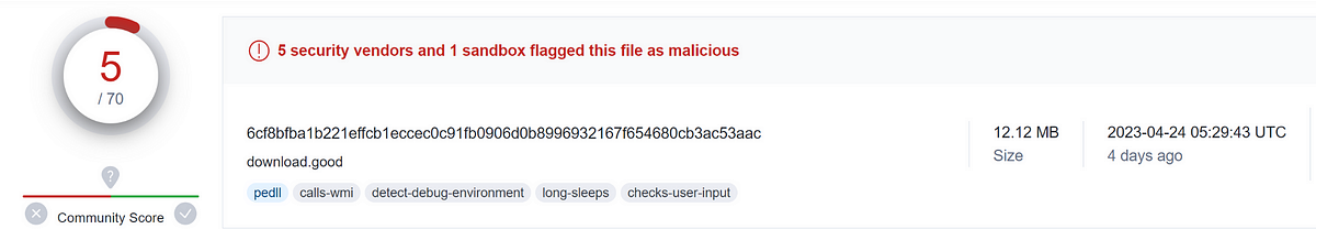
MetaStealer: String Decryption and DGA overview

medium.com/walmartglobaltech/metastealer-string-decryption-and-dga-overview-5f38f76830cd

Jason Reaves May 9, 2023



--

By: Jonathan McCay, Joshua Platt and Jason Reaves

Unit42[1] recently tweeted about a campaign starting with a malicious email link that downloads a OneNote file used to drop and execute MetaStealer. While investigating the MetaStealer sample[2], we noticed it attempts to connect to multiple domains that seemed to be randomly named. After landing on the C2 routine, instead of decrypting a static list of servers, the sample used a domain generation algorithm[3], (DGA) to derive the list. Predominantly used as a fall back mechanism, a domain from the DGA can be registered to communicate when the primary C2s are not available. The use of a DGA as a primary method of contacting the threat actor's infrastructure is interesting.

Sample:

String Decryption:

On top of the MetaStealer binary being highly obfuscated, the author inserted garbage code while also encoding the important onboard strings. The encoding used has been previously mentioned by NCC[4], but a lack of listing of the decoded strings in existing research was noticed. The strings are organized as DWORD values and pushed onto the stack along with the XOR key:

The PXOR instruction allows for using 64 bit(MMX) or 128 bit(XMM) length operands. In this case, you can see that most of the DWORD values are the same. This is because the encoded strings are NULL padded to make the length fit. Ultimately, in the screen above, the data is just XOR encoded:

```
>>> a = struct.pack('<IIII', 0x7D414F85, 0x677C8C64, 0x0AB4023C0, 0x0FF480D71)>>> b =
struct.pack('<IIII', 0x73837AB, 0x677C8C64, 0x0AB4023C0, 0x0FF480D71)>>> a =
bytearray(a)>>> b = bytearray(b)>>> for i in range(len(a)):...   a[i] ^= b[i]...>>>
abytearray(b'.xyz\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00')
```

Attempting to decode all or the majority of the strings in an automatic manner would require finding the relevant blocks of code. After some investigation, it seems longer strings use the VPXOR instruction, which can leverage 128 bit(XMM) or 256 bit(YMM) length operands.

A quick and dirty way to decode strings is to find all the PXOR and VPXOR instructions and then walk backwards to find all the DWORD stack loads to get most of the strings decoded.

```
rule_source = '''
rule meta_s
{
    meta:
        author = "sysopfb"
    strings:
        $snippet1 = {66 0? ef}
        $snippet2 = {c5 ?? ef}
    condition:
        ($snippet1 or $snippet2)
}
'''
def yara_scan(raw_data, rule_name):
    addresses = {}
    yara_rules = yara.compile(source=rule_source)
    matches = yara_rules.match(data=raw_data)
    for match in matches:
        if match.rule == 'meta_s':
            for item in match.strings:
                if item.identifier == rule_name:
                    addresses[item.identifier] = item.instances
    return addresses



data = open(sys.argv[1], 'rb').read()
ret = []
out = None

length_off = Falsesnippet = yara_scan(data, '$snippet1')prev_offset = 0for val in
snippet['$snippet1']: offset = val.offset test = data[prev_offset:offset] vals =
re.findall(b'''\xc7\x85..\xff\xff....''',test) if vals != []:  if len(vals) > 8:
temp = vals[-8:]  else:   temp = vals  try:   xdata = temp[0][-4:]   xdata += temp[1]
[-4:]   xdata += temp[2][-4:]   xdata += temp[3][-4:]   xkey = temp[4][-4:]   xkey +=
temp[5][-4:]   xkey += temp[6][-4:]   xkey += temp[7][-4:]   xdata = bytearray(xdata)
xkey = bytearray(xkey)   for i in range(len(xdata)):    xdata[i] ^= xkey[i]
print(b''.join(xdata.split(b'\x00')))  except:   pass prev_offset = offset
```

Then repeating a similar approach for PVXOR but accounting for larger assortment of strings, as previously mentioned this will only get a majority of the strings decoded but it is enough to determine that this stealer has much functionality including references to starting socks, backconnect, punching holes in the firewall and detonating shellcode:

Decrypted Strings:

```
syschromefirefoxedgenotepadippasswordcmdinputFG StartedshellcodemodeResult:
bc_addrportsocks started:1775uuid/api/client/newffoxdir=in
version.xyzokos_cryptencrypted_keyRtlGetVersionWindows 10Windows VistaWindows
7Windows 8Windows
8.1Proactionstatuspasswdloader_idfilesdurscriptFALSEtypetask_result/tasks/collectROOTn
Typecmd.exemicrosoft\\windows/c "echo start "" "Failed to create task
definitionaction=allow program="(\\d{1,3})\\.(\\d{1,3})\\Default\\Login
Data\\Default\\Web Data
```

A more exhaustive method for harvesting stack loaded and then decoded strings would be using a CPU emulator similar to how we showed against BazaLoader[5] previously.

Domain Generation Algorithm:

Additional domains the sample attempts to communicate with that appeared to be DGA.

IDA: Pushing seed, (0x1234) before calling DGA

IDA: DGA Routine

Converted to Python:

Now that the host names have been generated, the url can be built using the tld and uri from the decrypted strings. An attempt to connect to each host will be made until a response is received. At the time of writing this, only one of the derived hostnames, (wgcuwcgociewewoo.xyz) would resolve. The ip it resolves to, (pictured below) also contains another hostname that used to be active, (mmswgeewswyyywqk.xyz). Both of these domains are in the list derived by the DGA when using the seed value 0x1234. Going back further we can find additional seed values being used at different time periods; 0xabc8, 0x9b2f, 0x7b2f, 0xc17a, 0x2f73 and 0x4b9a.

When KELA[6] first observed the threat actor “_META_” offering a new stealer, it was marketed as having the same functionality and panel as RedLine[7] Stealer. After seeing references to backconnect, socks, and loader id’s in the decrypted strings, we can see that the improvements made to this tool now offer more than just credential theft.

IOCs

Endpoint:

```
powershell -inputformat none -outputformat none –NonInteractive -Command Add-
MpPreference -ExclusionExtension "exe"AppData\Local\Microsoft\Windows\hyper-
v.exehyper-v.ver
```

Network:

```
mmswgeewswyyywqk.xyz
wgcuwcgociewewoo.xyz
yiogqkksoyysqiky.xyz
ikuasuggwiewymsi.xyz
uosqysascuwmqgyk.xyz
uiouaqcqqcgueweg.xyz
uawqgawkguwiqeyk.xyz
kaewquswkswcmsim.xyz
aaycciywcgaqwkky.xyz
mkgcsmogqewauaiw.xyz
185.172.129.192
185.203.116.71
167.88.12.112

[a-z]{16}.xyz:1775/api/client/new[a-z]{16}.xyz:1775/api/client/verify[a-z]
{16}.xyz:1775/api/client_hello[a-z]{16}.xyz:1775/tasks/get_worker[a-z]
{16}.xyz:1775/tasks/collect[a-z]{16}.xyz:1775/avast_update
```

Full DGA dump for every currently known seed can be found at: https://github.com/sysopfb/open_mal_analysis_notes/tree/master/metastealer_dga

References

1: https://twitter.com/Unit42_Intel/status/1646940355936256000

2: https://www.virustotal.com/gui/file/6cf8bfba1b221effcb1eccec0c91fb0906d0b8996932167f654680cb3ac53aac

3: https://en.wikipedia.org/wiki/Domain_generation_algorithm

4: https://research.nccgroup.com/2022/05/20/metastealer-filling-the-racoon-void/

5: https://medium.com/walmartglobaltech/decrypting-bazarloader-strings-with-a-unicorn-15d2585272a9

6: https://www.bleepingcomputer.com/news/security/new-blackguard-password-stealing-malware-sold-on-hacker-forums/

7: https://www.proofpoint.com/us/blog/threat-insight/new-redline-stealer-distributed-using-coronavirus-themed-email-campaign