vysec / **Windows-SignedBinary**
forked from Mr-Un1k0d3r/Windows-SignedBinary

No description, website, or topics provided.

14 commits **1** branch **0** releases **1** contributor

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

This branch is 4 commits behind Mr-Un1k0d3r:master. Pull request Compare

Mr-Un1k0d3r Update README.md Latest commit 89ac9c9 on Sep 19, 2017

samples	Delete README.md	8 months ago

README.md	Update README.md	8 months ago
generate.py	Create generate.py	8 months ago
signcheck.bat	Create signcheck.bat	8 months ago
signtool.exe	Add files via upload	8 months ago

README.md

InvalidSign

The idea was to bypass endpoint solution that block known "malicious" signed application such as "regsvr32.exe". I wanted to find a way to get a valid signed file with a different hash.

The analysis

I may have misundertood the output after further analysis but the result remains the same.

I used `signtool verify /v /a cmd.exe`

```
C:\signcheck>signtool verify /a /v cmd.exe

Verifying: cmd.exe
File is signed in catalog: C:\windows\system32\CatRoot\{F750E6C3-38EE-11D1-85E5-
00C04FC295EE}\Microsoft-Windows-Foundation-Package~31bf3856ad364e35~amd64~~6.1.7
601.17514.cat
Hash of file (sha1): 7EB22CBAA74B208DF433C70C06A99280036A52F3
```

```
Signing Certificate Chain:
    Issued to: Microsoft Root Certificate Authority
    Issued by: Microsoft Root Certificate Authority
    Expires:   Sun May 09 19:28:13 2021
    SHA1 hash: CDD4EEAE6000AC7F40C3802C171E30148030C072
```

I thought the "Hash of file" was the SHA1 of cmd.exe based on the output (7EB22CBAA74B208DF433C70C06A99280036A52F3)

Further check revealed that the SHA1 of cmd.exe file was

```
$ sha1sum.exe cmd.exe
0f3c4ff28f354aede202d54e9d1c5529a3bf87d8 *cmd.exe
```

Interesting same file 2 different hashes.

Generating test files

At this point I suspected that the signature may not include all sections of the file.

I wrote a simple python script to generate test files.

```
import sys

orig = list(open(sys.argv[1], "rb").read())

i = 0
while i < len(orig):
```

```
        current = list(orig)
        current[i] = chr(ord(current[i]) ^ 0xde)
        path = "%d.exe" % i

        output = "".join(str(e) for e in current)
        open(path, "wb").write(output)
        i += 1

print "done"
```

`python generate.py cmd.exe` was then executed and generated more 300 Gb of new files.

Final step

We now need to validate each files we created to see if they pass the signature test.

A simple batch file can to that

```
FOR /L %%A IN (1,1,10000) DO (
        signtool verify /v /a %%A.exe
)
```

The binary 330.exe passed the signature check. in this case the file is different since the offset 330 was modified.

```
C:\signcheck>signtool verify /a /v 330.exe

Verifying: 330.exe
File is signed in catalog: C:\windows\system32\CatRoot\{F750E6C3-38EE-11D1-85E5-
00C04FC295EE}\Microsoft-Windows-Foundation-Package~31bf3856ad364e35~amd64~~6.1.7
```

```
601.17514.cat
Hash of file (sha1): 7EB22CBAA74B208DF433C70C06A99280036A52F3
```

```
$ sha1sum.exe 330.exe
4c05efb9d67291febe44f8c661db55a1ec06bc41 *330.exe

$ sha1sum cmd.exe
0f3c4ff28f354aede202d54e9d1c5529a3bf87d8 *cmd.exe
```

I have not finished testing all of the samples but so far I've found that the following bytes can be modified `330, 331, 408 - 412` for cmd.exe.

regsvr32.exe

The following offset can be modified without breaking the signature:

320 - 323, 400 - 407