Classification and Prevention Techniques of Buffer Overflow Attacks

1Seema Yadav, 2Khaleel Ahmad and Jayant Shekhar

CSE/IT Dept. S.I.T.E., SVSU,
Meerut-250002, India

1Seema_yadava@yahoo.com, 2khaleelamna@yahoo.co.in and 3jayant_shekhar@hotmail.com

ABSTRACT

In computer world there are many types of input validation attacks, in which “Buffer Overflow Attacks” is one of the most important types of attacks. Buffer overflow attacks create more dangerous to handle. Buffer Overflow is an anomaly where a programmer writes a data in a buffer, that overruns boundary of the buffer and overwrites the adjacent memory. This give the result erratic program types, such as memory access error, wrong results, a crash or break the computer security. In this paper, we discuss the classification of buffer overflow according to the generation, and prevention techniques of buffer overflow vulnerabilities.

KEYWORDS

Attacks; Buffer; stack; heap; Vulnerability; Input validation attacks

1- INTRODUCTION

The purpose of an attacker may range from intellectual gratification, denying valid users access to some internet websites, to stealing important and confidential information. Attackers exploit several types of vulnerabilities in the software systems driving computer and information systems. One of the most important and popular attack exploited by the attackers is the buffer overflow vulnerability. Buffer overflow is the results of some data or code area to be overwritten, thus destroying valid information [5]. Buffer overflows were understood as early as 1972, when the Computer Security Technology Planning Study (CSTPS) concerning the technique: “The code which is used to perform this function does not check the source and destination addresses in the proper way. It was one of several exploits used by the Morris worm to propagate itself over the Internet. A year later, in 1996, Elias Levy published it in a Phrack magazine the paper title is" Smashing the Stack for Fun and Profit", in this paper they give step-by-step knowledge about the overflow. After that earliest documented hostile exploitation of a buffer overflow was comes in 1988, name of that buffer overflow is internet worm. Fingered. Some years later: over 50% of all CERT advisories clarify that in 1997,28 attacks comes on network ,in which 16 attacks come under the buffer overflow, in 1998 .9 attacks come from 13 and .in 1999 six attacks come under this category of attack over 12.
Buffer overflows provides control to the hacker to launch the malicious codes on the target server. Malicious code includes some commands to

Steal the confidential information, password, alter system configurations, install backdoors, or launch other attacks.

There are two major internet worms have exploited buffer overflows to compromise a large number of internet systems. In 2001, in computer field come Code Red worm which exploited a buffer overflow in Microsoft's Internet Information Services (IIS) 5.0 and in 2003 the SQL Slammer worm compromised machines running Microsoft SQL Server 2000. Example of buffer over-flow is given in a program, with the help of this program we understand buffer overflow easily.

int main ()
{
    int buffer[150];
    buffer[200] = 150;
}
The above program which is given is a valid program, every compiler compile this program without any errors in each times. However, this program takes more memory to write data, which is larger than the allocation, which gives the result in unexpected way.

2- BACKGROUND

Before giving the detail over buffer overflow I give the basic details of input validation attacks. Input validation attacks are the attacks in which the attacker sends unusual input in the hopes of confusing the user. There are many types of input validation attacks which are given as follows. Buffer overflows, Canonicalization, cross-site scripting, SQL injection, HTML injection, and command execution. In this paper we discuss about buffer overflow and prevention technique. A buffer overflow is a condition in which data transferred to a buffer exceeds the storage capacity of the buffer and some of the data “overflows” into another buffer. Because buffers hold only a specific amount of data, when that capacity has been completed then the data has to flow some other place or buffer, which corrupted the data which is placed already in that buffer [1].

This is not wrong to saying that buffer overflows present one of the largest security problems today. The first worm attack on the Internet is Morris worm, was able to do so because of a buffer overflow. The overflow problem occurred because not enough memory was allocated before being passed to one of the standard string library functions.
The buffer overflow problems also occur in built-in functions of PHP, and JAVA language. For executing buffer overflow
attacks we give more and more data in to the input field. Perl is the best suited programming language for conducting buffer overflow attacks. Buffer overflow is also tested by sending the repeated request to the application and record response by the server.

3-CLASSIFICATION OF BUFFER OVERFLOW
Buffer overflows are mainly divided into multiple categories, based on both ease of exploitation and historical discovery of the technique. There is no any formal definition to the division of buffer overflow, it broken into three generations which is given as, first generation buffer overflows involve overwriting stack memory; second generation overflows involve heaps, function pointers, and off-by-one exploits; and finally, third generation overflows involve format string attacks and vulnerabilities in heap structure management. Taxonomy of buffer overflow according to generation is given as below.

3.1-FIRST GENERATION BUFFER OVERFLOW
First generation buffer overflows involve overflowing a buffer that is located on the stack. Stack overflow is also called stack smashing attacks

![Taxonomy of buffer overflow](image1.png)

3.1.1 Stack overflow-
Data or objects are placed (pushed) on the stack and inserted (popped) in a last-in and first out fashion. A stack is use to store automatic variables, these variables are declare only for that subroutine in which these are declared. Stack is used during subroutine linking in any program. Stack pushes the return address on the stack, when the subroutine is called. When stack returns, this received the save value from the stack and jumps to the finding address. Stack is accessed by the register that is called Stack Pointer, which indicate current top of the stack. In stack there are also another pointer (frame pointer) which is used to points to some fixed points in the frame structure, such as location of the return address. Stack buffer overflow bugs are caused when a program writes more data in a buffer located on the stack than there was actually allocated for that buffer. This almost always results in corruption of adjacent data on the stack, and in cases where the overflow was triggered by mistake, will often cause the program to crash or operate incorrectly.

Stack buffer overflow is generally known due to the Morris’ Internet Worm which comes in 1988 [12]. Stack smashing attack implementation program and diagram are given below.

![Overwriting data in to the Buffer](image2.png)
For example, the following program declares a buffer that is 128 bytes long. However, the program attempts to fill it with 256 bytes of the letter “A”

```c
int i;
void function(void)
{
    char buffer[128]; // create a buffer
    for(i=0;i<256;i++) // iterate 256 times
        buffer[i]='A'; // copy the letter A
}
```

Above program gives the attempt of stack overflow. Stack smashing attack are protected by two ways

A- Hardware boundary check and
B- Secure function call

Hardware boundary check gives better security as compare to secure function call.

### 3.2 SECOND GENERATION BUFFER OVERFLOW

The second generation of buffer overflows is known as heap, off-by-ones and function pointer overwrites. This is describing below as possible.

#### 3.2.1 Off-By-One Overflows

An off-by-one overflow specifies a one-byte buffer overflow. Such an error is made exceedingly often in loop conditions [6]. Programmers who attempt to use in safe functions in our program such as strncpy do not necessarily make their programs much more secure from overflows. Consider the following program where the programmer has mistakenly utilized “less than or equal to” in the place of “less than”.

```c
#include <stdio.h>
int i;
void vuln(char *foobar)
{
    char buffer [128];
    for (i=0;i<=128;i++)
        buffer[i]=foobar[i];
}
```

#### 3.2.2 Heap Overflows

A heap is memory that has been dynamically allocated. This memory is logically separate from the memory allocated for the stack and code. Heaps are dynamically created (e.g., new, malloc) and removed (e.g., delete, free). Heaps are generally used because the size of memory needed by the program is not known ahead of time, or is larger than the stack. The memories, where heaps are located, generally do not contain return addresses such as the stack. Thus, without the ability to overwrite saved return addresses, redirecting execution is potentially more difficult. However, that does not mean by utilizing heaps, one is secure from buffer overflows and exploitation.

There is a sample program with a heap overflow. The program dynamically allocates memory for two buffers. One buffer is filled with “C”’s. The other one is taken in from the command line. If one types too many characters on the command line, then an overflow will occur.

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

#### 3.2.3 Function Pointers

Another second generation overflow is the function pointers. A function pointer occurs mainly when callbacks occur in our program. If, in memory, a function pointer follows a buffer, there is the possibility to overwrite the function pointer if the buffer is unchecked. There are two types of function pointer such as, local and shared function pointer [14].

### 3.3 THIRD GENERATION BUFFER OVERFLOW

In third generation buffer overflows there are two types of overflows first is Format string attacks and second one isHeap management structure. Details are given as below.

#### 3.3.1 Format String Attacks

Format string vulnerabilities occur due to sloppy coding by programmer. A variety of C language means such as C and C++ functions allow printing the characters to files, buffers, and the screen. These functions not only place values on the screen, but can format them as well; the list which is given below contains ANSI format functions.

Format string danger discovered in June 2000. Examples of format string attacks are given below.

- BSD chpass: local root,
- wu-ftpd 2.* : remote root,
- Linux rpc.statd: remote root
- IRIX telnetd: remote root

Any function using a format string for printing and logging

#### 3.3.2 Printing function
4. BUFFER OVERFLOW PREVENTION TECHNIQUES

4.1 BUFFER OVERFLOW PROTECTION

Buffer overflow is using to detect most common buffer overflow attack by checking that when a function returns the stack has not been altered. If the stack altered then the program exist with a segmentation fault. There are some system which is used to buffer overflow protection is given as below.

1- Libsafe [11]- this is dynamic loaded library
2- StackGuard[14] –implemented as a GCC path

We find out the stranger stack protection by splitting stack in two parts such as Data and Function returns. The division of stack is done in fourth generation programming language, because of this reason this is not security based design decision. Because of this reason this is not a complete solution to buffer overflows, as sensitive data and other than the return address is still overwritten.

Libsafe is a library that provides a secure way to the functions, even if these functions are not available. When a buffer passes any arguments to any unsafe functions, for correcting the stack frame pointer follows the Libsafe.

Libsafe then check nearest return address distance, it make sure that given address are not overwritten at the time of function executing [11].

4.2-USE OF SAFE LIBRARIES

The problem of buffer overflow attack is same in C as well as C++ programming language. The reason of buffer overflow occurring is low level representation details of buffer as container for all data types. We avoid the buffer overflow to maintain a high degree of correctness in coding which also performs buffer management. In use of safe libraries we recommended to avoid standard library functions which is not check the bounds of array, such as gets, scanf and strcpy[14].

The libraries which is well written and tested abstract data types are automatically perform buffer management, includes bounds checking, reduce the occurrence of buffer overflow attacks. String and arrays are two main types of data in these languages in which buffer overflow mistily occurs. Thus the safe libraries prevents buffer overflow which occurs by these data types. Safe library implementations include "The Better String Library" Vstr and Erwin.

4.3. CHOICE OF PROGRAMMING LANGUAGE

The choice of programming language also effect to occurring of buffer overflow attacks. In 2008 more and more software are programmed in C and its derivative C++, C and C++provides no any type of built-in protection to access or overwrite the data in any part of memory, in others words in C and C++ there are no any checking done for data which is written in any array is under the boundary of the array or not. So when we use the programming language do very careful selection such as byte code languages, not a compilation language. The Java and .NET programming language are interpreted language and they have byte code (when Java compiler translate java program in an intermediate language is called java byte code) environment, these also require boundary checking for any type of array. Basically all the interpreted language doing work against buffer overflow attacks. Static code analysis is the technique which can remove many dynamic bounds and type checking [5][9],but bad implementation decreases the performance of any type of software, that call the buffer overflow attacks.

4.4. DEEP PACKET INSPECTION

Deep packet inspection is used to detect, at the network of computer, very basic remote attempts to remove buffer overflow attacks by using attack signature. Deep packet inspection (DPI) is able to block the packets which have the signature of any type of known attack, or if a No operation instructions (nop-sled) is detected, these are used in that time when the location of the exploit’s payload is slightly not static.

Deep packet inspection can be effective over buffer overflow attacks, denial of service attacks (Dos) attacks, and small percentage of worms that can be fit within a single packet. DPI engines are situated at network boundaries at that place security controls and bandwidth is logically implemented.

4.5. EXECUTABLE SPACE PROTECTION

Proceedings of the 5th National Conference; INDIACom-2011
It is a technique of buffer overflow attack protection which prevents execution of code in heap and stack. An attacker may use 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 the heap. An attacker may use buffer overflows to insert arbitrary code into the memory of a program, but with executable space protection, any attempt to execute that code will cause an exception.

New version of Microsoft Windows also support the executable space protection technique, there are two tools which is used to protect from buffer overflow

- BufferShield and StackDefender

Return-to-libc attacks are not generally protected by Executable space protection, and also not protect any other or attack which is not rely on the execution of the attacker’s coding. However, on 64-bit systems using ASLR, as de, executable space protection makes more difficult to execute such attacks.

4.6. ADDRESS SPACE LAYOUT RANDOMIZATION

Address space layout randomization (ASLR) is a security feature of computer data which involves arrange the positions of key data areas, basically including the position of libraries and base of the executable, heap, and stack, randomly in a process’ address space. ASLR is based on the theory that work on static values such as addresses which is contain specific operands or pointers to the known location in a buffer on the stack.

Virtual memory address on which variables and functions are retrieved can make removing of buffer overflow more difficult, but it is not impossible to remove it. ASLR also forces the attacker to doing the attempt on individual system, which foils the attempts of internet worms [15].

4.7. POINTER PROTECTION

Buffer overflow attacks work by manipulating the pointers (also stored addresses). PointGuard was proposed as a compiler-extension to prevent attackers from being able to reliably manipulate pointers and addresses. The approach works by having the compiler add code to automatically XOR-encode pointers before and after they are used. Because the attacker (theoretically) does not know what value will be used to encode/decode the pointer, he cannot predict what it will point to if he overwrites it with a new value. PointGuard [11] was never released, but Microsoft implemented a similar approach beginning in Windows XP SP2 and Windows Server 2003 SP1. Rather than implement pointer protection as an automatic feature, Microsoft added an API routine that can be called at the discretion of the programmer. This allows for better performance (because it is not used all of the time), but places the burden on the programmer to know when it is necessary.

5. CONCLUSION OF WORKS

In internet security buffer overflow is the more popular attacks done by the attacker. We presented in this paper classification and prevention techniques related to buffer overflow. In this paper we classify first input validation attacks then classify the buffer overflow according to the generation of buffer overflow. The buffer overflow attack is the most famous hacking technique in the past 10 years. Our approach is to uses all of the techniques used in prevention of buffer overflow. Our aim to design this paper is preventing internet security problem which is related to buffer overflow basically on heap and stack.

REFERENCES

[1]. Buffer-Overflow Protection: The Theory Krerk Piromsopa, Member, IEEE, and Richard J. Enbody, Member, IEEE.
[6]. On the Evolution of Buffer Overflows ,Matthias Vallentin,vallentin@icsi.berkeley.edu May 20, 2007
[8]. Classification of Buffer Overflow Vulnerability Monitors, Hosssain Shahriar and Mohammad Zulke-mdine, School of Computing,Queen’s University, Kingston, Canada.
[9]. Protecting global and static variables from buffer flow over attacks,Yves Younan Frank Piessens Wouter Joosen,Distrinet, Department of Computer Science,Katholieke Universiteit Leuven, Celestijn-enlaan 200a, B3001 Leuven, Belgium,E-mail:
[12]. Security Protection and Checking for Embedded System Integration against Buffer Overflow Attacks via Hardware/Software, Zili Shao, Member, IEEE,
[13]. Mutation-based Testing of Buffer Overflow Vulnerabilities, Hossain Shahriar and Mohammad Zulkernine, School of Computing, Queen's University, Kingston, Ontario, Canada.


[15]. A comparison a buffer overflow prevention, implementation and weakness written by: Peter Silverman and Richard Johnson.