- QUESTION: Find and discuss an example of a buffer overflow exploit. (Minimum word count 300 with references). How could you verify that an overflow condition exists?
- FIELD: COMPUTER SCIENCE

A buffer overflow exploit is a type of software vulnerability that occurs when a program tries to store data in buffer than it was designed to hold. In other words, it is an anomaly whereby a program writes data to a buffer beyond the buffer's allocated memory, overwriting adjacent memory locations.

Here is an example of a buffer overflow exploit:

Suppose there is a program that reads user input from command line and stores it in a temporary storage buffer of size 12. The program then prints the contents of the buffer to the output screen. Here is a sample C code I will use to demonstrate:

```
#include <stdio.h>
#include <string.h>
int main (int argc, char*argv){
Char buffer [12];
Strcpy(buffer,argv[1]);
Printf("Buffer information: %s\n",buffer);
Return 0;
}
```

From the above sample code, if an attacker could provide an input that is longer than 12 characters, the excess data would overflow into adjacent memory locations, potentially overwriting important data or code. For example, an attacker could provide the following input:

./program $(python –c 'print "B"*19')

This will cause the buffer to be filled with 19 "B" characters, consequently overflowing into adjacent memory locations. Depending on the specific program and system configuration, this could allow the attacker to execute arbitrary code or crash the program.

Verifying the presence of an overflow condition involves a proper analysis of the code and understanding the behavior of the program. As it can be generally understood, here are some of the approaches that can be used to verify the existence of an overflow condition

Review the code paying close attention to functions or methods that handle user input or carry out buffer manipulation. Look for scenarios where the size of the input is improperly checked

before transferring into the buffer. Conduct a quick check for functions like printf which are prone to buffer overflows.

Dynamic analysis involves execution of a program with input data that may result in buffer overflows. This can be used to verify the presence of a buffer overflow by looking at the behavior of the program during execution, in case of any erratic behavior or crashes. When a crash occurs or output is noticed, then there is a possible occurrence of buffer overflow.

According to a synopsis by the Editorial team (February 07, 2017), fuzz testing can aid in verification of an overflow. This involves providing the program with a large amount of random input that is generated by a system. Conduct an observation on how the program handles the strange inputs and if it leads to buffer overflows.

Profiling the memory by using memory profiling tools to monitor the program's usage of the memory and the storage. Observe the memory behavior of the memory and note any abnormal behavior. If the memory writes excessively beyond allocated buffers, then it indicates the presence of a buffer overflow.

Enable Address Space Layout Randomization (ASLR) to randomize the memory address of the program's components. This makes it difficult for attackers to predict where to insert the malicious codes in the program. This quick verification tip will be of importance in providing a comprehensive approach to determining the existence of a buffer overflow.

Lastly, according to Benjamin Kerestan (July 12, 2017), use of static analysis tools such as lint or specialized security scanners can aid in identifying potential buffer overflow vulnerabilities by conducting an analysis of the code without executing it. This will enable one to prevent any possible buffer overflow.

In conclusion, buffer overflow can cause serious problems to computer components and the system as a whole. The verification methods mentioned in this piece of writing should be looked at carefully to prevent any possible outcome of buffer overflow.

REFERENCES

Benjamin Kerestan (July 12, 2017)

Synopsis by the Editorial Team (February 07, 2017)

BOOKS

1. James C. Forster- Published by Syngress, Rockland, MA (2005)
2. Jason Deckard-*Buffer overflow attacks (2005).*