Research on Classification of Memory Attack

Gang Shi[1], Jiangtao Ru[1]

[1]Institute of Information Engineering, CAS, Beijing 100093, China;

Keywords: memory vulnerability; memory attack; categorization; buffer overflow; vulnerability database; attack database.

Abstract. Exploits on memory vulnerability has existed for two or three decades, it's always an important problem how to prevent memory attack. Vulnerability patch can't resolve the problem fundamentally, we should think about how to improve the memory security mechanism of OS, such as DEP, ALSR, etc. This paper builds the bridge between memory attack and defense by categorizing memory attack and memory vulnerability, which lay a foundation for studying better memory security mechanism. Also database of memory vulnerability and database of memory attack is built on the categorization, which can map the reality of memory attack and defense. What's more, the categorization could be extended for the future situation.

1. Introduction

With the development and popularization of computer technology and network, the problem of computer security has become more and more important. To improve the security of the computer is to reduce vulnerabilities. Through the research on computer vulnerabilities, we can put forward better security mechanism to eliminate the threat to computer system.

Memory errors in C and C++ programs are among the oldest classes of software vulnerabilities. Memory error exploitation have been around for over 25 years and still rank among the top 3 most dangerous software. To date, the research community has proposed and developed a number of different approaches to eradicate or mitigate memory threat and their exploitation [1].These underlying programming languages introduce memory vulnerabilities, the most typical is the buffer overflow vulnerability. It's not reasonable that we use buffer overflow to represent this type of vulnerability. The use of memory attack concept is more reasonable. Here memory attack is defined as exploiting memory corruption. Memory vulnerability is defined as a vulnerability which provides opportunity for memory attack such as buffer overflows, formatted string, etc.

Clarifying the memory vulnerability and memory attack is important for the research of better memory protection security mechanism. The classification of the memory vulnerability and attack are too superficial at present, this paper classifies the memory attacks and vulnerabilities on a deeper level to build a bridge between the attack and defense.

2. The Evolution of Memory Attack and Defense

2.1 The Arm-race of Memory Attack and Defense.

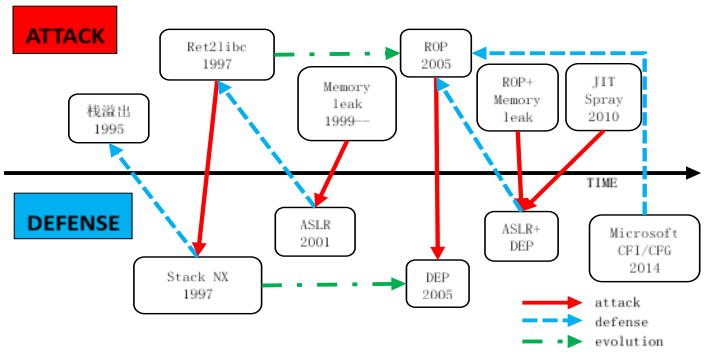

Fig.1 Arm-race of memory attack and defense

Memory errors is found as early as in 1972 [2].In 1995, Thomas Lopatic describes a specific method of stack overflow exploitation[3], then, the harm of stack overflow attack is more and more serious, the defense also has gradually been studied. In 1997, the non-executable stack (NX) is put forward, in the end all data segment non-executable has been introduced (In 2005 Microsoft introduces DEP (Data Execution Protection) [4]), which results in that the shellcode in the Data segment cannot be executed. Shortly after the mechanism of NX (No eXecute) is put forward, Solar Designer describes a new method of ret2libc [5], using the original code of function library in the system to bypass the NX protection. In 2005, Krahmer first put forward ROP (Return Oriented Programming) [6, 7] using instruction gadget on the basis of Ret2libc.ROP makes the mechanism of NX failure, at the same time, makes the defender harder to distinguish between normal code and malicious code, which increases the difficulty of the defense. Due to the address of injection shellcode and short instruction gadgets that constructs ROP are basically knowable, in 2001 PaX developed ASLR (Address Space Layout Randomization) [8], introducing the Address randomized to fend off attacks. With the wide use of ASLR, new means of attack against ASLR appeared. In 2004, SkyLined proposed Heap Spray [9], the repeated construction of shellcode in the heap makes the shellcode address can be predicted on some level. For now, the memory leak [10] is the biggest threat to ASLR, which will leak the layout of code and data in memory. Also memory leak can leak sensitive information in memory, such as password, keys, etc. The earliest vulnerability of memory leak were found in 1999, the vulnerability of format string [11], and now the problem is still very serious [12].

In order to cope with the more and more serious memory attack, a lot of memory security features are introduced in the operating system, such as DEP/ASLR, Stack Cookies, heap and SEH protection, etc. DEP and ASLR are put forward with the consideration of the common characteristics of all attacks, which are successful mechanism. Particularly DEP is supported by the hardware, which makes it more efficient. New attack methods such as JIT Spray [13, 14] makes the combination of DEP and ASLR failure. The combination of DEP and ASLR also can be bypassed by the combination of ROP and memory leaks. ASLR is not enough to defend ROP. ROP changed the order of the original instruction sequence, which destroys the Integrity of the Control Flow. Based on this, CFI (Control Flow Integrity) concept was put forward in 2005 [15].Because of the consideration of efficiency and compatibility, CFI is not yet widely used. In 2014, Microsoft adopted CFG (Control Flow Guard) (a kind of weakening CFI) [16], which effect has large limitation. As the threat of ROP attack is still serious, how to resist the ROP by CFI will be still the focus of research in the next few years.

2.2 The Purpose of this Paper.

From the history of memory attack and defense, we can see the mutual promotion of memory attack and defense. But we can't clearly see the connection between the attack and defense. An important purpose of this paper is to establish classification of the memory attack and memory vulnerability, cleaning up the link between the attack and defense, which lays the foundation for the research of better memory safety mechanism.

3. Classification of Memory Attack

Memory attack is just malicious use of the critical data in the memory reaching the purpose of directly or indirectly executing malicious code. Here, the attack point is defined as the critical data in memory, such as function pointers, data pointers, common data, etc. On the basis of the definition of attack point, security mechanism, exploiting method, technical means and trigger condition, the concept of attack mode is defined.

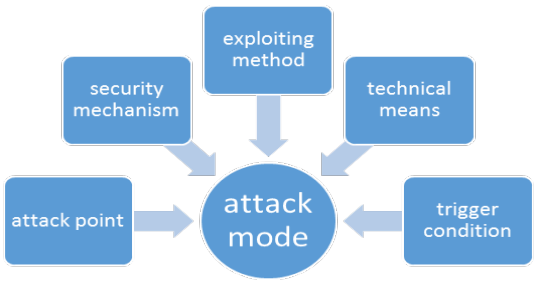

Fig.2 Definition of attack mode

Attack point: the critical data in the memory that can be exploited by the attacker, such as function pointer, data pointer, common data, etc.

Security mechanism: the security mechanisms of the OS that mitigate the memory attack, such as DEP, ASLR, stack cookie, etc.

Exploiting method: the methods that corrupt the attack point, such as tampering with data, data falsification, etc.

Technical means: the necessary technical means that make the memory attack success, such as Ret2libc, ROP/JOP, heap spray, etc.

Trigger condition: the condition on which the attack point can be attacked, such as the management of stack and heap, the management of function, etc.

Attack mode: the combination of attack point, security mechanism, exploiting method, technical means form the definition of attack mode. According to the definition of attack mode, we can distinguish kinds of memory attack.

According to the definition of attack mode, the memory attack can be classified as shown in the figure 3 in details:

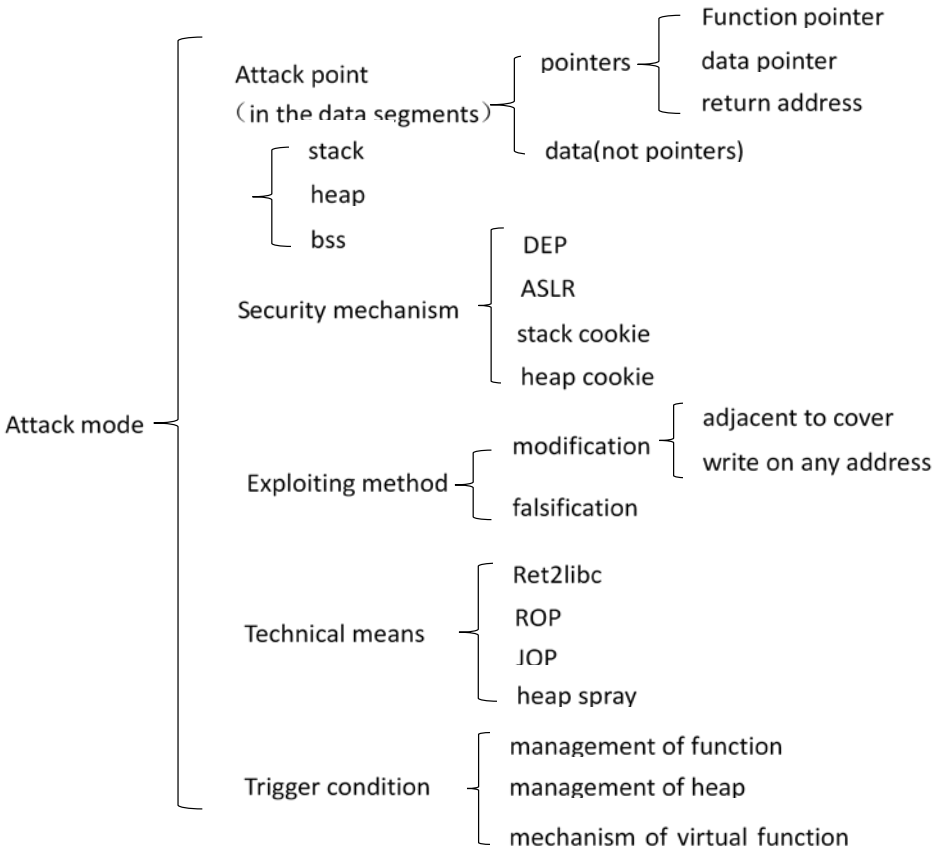

Fig.3 Details of attack mode

According to the details of attack mode above, the traditional memory attack can be summarized as shown below:

Tab.1 Summary of classic memory attack

Memory attack	Attack point	Security mechanism	Exploiting method	Technical means	Trigger condition
classical stack overflow	return address		adjacent cover		management of function
stack overflow bypassing DEP	return address	DEP	adjacent cover	ROP	management of function
heap overflow	data pointer		adjacent cover		management of heap
Use After Free	function pointer		falsification		mechanism of virtual function

This classification of memory attack can be extended in the future. As the research being deeper and deeper, the details of this classification of memory attack will be more complex and comprehensive. Here, the memory vulnerability is defined as the vulnerabilities which provide opportunities for memory attack. So we get the definition of memory attack and memory vulnerability at the same time.

From the above, the memory attack includes five aspects. Mitigating every aspect can mitigate the memory attack, which is the basis of the research of better memory security mechanism, for example, stack cookie prevents the adjacent cover, and integrity check of heap head prevents the malicious use of management of heap. Then, the bridge between memory attack and defense is built. Based on this, we can put forward more and more better memory mechanism. In reality, memory attack is inexhaustible. Under this classification of attack mode, the means of memory attack is exhaustible. At the same time, this classification can reflect the reality well.

4. The Establishment of the Memory Vulnerability and Attack Database

The memory vulnerability database and memory attack database are built on the base of the definition of memory vulnerability and memory attack, which provides the experimental environment for the study of better memory mechanism. As shown in figure 4, the framework of attack database is written by python which is a good language for writing attack program and the vulnerability database is compiled by GCC running on Ubuntu12.04.

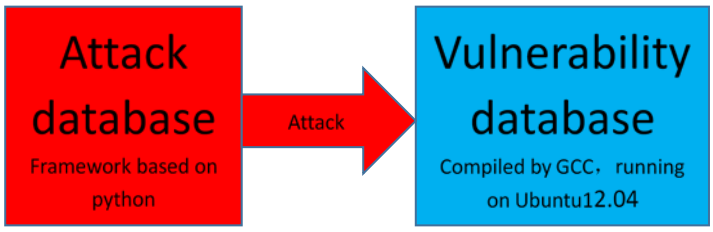

Fig.4 Attack database and vulnerability database

Here taking classic buffer overflow as example, how to construct memory vulnerability and attack program are explained as below.

Program with vulnerability of buffer overflow will first open a port to monitor request from the client, accepts data from the client, and then calls functions as shown in figure 5 (the vulnerability of buffer overflow is obvious).

```
void testVul(char *buff)
{
        char str[10];
        strcpy(str, buff);
}
```

Fig.5 Function with vulnerability of buffer overflow

Attack program which is written by python as shown in figure 6, first connects the server with the

vulnerability of buffer overflow, and then sends malicious data to exploit the vulnerability. After successful attack, a shell will be open as a symbol of success.

```
#coding:UTF-8
'''StackOverflow - RetOverwrite (DEP - off, ALSR - off)'''

import socket
import struct
import sys
sys.path.append("..")
import payloads
from payloads import *
# from pwn import *

class exploit(object):
    'exploit1'
    def __init__(self, options):
        self.options = options
        # self.info = {'ret':0xbffff180, 'offset':22}
        self.info = {'ret':0x08048814, 'offset':22}
    def test(self):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((self.options['dIP'], int(self.options['dPort'])))
        payload = eval(self.options['payload']).payload()
        ret_addr = struct.pack('<I', self.info['ret'])
        # ret_addr = p32(self.info['ret'])
        print ret_addr
        # print hex(ret_addr)
        data = 'A' * self.info['offset'] + ret_addr + payload.data
        print data
        s.send(data)
        s.close()

if __name__=='__main__':
    pass
```

Fig.6 Attack program

When writing the vulnerabilities and attack program, we should modularize the parts as much as possible, which makes the extension easier. All the vulnerability programs is put into a framework and all the attack programs is also put into the other framework.

After establishing vulnerability database and attack database, we can simulate the memory attack in reality, building an experiment platform for the research of better memory security mechanism.

5. Summary

At first, this paper present a brief history of memory attack and defense. Then, the concept of attack mode is defined on the base of the attack point, security mechanism, exploiting method, technical means and trigger condition. At the same time, the memory vulnerability database and memory attack database are built on the basis of the definition of attack mode and memory vulnerability. After this, we built the bridge between the memory attack and memory defense, which lay a foundation for the research of better memory mechanism.

Future work mainly focused on two things. First, establish the integrity mapping relationship of memory vulnerability between the database and the reality. Second, establish the integrity mapping relationship of memory attack between the database and the reality.

References

[1]. Van der Veen V, Cavallaro L, Bos H. Memory errors: the past, the present, and the future[M]//Research in Attacks, Intrusions, and Defenses. Springer Berlin Heidelberg, 2012: 86-106.

[2]. Anderson J P. Computer Security Technology Planning Study. Volume 2[R]. ANDERSON (JAMES P) AND CO FORT WASHINGTON PA, 1972.

[3]. T. Lopatic, “Vulnerability in NCSA HTTPD 1.3,” February 1995.

[4]. Microsoft, A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003[J].2006.

[5]. Designer S. Getting around non-executable stack (and fix)[J]. 1997.

[6]. Krahmer S. x86-64 buffer overflow exploits and the borrowed code chunks exploitation technique[J]. 2005.

[7]. Prandini M, Ramilli M. Return-oriented programming[J]. Security & Privacy, IEEE, 2012, 10(6): 84-87.

[8]. Team P. Design and implementation of PAGEEXEC[J]. 2000.

[9]. Sotirov A. Heap feng shui in javascript[J]. Black Hat Europe, 2007.

[10]. Planet C. A eulogy for format strings[J]. Phrack (Nov. 2010), 2010.

[11]. Twillman T. Exploit for proftpd 1.2. 0pre6[J]. Posting to Bugtraq Mailing List, http://seclists. org/bugtraq, 1999.

[12]. Serna F J. CVE-2012-0769, the case of the perfect info leak[J]. 2012.

[13]. Blazakis D. Interpreter Exploitation[C]//WOOT. 2010.

[14]. Wei T, Wang T, Duan L, et al. Secure dynamic code generation against spraying[C]//Proceedings of the 17th ACM conference on Computer and communications security. ACM, 2010: 738-740.

[15]. Abadi M, Budiu M, Erlingsson U, et al. Control-flow integrity[C]//Proceedings of the 12th ACM conference on Computer and communications security. ACM, 2005: 340-353.

[16]. MJ0011, Windows 10 Control Flow Guard Internals[J]. 2014.