Cataloguing and Avoiding the Buffer Overflow Attacks in Network Operating Systems

*P.Vadivelmurugan #K.Alagarsamy
*Research Scholar, Department of Computer Center, Madurai Kamaraj University, Madurai, Tamil Nadu, India
#Associate Professor, Department of Computer Center, Madurai Kamaraj University, Madurai, Tamil Nadu, India

Abstract:
The application software has a different dimension, size and intricacies is rising rapidly in current technology era and simultaneously increase a programming bugs also. The programming bugs cause vulnerabilities to the security systems. The large number of exploit is based on the buffer overflow vulnerability. In this paper, we classify the number of buffer overflow attacks with generation. Buffer overflow attacks are very harmful to current scenario; programmer writes a coding, in a buffer that overflows the boundary and overwrites in adjacent memory. This causes the erratic result and crash or breaks the computer security. We suggest the tools to prevent the buffer overflow vulnerability.

Keywords: *buffer overflow, stack smashing, heap overflow, corrupting memory, malicious code.*

I.INTRODUCTION

The growth of the networking and internet based applications, the number of security exploits are also increased simultaneously. The internet has interconnected the world and possible to share the ideas and resources. Unfortunately, the same technology permits the attacker to cause the vulnerability to the system. As a result software security is needed to detect and prevent the exploits caused by the attackers. Among the exploits Buffer overflow attack has been known for a long time, it has been from 1960[1] The most famous buffer overflow attack is the internet worm written by RoberT.Morris in 1988.A buffer overflow attack is done intentionally entering more data than a program was written to handle. Buffer overflow attacks exploit a dearth of boundary checking on the size of input being stored in a buffer. The extra data will overflow the memory set aside to accept it and overwrite another region of memory that was meant to hold some of the program's instruction. The result of this is a cascade, which can finally halt the application or the system it is running on. The outline values can be new

instructions, which could give the attacker control of the target machine depending on what was input.

There are two main internet worms have exploited buffer overflows to a large number of internet systems. In the year 2001, code Red worm which exploited a buffer overflow in Microsoft's Internet Information Services (IIS) 5.0 and in 2003 the SQL Slammer worm attacks machines running Microsoft SQL Server 2000. The given buffer overflow sample program, help us to understand the vulnerability easily

```
int main ()
{
int buffer[100];
buffer [200] = 300;
}
```

The above code which is given is a valid program, every compiler compile this program without any errors in each times. But, this program takes additional memory to write data, which is larger than the allocation, which gives the result in unexpected way [2].

I FUNDAMENTALS OF BUFFER OVERFLOW ATTACK

The researchers pointed out that there are three necessary conditions for bufferoverflow attacks to be successful: (i) injecting malicious code and (ii) redirecting the program control flow to execute that code (iii) writing the code with improper looping. In the most of attacks, control data is the objective of the attacker, so prevention methods have focused on control data. Control data can be divided into three types. They are return addresses, function pointers, and branch slots. Return addresses have been the primary target since their location can easily identify.

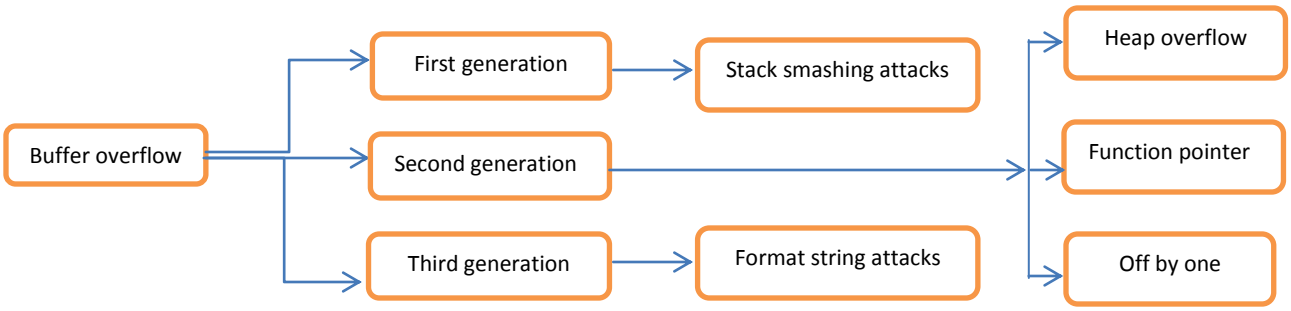
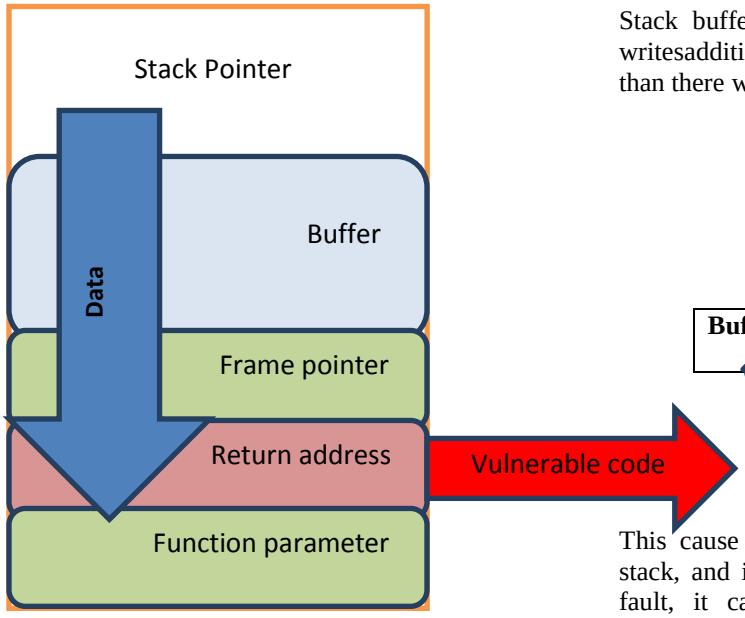


Figure 1: Classification of buffer overflow attack

The overflow problem transpired because not enough memory was assigned previous to being passed to one of the string library functions. The buffer overflow problems also occur in built-in functions of PHP, Ajax, Jquery and JAVA language. For performing buffer overflow attacks we give large amount of data in to the input field. Perl language is the one of the suited programming language for conducting a buffer overflow attacks. Buffer overflow is also tested by sending the repetitive request to the application and record response by the server. The server may irresponsible of the request and makes the memory overflow and it cause the server crash.

A. First generations - Stack smashing attacks

Injecting malicious code is not necessary since the code can be resident code found in library files. For example, jumping to resident shell code while in privileged mode is sufficient.



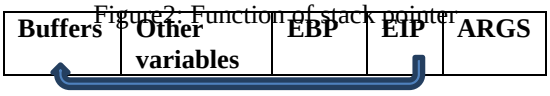

```
voidfunc (char *p, inti)
{
Int j=0
/* local variable*/
Vulnerable code…..;
Char b[128];
Strcpy(b,p);
}
```

```
Char b [128];  ---→ buffers


Int j=0
Vulnerable code…..;----→other
variables
```

Data can be inserted into the stack (popped) in a last-in and first out method. A stack is a place to store automatic variables, and these variables are considered only for that subroutine in which they are declared. Stack is used to linking the loops in a program. Stack pushes the return address on the stack and the subroutine is called. If the stack returns, the stored value from the stacks and jump to find the address. Stack is accessed by the registers that are called the Stack pointer, which indicates the current upper value of the stack. In the stack there is another pointer (Frame pointer) which is used to points to some static points in the frame structure, such as the place of return address Stack buffer overflow are initiated when a program writesadditional data in a buffer, located on the stack than there was actually assigned for that buffer.

Figure2: Function of stack pointer



This cause the corruption of contiguous data on the stack, and in cases where theoverflow was caused by fault, it causes the program to crash or function erroneously.

B. Second generations - Heap overflow

A buffer overflow attack on a heap works by corrupting data in the heap. Heap overflow attacks are normally harder to perform than a Stack based attack, because the overflow is not the only factor that regulates the realization, quite repeatedly the data in the heap must be corrupted not overwritten.

There are two ways a heap overflow attack is used, they are (1) To overwrite data stored in the heap (2) To overwrite a pointer this could be used to modify things akin to a pointer that points the code that is performed and change it to point to a malicious code from the attacker. [3]

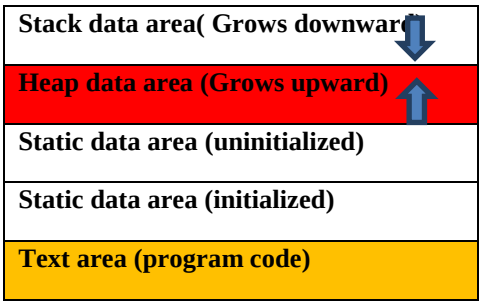


Figure 3: Stack and heap function the buffer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main(intargc, char **argv)
{
char *buffer = (char *) malloc(10);
char *input = (char *) malloc(10);
}
```

Off by one

An off-by-one overflow specifies a one-byte buffer overflow. Such an error is made exceedingly often in loop conditions [4]

Programmers who try to use in safe functions in the program such as strncpy, unnecessarily make their programs more secure from buffer overflows. Let consider the below program where the programmer has incorrectly used "less than or equal to [for (i=0; i<=128;i++)] " in the place of "less than" symbol [5]

```
#include <stdio.h>
#include <conio.h>
int x;
voidvuln(char *malicious)
{
char buffer [128];
for (x=0;x<=128;x++)
buffer[x]= malicious [x];
}
void main(intargc, char *argv[])
{
if (argc==2)
vuln(argv[1]);
}
```

Function pointer A function pointer error occurs if function in theprogram. In a memory, a function pointer follows a buffer; there is a chance to overwrite the function pointer if the buffer is unchecked. The local and shared function pointer may cause the vulnerable to the buffer.

C. Third generation Format string attack Crashing the program printf ("%s%s%s%s%s%s%s%s%s%s%s%s");

For every %s statement , printf() will fetch a number from the stack, it consider this number as an address, and print out the memory contents pointed by this address as a string, until a NULL character like (i.e., number 0, not character 0) is encountered.

- Since the number fetched by printf () might not be an address, the memory pointed by this number might not occur (i.e. no physical memory has been allocated to such an address), and the program will crash.

- It is also possible that the number happens to be a correct address, but the address space is secured (e.g. it is reserved for kernel memory). In this case, the program will also crash.

Viewing the stack printf ("%08x %08x %08x %08x %08x\n");

- This instructs the printf-function to retrieve five parameters from the stack and display them as 8-digit padded hexadecimal numbers. So a possible output may look like: 40012980 080628c4 bffff7a4 00000005 08059c04

III PROTECTION MECHANISM

There are several numbers of approaches that have been developed to make buffer overflow attacks more difficult to achieve; still the effective coding, that is not vulnerable by using strong libraries and strong languages. Since the developers are not making safe software's, a number of h approaches have been implemented and try to protect against the occasional error:

Address Space Randomization Address Space Layout Randomization isrearranging the order of the elements of the stack. It means the attacker cannot rely overwriting system specific information by overflowing a variable. This type of protection is effective against automated attacks because there is no way for the automated program to know the way that the information is laid out, however the attacker can try the attack a with different ways and find a way to deploy the program.

For security purposethe data which involves arrange the positionsof key data areas, mainly it includes the position of librariesand base of the executable stack and heap, randomly in aprocess' of address space. It is based on the static values such as addresses which arecontaining specificoperands or pointers to the known location in a buffer on thestack.Virtual memory address on which variables and functions aresaved can make removing the buffer overflow more difficult,but it is also possible to eliminate it. [6]

Canaries The canaries in computer are planned to work in between the data and the system information that could be targeted. In this way if an attacker write the coding to make overflow a value and tries to change the return pointer it will overwrite the canary too. If this value is known and if it is found to have changed, when the subprocess returns then the process fails and it does not call the return again.

The contents of the canaries vary on which of the following types they are:

(i) Random Canaries (ii) Terminator Canaries(iii) Random XOR Canaries

Deep Packet Inspection This type of attacks originating from network and often come from the internet,in a remote computer it's possible to use Deep Packet Inspection to find the attack and stop the packet earlier it gets to the program to become vulnerable.

It is used to detect, at the network computer, the remote attempts to remove bufferoverflow attacks by using attack signature. Deep packetinspection (DPI) is able to block the packets which have thesignature of any type of known attack, or if No OPeration instructions (NOP) are detected, these are used in that timewhen the location of the exploits payload is may not static. DPIengines are placed at network boundaries at that place of securitycontrols and bandwidth is logically implemented.

Executable Space Protection In a lot of attacks an attacker is writing in the code that they want executed into the variable that is used by the program and then over writing the return address to point to this code. Executable Space Protection plans at fixing this by not allowing assured parts of the stack or heap to be executed. This protects against this one type of attack however it still allows an attacker to run existing code and change variables that they wish. This is another piece of code intended to protect the user from vulnerable programs but that does not protect the program from malicious users.

In this method stack and heap are protected from buffer overflow. It is a technique of buffer overflow attack protection whichprevents execution of code in heap and stack. An attacker mayuse buffer overflows to inserting impartment code into the program memory, but when we use Execution space protection, any attempt to execute the code will cause the exception.Executable space protection is an approach to buffer overflow protection which prevents execution of code on the stack or theheap. An attacker may use buffer overflows to insert arbitrarycode into the memory of a program, but with executable spaceprotection, any attempt to execute that code will cause an exception. New operating system of Microsoft support the executable space protection technique, it contains two tools which is used to secure from buffer overflow.

Buffer Shield and Stack Defender Return-to-libc attacks are not usually confined by Executable space protection, and also not protect any other or attack which is not rely on the implementation of the attacker's coding. However, on 64-bit systems using Address Space Layout Randomization , executable memory space protection makes more difficult to execute such attacks.

Pointer Protection Pointer protection is a planned process of encoding and decoding of pointers. In XOR a pointer with a known value when it is saved into the stack or heap and then XOR it once more when it is used by the program. This makes it so that if the attacker alters the pointers they have to XOR it with the known value and if they don't know they value and just input it the program would crash from the null pointer exception. As with most all of the security methods it is possible to overcome this protection even though this protection is one of the few that if used does protect the program from the user as well as the user from a susceptible program from the user.

It protects the program from the user because even if a user uses a debugger and examines the stack it will be hard for them to know that this is being used.

Use of safe libraries The libraries which is written and tested data types are automatically perform the buffer management, includes boundary checking, reduce the incident of buffer overflow attacks. String and arrays are two main types of data in the C, C++ languages in which buffer overflow unclearly occurs. Thus the safe libraries prevents buffer overflow which occurs by these type of data types.

IV RESEARCH CONCLUSION

In the systems, buffer overflow attack is done by the attacker is most vulnerable in the recent era. In this research work is categorizing the input validation attacks after that catalog of buffer overflow according to the generation of buffer overflow. The buffer overflow attack is the most famous vulnerable attack in past decade. Here we move towards to use all the security method to prevent buffer overflow, and the research work is contribute to solve the problems caused by the buffer overflow in stack and heap.

V REFERENCES

[1] Crispin Cowan. "Posting to bug mailing" List. http://geek-girl.com/bugtraq/1999_1/0481.html

[2] "Classification and Prevention Techniques of Buffer Overflow Attacks" Seema Yadav, Khaleel Ahmad and Jayant Shekhar Proceedings of the 5th National Conference; INDIACom-2011 Computing for Nation Development, March 10 – 11, 2011 Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

[3] https://sites.google.com/site/bufferattack/attacks/heap

[4] "On the Evolution of Buffer Overflows", MatthiasVallentin,vallentin@icsi.berkeley.edu May 20, 2007

[5] "Classification and Prevention Techniques of Buffer Overflow Attacks" Proceedings of the 5th National Conference; INDIACom-2011 Computing For Nation Development, March 10 – 11, 2011 Seema Yadav, Khaleel Ahmad and Jayant Shekhar.

[6] "A comparison buffer overflow prevention, implementation and weakness" written by: peter Silverman and Richard Johnson

[7] "Buffer Overflow Attack Vulnerability in Stack." P. Vadivel Murugan, and K. Alagarsamy, International Journal of Computer Applications 13.5 (2011): 1-2.

[8] 8 **"**A Lightweight Buffer Overflow Protection Mechanism with Failure-Oblivious Capability", Tz-Rung Lee1, Kwo-Cheng Chiu1, and Da-Wei Chang2 A. Hua and S.-L. Chang (Eds.): ICA3PP 2009, LNCS 5574, pp. 661–672, 2009. Springer - Verlag Berlin Heidelberg 2009

[9] "Averting Buffer Overflow Attack in Networking OS using – BOAT Controller", Vadivel Murugan.P K.Alagarsamy, International Journal of Computer Trends and Technology (IJCTT) – volume 4 Issue 7–July 2013

[10] "Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques" V.Selvi Dr.R.Umarani, International Journal of Computer Applications (0975 – 8887) Volume 5– No.4, August 2010

[11] "A comparative analysis of methods of defense against buffer overflow attacks". I. simon. http://www.mcs.csuhayward.edu/˜simon/security/boflo.html, January 2001.

[12] "Take Two Aspirin, and Patch That System – Now", J. McCarthy, *SecurityWatch*, August 31, 2001.

[13] "A Robust Kernel- Based Solution to Control-Hijacking Buffer Overflow Attacks" Li-Han Chen, Fu-Hau Hsu, Cheng-Hsien Huang, Chih- Wen Ou,Chia-Jun Lin And Szu-Chi Liu Journal Of Information Science And Engineering 27, 869-890 (2011).