CTF 中的 PWN 题目探析

Analysis of PWN in CTF

钱泰丞　王影 *

Taicheng Qian　Ying Wang*

山东工商学院　中国·山东 烟台 264005

Shandong Technology and Business University, Yantai, Shandong, 264005, China

摘　要：CTF（Capture The Flag）比赛起源于 1996 年 DEFCON 全球黑客大会，以代替之前黑客们通过互相发起真实攻击进行技术比拼的方式。该比赛发展至今，已经成为全球范围网络安全界流行的竞赛形式。PWN 是 CTF 比赛中专门设计出来模拟现实中服务器被入侵情况的比赛项目。该项目考验选手漏洞的挖掘和利用的能力，利用反汇编器分析二进制文件，得到“伪代码”，寻找其中的漏洞，然后编写漏洞利用程序脚本。在比赛中完成模拟入侵系统，完成攻击后取得相应的分数。论文就 PWN 问题进行深度研究，对于保证服务器安全、网络空间安全具有重大意义。

Abstract: CTF(Capture The Flag) competition originated from DEFCON Global Hacking Conference in 1996, to replace the previous way in which hackers competed with each other by launching real attacks. Up to now, this competition has become a popular competition form in the global network security. PWN is a competition specially designed in CTF competition to simulate the intrusion of servers in reality. This project tests the player's ability to exploit and exploit vulnerabilities. By analyzing binary files with disassembler, we can get "pseudo-code", find the vulnerabilities in it, and then write the exploit program script. Complete the simulated intrusion system in the competition, and score corresponding points after the attack. In this paper, the in-depth study of PWN is of great significance to ensure the security of servers and cyberspace.

关键词：程序逆向；漏洞利用；PWN；CTF

Keywords: program reversal; exploitation of loopholes; PWN; CTF

基金项目：教育部产学合作协同育人项目《现代学徒制在网络空间安全专业的应用与推广》（项目编号：201902039011）。

DOI: 10.12346/sde.v4i6.6579

1 引言

CTF（Capture The Flag）中文一般译作“夺旗赛”，是网络安全领域一种较为流行的信息安全竞赛形式。该竞赛起源于 1996 年 DEFCON 全球黑客大会，以代替之前黑客们通过互相发起真实攻击进行技术比拼的方式。参赛团队通过进行攻防对抗、程序分析等形式，率先从主办方给出的比赛环境中得到一串具有一定格式的字符串或其他内容，并将其提交给主办方，从而夺得分数。这段字符串或者其他内容被称为“Flag”[1]。仅在 2013 年，全球举办了超过五十场国际性 CTF 赛事。近年来，中国 CTF 赛事在国内各大高校、信息安全界也掀起了热潮。

笔者大二与大三期间参加本专业 CTF 兴趣小组，并多次参加比赛，在多种方向进行尝试后选择了难度较大具有挑战性的 PWN（攻破）方向。

2 PWN 简介

PWN 在黑客俚语中代表着攻破、破解、获取权限。该词汇可以作为一个动词被使用，如“PWN”掉了一个程序，

【作者简介】钱泰丞（2001-），男，回族，中国山东淄博人，在读本科生，从事信息管理与信息系统（网络安全）研究。
【通讯作者】王影（1976-），女，中国辽宁鞍山人，硕士，副教授，从事信息管理与信息系统研究。

“PWN”掉了一个服务器，“PWN”掉了一台电脑等 [2]。在 CTF 比赛中它代表着溢出类的题目，主要考查参赛选手对漏洞的利用能力其中常见类型溢出漏洞有整数溢出、栈溢出、堆溢出等。

一句话总结 PWN 的核心：漏洞的挖掘和利用。

PWN 研究的漏洞不是与 WEB 漏洞、PHP 漏洞等类似在高级的应用层编译代码出现的缺陷，而是在数据链路层，是已经编译成机械码的漏洞、二进制与程序交互中出现的破绽。

一般比赛中，主办方会在服务器上某个端口挂一个二进制服务，并把对应的二进制程序交给选手分析。选手首先通过利用反汇编器分析二进制文件，得到伪代码，寻找其中的漏洞，然后编写漏洞利用程序脚本；然后运行这个脚本攻击远程服务器，得到服务器的 SHELL（交互模式），至此，攻击任务完成，选手获得服务器的远程控制权；在比赛中则需要找到服务器上的 FLAG 文件并将其提交给举办方来拿到分数。这就是一个 PWN 问题的解决过程。真正的攻击过程用时不长，但是漏洞挖掘和脚本编写需要耗费大量的时间精力。

PWN 题型的出题思路大多源于实践，一般是将出现过或者将来可能出现的漏洞的情况进行简化，形成小规模便于分析的程序，考察解题人挖掘漏洞和利用漏洞的能力。

3 PWN 常用工具

3.1 IDA 交互式反编译器（Interactive Disassembler Professional）

这是一款功能很丰富很复杂很强大的工具，应用十分广泛，属于逆向辅助工具。

3.2 OD（OLLYDBG）

一个新的动态追踪工具，将 IDA 与 SOFTICE 结合起来的思想，Ring3 级的调试器，已代替 SOFTICE 成为当今最为流行的调试解密工具了。同时还支持插件扩展功能，是目前最强大的调试工具。

3.3 PWNTOOLS

一个 CTF 框架和漏洞利用开发库，由 RAPID 设计，模块很丰富，方便使用者快速开发 exploit，属于编写利用工具。

4 PWN 题目实践

IDA 用于静态分析，OD 用于动态分析。取得题目的第一步是对题目的二进制代码进行反编译，从而将其变为可视的代码。PWN 的先导能力是逆向，对程序进行逆向操作主要是便于分析人员看懂程序，但是逆向出的代码并不与编译前完全一样，只是实现的功能相同，一般被称为伪代码。很多情况下，PWN 题型对于逆向能力的要求不是特别高，有时只要分析部分代码即可。在个别情况下，文件也会“加壳”，即为可执行程序资源压缩，压缩后的程序可以直接运行；另一种常用的方式是在二进制程序中植入一段代码，在运行的时候优先取得程序的控制权，之后再把控制权交还给原始代码，这样做的目的是隐藏程序真正的 OEP（入口点，防止被破解）。大多数病毒就是基于此原理，这种技术也常用来保护软件版权，防止软件被破解 [3]。

脱壳有以下三种方法：

①自动静态脱壳：运用脱壳工具即可，相当于解压缩，该方法最为方便快捷。

②自动动态脱壳：运行可执行文件，让脱壳存根的文件脱出原始的可执行文件。

③手动脱壳：找到加壳算法程序，手动分析源程序进行脱壳。

脱壳后先用 file 命令看一下文件为 32 位还是 64 位，再使用 CHECKSEC 指令查看可执行文件的保护措施开启情况，运行一下查看文件运行流程，之后用 IDA 和 OD 打开。

分析程序主要是为了厘清程序内部的逻辑关系，以便于分析程序的脆弱性（即查找漏洞点）并构造出触发的条件。漏洞查找通常有一定的方法，这依赖攻击者的经验积累和知识储备。

获取 SHELL 最简单的方法是利用程序员在编写程序时给自己留的“后门”程序，只要控制服务的执行流到这个“后门”地址，即可快速获取 SHELL。如果没有后门，那么就要想办法运行系统命令：

system(“/bin/sh”)；

需要手动向程序中输入。

5 漏洞原理

PWN 漏洞类型可主要分为栈漏洞、堆漏洞、格式化字符穿漏洞、整型漏洞、逻辑漏洞等。很多时候，这些漏洞类型需要相互结合，构造出复杂条件。同样这些漏洞类型的利用也可以互相转化，以便写出更好更快地利用脚本。

想要了解漏洞的存在原理就要先了解可执行文件的生成过程：编译，由 C 语言等高级语言生成汇编代码。汇编，再由汇编代码生成机器码。链接，将多个机器码的目标文件链接成一个可执行文件。其中机器码就是二进制代码，一般以十六进制表示。传统的缓冲区溢出涉及二进制编码的漏洞统称为二进制漏洞，根据缓冲区所处的不同内存空间以及分配方式的不同，缓冲区溢出即为栈溢出和堆溢出。

5.1 栈溢出漏洞

栈是一种基本的数据结构，是由编译器自动进行分配、释放的。栈遵循先进后出的规则，生长方向为从高向低生长，也就是地址由高到低栈中存放的数据对于要调用的子函数来说，存放子函数的参数、返回地址、局部变量等重要信息。函数可以通过栈来方便的对局部变量进行读取，栈会用到以

下寄存器。

ESP 用来储存栈顶地址，在压栈和退栈时候会变化 EIP 存储着下一条指令的地址，每执行一条指令，该寄存器变化一次。

EBP 存储着当前函数栈底的地址，栈低通常作为基址，可以通过栈底地址和偏移相加减来获取变量地址（很重要）。调用一个函数需要以下步骤：首先将参数放到栈上，然后将当前的值放到栈上进行保存，当作函数调用之后的返回地址，最后执行子函数的函数体。

程序在调用子函数的时候（可以通过 OD 等工具看汇编代码），会先将 EBP 压栈调整栈结构，在栈上开辟一段空间即子函数的空间。在实际的函数调用过程中，局部变量是不断变化的。需要注意的是：栈是向低地址的方向生长的，但是局部变量是向高地址生长的。

由于变量是向下“吞噬”的，就会产生一个问题：当变量向下“吞噬”的时候，“一不小心”把下边 EBP 和 EIP 的空间也“吞噬”了，这个时候就会产生栈溢出现象。EIP 的值是用来函数执行完之后返回用的，但是在被“吞噬”之后，栈空间的值已经发生改变，就会返回错误。此时，攻击者就控制了程序的 EIP，从而为所欲为。

什么是造成栈溢出的主要原因呢？代码编写过程不规范，例如：在执行数据复制的操作的时候，不检查源数据的长度直接进行复制，很有可能会因为目的空间不够而导致这种情况的产生。

5.2 整型溢出漏洞

整形溢出漏洞主要是指在计算机中的数据类型问题导致的溢出漏洞，整型作为一种数据类型，不仅仅存在正负的问题还存在数据范围的问题甚至还包括不同整型数据之间的赋值和转换问题，而在其中容易产生一系列不严谨的算法，可能会导致溢出漏洞的产生。

5.3 存储溢出漏洞

存储溢出相对比较简单易懂，就是由于使用不同的数据类型来存储整型数造成的。举个例子：

下边这个代码段，在实际的赋值过程中：作者想要达到的目的可能是将 a 的值赋给 b，但是在实际的输出过程中，我们可以看到，实际的输出结果并不是 a 的值，输出结果如下：

```
int a=0x10000;
short b=a;
printf(“%d”,b);
```

输出值 b 得 0，并不是赋予它的 0x10000，这是因为，不管哪一种数据类型都是由存储范围的限制的，int 型是 32 位长度的，但是 short 是 16 位的数据长度，因此在进行赋值操作的时候，short 型的数据没办法将 int 型数据的所有位数全盘接受，这样就会导致我们之前说的存储溢出漏洞。

5.4 运算溢出漏洞

运算溢出，顾名思义，就是在整型数据在运算过程中导致的溢出。数据类型的范围长度以及范围是有限的，因此在定义数据类型的应谨慎使用各类数据类型，防止超出数据类型的范围。但是由于注意力往往集中在赋值以及定义变量，定义之后，对该数据的各类运算是否会超出数据类型的范畴通常不关注，因为大多情况下关注的都是运算的结果。这样很多漏洞都是通过这类型的溢出导致的。

5.5 符号问题导致的漏洞

整型数据分为无符号整型数据和有符号整型数据，如果在编写程序代码的时候，忽略符号问题造成的影响，那么就极有可能造成安全事故。

6 漏洞利用的攻击脚本

例如：在IDA中分析伪代码发现了一个程序员留下的“后门”函数如下：

```
INT get_shell()
{
 System(“/bin/sh”);
 Return 0;
}
```

这个 GET_SHELL 函数只要执行，就可以拿到服务器的 SHELL，我们利用这个存在栈溢出漏洞的函数来想办法执行 GET_SHELL。

```
INT vulnerable（ ）
{
 Char buffer[8];
 Gets(buffer);
 Return 0;
}
```

在这个函数中，声明了一个长度为八的数组，也就是变量“buffer”, 在动态 OD 中可以看到它存储在一个栈中，可以看到 ESP,EAX,EBP 三个寄存器的位置，在栈底也就是 EBP 的后面紧跟着 return 返回地址。

我们只要写入超过数组长度的字符，想办法覆盖掉 EBP 和后面跟着的返回地址，把这个地址改成 GET_SHELL 函数的地址，我们的攻击就完成了。下面根据这个思路来编写攻击脚本。

```
From pwn import *（导入 python 中的 pwn 库）
Io = process(“./ret2text”);（打开与文件的交互）
Io.recvline(); ( 接受该程序输出的内容 )
Payload = b’A’* 16 + b’BBBB’+ p32(0x8048522)
```

(首先填充十六字节垃圾数据填满栈空间，另外再加上四字节的数据覆盖掉 EBP 就到了返回地址位置，把返回地址填写成需要的目的函数地址)

Io.sendline(payload); (发送构造的数据)

Io.interactive(); (打开交互模式)

运行完成此脚本就会进入目标的 SHELL 模式，攻击就完成了。

这个脚本是对针对本地文件的攻击，如果打远程服务器上的程序的话，只需要把 process 函数改成 remote 函数即可 [4]。

7 PWN 问题探索的意义

没有绝对安全的系统，没有不为人控的程序，数据和操作系统本身可以被任何人出于任何目的读取和利用。PWN 这一方向始终提醒着程序开发人员保留着对程序基本原理的重视。PWN 题目可以考察程序逆向能力、漏洞查找能力、利用代码编写能力等。

熟能生巧，经过反复训练，新手可以解决接近于现实中的难题而不再拘泥于应试技巧。

8 结语

我们不能只使用便利的现代图形化界面操作系统，而忘记了它的基本原理。前人给我们留下了好用的工具，如果我们只会使用它，而不了解它是怎么做出来的，那么我们进步的脚步就会轻飘飘的，落不稳，我们应当扎实的继承前人的知识，走好我们进步的每一步。

PWN 的学习研究，入门难，提高难，需要基础知识非常扎实。同时PWN涉猎的范围足够大，有计算机原理、内存存储、python 脚本、Linux 的 shell 基础，保证了技术的全面性。

参考文献

[1] 李艺.FLAPPYPIG战队CTF特训营[M].北京:机械工艺出版社,2020.

[2] 九层台PWN总结[Z].CSDN博客,2018.

[3] Magicknight.从零开始学习PWN[J].简书,2018(8):2.

[4] 董付国.PYTHON程序设计基础[M].北京:清华大学出版社,2015.