in2al5dp3in4er Loader

research.openanalysis.net/in2al5dp3in4er/loader/analysis/sandbox/invalid printer/2023/04/23/in2al5dp3in4er.html

OALABS Research April 23, 2023

Overview

This new? loader was exposed by Morphisec. According to the post, the loader is compiled with Embarcadero RAD Studio and employs a graphics card check to ensure it is not running in a sandbox before deploying its embedded payload (the loader). The loader is simply used to download and execute a final payload (main functionality).

References

Samples

- `66383d931f13bcdd07ca6aa50030968e44d8607cf19bdaf70ed4f9ac704ac4d1` UnpacMe

Analysis

```
data = open('/tmp/blob.bin', 'rb').read()

out = []
for i in range(len(data)):
    tmp = data[i]
    tmp = (tmp - 52) & 0xff
    tmp ^= 0x55
    tmp = (tmp + i - 18) & 0xff
    out.append(tmp)

out = bytes(out)
out[:100]
open('/tmp/out.bin','wb').write(out)
```

```
3168770
```

Aurora Stealer

The extracted 2nd stage is the golang stealer sold as "Aurora Stealer" malpedia.

`21545028cac12fc9e8692a71247040718e6d640ee6117d1b19f4521f886586be`UnpacMe

Packer ID

We can make a simple yara rule based on the following

riid for `CreateDXGIFactory` call

```
EC 66 71 7B C7 21 AE 44 B2 1A C9 AE 32 1A E3 69
```

imports

`CreateDXGIFactory` from `DXGI.dll`

checks

```
cmp     eax, 887A0002h
3D 02 00 7A 88
```

gfx whitelist ids

```
{29 9? 01 00}
```

Rule

```
import "pe"
import "math"

rule riid_hunt {

    strings:
        $riid = { EC 66 71 7B C7 21 AE 44 B2 1A C9 AE 32 1A E3 69 }
        $embarcadero = "This program must be run under Win32" ascii
        $import = "CreateDXGIFactory" ascii wide
    condition:
        all of them and
        for any i in (0..(pe.number_of_sections)-1) :
        (
           pe.sections[i].name == ".data" and
            math.entropy(pe.sections[i].raw_data_offset,
pe.sections[i].raw_data_size) >= 7
        )

}
```

Unpacking

```
48 8D 05   9A 94 16 00                                        lea     rax, blob
48 B9   EE EE DE DD CD CC BB 0A                               mov     rcx,
0ABBCCCDDDDEEEEEh
48 BA   55 55 45 44 34 23 12 00                               mov     rdx,
12233444455555h
49 B8   CC CC B3 BB A2 1A 00 00                               mov     r8,
1AA2BBB3CCCCh
4C    63 4D E0                                                 movsxd  r9,
[rbp+var_20]

48 8D 05   D1 93 16 00                                        lea     rax, blob
48 B9   81 FD A9 98 F6 50 00 00                               mov     rcx,
50F698A9FD81h
48 BA   1B 06 AC 5D DE F8 ED 00                               mov     rdx,
0EDF8DE5DAC061Bh
49 B8   04 68 7C AA 99 9D 0B 00                               mov     r8,
0B9D99AA7C6804h
4C   63 4D E8                                                 movsxd  r9,
[rbp+var_18]
```

```
import re
import struct
import pefile

file_data = open('/tmp/pointer.bin', 'rb').read()

pe = pefile.PE(data=file_data)

crypto_egg =
rb'\x48\x8D\x05(....)\x48\xB9(.).......\x48\xBA(.).......\x49\xB8(.).......\x4C'

match = re.search(crypto_egg, file_data, re.DOTALL)

assert match is not None

match_offset = match.start()
payload_offset = struct.unpack('<i', match.group(1))[0]
match_rva = pe.get_rva_from_offset(match_offset)
blob_rva = match_rva + 7 + payload_offset
blob_offset = pe.get_offset_from_rva(blob_rva)
add_inc_key = struct.unpack('B', match.group(2))[0]
xor_key = struct.unpack('B',match.group(3))[0]
add_key = struct.unpack('B',match.group(4))[0]

print(f"add_inc_key: {hex(add_inc_key)}")
print(f"xor_key: {hex(xor_key)}")
print(f"add_key: {hex(add_key)}")
print(f"blob_rva: {hex(blob_rva)}")
print(f"blob_offset: {hex(blob_offset)}")
```

```
add_inc_key: 0xee
xor_key: 0x55
add_key: 0xcc
blob_rva: 0x172ef0
blob_offset: 0x171af0

tmp_data = file_data[blob_offset:]
blob_data = tmp_data.split(b'\x00\x00\x00\x00')[0]
blob_data[:100].hex()

'3e72298e788c7992939091868485858a0c8889dedfdcdde2a3e0e1d6d7d4d5dadbd8d9eeefecedf2f3f0f

def decrypt(data, key1, key2, key3):
    out = []
    for i in range(len(data)):
        tmp = data[i]
        tmp = (tmp + key1) & 0xff
        tmp ^= key2
        tmp = (tmp + i + key3) & 0xff
        out.append(tmp)
    out = bytes(out)
    return out

out = decrypt(blob_data, add_key, xor_key, add_inc_key)


tmp_pe = pefile.PE(data=out)
pe_size = pe.sections[-1].PointerToRawData + pe.sections[-1].Misc_VirtualSize
final_pe = out[:pe_size]
open('/tmp/testpe.bin', 'wb').write(final_pe)

3114235
```

```
def extract(file_path):
    file_data = open(file_path, 'rb').read()
    pe = pefile.PE(data=file_data)
    crypto_egg =
rb'\x48\x8D\x05(....)\x48\xB9(.).......\x48\xBA(.).......\x49\xB8(.).......\x4C'
    match = re.search(crypto_egg, file_data, re.DOTALL)

    assert match is not None

    match_offset = match.start()
    payload_offset = struct.unpack('<i', match.group(1))[0]
    match_rva = pe.get_rva_from_offset(match_offset)
    blob_rva = match_rva + 7 + payload_offset
    blob_offset = pe.get_offset_from_rva(blob_rva)
    add_inc_key = struct.unpack('B', match.group(2))[0]
    xor_key = struct.unpack('B',match.group(3))[0]
    add_key = struct.unpack('B',match.group(4))[0]
    tmp_data = file_data[blob_offset:]
    blob_data = tmp_data.split(b'\x00\x00\x00\x00')[0]
    out = decrypt(blob_data, add_key, xor_key, add_inc_key)
    assert out[:2] == b'MZ'
    tmp_pe = pefile.PE(data=out)
    pe_size = pe.sections[-1].PointerToRawData + pe.sections[-1].Misc_VirtualSize
    final_pe = out[:pe_size]
    open(file_path+'_extracted.bin', 'wb').write(final_pe)

# import required module
import os
# assign directory
directory = '/tmp/samples'

# iterate over files in
# that directory
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
        try:
            print(f)
            extract(f)
        except:
            continue
```

```
/tmp/samples/66383d931f13bcdd07ca6aa50030968e44d8607cf19bdaf70ed4f9ac704ac4d1
/tmp/samples/.DS_Store
/tmp/samples/a4cab01d61d8c18876d4b53d52de365fb9b512430371fd4217359159f3c507f6
/tmp/samples/66383d931f13bcdd07ca6aa50030968e44d8607cf19bdaf70ed4f9ac704ac4d1_extracte

/tmp/samples/5e4b6272dc2d955c5e52c755ea598f44e324b04466a4e3bacf6c9d845345322b
/tmp/samples/cdb09a5df36fece23bc3c9df101fe65724327b827ec43aa9ce0b3b76bdcc3101
/tmp/samples/2c540f5220b7ba3cd6efcd2fe8091fc24f8da11be4b1782c4e502261ef48da82
```