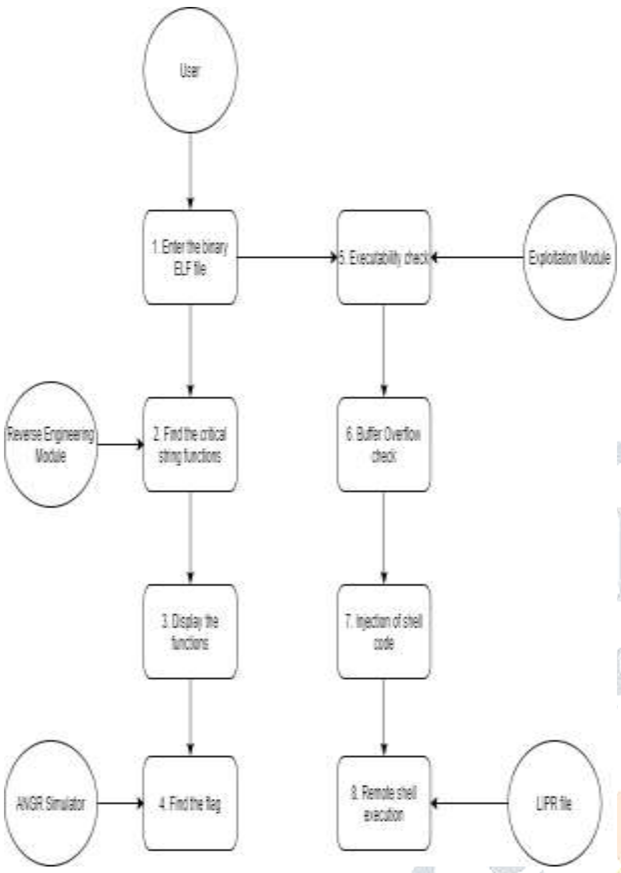
A complex method which involves a 64 bit binary with a memory leak function that has more than 2 arguments to leak memory addresses. And then use the memory addresses to calculate offset to perform a classical ret2libc.

IV. RESULTS AND DISCUSSION

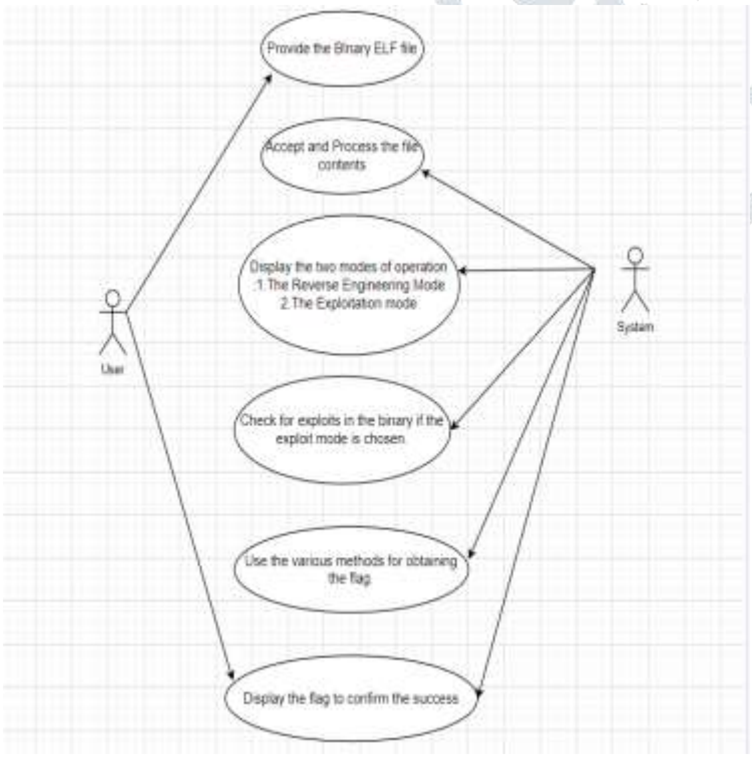
A. System Architecture



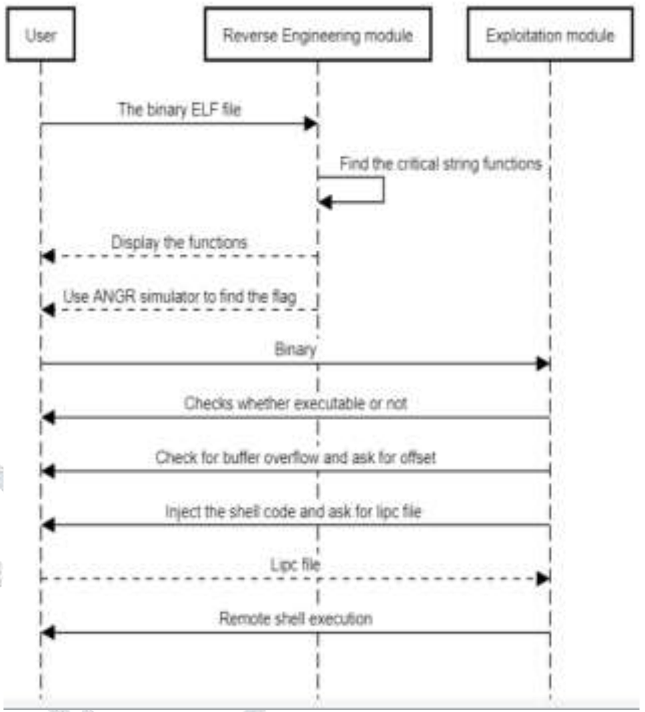
B. Data Flow Diagram


User
1. Enter the binary ELF file
5. Executability check
Exploitation Module
Reverse Engineering Module
2. Find the critical string functions
6. Buffer Overflow check
3. Display the functions
7. Injection of shell code
ANGR Simulator
4. Find the flag
8. Remote shell execution
LIPR file

C. Use Case Diagram


Provide the Binary ELF file
Accept and Process the file contents
Display the two modes of operation
1. The Reverse Engineering Mode
2. The Exploitation mode
Check for exploits in the binary if the exploit mode is chosen
Use the various methods for obtaining the flag
Display the flag to confirm the success
User
System

D. Sequence Diagram


User
Reverse Engineering module
Exploitation module
The binary ELF file
Find the critical string functions
Display the functions
Use ANGR simulator to find the flag
Binary
Checks whether executable or not
Check for buffer overflow and ask for offset
Inject the shell code and ask for lipc file
Lipc file
Remote shell execution

V. OUTCOMES OF THE PROJECT

1. Analyze the ELF Binary file.

2. Group the printable characters in ELF binary file in a way that is optimized for user to get the idea about the strings, compiler version, sometimes OS info, user defined function names and predefined function names, section headers and source code name.

3. Automate the process of stack buffer overflow.

4. Detects whether the ELF binary is vulnerable to a buffer overflow attack.

5. Exploits the buffer overflow with different methods based on the binary's exploit mitigation present like ret2win, shellcode injection, ret2libc, ret2csu→ ret2libc.

6. Reverses the program to get knowledge about the functions used in binaries.

7. Fetches information about the library calls this is critical to get arguments used by string functions related to user input compare and system calls made by the binary.

8. Uses various method to find the keygen or flag present in the binary if for any CTF challenges.

VI. CONCLUSION

In summary, the term 'capture the flag', shortly 'CTF' and the concepts used in it have been discussed. Also the program is designed to do two different operations, one to find the flag using Reverse Engineering and another one to check vulnerability for any buffer overflow attacks using Binary Exploitation Method.

VII. REFERENCES

Popa Marius, Binary Code Disassembly for Reverse Engineering, December 2012, [https://www.researchgate.net/publication/235611044_Binary_Code_Disassembly_for_Reverse_Engineering]

Aleph One, Smashing The Stack For Fun And Profit, Phrack Magazine, November 1996[http://phrack.org/issues/49/14.html]

Nergal, The Advanced return-into-lib(c) exploits (PaX case study), December 2001, [http://phrack.org/issues/58/4.html]

Dr. Hector Marco-Gisbert and Dr. Ismael Ripoll-Ripoll, Address Space Layout Randomization (ASLR), 2018, [https://i.blackhat.com/briefings/asia/2018/asia-18-Marco-return-to-csu-a-new-method-to-bypass-the-64-bit-Linux-ASLR-wp.pdf]

ANGR – A Python Framework for analyzing binaries, [https://angr.io/]