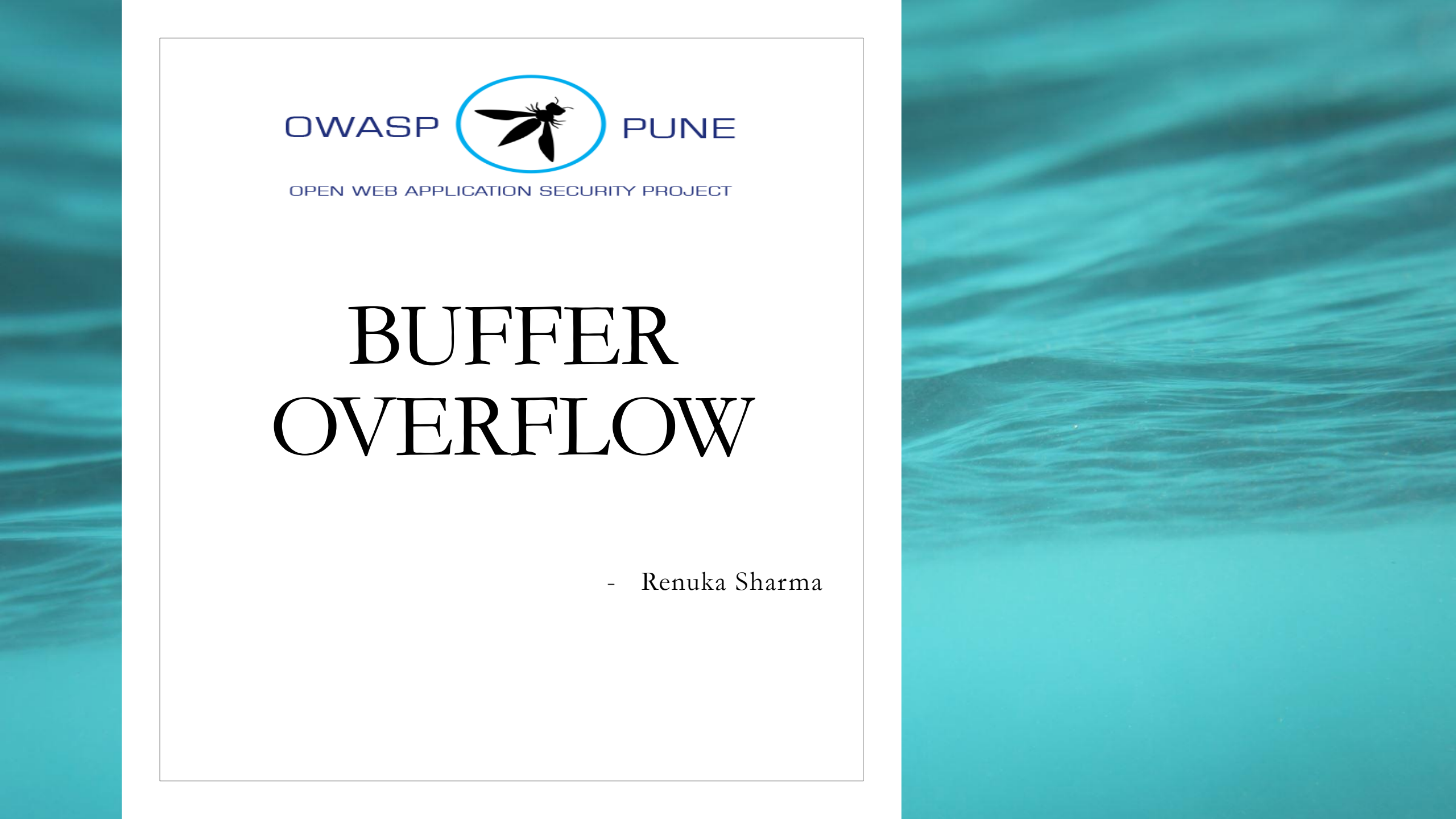
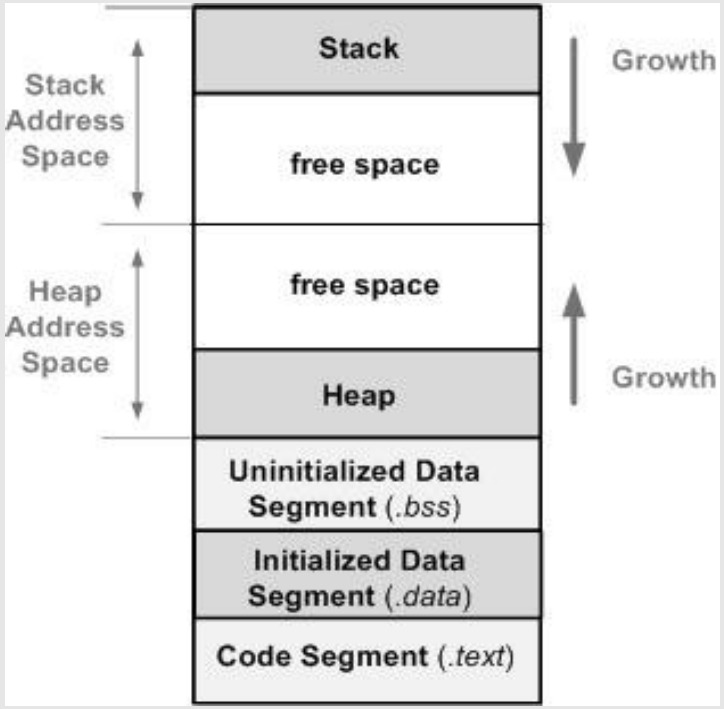


BUFFER OVERFLOW

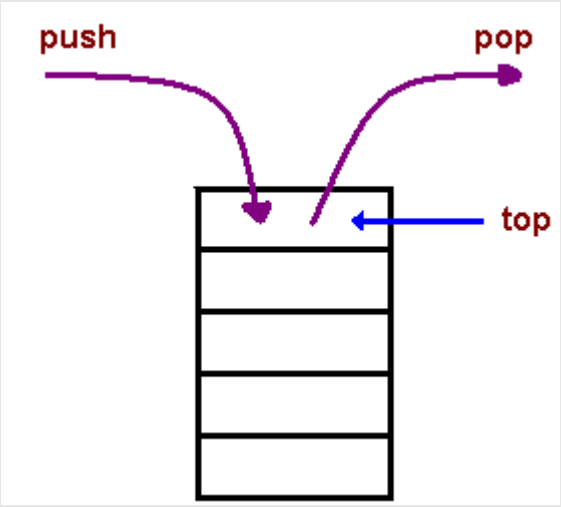
- Renuka Sharma

Memory Layout of C Program

- Several segments of program
  - text
  - Initialized Data Segment (.data)
  - Uninitialized Data Segment
  - Heap
  - Stack



What is Stack

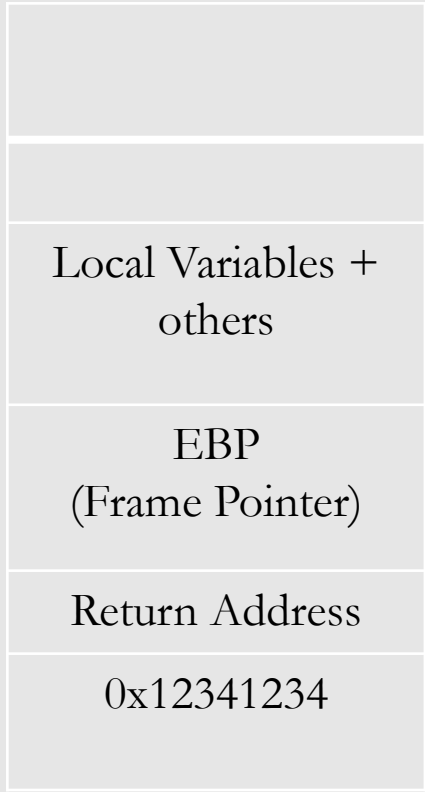


- A reserved area of memory used to keep track of a program's internal operations, including functions, return addresses, passed parameters
- A stack is work as "last in, first out" (LIFO) data s) data structure
- When you PUT something ONTO the stack (PUSH onto the stack), the SP is decremented before the item is placed on the stack.
- When you take something OFF of the stack (PULL from the stack), the SP is incremented after the item is pulled from the stack.

How Stack Memory Works

```
Main(……)
{
…….
Function1(0x12341234)
……..        <-------  Return Address
}

Function1()
{
……..
}
```



Frame Pointer

- Stack frame contains
  - Parameters to the function
  - Local Variables
  - The data necessary to recover previous stack frame
  - Value of Instruction Pointer
- Frame pointer points to fixed location in the frame

Buffer overflow Attacks

- A buffer overflow occurs when a program or process attempts to write more data to a fixed length block of memory, or buffer than the buffer is allocated to hold.

- The most important techniques to identify buffer overflow bugs are as follows:
- Reverse Engineering
- Source Code Analysis.
- Fuzzing.

Vulnerable code sample

```
  /* vuln.c */
#include<stdio.h>
  int main(int args,char *argv[1])
  {
  char buffer[64];
  strcpy(buffer,argv[1]);
  return(0);
  }
```

python –c ‘print “A”*80’ | ./vuln

Disabling stack protection

Compiling the program

- Disable ASLR(Address Space Layout Randomization)
- Memory protection process for operating system
- Memory addresses associated with the running processes on system are not predictable.
- Vulnerabilities associated with processes will be difficult to exploit.

Sudo bash –c 'echo 0 > /proc/sys/kernel/randomize_va_space'

- Disable GCC stack smashing protection

gcc login.c -o login -fno-stack-protector -m32

-m32 : tells gcc to compile for 32 bit machines

Controlling EIP

- EIP register controls the execution flow of program
- We need to locate the exact position of EIP in our buffer
- Send Unique string input , then locate the position of this bytes
- Script for generating unique string :

  /usr/share/Metasploit-framework/tools/pattern_create.rb 88
- To discover exact position of bytes overwriting EIP :

  /usr/share/Metasploit-framework/tools/pattern_offset.rb unique4byte
- Buffer = “A”*72+”BBBB”+”C”*12

Bad characters

- List of characters that can break the shell codes.
- Some of the very common bad characters are:

- 00 for NULL
- 0A for Line Feed \n
- 0D for Carriage Return \r
- FF for Form Feed \f

Identifying Bad Characters

- The steps to identifying the bad characters are given below.
- Send the full list of the characters from 0x00 to 0xFF as input into the program.
- Check using debugger if input breaks
- If so, find the character that breaks it.
- Remove the character from the list and go back to first step again.
- If input no longer breaks, the rest of the characters could be used to generate the shell code.

Program execution control

- At the time of crash, ESP always points to the beginning of buffer
- Looking for naturally occurring instruction JMP ESP in supporting dll of program
- open any DLL and search for the JMP ESP instruction that does not have the Null Byte in the address.
- Find corresponding address of JMP ESP instruction in DLL
- Now replace the location of EIP with the address of JMP ESP in reverse order

  4B7CABD4 - \xD4\xAB\x7C\x4B

Shellcode

- Generate Shellcode with msfvenom
- Msfvenem also gives the flexibility to exclude the bad character

  msfvenom -p windows/shell_bind_tcp -f c -a x86 -b “\x00”
- You can Use netcat to get reverse shell

Conclusion

- Buffer overflow are the result of stuffing more information in buffer than it meant to be hold.
- The standard C library provides a number of functions to copying strings or receiving string, that does not follow boundary checking including strcpy(),strcat(), sprint(),gets() etc.
- After the attacker locates these weak functions, they try to see what amount of input is needed to overwrite IP or return pointer. To do this, the attacker passes some amount of similar input like a series of 'A' into the input field
- Attacker should also make sure that the exploit code will fit into the buffer and does not contain characters that are filtered out

References

- **Binary Exploitation / Memory Corruption by LiveOverflow**

  https://www.youtube.com/watch?v=8QzOC8HfOqU
- https://exploit.education/
- https://resources.infosecinstitute.com/stack-based-buffer-overflow-in-win-32-platform-part-1/
- https://blog.securelayer7.net/static-analysismemory-forensics-reverse-engineering-thick-client-penetration-testing-part-4/
- https://medium.com/@n0auth/buffer-overflows-0x01-67664959a256
- http://www-inst.eecs.berkeley.edu/~cs161/fa08/papers/stack_smashing.pdf

Thank You . .