[RElog] Shellcode Formatter (sheadovas/artykuly/relog/shellcode-formatter/)

Lut 23, 2016 / RElog (sheadovas/category/artykuly/relog/)

Witam Was w nowej serii artykułów poświęconych moim zabawom z *Reverse Engineeringiem*[1]. Słowo „zabawa" nie pojawiło się bez przypadku, ponieważ to nie jest seria poradników, a raczej luźne pokazanie artykułów, które jakoś się wiążą z RE.

Głównie to będą jakieś ciekawostki, wnioski, ewentualnie proste programy/skrypty.

Shellcode Formatter

Z racji, że ostatnio robiłem całkiem sporo zadań polegających głównie na wstrzyknięcie jakiegoś *shellcode'u*[2] do programu, to intensywnie korzystałem ze strony [shell-storm.org (http://shell-storm.org/shellcode/)], na której można znaleźć gotowy shellcode. Jest on zazwyczaj w postaci tablicy char'a, podzielonej na wiele linii w celu zwiększenia czytelności.

Linux/x86 - execve(/bin/sh) - 28 bytes by Jean Pascal Pereira PHP

```
/*
Title:  Linux x86 execve("/bin/sh") - 28 bytes
Author: Jean Pascal Pereira <pereira@secbiz.de>
Web:    http://0xffe4.org


Disassembly of section .text:

08048060 <_start>:
 8048060: 31 c0                xor    %eax,%eax
 8048062: 50                   push   %eax
 8048063: 68 2f 2f 73 68       push   $0x68732f2f
 8048068: 68 2f 62 69 6e       push   $0x6e69622f
 804806d: 89 e3                mov    %esp,%ebx
 804806f: 89 c1                mov    %eax,%ecx
 8048071: 89 c2                mov    %eax,%edx
 8048073: b0 0b                mov    $0xb,%al
 8048075: cd 80                int    $0x80
 8048077: 31 c0                xor    %eax,%eax
 8048079: 40                   inc    %eax
 804807a: cd 80                int    $0x80



*/

#include <stdio.h>

char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";

```

O ile ta forma kodu jest akceptowalna przy normalnym skryptowaniu w Pythonie, tak w przypadku uruchamianiu programu z linii poleceń, to bez zapisania skryptu na dysku jest już niewygodna, dlatego postanowiłem napisać skrypt, który z czegoś takiego:

Przykładowe wejście programu C

```
"\x31\xc0\x50\x68\x2f\x2f\x73"
"\x68\x68\x2f\x62\x69\x6e\x89"
"\xe3\x89\xc1\x89\xc2\xb0\x0b"
"\xcd\x80\x31\xc0\x40\xcd\x80";
```

Z kolei pożądany efekt wygląda następująco:

Przykładowe wyjście programu C

```
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80"
```

Jasne, niby można wszystko robić ręcznie, ale po co? Skoro mogę cały proces w bardzo prosty sposób zautomatyzować pisząc skrypt w nie więcej niż 30min, a potencjalnie oszczędzający go bardzo wiele, to to robię. Jakby nie było jestem programistą ;)

Aby było jeszcze ciekawiej (i wygodniej) postanowiłem, że kod będzie pobierał wejście ze schowka systemowego oraz po wykonaniu prostej konwersji zapisywał w nim wyjście. Sam kod prezentuje się następująco:

Shellcode Formatter Python

```python
import pyperclip
import re
import sys

# get raw code
raw = pyperclip.paste()

# create pattern & extract shellcode
pattern = re.compile(r'"[\\|x|\d|a-f|A-F]*"')
matches = pattern.findall(raw)

if len(matches) is 0:
    print 'Error: Invalid arg!'
    sys.exit(1)


# format
res = ""
for line in matches:
    res += line

res = res.replace("\"", "")


# copy final string to clipboard
pyperclip.copy('"' + res + '"')
```

Jak widać został napisany przy użyciu Python'a 2 oraz do poprawnego działania potrzebuje zainstalowanego modułu [pyperclip (https://pypi.python.org/pypi/pyperclip)] który umożliwia zarządzanie schowkiem.

W pierwszej ‚fazie' działania program kopiuje ‚surowe' wejście do stringa o nazwie *raw*, następnie korzystając z prostego *wyrażenia regularnego*[3] (ale wystarczającego do tego zadania) przeszukuje string w poszukiwaniu wystąpienia shellcode'u, tzn wyrażeń w których pomiędzy znakami *"„* posiadających dowolną ilość znaków składających się wyłącznie ze znaków: \, *x*, cyfr oraz liter {*a..f*, *A..F*}.

Dość łatwo można znaleźć przykładowy tekst składający się z tych znaków, lecz nie będący shellcode'm, jednakże ten skrypt będzie używany w ściśle określonych sytuacjach więc jest wystarczający (np. można dodać warunek, mówiący że początek musi zaczynać się od ‚\x', może w przyszłych iteracjach skryptu udoskonalę to wyrażenie).

W przypadku gdy wzorzec nie zostanie odnaleziony, to skrypt kończy swoje działanie, dzięki czemu schowek pozostanie w stanie nienaruszonym.

Ostatnim etapem jest przekopiowanie całej listy do pojedynczego string'a oraz usunięcie znaków *„"* (przy wklejaniu wyniku do schowka string zostaje umieszczony w cudzysłowach).

Najbardziej aktualną wersję skryptu można znaleźć w repo [py-tools-public (https://github.com/sheadovas/py-tools-public)] na moim *Githubie* (jeżeli chcecie to możecie podsyłać pull requesty).

Materiały dodatkowe

1. Reverse Engineering (https://pl.wikipedia.org/wiki/In%C5%BCynieria_odwrotna) – definicja inżynierii wstecznej.
2. Shellcode (https://pl.wikipedia.org/wiki/Shellcode)
3. Wyrażenia regularne (https://pl.wikipedia.org/wiki/Wyra%C5%BCenie_regularne), tzw regex'y

Code ON!