TrueBot Analysis Part IV - Config Extraction

13 Jul 2023 » malware_analysis, reverse_engineering, config_extraction

In the last post of the TrueBot series, I described some of TrueBot’s capabilities in more detail. In this post I will use this information to write a config extractor and extract the RC4 key for the C2, the mutex and the C2 IPs/domains.

Like in all config extractors, I need to somehow find the relevant things I want to extract.

Usually, I use YARA to navigate within the binary but this time, I wanted to try something different.
TL;DR - It’s not easier than YARA and does not work in all cases, but only in a specific case.

My idea was to use SMDA from Daniel Plohmann to identify all RC4-/Base64 and CreateMutex calls based on their structure (amount of basic blocks, incoming/outgoing calls, etc.). This approach works for different sets of samples but unfortunately not for all samples, so I found myself implementing a lot of exceptions for certain sets of samples. The code thus became very unreadable and unnecessarily complicated. I therefore decided to use SMDA only for finding the CreateMutex call and to find the other things in a more “simple” way. For the second RC4 key, the one used to decrypt downloaded payloads, I haven’t found an easy solution to extract it reliably for all samples. It is easy to find in the binary, but not easy to extract programmatically without implementing a lot of edge cases. I have therefore decided to not extract the second RC4 key.

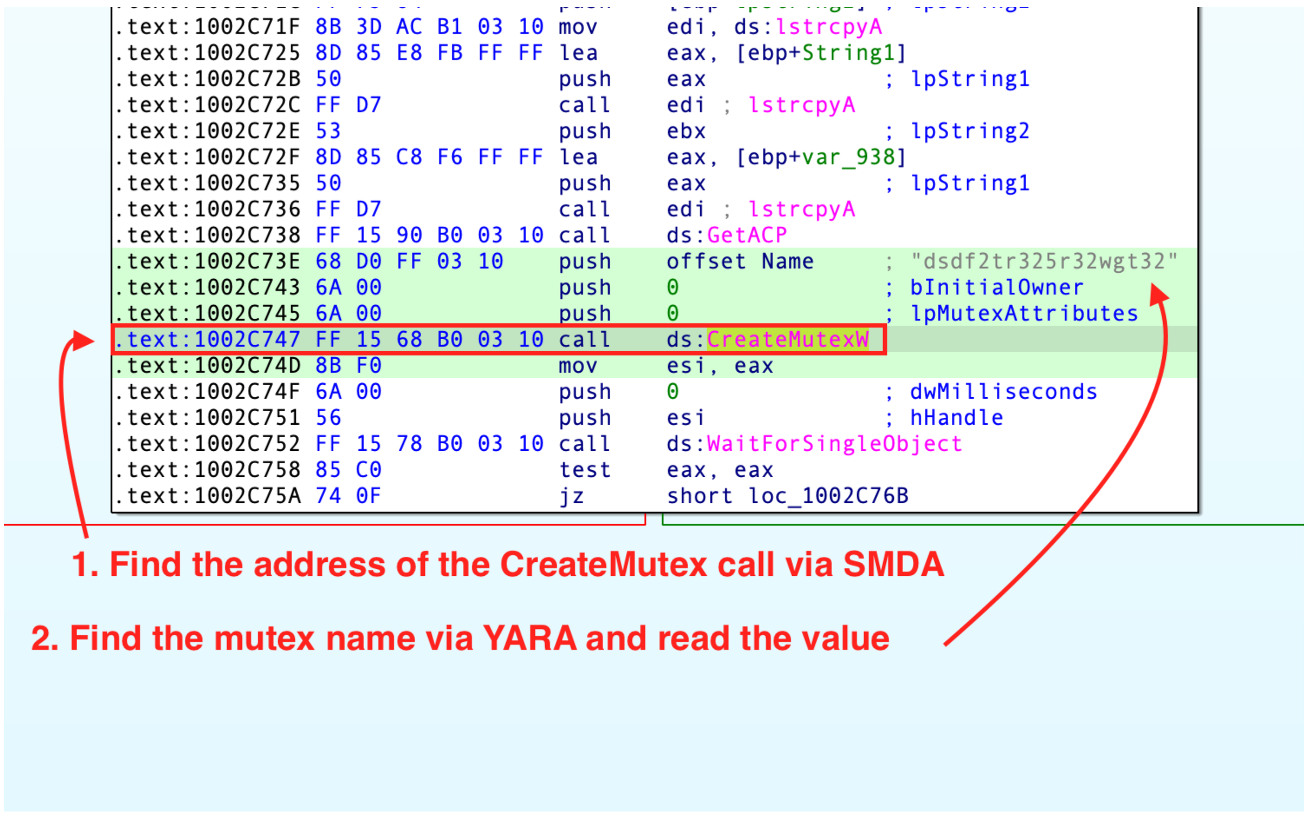
CreateMutex() and arguments

To find the address of the call to `CreateMutex`, I use SMDA to get all API calls available in the binary and look for the string `kernel32.dll!CreateMutex`. In all observed TrueBot samples, there is only one call to `CreateMutex`, so I don’t have to deal with multiple calls.

```
disassembler = Disassembler()
report = disassembler.disassembleFile(args.file)
functions = report.getFunctions()

for fn in functions:
    for addr, name in fn.apirefs.items():
            if 'kernel32.dll!CreateMutex' in name:
            # Find the argument for the CreateMutex call
            ...
```

When found the address, I use YARA to find the push instruction nearby the call to `CreateMutex` and just read the argument with Malduck.



```
.text:1002C71F 8B 3D AC B1 03 10 mov     edi, ds:lstrcpyA
.text:1002C725 8D 85 E8 FB FF FF lea     eax, [ebp+String1]
.text:1002C72B 50                push    eax             ; lpString1
.text:1002C72C FF D7             call    edi ; lstrcpyA
.text:1002C72E 53                push    ebx             ; lpString2
.text:1002C72F 8D 85 C8 F6 FF FF lea     eax, [ebp+var_938]
.text:1002C735 50                push    eax             ; lpString1
.text:1002C736 FF D7             call    edi ; lstrcpyA
.text:1002C738 FF 15 90 B0 03 10 call    ds:GetACP
.text:1002C73E 68 D0 FF 03 10    push    offset Name     ; "dsdf2tr325r32wgt32"
.text:1002C743 6A 00             push    0               ; bInitialOwner
.text:1002C745 6A 00             push    0               ; lpMutexAttributes
.text:1002C747 FF 15 68 B0 03 10 call    ds:CreateMutexW
.text:1002C74D 8B F0             mov     esi, eax
.text:1002C74F 6A 00             push    0               ; dwMilliseconds
.text:1002C751 56                push    esi             ; hHandle
.text:1002C752 FF 15 78 B0 03 10 call    ds:WaitForSingleObject
.text:1002C758 85 C0             test    eax, eax
.text:1002C75A 74 0F             jz      short loc_1002C76B
```

1. Find the address of the CreateMutex call via SMDA

2. Find the mutex name via YARA and read the value

Base64 decode calls and strings

As mentioned before, the usual idea was to find the Base64 decoding calls in the binary via SMDA. However, it turned out that this did not really work reliably for all samples available to me without implementing a number of exceptions.
I therefore decided to use a trivial approach, which, however, works surprisingly reliably.

To find all Base64 strings, I just search for all strings in the binary and try to Base64 decode them. If this succeeds without an exception, I have found a base64 string and can use it later on.

```
    def extract_ascii_strings(data, min_len=16):

        chars = b" !\"#\$%&\'\(\)\*\+,-\./0123456789:;<=>\?@ABCDEFGHIJKLMNO" \
                b"PQRSTUVWXYZ\[\]\^_`abcdefghijklmnopqrstuvwxyz\{\|\}\\\~\t"

        string_list = []
        regexp = b'[%s]{%d,}' % (chars, min_len)
        pattern = re.compile(regexp)
        for s in pattern.finditer(data):
            string_list.append(s.group().decode())
        return string_list

        ...

        lazy_b64_pattern = re.compile('^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]'
                                      '[AQgw]==|[A-Za-z0-9+/]{2}
[AEIMQUYcgkosw048]=)?$')

        b64_strings = []
        extracted_strings = extract_ascii_strings(data, min_len=16)

        for s in extracted_strings:
            for a in lazy_b64_pattern.finditer(s):
                tmp = a.group()
                try:
                    decoded = base64.b64decode(tmp).decode("utf-8")
                    logger.info(f'Found {tmp} string.')
                    b64_strings.append(decoded)
                except:
                    # ignore all non base64 strings
                    pass
```

Decrypt the C2

Similar to the Base64 decode calls, finding the RC4 Calls with SMDA and then navigating to the key via YARA is a pain because the structure is different in a lot of samples and it just does not work reliably. Since I already found the Base64 strings before, I just need to b64decode them and then URL decode the result. Then I can bruteforce the result with all strings I collected before. Since I know, how the decrypted string will look like (e.g. must be a valid domain, always includes `.php`), it's quite easy to find the domain and the page.

```
    c2 = ''
    rc4_key = ''
    for item in extracted_strings:
        item = item.encode('utf-8')
        for decoded_b64_string in b64_strings:
            decoded = urllib.parse.unquote_to_bytes(decoded_b64_string)
            try:
                decrypted = malduck.rc4(item, decoded)
                if decrypted:
                    decrypted = decrypted.decode('ascii')
                    if c2:
                        if '.php' in decrypted:
                            c2 += decrypted
                            break
                    elif '.' in decrypted:  # lazy check
                        logger.debug(f'Successfully decrypted with key {item}:
{decrypted}')
                        c2 += decrypted
                        rc4_key = item
            except:
                pass
```

Testing

I tested my code against the following hashes. Please note that the code is kinda slow because of SMDA. If you don’t need the mutex, just comment it out and it will run much faster.

```
05c72e77d14cee079ac94706759dfe77c27fe51731a1eca22b03352190087e9e
80b9c5ec798e7bbd71bbdfffab11653f36a7a30e51de3a72c5213eafe65965d9
0e3a14638456f4451fe8d76fdc04e591fba942c2f16da31857ca66293a58a4c3
894a7d13d4bd0925e4ec64a401b818ab11ddccac96111d54e10ec32b221d198a
1415f335a0c29fecc3309c8370c8bebefab590de35f206aa9d83861e38d0b74b
97bae3587f1d2fd35f24eb214b9dd6eed95744bed62468d998c7ef55ff8726d4
20af4f3b3d38c770a6539ea716d505fe17962d26a7ad7fa9d5e15dae0838618d
97d0844ce9928e32b11706e06bf2c4426204d998cb39964dd3c3de6c5223fff0
22e3f4602a258e92a0b8deb5a2bd69c67f4ac3ca67362a745178848a9da7a3cc
a0dc543073acd80e4cd97aefb057f030d419787647c7d2a3adb3f32efa9c22a6
32ae88cddeeeec255d6d9c827f6bffc7a95e9ea7b83a84a79ff793735a4b4ed7
a30e1f87b78d1cd529fbe2afdd679c8241d3baab175b2f083740263911a85304
36d89f0455c95f9b00a8cea843003d0b53c4e33431fe57b5e6ec14a6c2e00e99
a67df0a8b32bdc5f9d224db118b3153f66518737e702314873b673c914b2bb5c
3cd5c0ae2e8bb1397a8e89ad3539606f692d2570a50c7a282e47551dd801b3ab
af21e8bbd82c03bf72dffc3ef14fcdce25f3b42aec57cf23812d402332ffeb2e
459016820777fea8602b9a58c5f8d21b8fc4574aa5913390a843fedae2eac3e0
b95a764820e918f42b664f3c9a96141e2d7d7d228da0edf151617fabdd9166cf
47f962063b42de277cd8d22550ae47b1787a39aa6f537c5408a59b5b76ed0464
c042ad2947caf4449295a51f9d640d722b5a6ec6957523ebf68cddb87ef3545c
4862618fcf15ba4ad15df35a8dcb0bdb79647b455fea6c6937c7d050815494b0
c0f8aeeb2d11c6e751ee87c40ee609aceb1c1036706a5af0d3d78738b6cc4125
4e04e8b780acaf693f860184db29d3420f6b6d8176b8b3c73e3c813de4550e62
c944a6a872f16a744ec3a83d1bb339ebc31313ad71eecc4784bb49abc97e0ba4
51fa720e8789821ef57e31381ebae5b70999402320efe7f50b952ace6968f4a2
c9b874d54c18e895face055eeb6faa2da7965a336d70303d0bd6047bec27a29d
54f33d06e2898f0968cb7ba552bc71e4459832637f154e21a4825d22eb9336eb
d408df352b4b9e27c217b8fecdf1136174e15c5164267eddf88e35094093bb36
594ade1fb42e93e64afc96f13824b3dbd942a2cdbc877a7006c248a38425bbc1
dfde0f94a69d0f68a8846e400748bb89bc8900059a64b1dd05e6a3226db2ca92
5cc8c9f2c9cee543ebac306951e30e63eff3ee103c62dadcd2ce43ef68bc7487
e0178ab0893a4f25c68ded11e74ad90403443e413413501d138e0b08a910471e
6a565088c66a78dd0362af9766b8ddf424afcbee20e167c0aa1131f8a518baa7
ed38c454575879c2546e5fccace0b16a701c403dfe3c3833730d23b32e41f2fe
6b646641c823414c2ee30ae8b91be3421e4f13fa98e2d99272956e61eecfc5a1
f9f649cb5de27f720d58aa44aec6d0419e3e89f453730e155067506ad3ece638
717beedcd2431785a0f59d194e47970e9544fbf398d462a305f6ad9a1b1100cb
ff3c79e793f5b803554542435d164867aa0d3672897e131832c3c3ba15bbe9ae
7c607eca4005ba6415e09135ef38033bb0b0e0ff3e46d60253fc420af7519347
ff8c8c8bfba5f2ba2f8003255949678df209dbff95e16f2f3c338cfa0fd1b885
7c79ec3f5c1a280ffdf19d0000b4bfe458a3b9380c152c1e130a89de3fe04b63
```

Results:

```
sha256:05c72e77d14cee079ac94706759dfe77c27fe51731a1eca22b03352190087e9e
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:0e3a14638456f4451fe8d76fdc04e591fba942c2f16da31857ca66293a58a4c3
c2:qweastradoc.com/gate.php
rc4_key_c2:duwureLycirifysy
mutex:IFjwi312fu321321rfewfew

sha256:1415f335a0c29fecc3309c8370c8bebefab590de35f206aa9d83861e38d0b74b
c2:midnigthwaall.com/gate.php
rc4_key_c2:NevucyNyUaXyraIy
mutex:FuckingShitonAllEarth#666

sha256:20af4f3b3d38c770a6539ea716d505fe17962d26a7ad7fa9d5e15dae0838618d
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:GLOIUTWISPFKr2tfsg432

sha256:22e3f4602a258e92a0b8deb5a2bd69c67f4ac3ca67362a745178848a9da7a3cc
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:32ae88cddeeeec255d6d9c827f6bffc7a95e9ea7b83a84a79ff793735a4b4ed7
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:36d89f0455c95f9b00a8cea843003d0b53c4e33431fe57b5e6ec14a6c2e00e99
c2:ronoliffuion.com/dns.php
rc4_key_c2:TiCacyTumoQifixu
mutex:LjdDlkfdslkfj328ewfujsifj32oirew

sha256:3cd5c0ae2e8bb1397a8e89ad3539606f692d2570a50c7a282e47551dd801b3ab
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:459016820777fea8602b9a58c5f8d21b8fc4574aa5913390a843fedae2eac3e0
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:47f962063b42de277cd8d22550ae47b1787a39aa6f537c5408a59b5b76ed0464
c2:dremmfyttrred.com/dns.php
rc4_key_c2:WoOoHequZeMyNusa
mutex:KisujIIs3fsfsSOFldsfds

sha256:4862618fcf15ba4ad15df35a8dcb0bdb79647b455fea6c6937c7d050815494b0
c2:essadonio.com/538332.php
```

```
rc4_key_c2:qaTuMuseBaMuQoNe
mutex:OrionStartWorld#666

sha256:4e04e8b780acaf693f860184db29d3420f6b6d8176b8b3c73e3c813de4550e62
c2:185.55.243.110/gate.php
rc4_key_c2:HirisuTiZoKaMyEe
mutex:GLOIUTWISPFKr2tfsg432

sha256:51fa720e8789821ef57e31381ebae5b70999402320efe7f50b952ace6968f4a2
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:54f33d06e2898f0968cb7ba552bc71e4459832637f154e21a4825d22eb9336eb
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:594ade1fb42e93e64afc96f13824b3dbd942a2cdbc877a7006c248a38425bbc1
c2:dremmfyttrred.com/dns.php
rc4_key_c2:WoOoHequZeMyNusa
mutex:KisujIIs3fsfsSOFldsfds

sha256:5cc8c9f2c9cee543ebac306951e30e63eff3ee103c62dadcd2ce43ef68bc7487
c2:jirostrogud.com/gate.php
rc4_key_c2:KuXoZowywoCyKawi
mutex:IFjwi312fu321321rfewfew

sha256:6a565088c66a78dd0362af9766b8ddf424afcbee20e167c0aa1131f8a518baa7
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:6b646641c823414c2ee30ae8b91be3421e4f13fa98e2d99272956e61eecfc5a1
c2:nomoresense.com/checkinfo.php
rc4_key_c2:HeSaXuEyfoEaKiTy
mutex:vxzcsdbfhk523wfesfFESRSUDHD

sha256:717beedcd2431785a0f59d194e47970e9544fbf398d462a305f6ad9a1b1100cb
c2:essadonio.com/538332.php
rc4_key_c2:qaTuMuseBaMuQoNe
mutex:OrionStartWorld#666

sha256:7c607eca4005ba6415e09135ef38033bb0b0e0ff3e46d60253fc420af7519347
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:GLOIUTWISPFKr2tfsg432

sha256:7c79ec3f5c1a280ffdf19d0000b4bfe458a3b9380c152c1e130a89de3fe04b63
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:GLOIUTWISPFKr2tfsg432
```

```
sha256:80b9c5ec798e7bbd71bbdfffab11653f36a7a30e51de3a72c5213eafe65965d9
c2:jirostrogud.com/gate.php
rc4_key_c2:KuXoZowywoCyKawi
mutex:dsdf2tr325r32wgt32

sha256:894a7d13d4bd0925e4ec64a401b818ab11ddccac96111d54e10ec32b221d198a
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:97bae3587f1d2fd35f24eb214b9dd6eed95744bed62468d998c7ef55ff8726d4
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:kknfexkseiK

sha256:97d0844ce9928e32b11706e06bf2c4426204d998cb39964dd3c3de6c5223fff0
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:a0dc543073acd80e4cd97aefb057f030d419787647c7d2a3adb3f32efa9c22a6
c2:midnigthwaall.com/gate.php
rc4_key_c2:NevucyNyUaXyraIy
mutex:FuckingShitonAllEarth#666

sha256:a30e1f87b78d1cd529fbe2afdd679c8241d3baab175b2f083740263911a85304
c2:hiperfdhaus.com/gate.php
rc4_key_c2:mufaKanuIuKoQiCy
mutex:LKJFggwithj24ikjofw23

sha256:a67df0a8b32bdc5f9d224db118b3153f66518737e702314873b673c914b2bb5c
c2:dremmfyttrred.com/dns.php
rc4_key_c2:WoOoHequZeMyNusa
mutex:LjdDlkfdslkfj328ewfujsifj32oirew

sha256:af21e8bbd82c03bf72dffc3ef14fcdce25f3b42aec57cf23812d402332ffeb2e
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:b95a764820e918f42b664f3c9a96141e2d7d7d228da0edf151617fabdd9166cf
c2:hiperfdhaus.com/gate.php
rc4_key_c2:mufaKanuIuKoQiCy
mutex:LKJFggwithj24ikjofw23

sha256:c042ad2947caf4449295a51f9d640d722b5a6ec6957523ebf68cddb87ef3545c
c2:qweastradoc.com/gate.php
rc4_key_c2:duwureLycirifysy
mutex:IFjwi312fu321321rfewfew

sha256:c0f8aeeb2d11c6e751ee87c40ee609aceb1c1036706a5af0d3d78738b6cc4125
```

```
c2:ber6vjyb.com/dns.php
rc4_key_c2:QimuKexufeUeDoti
mutex:ASPODIKAFLKJoieLUFESJFuiewr

sha256:c944a6a872f16a744ec3a83d1bb339ebc31313ad71eecc4784bb49abc97e0ba4
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:c9b874d54c18e895face055eeb6faa2da7965a336d70303d0bd6047bec27a29d
c2:qweastradoc.com/gate.php
rc4_key_c2:duwureLycirifysy
mutex:IFjwi312fu321321rfewfew

sha256:d408df352b4b9e27c217b8fecdf1136174e15c5164267eddf88e35094093bb36
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:kknfexkseiK

sha256:dfde0f94a69d0f68a8846e400748bb89bc8900059a64b1dd05e6a3226db2ca92
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:e0178ab0893a4f25c68ded11e74ad90403443e413413501d138e0b08a910471e
c2:dragonetzone.com/gate_info.php
rc4_key_c2:NacuMydaguxoleba
mutex:FuckingShitonAllEarth#666

sha256:ed38c454575879c2546e5fccace0b16a701c403dfe3c3833730d23b32e41f2fe
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:kknfexkseiK

sha256:f9f649cb5de27f720d58aa44aec6d0419e3e89f453730e155067506ad3ece638
c2:nomoresense.com/checkinfo.php
rc4_key_c2:HeSaXuEyfoEaKiTy
mutex:vxzcsdbfhk523wfesfFESRSUDHD

sha256:ff3c79e793f5b803554542435d164867aa0d3672897e131832c3c3ba15bbe9ae
c2:nefosferta.com/gate.php
rc4_key_c2:OumaOyIuRymuZyOi
mutex:xUjfUjUtazdabr325

sha256:ff8c8c8bfba5f2ba2f8003255949678df209dbff95e16f2f3c338cfa0fd1b885
c2:qweastradoc.com/gate.php
rc4_key_c2:duwureLycirifysy
mutex:(u3qkfewi3ujrk32lqpti32ofwq
```

Code can be found here.

If you find bugs or other samples, you know the drill ;-).

Related Posts

- TrueBot Analysis Part III - Capabilities (Categories: malware_analysis, reverse_engineering)
- TrueBot Analysis Part II - Static unpacker (Categories: malware_analysis, reverse_engineering)
- TrueBot Analysis Part I - A short glimpse into packed TrueBot samples (Categories: malware_analysis, reverse_engineering)
- Python stealer distribution via excel maldoc (Categories: malware_analysis, reverse_engineering)
- Having fun with an Ursnif VBS dropper (Categories: malware_analysis, reverse_engineering)
- Trickbot tricks again [UPDATE] (Categories: malware_analysis, reverse_engineering)

« TrueBot Analysis Part III - Capabilities