

Taidoor - a truly persistent threat

 blog.reversinglabs.com/blog/taidoor-a-truly-persistent-threat



Taidoor RAT



Threat Research | September 22, 2020



Blog Author
Karlo Zanki, Reverse Engineer at ReversingLabs. [Read More...](#)

When malware lasts longer than your washing machine



Taidoor RAT



Introduction

Government-supported actors usually conduct long-lasting activities in cyberspace, and to simplify such continuous processes, they often develop malicious tools with the intention of using them for a long period of time. Like any other malicious tool, they need to be stealthy, and when they get detected, some modifications are necessary to become undetectable again. Sometimes it can be as simple as changing the compromised infrastructure, while at other times creating a new version of the tool is required. When organizations put a lot of effort into creating a tool like that, they probably don't plan to use it in massive campaigns. It is expected to be used in smaller, targeted attacks; therefore, researchers won't have too many samples for analysis at their disposal.

An example of such a tool is Taidoor RAT (remote access trojan), dating all the way back to 2008, whose new version was recently discovered and presented in a [technical report](#) released by the US government institutions. Taidoor is described as a Remote Access Trojan developed and used by cyber actors supported by the Chinese government. The new version of the RAT consists of two parts - a loader in a DLL form, and a main RAT module that comes as RC4-encrypted binary data. The loader first decrypts the encrypted main RAT module, and then executes its exported Start function. The report provides two samples for both the loader and the main encrypted RAT module. These samples come with only two C2 domains and one C2 IP.

In this blog, you will learn how our Titanium Platform can help you find more malware samples to extend the IOC lists and power up your defenses.

Taidoor

As already mentioned, the Taidoor RAT consists of two parts, a loader and the main RAT module. A few pivoting attempts on the loader samples didn't produce any results, so we focused on the samples of the main RAT module. This makes sense, as they contain the malware configuration including C2 domains and IPs.

Two samples of the main RAT available from the threat report are encrypted with the RC4 algorithm, and aren't suitable for pivoting in that form. The first step is to decrypt them and get them to their normal DLL format suitable for metadata extraction. Publicly available tools like [CyberChef](#) can be used to decrypt various encryption algorithms, including RC4. Once decrypted, the DLL is processed with Titanium Platform, and its metadata is extracted. The first thing to do is look at the files grouped into the same buckets based on the RHA1, our functional file similarity algorithm, for each of the samples.

d31191689682d5702371812613dbf40f904c...
Preview Sample

Size: 154.5 KB
Type: PE / Dll
Format: --
Threat: ● Win32.Trojan.Taidoor
First seen (cloud): 21 days ago
Last seen (local): 24 minutes ago
User uploads: 1

RHA1 Malicious: ● 2 Suspicious: ● 6 Known: ● 0
Pivoting

Summary

TitaniumCore

TitaniumCloud

Extracted Files (1)

All threats ▼ All Local TiCloud Fetch All

<input type="checkbox"/>	Time	Threat	Name	Format	Files	Size
<input type="checkbox"/>	24 minutes ago	Win32.Trojan.Taidoor	d31191689682d5702371812613dbf40f904c4596	PE/Dll	2	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulise	06bee0d766ec4cdf53837b862c3f15a932110921	PE/Dll	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulise	29aa93880afa08f75a6368dfa551d0e8d46f270a	PE/Dll	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Occamy	3fa7a7af944bf2d234e270df86889d9894b0201	PE/Dll	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	aa7efb1265a1632efc65a4365bb0e31d9e64618a	PE/Dll	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulise	de7b0889fce6e38ac4f902e2399c9a794f8f00df	PE/Dll	1	154.5 KB
<input type="checkbox"/>	1 year ago	Win32.Trojan.Generic	dee5ad6a101be6be0b917cef3466a7aef727dbd7	PE/Dll	1	154.5 KB
<input type="checkbox"/>	1 year ago	Win32.Trojan.Generic	deedc8ca9d96d6fa8601f48d76d20c8514e52c88	PE/Dll	1	154.5 KB

539912d9a50fd67cff736581b5f2d75efb519d...
Preview Sample

Size: 179.5 KB
Type: PE+ / Dll
Format: --
Threat: ● Win64.Trojan.Taidoor
First seen (cloud): 13 days ago
Last seen (local): 16 minutes ago
User uploads: 1

RHA1 Malicious: ● 1 Suspicious: ● 3 Known: ● 0
Pivoting

Summary

All threats ▼ All Local TiCloud Fetch All

<input type="checkbox"/>	Time	Threat	Name	Format	Files	Size
<input type="checkbox"/>	16 minutes ago	Win64.Trojan.Taidoor	539912d9a50fd67cff736581b5f2d75efb519d9d	PE+/Dll	2	179.5 KB
<input type="checkbox"/>	2 years ago	Win64.Trojan.Srvstr	07b108f8ea4fe1ea8bcl63b8650ad85b9ab4b59	PE+/Dll	1	179.5 KB
<input type="checkbox"/>	1 year ago	Win64.Trojan.Generic	0a64574b36d7730a6a71d9d5e5475fd494c8914a	PE+/Dll	1	179.5 KB
<input type="checkbox"/>	2 years ago	Win64.Trojan.Srvstr	22c55ded3486614728eaa29a7526d760ac496b20	PE+/Dll	1	179.5 KB

Similar files grouped by RHA1 algorithm

The RHA1 algorithm reveals ten more samples dating back to 2018 and 2019. However, there are additional options for pivoting. Looking at the samples' exports shows a specific combination of functions and original file name.

539912d9a50fd67cff736581b5f2d75efb519d...
Preview Sample

Size: 179.5 KB
Type: PE+ / Dll
Format: --
Threat: ● Win64.Trojan.Taidoor
First seen (cloud): 2020-08-12 13:08 UTC
Last seen (local): 2020-08-25 13:12 UTC
User uploads: 1

Exports

mm.dll
Service
Start

Samples exports

Running a Titanium Platform cloud search query with this specific combination finds the samples already discovered using the RHA1 file similarity algorithm, and also 3 additional samples.

pe-export:Start AND pe-export:Service AND pe-original-name:mm.dll

Local (0) Cloud Shareable (14) Private (1) Export

<input type="checkbox"/>	First Seen	Threat	Name	Format	Files	Size
<input type="checkbox"/>	5 days ago	Win64.Trojan.Taidoor	539912d9a50fd67c7f736581b5f2d75efb519d9d	PE+/DII	1	179.5 KB
<input type="checkbox"/>	2 weeks ago	Win32.Trojan.Taidoor	d31191689682d5702371812613dbf40f904c4596	PE/DII	1	154.5 KB
<input type="checkbox"/>	1 year ago	Win32.Trojan.Generic	deedc8ca9d96d65fa8601f48d76d20c8514e52c88	PE/DII	1	154.5 KB
<input type="checkbox"/>	1 year ago	Win64.Trojan.Generic	0a64574b36d7730a5a71d9d5e5475fd494c8914a	PE+/DII	1	179.5 KB
<input type="checkbox"/>	1 year ago	Win32.Trojan.Generic	dee5ad6a101be6be0b917cef3466a7aef727dbd7	PE/DII	1	154.5 KB
<input type="checkbox"/>	1 year ago	Win32.Trojan.Ulise	3fa7a7af944bf2d234e270df86889d99894b0201	PE/DII	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	aa7efb1285a1632efc65a4365bb0e31d9e64618a	PE/DII	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win64.Trojan.Srvstr	07b108f8bea4fe1ea8bc1d63b865ad85b9ab4b59	PE+/DII	1	179.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulise	06bee0d766ec4cdf53837b862c3f15a932110921	PE/DII	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulise	de7b0889fce6e38ac4f902e2399c9a794f8f00df	PE/DII	1	154.5 KB
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulise	29aa93880afa88f75a8368dfa551d0e8d46f270a	PE/DII	1	154.5 KB
<input type="checkbox"/>	4 years ago	Win32.Trojan.Generic	72ff757e2fc38532fb244773204d659ee279bd10	PE/DII	1	155.5 KB
<input type="checkbox"/>	4 years ago	Win32.Trojan.Generic	c4665f5f10b7a58be1d4b20edc116863f4abad44	PE/DII	1	155.5 KB
<input type="checkbox"/>	5 years ago	Win32.Trojan.Generic	f1a1ea963ae8aca3a4623912c405cc97df510c07	PE/DII	1	155.5 KB

1 14 results 100

Pivoting on exports and original name

Analyzing these three samples shows that they are using the same AES key as described in the threat report. The significant difference is that they have another layer of encryption used to hide the part of the code responsible for configuration decryption, including the AES key and S-Box initialization.

1001901C	68 58200210	push f1.10022058	
10019021	8D8D 70C3FFFF	lea ecx,dword ptr ss:[ebp-3C90]	
10019027	51	push ecx	
10019028	56	push esi	
10019029	50	push eax	
1001902A	FFB5 60C3FFFF	push dword ptr ss:[ebp-3CA0]	[ebp-3CA0]: "Hoi"
10019030	FF15 345E0210	call dword ptr ds:[<&RegQueryValueExA>]	
10019036	56	push esi	
10019037	6A 01	push 1	
10019039	8D8D A0CAFFFF	lea ecx,dword ptr ss:[ebp-3560]	[ebp-3560]: "L"-0.d11"
1001903F	C645 FC 00	mov byte ptr ss:[ebp-4],0	
10019043	E8 2698FEFF	call f1.1000286E	
10019048	FFB5 60C3FFFF	push dword ptr ss:[ebp-3CA0]	[ebp-3CA0]: "Hoi"
1001904E	FF15 385E0210	call dword ptr ds:[<&RegCloseKey>]	
10019054	8D45 DC	lea eax,dword ptr ss:[ebp-24]	
10019057	50	push eax	
10019058	8D8D OCC5FFFF	lea ecx,dword ptr ss:[ebp-3AF4]	
1001905E	C745 DC 287E1516	mov dword ptr ss:[ebp-24],16157E28	
10019065	C745 E0 28AED2A6	mov dword ptr ss:[ebp-20],A6D2AE28	
1001906C	C745 E4 ABF71588	mov dword ptr ss:[ebp-1C],8815F7AB	
10019073	C745 E8 09CF4F3C	mov dword ptr ss:[ebp-18],3C4FCF09	
1001907A	E8 DE83FEFF	call f1.1000145D	
1001907F	68 50060000	push 650	
10019084	68 58200210	push f1.10022058	
10019089	8D8D OCC5FFFF	lea ecx,dword ptr ss:[ebp-3AF4]	
1001908F	C645 FC 05	mov byte ptr ss:[ebp-4],5	
10019093	E8 C78AFEFF	call f1.1000185F	
10019098	68 02020000	push 202	
1001909D	68 A8220210	push f1.100222A8	
100190A2	68 F8780210	push f1.100278F8	
100190A7	E8 44C3FEFF	call f1.100053F0	
100190AC	83C4 0C	add esp,C	
100190AF	68 C0C30110	push f1.1001C3C0	1001C3C0: "system"
100190B4	56	push esi	
100190B5	FF15 405E0210	call dword ptr ds:[<&OpenEventLogA>]	
100190BB	8BD8	mov ebx,eax	
100190BD	3BDE	cmp ebx,esi	
100190BF	0F84 89000000	je f1.1001914E	
100190C5	8D85 54C3FFFF	lea eax,dword ptr ss:[ebp-3CAC]	
100190C8	50	push eax	
100190CC	53	push ebx	
100190CD	8D8D BCCAFFFF	lea edi,dword ptr ss:[ebp-3544]	
100190D3	FF15 445E0210	call dword ptr ds:[<&GetOpenEventLogRecords>]	

AES key

Loading of AES decryption key in previously decrypted layer of code

Pivoting on these three samples didn't return any new results. However, there is something uncommon in this exported file name - "mm.dll". It is probably a shorthand for something.

What if there were some variations in the functionality of these samples? There is a chance that this *mm* prefix could have something appended to it. To check this assumption, a search query is constructed using the wildcard character to match samples that start with the *mm* prefix and at the same time contain the aforementioned exported functions *Start* and *Service*.

pe-export:Start AND pe-export:Service AND pe-original-name:mm*

Local (3) Cloud - Shareable (43) Private (64) Export

<input type="checkbox"/>	<input type="radio"/> First Seen	Threat	Name	Format	Files	Size	^
<input type="checkbox"/>	1 year ago	Win32.Trojan.Johnnie	4118cc4ee6e22bca1933b0033cfe07924293b6bb	PE/DLL	1	134 KB	☰
<input type="checkbox"/>	3 years ago	Win32.Backdoor.Generic	f2320ce302bcfc8397b010556fcf5b19fb7304ec	PE/DLL	1	134 KB	☰
<input type="checkbox"/>	3 weeks ago	Win32.Trojan.Taidoor	d31191689682d5702371812613dbf40f904c4596	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	1 year ago	Win32.Trojan.Generic	deedc8ca9d96d6fa8601f48d76d20c8514e52c88	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	1 year ago	Win32.Trojan.Generic	dee5ad6a101be6be0b917cef3466a7aef727dbd7	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Occamy	3fa7a7af944bf2d234e270df86889d99894b0201	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	aa7efb1285a1632efc65a4365bb0e31d9e64618a	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulisse	06bee0d766ec4cdf53837b862c3f15a932110921	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulisse	de7b0889fce6e38ac4f902e2399ca9794f8f00df	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Ulisse	29aa93880afa88f75a8368dfa551d0e8d46f270a	PE/DLL	1	154.5 KB	☰
<input type="checkbox"/>	4 years ago	Win32.Trojan.Generic	c4665f5f10b7a58be1d4b20edc116863f4abad44	PE/DLL	1	155.5 KB	☰
<input type="checkbox"/>	5 years ago	Win32.Trojan.Generic	f1a1ea963ae8aca3a4623912c405cc97df510c07	PE/DLL	1	155.5 KB	☰
<input type="checkbox"/>	4 years ago	Win32.Trojan.Generic	72ff75e2fc38532fb244773204d659ee279bd10	PE/DLL	1	155.5 KB	☰
<input type="checkbox"/>	2 weeks ago	Win64.Trojan.Taidoor	539912d9a50fd67c7f36581b5f2d75efb519d9d	PE+/DLL	1	179.5 KB	☰
<input type="checkbox"/>	1 year ago	Win64.Trojan.Generic	0a64574b36d7730a5a71d9d5e5475fd494c8914a	PE+/DLL	1	179.5 KB	☰
<input type="checkbox"/>	2 years ago	Win64.Trojan.Srvstr	07b108fba4fe1ea8bc1d63b8650ad85b9ab4b59	PE+/DLL	1	179.5 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	4bfa15f1cf1b3617b01eb79974649e0d4986f212	PE/DLL	1	466 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	fc589d1b50eb46d839033737f29777b138446de1	PE/DLL	1	466 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	859e0f0ccbcafd25b0877a0c6df0c94cd84d2433	PE/DLL	1	466 KB	☰
<input type="checkbox"/>	2 years ago	Win32.Trojan.Generic	5a874820aa1c14e02596de6d97ab9ac604882239	PE/DLL	1	466 KB	☰

Pivoting on exports and original name using wildcard character

Ordering these search results by their size - and ignoring clean software and previously discovered samples - leaves just a few hashes we need to look at. These hashes are split in two groups based on their size. Looking at their exports reveals two new original file names - `mm_tcp_dll.dll` and `mm_tcp_svchost.dll`.

DLL **5a874820aa1c14e02596de6d97ab9ac60488...**

[Preview Sample](#)

Size: 466.0 KB

Type: PE / DLL

Format: --

Threat: ● Win32.Trojan.Generic

First seen (cloud): 2018-04-26 09:41 UTC

Last seen (local): 2020-08-26 12:09 UTC

User uploads: [1](#)



Exports

mm_tcp_dll.dll

Service

Start

DLL **4118cc4ee6e22bca1933b0033cfe07924293...**

[Preview Sample](#)

Size: 134.0 KB

Type: PE / DLL

Format: --

Threat: ● Win32.Trojan.Johnnie

First seen (cloud): 2019-06-06 12:18 UTC

Last seen (local): 2020-08-20 07:48 UTC

User uploads: [1](#)



Exports

mm_tcp_svchost.dll

Service

Start

Samples exports

Analyzing these samples in disassembler confirms these are indeed Taidoor samples, since they perform the same AES decryption and use the same AES key. The 466 KB samples also have a layer of encryption used to hide the part of the code responsible for configuration decryption, as already described in the previous case.

```
loc_10003E7D:
lea    ecx, [ebp+var_5430]
push  ecx
lea    ebx, [ebp+var_59DC]
mov    [ebp+var_5430], 16157E28h
mov    [ebp+var_542C], 0A6D2AE28h
mov    [ebp+var_5428], 8815F7ABh
mov    [ebp+var_5424], 3C4FCF09h
call   sub_10001000
push  404h
push  offset unk_100201A0
mov    edi, ebx
mov    byte ptr [ebp+var_4], 7
call   sub_10001740
cmp    word_100203A0, 0
mov    ecx, 80h
mov    esi, offset word_100203A0
mov    edi, offset word_10022ED8
rep    movsd
push  offset aSystem ; "system"
setnz dl
push  0
mov    dword_10022AD4, offset unk_100201A0
movsw
mov    byte_100229CD, dl
call   dword_10023100
mov    edi, eax
test   edi, edi
jz     loc_10003FB0
```

Loading of AES decryption key in 134KB samples

66925432	FF15 FC119466	call dword ptr ds:[<&RegCloseKey>]	
66925438	3BF4	cmp esi,esp	
6692543A	E8 96DEFBFF	call 85.668E32D5	
6692543F	C685 ECFCFFFF 28	mov byte ptr ss:[ebp-314],28	28: '+'
66925446	C685 EDFCFFFF 7E	mov byte ptr ss:[ebp-313],7E	7E: '~'
6692544D	C685 EEFCEFFF 15	mov byte ptr ss:[ebp-312],15	
66925454	C685 EFFCFEFFF 16	mov byte ptr ss:[ebp-311],16	
6692545B	C685 F0FCFFFF 28	mov byte ptr ss:[ebp-310],28	28: '('
66925462	C685 F1FCFFFF AE	mov byte ptr ss:[ebp-30F],AE	
66925469	C685 F2FCFFFF D2	mov byte ptr ss:[ebp-30E],D2	
66925470	C685 F3FCFFFF A6	mov byte ptr ss:[ebp-30D],A6	
66925477	C685 F4FCFFFF A8	mov byte ptr ss:[ebp-30C],A8	
6692547E	C685 F5FCFFFF F7	mov byte ptr ss:[ebp-30B],F7	
66925485	C685 F6FCFFFF 15	mov byte ptr ss:[ebp-30A],15	
6692548C	C685 F7FCFFFF 88	mov byte ptr ss:[ebp-309],88	
66925493	C685 F8FCFFFF 09	mov byte ptr ss:[ebp-308],9	9: '\t'
6692549A	C685 F9FCFFFF CF	mov byte ptr ss:[ebp-307],CF	
669254A1	C685 FAFCEFFF 4F	mov byte ptr ss:[ebp-306],4F	4F: 'O'
669254A8	C685 FBFCFFFF 3C	mov byte ptr ss:[ebp-305],3C	3C: '<'
669254AF	8D85 ECFCFFFF	lea eax,dword ptr ss:[ebp-314]	
669254B5	50	push eax	
669254B6	8D8D 30FAFFFF	lea ecx,dword ptr ss:[ebp-5D0]	
669254BC	E8 518DFAFF	call 85.668D1212	
669254C1	8985 40C0FFFF	mov dword ptr ss:[ebp-3FC0],eax	
669254C7	C645 FC 06	mov byte ptr ss:[ebp-4],6	
669254CB	68 50060000	push 650	
669254D0	68 78C09366	push 85.6693C078	
669254D5	8D8D 30FAFFFF	lea ecx,dword ptr ss:[ebp-5D0]	
669254DB	E8 D3BCFAFF	call 85.668D1183	
669254E0	C785 24FAFFFF 78C09366	mov dword ptr ss:[ebp-5DC],85.6693C078	
669254EA	68 02020000	push 202	
669254EF	8B85 24FAFFFF	mov eax,dword ptr ss:[ebp-5DC]	
669254F5	05 50020000	add eax,250	
669254FA	50	push eax	
669254FB	68 E8309466	push 85.669430E8	
66925500	E8 68DAFBFF	call 85.668E2F70	
66925505	83C4 0C	add esp,C	
66925508	8BF4	mov esi,esp	
6692550A	68 0C1E9366	push 85.66931E0C	
6692550F	6A 00	push 0	66931E0C: "system"
66925511	FF15 04129466	call dword ptr ds:[<&OpenEventLogA>]	
66925517	3BF4	cmp esi,esp	

Loading of AES decryption key in previously decrypted layer of code in 466KB samples

They also include the same strings present in the Taidoor samples mentioned in the referenced threat report.

```
0x040x230x190x340xfe0xc1
```

```
%%temp%%\%u
```

```
CONNECT %s:%d HTTP/1.0
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
```

```
Host: %s
```

```
Accept: */*
```

```
Pragma: no-cache
```

```
Proxy-Connection: Keep-Alive
```

```
Content-Length: 0
```

Some of the strings found in Taidoor samples

One interesting thing that catches the eye are the pdb paths. Samples of the 134 KB size contain the pdb path "C:\Users\john\Desktop\KD17.6_20170628\Release\mm_tcp_svchost.pdb" and have the compilation timestamp set to 2017-07-04T09:20:21. Samples of the 466 KB size have their compilation timestamp set to 2016-04-06T08:44:22 and don't include any pdb paths, but looking at the contained strings reveals one very similar to the mentioned pdb path - "c:\users\develop\desktop\kd15.1_aes_20160321\mm_tcp_dllsvchost.cpp". We can assume that in the "KDVV.v_YYYYMMDD" part of the string, the VV.v represents the malware version, and that YYYYMMDD represents the creation date. Unfortunately, pivoting variations on these pdb paths didn't result in any other samples besides the ones previously found.

There are still two discovered original file names left to pivot on. Pivoting on mm_tcp_svchost.dll doesn't result in anything new. On the other hand, pivoting on mm_tcp_dll.dll reveals around 70 more samples. Looking at them in detail indicates they are older versions of Taidoor dating all the way back to 2011. These samples have configuration

encrypted with the RC4 algorithm, and not with AES like the version from the [referenced](#) threat report. We won't describe these samples here because they are a bit outdated and are more similar to the older versions of Taidoor already covered with technical reports from [FireEye](#) and [TrendMicro](#).

But if there is a sample named *mm_tcp_dll.dll*, is it possible that there is a sample using the http protocol for communication? We can guess that it would be named *mm_http_dll.dll* if it follows the naming convention. A search query indeed finds 17 samples with this name, and analyzing them in disassembler shows they also have the configuration encrypted with the same “**0xA1 0xA2**” RC4 key. But as illustrated in the following image, these samples are also 8 years old, and are not in the focus of interest of this blog post.

The screenshot shows a search interface with the query 'pe-original-name:mm_http_dll.dll'. The results are displayed in a table with columns for selection, first seen, threat, name, format, files, and size. All results are Win32.Trojan.Cryect or Win32.Trojan.Graftor, dated between 6 and 9 years ago, with a size of 31.5 KB or 32 KB.

<input type="checkbox"/>	<input type="radio"/> First Seen	Threat	Name	Format	Files	Size
<input type="checkbox"/>	6 years ago	Win32.Trojan.Graftor	25192685d38c48a607d0672185a5138747765908	PE/DLL	1	32 KB
<input type="checkbox"/>	7 years ago	Win32.Trojan.Cryect	e8cced00055d13cd10c60dc9dd0b0f19fc28cf6f	PE/DLL	1	31.5 KB
<input type="checkbox"/>	7 years ago	Win32.Trojan.Cryect	c28d18fb9c130b595395fb62ea499357ef72a636	PE/DLL	1	31.5 KB
<input type="checkbox"/>	8 years ago	Win32.Trojan.Graftor	e51b348ae713baeedf87fe6e0ef5cad3ed247d86	PE/DLL	1	31.5 KB
<input type="checkbox"/>	8 years ago	Win32.Trojan.Cryect	835df53037be9ec1eab34c6b76edb47738fc3a52	PE/DLL	1	31.5 KB
<input type="checkbox"/>	8 years ago	Win32.Trojan.Cryect	65b95a28ec8c89ace14b9c65901b6175d2f084dd	PE/DLL	1	32 KB
<input type="checkbox"/>	8 years ago	Win32.Trojan.Cryect	6904f1a6db1a0984560c50707e5edbca30825edc	PE/DLL	1	32 KB
<input type="checkbox"/>	8 years ago	Win32.Trojan.Graftor	f7304c6a6be6e49e3ece2fe6772a23abb161d0f3	PE/DLL	1	32 KB
<input type="checkbox"/>	9 years ago	Win32.Trojan.Graftor	2b0765de29f97478ab7772eb80002d5e8aa2cda1	PE/DLL	1	32 KB

Pivoting on original name

Another very interesting sample exists, originally named *mm_udt_dll.dll*, suggesting usage of the UDT protocol for the C2 communication. Since it is also quite old, it won't be examined in detail, but it uses the same RC4 key for configuration encryption, and it is likely that the only difference is the network protocol used for the communication.

The screenshot shows a search interface with the query 'pe-original-name:mm_udt_dll.dll'. The results are displayed in a table with columns for selection, first seen, threat, name, format, files, and size. The result is Win32.Trojan.Ursu, dated 9 years ago, with a size of 122 KB.

<input type="checkbox"/>	<input type="radio"/> First Seen	Threat	Name	Format	Files	Size
<input type="checkbox"/>	9 years ago	Win32.Trojan.Ursu	c18797fc7f6c96d208fdef3b80ec349cc4869648	PE/DLL	1	122 KB

Pivoting on original name

All collected samples have their configuration placed in the *.data* section, and use the same AES key “**2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C**” IV: “**00**”. Configurations were extracted from all the samples and subsequently decrypted with the provided key. The last thing to do was to collect decrypted C2 domains and IPs, and create an IOC list provided at the end of this blog.

Development history

The samples were grouped based on the described findings, and one representative sample from each group was put into a table. The table was then sorted based on the compilation timestamp.

SHA1	Compile timestamp	First seen timestamp	Size	File Type	Original name
f1a1ea963ae8aca3a4623912c405cc97df510c07	2015-12-22 T07:15:44	12/23/2015 4:40	155.5KB	PE/DLL	mm.dll
859e0f0ccbcafd25b0877a0c6df0c94cd84d2433	2016-04-06 T08:44:22	7/18/2018 1:03	466KB	PE/DLL	mm_tcp_dll.dll
4118cc4ee6e22bca1933b0033cfe07924293b6bb	2017-07-04 T09:20:21	6/6/2019 12:18	134KB	PE/DLL	mm_tcp_svchost.dll
de7b0889fce6e38ac4f902e2399c9a794f8f00df	2018-04-25 T01:36:11	8/3/2018 13:31	154.5KB	PE/DLL	mm.dll
22c55ded3486614728eaa29a7526d760ac496b20	2018-09-21 T01:51:04	11/20/2018 13:19	179.5KB	PE+/DLL	mm.dll

Two oldest groups of samples have an extra layer of encryption used for hiding the part of the code responsible for AES decryption. At some point in 2017, the malware developers decided to simplify the code and move that extra protection layer, probably to another PE artifact (the loader component). The fact that the same AES key has been used for more than 4 years is a lucky coincidence which simplifies the IOC extraction for researchers and makes correlating samples easier. Older versions of Taidoor mentioned in this blog use a different encryption algorithm, but have a highly similar set of exported functions, and also very similar high-level program logic.

Conclusion

Like any other software product, malware families require a lot of maintenance and improvement to achieve long-term operability. Even though such continuous upgrading helps malware avoid detection mechanisms, it also results in related malware versions. These related versions must have some similarities, and this is an opportunity for security researchers to establish correlations between various malware campaigns and quickly find more threat samples.

As part of this research, and based on the analysis provided in the referenced threat report, 23 related samples were found and 40 new C2 IPs and domains extracted from their configurations.

Titanium Platform helps researchers get a more detailed insight into malware samples. Various search queries can be constructed from the extracted metadata, and can be used to find related threats. With Titanium Platform you can also establish time relationships between the samples, and together with the differences in malware functionality, this can enable you a better understanding of malware development history.

IOC list

The following link contains the data extracted from the newly discovered samples related to the Taidoor malware.

https://blog.reversinglabs.com/hubfs/Blog/Taidoor_SHA1_list.txt

https://blog.reversinglabs.com/hubfs/Blog/Taidoor_C2_list.txt

MORE BLOG ARTICLES
