

## 渗透测试二

### 0x01 目的

针对具体的漏洞，实现 Metasploit 漏洞利用模块，理解漏洞发现、漏洞利用的过程。

### 0x02 漏洞介绍

<https://nvd.nist.gov/vuln/detail/CVE-2008-1498>

Stack-based buffer overflow in the IMAP service in NetWin Surgemail 3.8k4-4 and earlier allows remote authenticated users to execute arbitrary code via a long first argument to the LIST command.

<https://www.exploit-db.com/exploits/5259>

NetWin Surgemail 0DAY (IMAP POST AUTH) Remote LIST Universal Exploit

下载程序：

[https://www.exploit-db.com/apps/7ca26524988a95dd8ea2265fa387a852-surgemail\\_38k4\\_windows.exe](https://www.exploit-db.com/apps/7ca26524988a95dd8ea2265fa387a852-surgemail_38k4_windows.exe)

漏洞利用代码：

<https://www.exploit-db.com/exploits/5259>

### 0x03 SEH 漏洞利用原理

SEH (Structured Exception Handling) 是 Windows 提供的异常处理机制。通常所用的异常处理（比如 C++ 的 throw、try、catch）都是编译器在操作系统提供的异常处理机制的增强版本。

#### 1) 参考资料

Exploit 编写教程第三篇：基于 SEH 的 Exploit

对应英文资料：Exploit writing tutorial part 3 : SEH Based Exploits

<https://www.corelan.be/index.php/2009/07/25/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-3-seh/>

Exploit 编写教程第三篇 b: 基于 SEH 的 Exploit-又一个实例

对应英文资料: Exploit writing tutorial part 3b : SEH Based Exploits – just another example

<https://www.corelan.be/index.php/2009/07/28/seh-based-exploit-writing-tutorial-continued-just-another-example-part-3b/>

How to Write a Metasploit Module – Part 2 (重点参考)

<https://cybercruddotnet.wordpress.com/2012/05/22/how-to-write-a-metasploit-module-part-2/>

Metasploit 渗透测试指南: 第 14 章 创建你自己的渗透攻击模块 (重点参考)

WRITING OUR OWN IMAP FUZZER TOOL

<https://www.offensive-security.com/metasploit-unleashed/simple-imap-fuzzer/>

WRITING AN EXPLOIT MODULE:

<https://www.offensive-security.com/metasploit-unleashed/writing-an-exploit/>

<https://www.offensive-security.com/metasploit-unleashed/shell/>

POP/POP/RET 的作用:

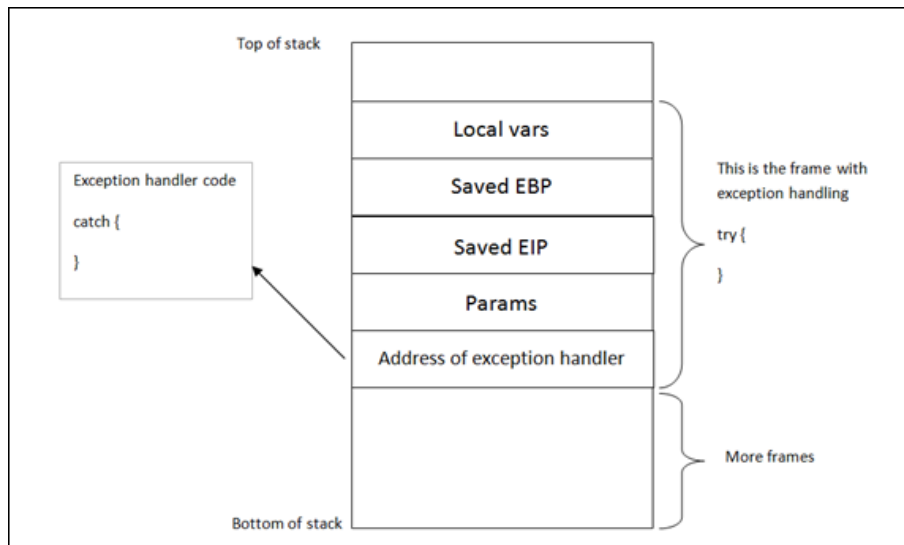
The need for a POP POP RET instruction sequence

<https://dkalemis.wordpress.com/2010/10/27/the-need-for-a-pop-pop-ret-instruction-sequence/>

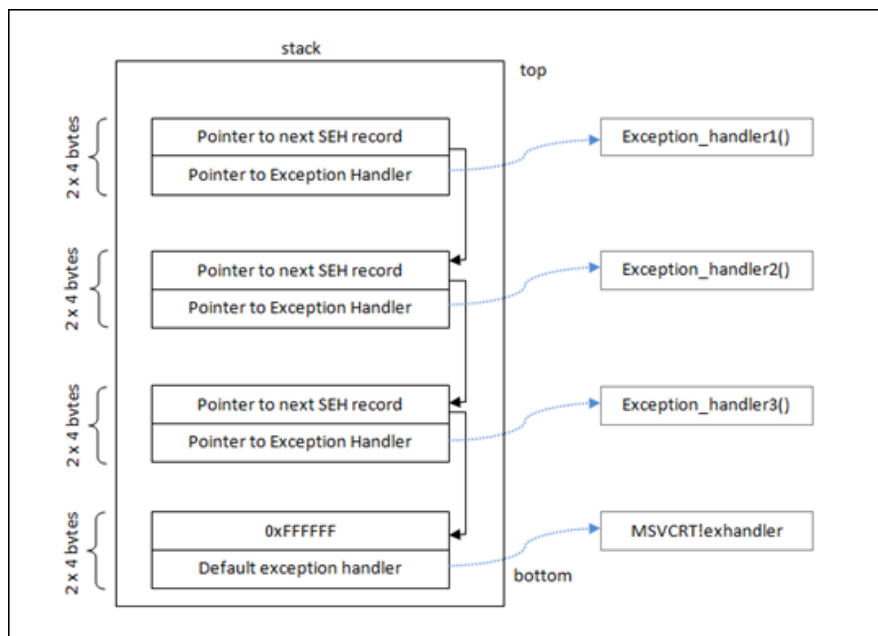
对应中文翻译: <https://blog.csdn.net/youb11/article/details/45113011>

## 2) Exception Handling

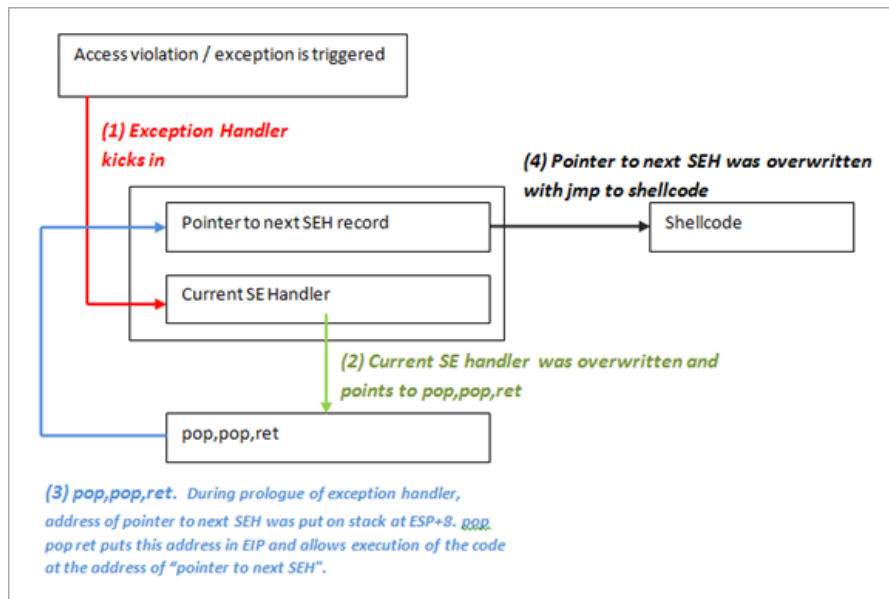
Stack 结构



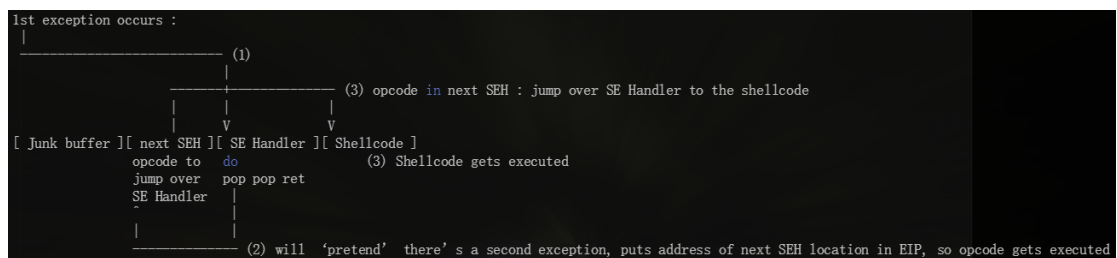
Exception Handler 结构:



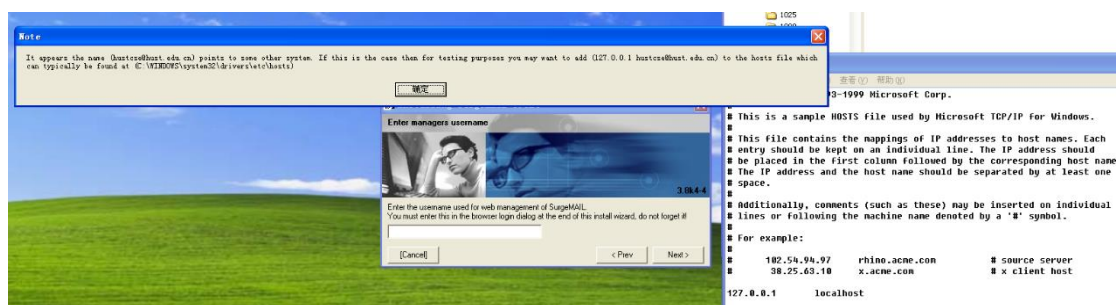
Exception Handling 过程:



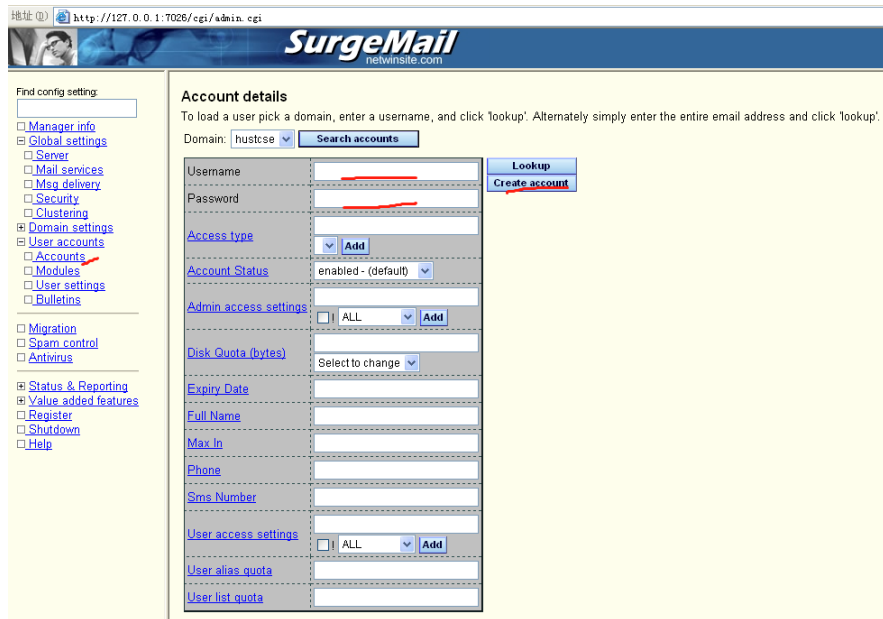
可能的恶意输入：



## 0x04 surgemail 程序安装



创建一个测试用户：



实验过程中需要重启服务（比如添加账号后）：

```
net stop surgemail
```

```
net start surgemail
```

查看链接情况：

```
netstat -na
```

surgemail 管理界面，可以添加用户，或者查询服务是否在线：

<http://127.0.0.1:7026/>

## 0x05 实验要求

按照“渗透测试指南：第 14 章” 进行实验。

1) 漏洞发现：

在 Kali 端，通过构造简单的 IMAP 协议 Fuzz 测试器；

```
sudo vi /usr/share/metasploit-framework/modules/auxiliary/fuzzers/fuzz_imap.rb
```

需要更改参考的 ruby 脚本最前面部分：

1. 删除 `require 'msf/core'`;

2. `class Metasploit3 > Msf::Auxiliary` 改成如下形式

```
class MetasploitModule < Msf::Auxiliary
  include Msf::Exploit::Remote::Imap
  include Msf::Auxiliary::Dos
```

修改脚本后，可以通过 reload 重新加载

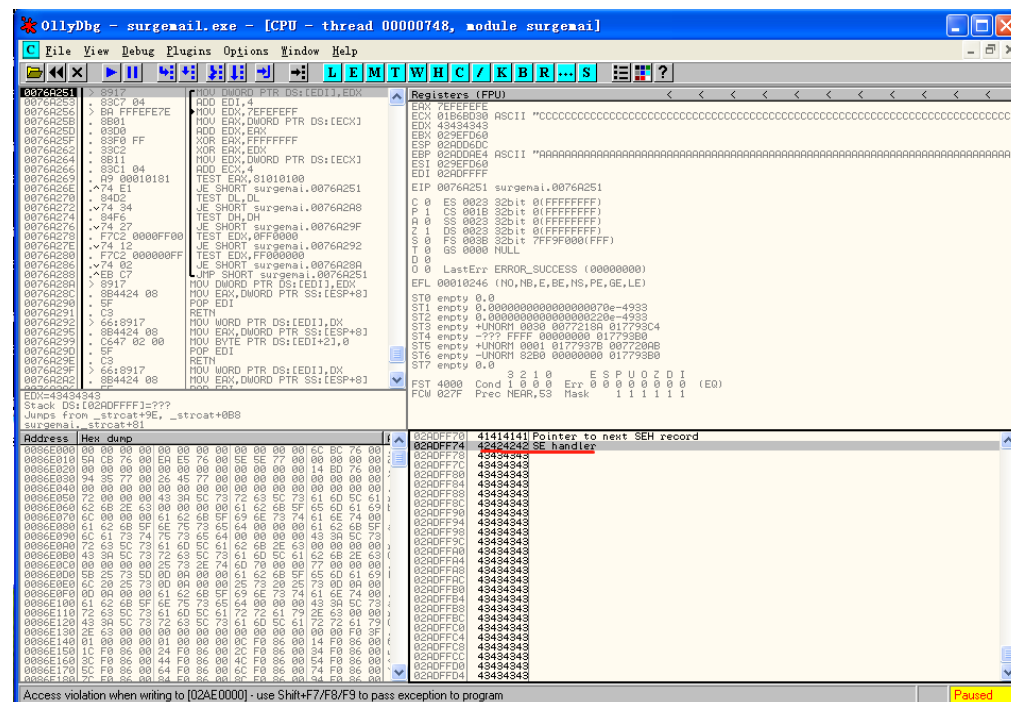
reload auxiliary/fuzzers/fuzz\_imap

## 2) 漏洞利用

包括两个任务：定位 SEH handler、创建恶意输入

在 Windows 端，使用 OllyDBG (Immunity Debugger)。

定位 SEH handler:



/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -q 684E3368 -l 11000

## 创建恶意输入：

两种方式：

[一段任意的缓冲区填充 | NOP 空指令滑翔区 | Shellcode | [近跳转](#) | [短跳转](#) | PPR]

[一段任意的缓冲区填充 | [短跳转](#) | PPR | NOP 空指令滑翔区 | Shellcode]

基于 Metasploit 框架，创建 exploit，完成 Surgemail 程序的漏洞利用。

```
sudo vi /usr/share/metasploit-framework/modules/exploits/windows/imap/surgemail_list.rb
```

## 说明一：

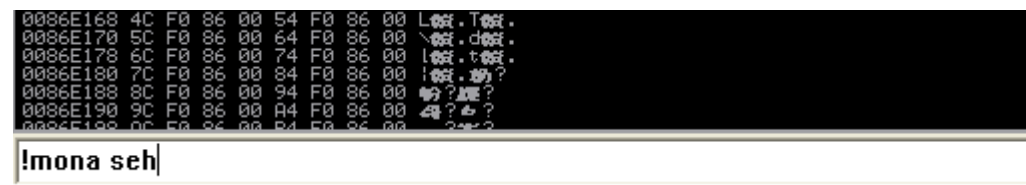
在 surgemail.exe 中定位一个 pop-pop-retn 指令序列的方法：

使用 Mona 脚本

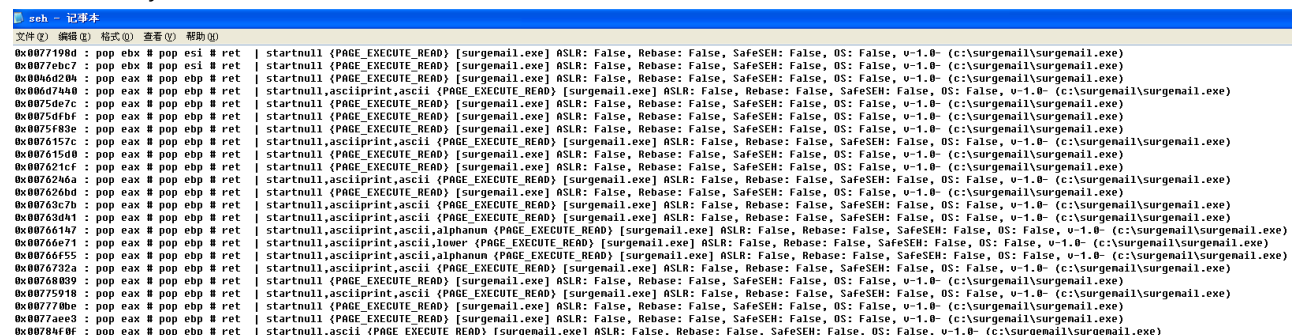
<https://www.corelan.be/index.php/2011/07/14/mona-py-the-manual/>

将 mona 脚本保存到：C:\Program Files\Immunity Inc\Immunity Debugger\PyCommands

在 Immunity Debugger 打开 surgemail.exe，执行 !mona seh



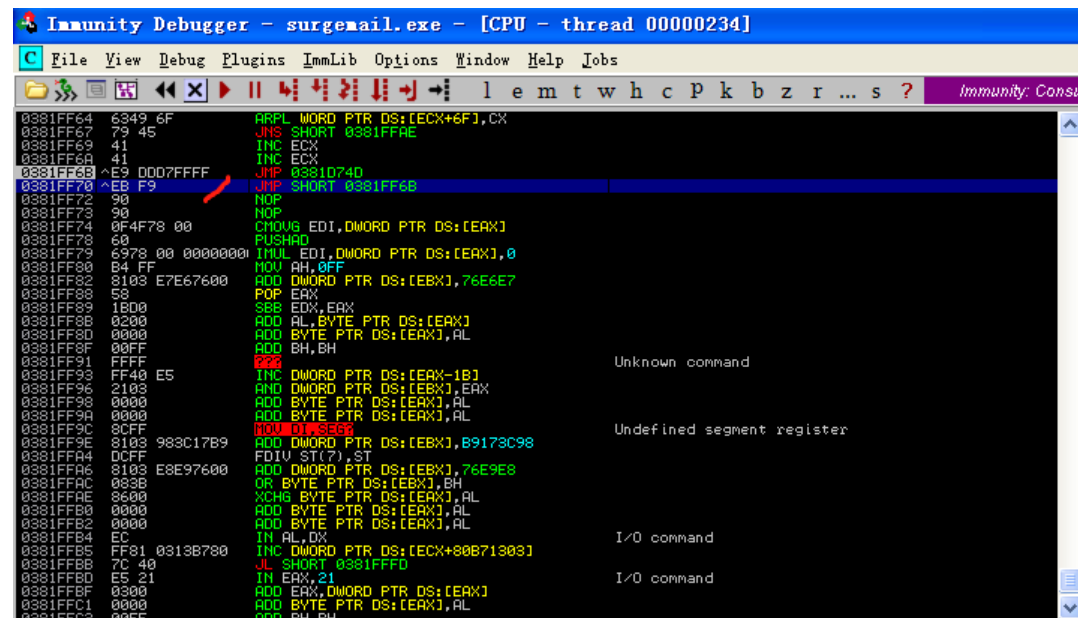
在 Immunity 安装目录下，找到 seh 文件。



## 说明二：

调试 OllyDBG/Immunity Debugger, File->Attach, 将调试器 Attach 到 surgemail 服务程序上。

虚拟机内, 调试快捷键使用时, 需要加上 Fn 键。如: F2—> Fn+F2



```
Immunity Debugger - surgemail.exe - [CPU - thread 00000234]
File View Debug Plugins ImmLib Options Window Help Jobs
0381FF64 6349 6F ARPL WORD PTR DS:[ECX+6F],CX
0381FF67 79 45 JNS SHORT 0381FFAE
0381FF69 41 INC ECX
0381FF6A 41 INC ECX
0381FF6B ^E9 D007FFFF JMP 03810740
0381FF70 ^EB F9 JMP SHORT 0381FF68
0381FF72 90 NOP
0381FF73 90 NOP
0381FF74 0F4F78 00 CMOVG EDI,DWORD PTR DS:[EAX]
0381FF78 60 PUSHAD
0381FF79 6978 00 00000000 IMUL EDI,DWORD PTR DS:[EAX],0
0381FF80 B4 FF MOV AH,0FF
0381FF82 8103 E7E67600 ADD DWORD PTR DS:[EBX],76E6E7
0381FF83 58 POP EAX
0381FF89 1B00 SBB EDX,EAX
0381FF8B 0200 ADD AL,BYTE PTR DS:[EAX]
0381FF8D 0000 ADD BYTE PTR DS:[EAX],AL
0381FF8F 00FF ADD BH,BH
0381FF91 FFFF INC DWORD PTR DS:[EAX-1B]
0381FF93 FF40 AND DWORD PTR DS:[EBX],EAX
0381FF96 2103 AND BYTE PTR DS:[EAX],AL
0381FF98 0000 ADD BYTE PTR DS:[EAX],AL
0381FF9A 0000 ADD BYTE PTR DS:[EAX],AL
0381FF9C 8CFF ADD DWORD PTR DS:[EBX],B9173C98
0381FF9E 8103 983C17B9 ADDI SI(7),SI
0381FFA4 DCFF ADD DWORD PTR DS:[EBX],76E9E8
0381FFA6 8103 E8E97600 OR BYTE PTR DS:[EBX],BH
0381FFAC 083B XCHG BYTE PTR DS:[EAX],AL
0381FFAE 8600 ADD BYTE PTR DS:[EAX],AL
0381FFB0 0000 ADD BYTE PTR DS:[EAX],AL
0381FFB2 0000 IN AL,DX
0381FFB4 EC INC DWORD PTR DS:[ECX+80B71903]
0381FFB6 FF81 0313B780 JLE SHORT 0381FFFD
0381FFB8 7C 40 IN EAX,21
0381FFBD E5 21 ADD EAX,DWORD PTR DS:[EAX]
0381FFBF 0300 ADD BYTE PTR DS:[EAX],AL
0381FFC1 0000 ADD BH,BH
0381FFC3 00FF
```

## 说明三：

修改 exploit, 为了测试其是否有错误, 可以用 ruby exploit\_blabla.rb 进行测试。  
在 msfconsole 里面, 可以用 rexploit 重新加载该 module, 不需要重新设置 options。

## 说明四：

Authentication 成功后, 才会执行到漏洞所在的代码;  
可能多次 Authentication 失败后, 才会成功;

## 说明五：

Crash 后, 关闭 debugger; 重新启动服务; 重新 attach



## 说明六：

Debugger 中：

View->SEH handler->Follow address in stack

