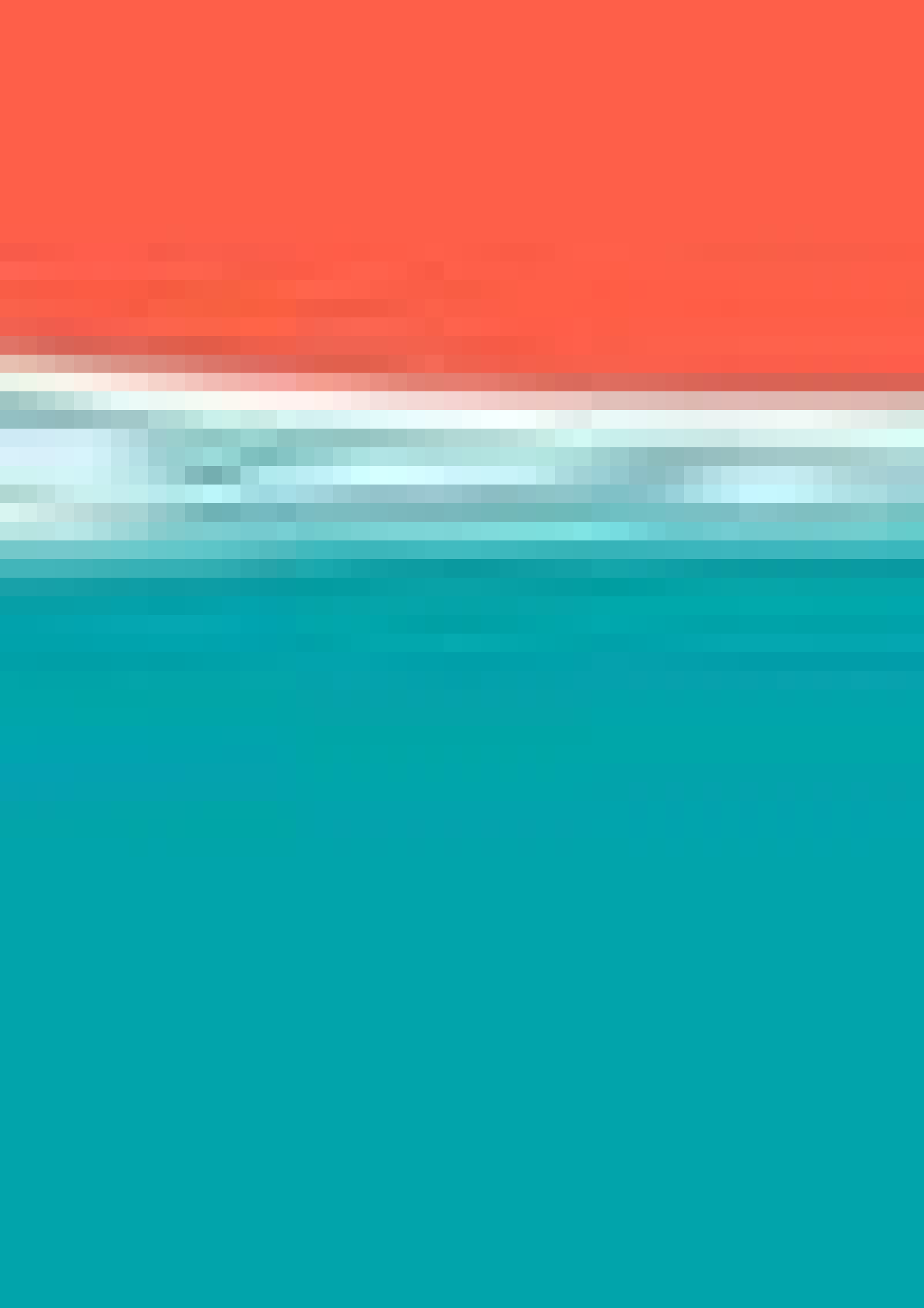

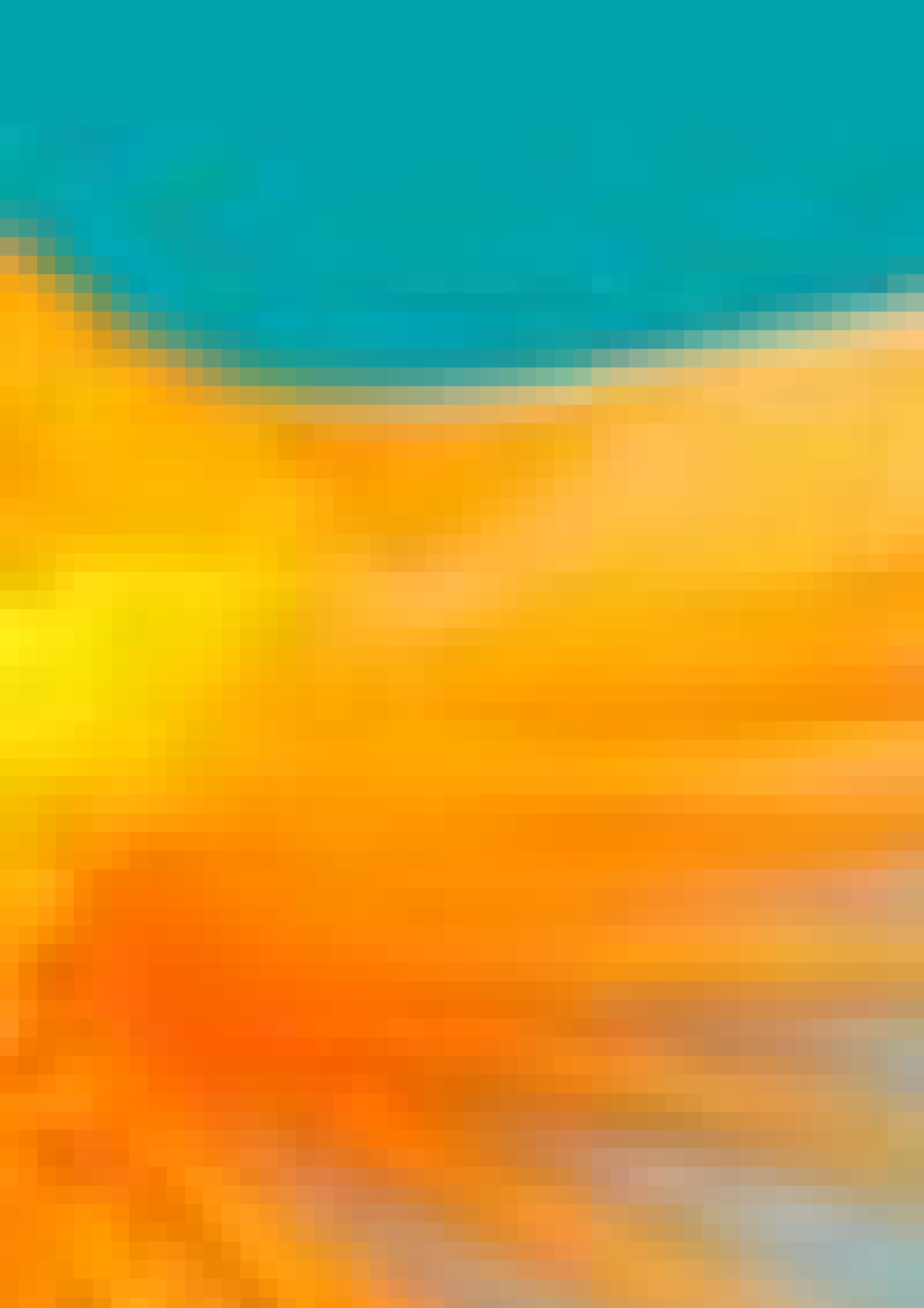

Fake or Fake: Keeping up with OceanLotus decoys

ESET researchers detail the latest tricks and techniques OceanLotus uses to deliver its backdoor while staying under the radar

Romain Dumont 20 Mar 2019 - 11:28AM

Share

This article will first describe how the OceanLotus group (also known as APT32 and APT-C-00) recently used

one of the publicly available exploits for CVE-2017-11882, a memory corruption vulnerability present in Microsoft Office software, and how OceanLotus malware achieves persistence on compromised systems without leaving any traces. Then the article describes how, since the beginning of 2019, the group has been leveraging self-extracting archives to run code.

Context

Following OceanLotus' activities is taking a tour in the world of deception. This group is known to lure victims by forging appealing documents to entice potential victims into executing the group's backdoor, and keeps coming up with new ideas to diversify its toolset. The techniques employed for the decoys range from files with so-called double extensions, self-extracting archives and macro-enabled documents, to reusing known exploits. On top of that, they are very active and relentlessly continue to raid their favourite victims, South East Asian countries.

Summing up the Equation Editor exploit

In mid-2018, OceanLotus carried out a campaign using documents abusing the weakness exposed by the CVE-2017-11882 vulnerability. Indeed, several Proofs-of-Concept were made available. The vulnerability resides in the component responsible for rendering and editing mathematical equations. One of the malicious documents used by OceanLotus was analysed by 360 Threat Intelligence Center (in Chinese) and includes details about the exploit. Let's take a look at a similar document.

First stage

This document FW Report on demonstration of former CNRP in Republic of Korea.doc (`SHA-1: D1357B284C951470066AAA7A8228190B88A5C7C3`) is similar to the one mentioned in the article above, and also interesting as it really targets people interested in Cambodian politics (the CNRP – Cambodia National Rescue Party – political party was dissolved in late 2017). Despite its .doc extension, the document is actually in RTF format (see Figure 1), contains many garbage groups, and is also malformed.

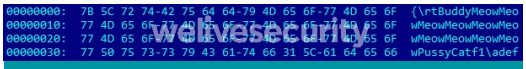

Figure 1 — RTF garbage fields

Despite the presence of malformed elements, Word successfully opens this RTF file. As seen in (Figure 2), at offset 0xC00 there is an EQNOLEFILEHDR structure, followed by the MTEF header and then an MTEF record (Figure 3) for a font.

Figure 2 — FONT record values

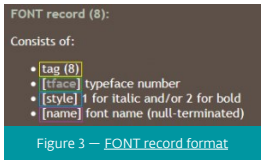

Figure 3 — FONT record format

An overflow in the *name* field is possible because its size isn't checked before being copied. A name that is too long triggers the vulnerability. As seen in the RTF file content (offset 0xC26 in Figure 2), the buffer is filled with shellcode followed by a NOP (0x90) sled and the return address `0x402114`. That address is a gadget in `EQNEDT32.exe` pointing to a `RET` instruction. This results in `EIP` pointing at the beginning of the *name* field which contains the shellcode.

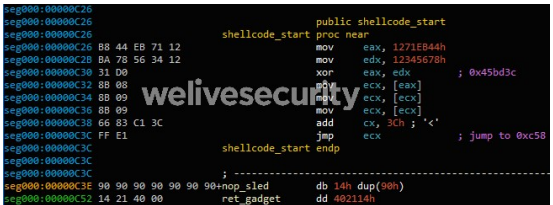

Figure 4 — Start of the exploit shellcode

The address `0x45BD3C` stores a variable that is dereferenced until it reaches a pointer to the currently loaded `MTEFData` structure. That is where the rest of the shellcode resides.

The purpose of the shellcode is to execute a second piece of shellcode, embedded inside the open document. First, the initial shellcode tries to find the handle of the open document file by iterating through all the system's handles (`NtQuerySystemInformation` with the `SystemExtendedHandleInformation` argument) and checking if the handle's *PID* matches the *PID* of a `WinWord` process and if the document was opened with the following access mask: `0x12019F`. To confirm it found the right handle and not the handle of another open document, the content of the file is mapped with the `CreateFileMapping` function and the shellcode checks if the last four bytes of the document are "`yyyy`"; this technique is called "Egg Hunting". Once it finds a match, the document is copied to a temporary folder (`GetTempPath`) as `ole.dll`. Then the last 12 bytes of the document are read.

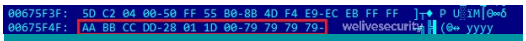

Figure 5 — Markers at the end of the document

The 32-bit value between the `AABBCCDD` and `yyyy` markers is the offset to the next shellcode. It is invoked using the `CreateThread` function. The extracted shellcode is the same that the OceanLotus group has been using for a while now. The Python emulator script we released in March 2018 still works to dump the next stage.

Second stage

Extracting the components

The filenames and directories are chosen dynamically. The code randomly selects the filename of an executable or DLL file located in `C:\Windows\system32`. It will then query its resources and extract the `FileDescription` field to use as a folder name. If this does not work, the code randomly chooses a folder name from the `%ProgramFiles%` or `C:\Windows` (from GetWindowsDirectoryW) directories. It avoids using a name that may clash with existing files by making sure it does not contain: `windows, Microsoft, desktop, system, system32` or `syswow64`. If the directory already exists, the directory name is appended with "NLS_{6 digits}".

The stage's `0x102` resource is parsed and the files are dropped in either `%ProgramFiles%` or `%AppData%` in the randomly chosen folder. The creation times are changed to have the same values as `kernel32.dll`.

For example, here is a folder and a list of files created by picking the `C:\Windows\system32\TCPSVCS.exe` executable as a source of data.

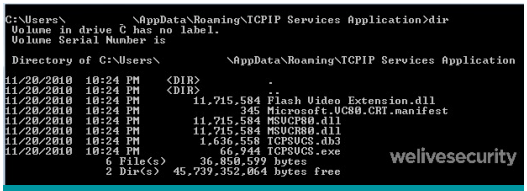

Figure 6 — Extraction of the different components

The structure of the resource 0x102 in the dropper is quite complex. In a nutshell, it contains:

- filenames
- files' size and content
- compression format (`COMPRESSION_FORMAT_LZNT1` used by `RtlDecompressBuffer` function)

The first file is dropped as `TCPSVCS.exe` which is in fact Adobe's legitimate `AcroTranscoder.exe` (according to its `FileDescription`, SHA-1: `2896738693A8F36CC7AD83EF1FA46F82F32BE5A3`).

You may have noticed that the file size of some DLLs exceeds 11MB. This is because a large contiguous buffer of random data is placed inside the executable. It is possibly a way to evade detection by some security products.

Achieving persistence

The resource `0x101` of the dropper contains two 32-bit integers that dictate how the persistence should be implemented. The value of the first one specifies how the malware will achieve persistence without administrator privileges.

First integer value	Persistence mechanism
0	Do not achieve persistence
1	Scheduled task as current user

First integer value	Persistence mechanism
2	(HKLM|HKCU)\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
3	Creation of a shortcut file (with a .lnk extension) in the subdirectory Microsoft\Windows\Start Menu\Programs\Startup under one of the environment variables: %ALLUSERSPROFILE%, %APPDATA% or %USERPROFILE%

The value of the second integer specifies how the malware should try to achieve persistence if it runs with elevated privileges.

Second integer value	Persistence mechanism
1	Scheduled task as administrator
2	Creation of a service

The service name is the filename without extension; the display name is the folder name but if it already exists then the string "`Revision 1`" is appended (the number is incremented until it finds an unused name). The operators made sure the persistence through the service would be resilient: on service failure, the service should restart after 1 second. Then, the registry value WOW64 of the new service key is set to 4 which indicates that it's a 32-bit service.

The scheduled task is created via several COM interfaces: `ITaskScheduler`, `ITask`, `ITaskTrigger`, `IPersistFile` and `ITaskScheduler`. Essentially, the malware creates a hidden task, sets the account information with the current user or the administrator information and sets the trigger.
This is a daily task with a duration of 24 hours and the interval between two executions is set to 10 minutes, which means it will run all the time.

The malicious bit

In our example, the executable `TCPSVCS.exe` (`AcroTranscoder.exe`) is legitimate software side-loading the DLLs that were dropped with it. In this case, the `Flash Video Extension.dll` is the interesting one.

Its `DLLMain` function just calls a single function. Some opaque predicates are present:

```
result = 0;
if ( "0E8j2kP9zUe9VdsFEfg2H1BV3EZAbhSKZjI52IH2pvAGji18Bi7abkTQf0ebqLPbL3erPszpTar6BuA" )
{
  result = StrStrIA((LPCSTR)"0E8j2kP9zUe9VdsFEfg2H1BV3EZAbhSKZjI52IH2pvAGji18Bi7abkTQf0ebqLPbL3erPszpTar6BuA", "0");
  if ( result )
  {
    if ( result <= (LPSTR)"2kP9zUe9VdsFEfg2H1BV3EZAbhSKZjI52IH2pvAGji18Bi7abkTQf0ebqLPbL3erPszpTar6BuA" )
```


Figure 7 — Opaque predicates

After these deceptive checks, the code gets the `.text` section of `TCPSVCS.exe`, changes its protection to `PAGE_EXECUTE_READWRITE` and overwrites it with do-nothing instructions that have no side effects:

```
nopsled:
        dec     ecx
        push    edi
        push    edi
        nop
        xchg    eax, ebx
        dec     ecx
        xchg    eax, ebx
        dec     ebx
        xchg    eax, ebx
        xchg    eax, ebx
        dec     edx
        dec     edx
        xchg    eax, ebx
        dec     edx
        xchg    eax, ebx
        push    ebp
        push    ecx
        inc     ebx
        inc     edx
        dec     eax
        nop
```


Figure 8 — Sequence of instructions without side effects

This function simply creates a mutex starting with `{181C8480-A975-411C-AB0A-630DB8B0A221}` and followed by the current username. Then, it reads the dropped file with the `.db3` extension, which contains position-independent code, and uses `CreateThread` to execute its content.

The content of the .db3 file is shellcode commonly used by OceanLotus. Again, we successfully unpacked its payload using the emulator script we published on GitHub.

The script extracts the final stage. This component is the backdoor that we already analysed in this white paper:

OceanLotus: Old techniques, new backdoor. It is recognizable as such from the GUID `{A96B020F-0000-466F-A96D-A91BBF8EAC96}` that is present in the binary. The configuration of the malware is still encrypted in a PE resource. It contains almost the same configuration but the C&C servers are different from the ones that were already published:

- `andreagahuvrauvin[.]com`
- `byronorenstein[.]com`
- `stienollmache[.]xyz`

Once again OceanLotus showcases a large combination of techniques to stay under the radar. They came back with a "better" version of the infection process. By choosing random names and filling executables with random data, they reduce the number of reliable IoCs (hash-based and filename-based). Moreover, since they're using DLL side-loading, the attackers only have to drop the legitimate `AcroTranscoder` binary as-is.

Self-Extracting archives

After using RTF files, the group started using self-extracting (SFX) archives that use common document icons in an attempt to further mislead their victims. It was briefly documented by Threatbook (in Chinese). When run, these self-extracting RAR files drop and execute DLL files (with a .ocx extension) with the final payload being the previously documented `{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll`. Since the middle of January 2019, OceanLotus began reusing the technique but changed some configuration over time. This section will describe the technique and what they have altered to achieve their goal.

Falling for the decoy

The document `THICH-THONG-LAC-HANH-THAP-THIEN-VIET-NAM (1).EXE` (meaning "FAVORITE RELATIONSHIP OF VIETNAMESE PERFORMANCE" according to Google Translate, SHA-1: `AC10F5B1D5ECAB22B7B418D6E98FA18E32BBDEAB`) was first seen in 2018. This SFX file is cleverly crafted, as the description (*Version Info*) states it's a "JPEG Image". The script of the SFX is the following:

```
;The comment below contains SFX script commands

Setup=regsvr32 /s /i {9ec60ada-a200-4159-b310-8071892ed0c3}.ocx
Setup=regsvr32 /s /i {9ec60ada-a200-4159-b310-8071892ed0c3}.ocx
Setup="2018 thich thong lac.jpg"
TempMode
Overwrite=1
```


Figure 9 — SFX commands

The malware drops `{9ec60ada-a200-4159-b310-8071892ed0c3}.ocx` (SHA-1: `EFAC23B0E6395B1178BCF7086F72344B24C04DCC`) as well as the image 2018 `thich thong lac.jpg`.

The decoy image is the following:

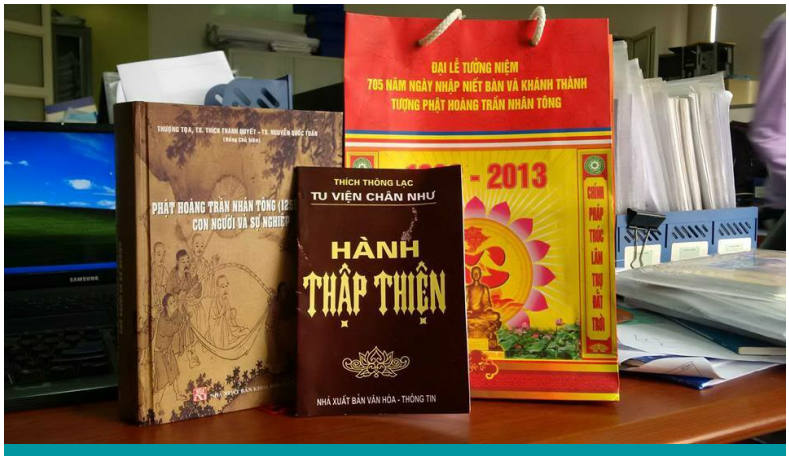

Figure 10 — Decoy image

You may have noticed the first two lines in the SFX script invoke the OCX file twice, but it is not a mistake...

{9ec60ada-a200-4159-b310-8071892ed0c3}.ocx (ShLd.dll)

The OCX file's control flow is very similar to other OceanLotus components: there are a lot of `JZ/JNZ and PUSH/RET` instruction sequences interleaved with junk code.

```
loc_10002321:                           ; CODE XREF: .text:10002634↓j
                                        ; DATA XREF: .text:1000265E↓o
                lea     eax, [esp+8]
                push    eax
                jmp     loc_10002384
; ---------------------------------------------------------------------------
                push    0B7C020h
                mov     eax, 1053668Ch
                mov     edx, ds:1153FACCh
```

```
                cmp     esi, ds:4116F0h
                push    0A7F6D4h
                mov     dx, [esi+9]
                jmp     dword ptr ds:0D8EC1Ch
; ---------------------------------------------------------------------------
                dd 0FE7485C7h, 29DCFFFFh, 85890049h, 0FFFFFCE4h, 081C1588h
                dd 688800CDh, 8400F41Ah, 0A98068D8h, 0BF76009Fh, 6537D6C1h
                dd 5000AF5Ah, 0DFC880Ah, 25596AFEh
; ---------------------------------------------------------------------------

loc_10002384:                           ; CODE XREF: .text:10002326↑j
                push    0
                push    0F003Fh
                push    0
                push    0
                jz      loc_100020B6
                jnz     loc_100020B6
                mov     eax, ds:11558650h
                push    114796FCh
                push    68EBD569h
                sbb     [esi+18h], ebp
                mov     eax, ds:11541618h
                mov     ds:1153FE9Ch, eax
                mov     [ebx+50h], eax
                fld     qword ptr ds:1141CE08h
```


Figure 11 — Obfuscated code

After filtering the junk code, the export `DllRegisterServer` called by `regsvr32.exe` looks like this:

```
LSTATUS sub_10001F70()
{
  LSTATUS result; // eax
  LSTATUS v1; // edi
  int DllBaseAddress; // esi
  BYTE Data[4]; // [esp+7962h] [ebp-10h]
  HKEY phkResult; // [esp+7966h] [ebp-Ch]
  DWORD cbData; // [esp+796Ah] [ebp-8h]
  DWORD Type; // [esp+796Eh] [ebp-4h]

  phkResult = 0;
  result = RegCreateKeyExW(
             HKEY_CURRENT_USER,
             L"SOFTWARE\\Classes\\CLSID\\{E08A0F4B-1F65-4D4D-9A09-BD4625B9C5A1}",
             0,
             0,
             0,
             KEY_ALL_ACCESS,
             0,
             &phkResult,
             0);
  if ( phkResult )
  {
    DllBaseAddress = f_GetDllBaseAddress();
    Type = 0;
    *(_DWORD *)Data = 0;
    cbData = 4;
    v1 = RegQueryValueExW(phkResult, L"Model", 0, &Type, Data, &cbData);
    Sleep(0x3E8u);
    if ( v1 || !*(_DWORD *)Data || cbData < 4 || Type != 4 )
    {
      *(_DWORD *)Data = 0x125211 - DllBaseAddress + f_Ret_10001DE0();
      RegSetValueExW(phkResult, L"Model", 0, 4u, Data, 4u);
      result = RegCloseKey(phkResult);
    }
    else
    {
      RegDeleteValueW(phkResult, L"Model");
      *(_DWORD *)Data = *(_DWORD *)Data + DllBaseAddress - 0x125211;
      (*(void (__stdcall **)(int))Data)(DllBaseAddress);
      result = RegCloseKey(phkResult);
    }
  }
  return result;
```


Figure 12 — Main code of the installer

Basically, the first time the `DllRegisterServer` is called, it sets the registry value `HKCU\SOFTWARE\Classes\CLSID\{E08A0F4B-1F65-4D4D-9A09-BD4625B9C5A1}\Model` to an encoded offset in the DLL (`0x10001DE0`).
The second time the function is called, it reads this very same value and executes the function at that address. From there, the resource is read and executed and many in-memory operations are executed.

The shellcode is the same PE loader used in the earlier OceanLotus campaigns. It can be emulated with our miasm emulation script. Ultimately, it drops `db293b825dcc419ba7dc2c49fa2757ee.dll`, loads it into memory and executes `DllEntry`.

The DLL retrieves the content of its resource, decrypts (AES-256-CBC) and decompresses it (LZMA). The resource has a specific format that is quite easy to reverse engineer.

```
[-] [root]
  [.] total_length = 1126732
  [-] data_n
    [.] folders_len = 174
    [-] folders
      [-] folder (2 = 0x2 entries)
        [-] 0
          [.] str_len = 90
          [.] str_buf = "%appdata%\\Intel\\logs\\BackgroundUploadTask.cpl"
        [-] 1
          [.] str_len = 76
          [.] str_buf = "%%windir%%\\%s\\BackgroundUploadTask.cpl"
    [-] binary
      [.] bin_len = 1126400
      [.] bin_data = 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    [.] config_len = 146
    [-] control
      [.] str_len = 58
      [.] str_buf = "%windir%\\System32\\control.exe"
    [-] name
      [.] str_len = 80
      [.] str_buf = "BackgroundUploadTask\u009D\u009D\u009D\u00A0 \u009D \u00A0\u00A0 \u00A0\u00A0 \u00A0\u009D  \u00A0\u00A0\u009D"
```


Figure 13 — Structure of the installer configuration (KaitaiStruct Visualizer)

The configuration is explicit: depending on the privilege level, the binary data will be written to either `%appdata%\Intel\logs\BackgroundUploadTask.cpl` or `%windir%\System32\BackgroundUploadTask.cpl` (or `SysWOW64` for 64-bit systems).
Next, persistence is achieved by creating a task named `BackgroundUploadTask[junk].job` where a `[junk]` is a collection of `0x9D` and `0xA0` bytes.
The application name of the task is `%windir%\System32\control.exe` and the parameter value is the path of the dumped binary. The hidden task is set to run every day.

Structurally, the CPL file is a DLL whose internal name is `ac8e06de0a6c4483af9837d96504127e.dll` and that exports a `CPlApplet` function. This file decrypts its only resource `{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll`, then loads that DLL and calls its only export, `DllEntry`.

Backdoor configuration file

The backdoor has an encrypted configuration embedded in its resources. The structure of the configuration file is quite similar to the previous one.

Figure 14 — Structure of the backdoor configuration (KaitaiStruct Visualizer)

Despite the structural similarity, of the values in many of these fields have been updated comparing this to that in our white paper from March 2018.
The first element of the binaries array contains a DLL (`HttpProv.dll` MD5: `2559738D1BD4A999126F900C7357B759`) identified by Tencent but as the export name has been removed from the binary, the hashes don't match.

Going the extra mile

While hunting for samples, a few characteristics stood out. The sample just analysed appeared around July 2018 and other similar were found very recently in mid-January through early-February 2019. The infection vector used was an SFX archive dumping a legitimate, decoy document and a malicious OCX file.

Even though OceanLotus uses fake timestamps, it has been observed that the timestamp of the SFX and OCX files are always the same (`0x57B0C36A` (08/14/2016 @ 7:15pm UTC) and `0x498BE80F` (02/06/2009 @ 7:34am UTC) respectively). This probably means that they have some kind of "builder" that reuses the same templates and just changes some characteristics.

Among the documents we analysed since early-2018, we saw different document names suggesting country-related targeting:

- *The New Contact Information Of Cambodia Media(New).xls.exe*
- *李建香 （个人简历）.exe* (fake pdf document of a CV)
- *feedback, Rally in USA from July 28-29, 2018.exe*

Since the discovery of the `{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll` backdoor and its public analysis by multiple researchers, we observed some changes in the malware's configuration data.
First, the authors started removing the names from the helper DLLs (`DNSprov.dll` and the two versions of `HttpProv.dll`).
Then the operators stopped packaging the third DLL (second version of `HttpProv.dll`), choosing to embed just one.

Second, a lot of the backdoor configuration fields have been changed, perhaps to avoid detection, since many IoCs became available.
The important fields that changed are the following:

- the "AppX" registry key changed (see IoCs)
- the mutex encoding string ("def", "abc", "ghi")
- the port number

Finally, all the new variants analysed have new C&C servers, which are listed in the IoCs section.

Conclusion

OceanLotus is very active and keeps evolving. The group really focuses on varying their toolsets and decoys. They cleverly wrap their payloads with attractive documents based on current events that are likely to be of interest to their intended victims. They keep coming up with different techniques and even reuse and readapt publicly available exploit code such as for the Equation Editor exploit. Moreover, they keep improving their techniques to reduce the number of artefacts left on their victims' machines, thereby reducing the odds of detection by security products. As we have shown, a lot of in-memory operations are involved, filenames are randomly generated and the OceanLotus operators have modified their binaries to avoid being detected. Another very interesting point is that some domain names seem to be derived from a dictionary. OceanLotus is making the extra effort to continue carrying out their campaigns, but don't hold your breath...

Indicators of Compromise (IoCs)

The IoCs in this blogpost, as well as the MITRE ATT&CK attributes, are also available from our GitHub repository.

Registry keys/values:

- `HKCU\SOFTWARE\Classes\CLSID\{E08A0F4B-1F65-4D4D-9A09-BD4625B9C5A1}\Model`
- `[HKCU|HKLM]\SOFTWARE\App\`
  - `AppXbf13d4ea2945444d8b13e2121cb6b663\`
    - `Application`
    - `DefaultIcon`
  - `AppX70162486c7554f7f80f481985d67586d\`
    - `Application`
    - `DefaultIcon`
  - `AppX37cc7fdccd644b4f85f4b22d5a3f105a\`
    - `Application`
    - `DefaultIcon`

Mutexes:

- `{181C8480-A975-411C-AB0A-630DB8B0A221}_` (+ username)

Domain names
aliexpresscn[.]net
andreagahuvrauvin[.]com
andreagbridge[.]com
aol.straliaenollma[.]xyz
beaudrysang[.]xyz
becreybour[.]com
byronorenstein[.]com
chinaport[.]org
christienoll[.]xyz
christienollmache[.]xyz
cloud.360cn[.]info
dieordaunt[.]com
dns.chinanews[.]network
illagedrivestralia[.]xyz
karelbecker[.]com
karolinblair[.]com
lauradesnoyers[.]com
ntop.dieordaunt[.]com
office.ourkekwiciver[.]com
ourkekwiciver[.]com
sophiahoule[.]com
stienollmache[.]xyz

Domain names
straliaenollma[.]xyz
ursulapapst[.]xyz

Files:

Documents exploiting CVE-2017-11882:
SHA-1 hashes
D1357B284C951470066AAA7A8228190B88A5C7C3
49DFF13500116B6C085C5CE3DE3C233C28669678
9DF3F0D8525EDF2B88C4A150134C7699A85A1508
50A755B30E8F3646F9476080F2C3AE1347F8F556
BB060E5E7F7E946613A3497D58FBF026AE7C369A
E2D949CF06842B5F7AE6B2DFFAA49771A93A00D9
ESET detection names
Win32/Exploit.CVE-2017-11882.BU
Win32/Exploit.CVE-2017-11882.A
Win32/Exploit.Agent.KT
Win32/Exploit.Agent.LT
Win32/Exploit.CVE-2017-11882.EI
SFX archives and OCX droppers:
SHA-1 hashes
AC10F5B1D5ECAB22B7B418D6E98FA18E32BBDEAB
7642F2181CB189965C596964D2EDF8FE50DA742B
CD13210A142DA4BC02DA47455EB2CFE13F35804A
377FDC842D4A721A103C32CE8CB4DAF50B49F303
B4E6DDCD78884F64825FDF4710B35CDBEAABE8E2
BD39591A02B4E403A25AAE502648264308085DED
B998F1B92ED6246DED13B79D069AA91C35637DEC
CC918F0DA51794F0174437D336E6F3EDFDD3CBE4
83D520E8C3FDAEFB5C8B180187B45C65590DB21A
EFAC23B0E6395B1178BCF7086F72344B24C04DCC
8B991D4F2C108FD572C9C2059685FC574591E0BE
B744878E150A2C254C867BAD610778852C66D50A
3DFC3D81572E16CEAAE3D07922255EB88068B91D
77C42F66DADF5B579F6BCD0771030ADC7AEFA97C
ESET detection names
Win32/Agent.ZUR

MITRE ATT&CK techniques

Tactic	ID	Name	Description
Initial Access	T1193	Spearphishing Attachment	Deceitful RTF documents and self-extracting archives are sent to potential victims.
Execution	T1204	User Execution	The user needs to execute the self-extracting archive or open the RTF document.
	T1117	Regsvr32	The self-extracting archives execute `regsvr32` to run the OceanLotus' backdoor.
	T1035	Service Execution	The second stage of the exploit tries to run OceanLotus' backdoor as a service.
Persistence	T1050	New Service	The second stage of the exploit tries to achieve persistence by creating a service.
	T1060	Registry Run Keys / Start Folder	The second stage of the exploit tries to achieve persistence by adding a value in the Run registry key.
	T1053	Scheduled Task	The second stage of the exploit tries to achieve persistence by creating a schedule task.

Tactic	ID	Name	Description
Defense Evasion	T1009	Binary Padding	The second stage of the exploit fills dropped executables with random data.
	T1073	DLL Side-Loading	OceanLotus' backdoor is side-loaded by dropping a library and a legitimate, signed executable (AcroTranscoder).
	T1112	Modify Registry	OceanLotus' backdoor stores its configuration in a registry key.
	T1027	Obfuscated Files or Information	The second stage of the exploit drops an encrypted shellcode.
	T1099	Timestomp	The creation time of the files dropped by the second stage of the exploit is set to match the creation time of `kernel32.dll`.
Discovery	T1083	File and Directory Discovery	OceanLotus' backdoor can list files and directories.
	T1012	Query Registry	OceanLotus' backdoor can query the Windows Registry to gather system information.
	T1082	System Information Discovery	OceanLotus' backdoor captures system information and sends it to its C&C server.
Exfiltration	T1002	Data Compressed	OceanLotus' backdoor uses LZMA compression before exfiltration.
	T1022	Data Encrypted	OceanLotus' backdoor uses RC4 encryption before exfiltration.
	T1041	Exfiltration Over Command and Control Channel	Data exfiltration is done using the already opened channel with the C&C server
	T1203	Exploitation for Client Execution	The RTF document includes an exploit to execute malicious code. (CVE-2017-11882)
Command And Control	T1094	Custom Command and Control Protocol	OceanLotus' backdoor can exfiltrate data by encoding it in the subdomain field of DNS packets.
	T1065	Uncommonly Used Port	OceanLotus' backdoor use HTTP over an uncommon TCP port (14146). Port is specified in the backdoor configuration.

Romain Dumont 20 Mar 2019 - 11:28AM

Similar Articles

Tracking Turla: New backdoor delivered via Armenian watering holes

Guildma: The Devil drives electric

Up close and personal with Linux malware

Linux and malware: Should you worry?

Discussion