Tracking Seven Years' Evolution of the Enfal RAT

EXPOSING THE INNER WORKINGS OF HIGHLY EVASIVE MALWARE

Research Report | July 2016

VERINT

TABLE OF CONTENTS

ABSTRACT

The Enfal malware was first spotted around 2004 and gained serious traction in cyber attack operations in 2008. This advanced malware has continued to evolve over the past seven or eight years, particularly in terms of the sophistication of its encryption methods. In recent years, different versions of Enfal (aka GoldSun) have been involved in a number of high-profile cyber attacks. The PittyTiger APT campaign exposed by Airbus Defense and Space in 2014, which targeted private sector and government organizations, featured a RAT that used the Enfal protocol for communications, but with RC4 algorithms to encrypt data. This variant of the Enfal malware was named MM RAT.

Furthermore, our research revealed deep connections between Enfal and the notorious Taidoor APT backdoor groups. Since 2008, Taidoor malware has been used in cyber espionage campaigns launched against corporations and government agencies with interests in Taiwan. During our research, we came across some Taidoor backdoors that scan the availability of Enfal's Command & Control (C2) IP, implying that the two malwares may use the same C2 protocol.

The Verint Research team has been monitoring the Enfal malware since 2008. To our knowledge, we were the first to discover and track the evolution of this constantly morphing malware. Moreover, we have been observing the differences between variants of the Enfal malware over the past seven years. Within the framework of our security investigations, we were able to access a compromised Enfal C2 server, which enabled us to discover and analyze the Enfal backend. We performed detailed forensics on the C2 server, and were able to reverse engineer the inner workings of Enfal's communication protocol.

The comprehensive technical analysis in this report is based on our observation and research into different versions of Enfal over an extended period of time. The report introduces findings related to sophisticated API obfuscation techniques used to evade malware detection, communications protocols and backend functionality that have yet to be published in other reports on Enfal, most of which cover malware samples and datasets of a particular attack.

ENFAL MALWARE OVERVIEW

The Enfal malware received its name based on the fact that its earlier instances contained the "NFal" string in the code. This is no longer the case in most recent findings. Basic functions of the Enfal malware include collecting details of compromised computers, accessing command files, running programs, enumerating files, file transmission, opening command shell, and so on. In the following sections, we will describe the distinct characteristics of Enfal, from its early versions to the more recent PittyTiger MM RAT. Some AV vendors also refer to Enfal as GoldSun, as it has also named its mutex sample as "GoLdSuN."

The source code of Enfal's backend can in fact be searched and downloaded (HttpServer_VC, Fig. 1). This source code only contains the controlling part of Enfal, and its contents are as simple as a student project. However, by virtue of this simplicity, the need for unsynchronized communications between client and server is greatly minimized, and most communications can be transmitted through HTTP. Later versions are based on this source code, adding encryption and decryption functions, and are still being used in APT attacks to this day. In previous reports related to Enfal, insufficient evidence was found to determine which groups are behind the attacks. Verint's team discovered that a control component of Taidoor scans the Enfal C2, enabling us to deduce that the Enfal malware is being utilized by the Taidoor group as a weapon.

```
HttpServer_VC\app.config
.............\Buffer.h
.............\CGI.h
.............\HttpServer.cpp
.............\HttpServer.dll
.............\HttpServer.dsp
.............\HttpServer.dsw
.............\HttpServer.h
.............\HttpServer.plg
.............\HttpServer.rc
.............\HttpServer.sln
.............\HttpServer.suo
.............\HttpServer.vcproj
.............\icon1.ico
.............\KMHook.h
.............\new.reg
.............\old.reg
.............\ReadMe.txt
.............\..lease1\BuildLog.htm
.............\........\httpdocs\mm\computer_00-0C-29-8A-D2-17\Ctime.txt
.............\........\........\..\.........................\Lasttime.txt
.............\........\........\..\.........................\Online.txt
.............\........\........\..\.........................\Reply.sec
```

Figure 1. Some source code of Enfal's backend

ANALYZING SEVEN YEARS OF ATTACK ACTIVITY

We analyzed the compromised data beginning from 2008, and found that the initial attacks were mostly targeted at the US and East Asian countries, such as China and Taiwan. In 2015, the attacks were directed at Taiwan, Indonesia, Malaysia, and Korea, mostly targeting diplomatic organizations and NGOs. Attesting to the stealthiness of the Enfal malware, these organizations were completely unaware that they were under attack prior to being notified by us. Some of these organizations had been compromised for seven years, since the beginning of Enfal's active period.

In March of 2016, Enfal activities were once more discovered in Taiwan government units. Some changes were made to the communication URL, but otherwise most functions remained the same. Using our forensics technology, we were able to cross-check the basic timestamp metadata (create time, last access, last write) against other data sources to confirm that the metadata had been tampered with and that this tampering most likely took place on March 1, 2016. To date, this malware has not yet been seen on VirusTotal.

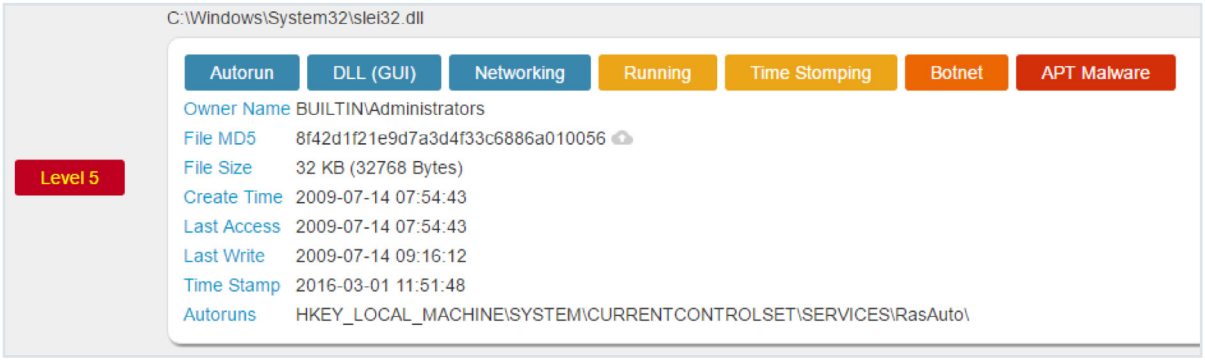


Figure 2. The 2016 version of Enfal (indicating it is still active today)

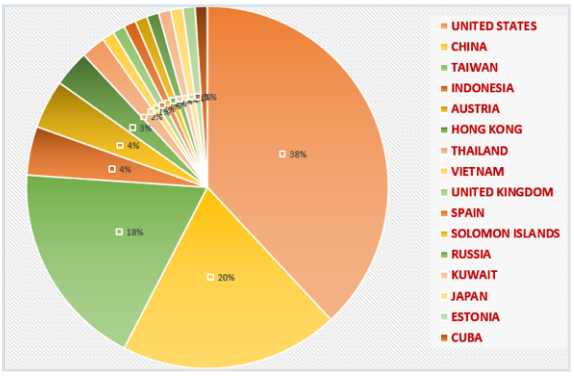


Figure 3. Countries attacked by Enfal in 2008

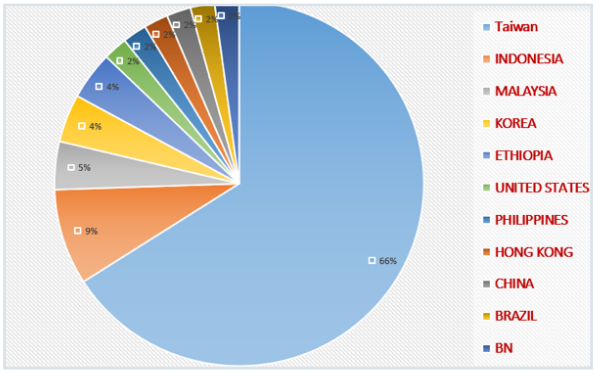


Figure 4. Countries attacked by Enfal in 2015

```
[AMCHAMSERVER01 345-Cs
CreateTime=Thu Aug 28 18:57:38 2008
AMC 2008
95.62.5
RefreshTime=Sat Aug 30 09:47:36 2008
I=192.168.110.1
O=2003 server Standard Edition (Service Pack 2)
L=English(US)
V=3.4.5
M=kchdav.exe
B=selcon.exe
sen=N
Prxy=N
Lg=n
Online='n'
Online=n
WebIpAddress=211.95.62.5
RefreshTime=Sat Aug 30 09:47:36 2008
I=192.168.110.1
O=2003 server Standard Edition (Service Pack 2)
L=English(US)
V=3.4.5
M=kchdav.exe
B=selcon.exe
sen=N
Prxy=N
Lg=n
```

```
[AMC-MAINSERVER
CreateTime=Mon Dec 29 23:52:19 2014
AMC 2015
.251.11.1
RefreshTime=Mon Mar 16 05:46:30 2015
I=192.168.
O=2008S(Service Pack 1)
L=Chinese (Simplified)(CN)
V=3.0.0
M=
B=pmss.dll
sen=N
Prxy=N
Lg=n
Online=y
Online=n
WebIpAddress=114.251.11.1
RefreshTime=Mon Mar 16 05:46:30 2015
I=192.168.88.7
O=2008S(Service Pack 1)
L=Chinese (Simplified)(CN)
V=3.0.0
M=
B=pmss.dll
sen=N
Prxy=N
Lg=n
```

Figure 5. AMC was suspected to have been compromised for seven years

ENFAL TECHNICAL ANALYSIS

Installation

We discovered two methods for installing Enfal malware. The first method is to execute the Enfal malware using a single executable file. The malware would find a specific legitimate process (e.g., explorer.exe) or program (e.g., services.exe or svchost.exe) and inject its own functions and data into the legitimate process. The second method is to drop and inject a DLL file into the specific process. To continue executing the dropper or the malware itself after a reboot, autorun correlated keys are written into the registry.

Possibly Modified AutoRun Registry Path
HKU\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run\Service
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman
HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

API Name Obfuscation

As mentioned, one method used for installing Enfal is to inject its functions into other processes. Prior to injection, the malware would use LoadLibrary and GetProcAddress functions to access the API addresses it needs, collecting them into an array. Then it injects the API array, along with the functions, into the target process. The API is indirectly called by other functions using the array. Thus, we thought that if we could analyze the offsets of API address array and their relations with the respective API, we could learn which entry has access to which API, and there would be no need to reverse engineer the original API. However, further analysis showed that some Enfal malware would obfuscate DLL names with API names, and the ways of doing this varied over different time periods.

These sophisticated obfuscation tricks are designed to evade detection by static AV analysis tools, most of which use signature-based approaches. Obfuscation renders these tools ineffective by disguising the suspicious patterns these tools look for. In our research, we observed six distinct API obfuscation and location methods:

I. Moving 4 bytes multiple times, and setting the terminator character

```
*(_DWORD *)&Text[8] = 'Eeue';
*(_DWORD *)Text = 'tirW';
*(_DWORD *)&Text[4] = 'liFe';
Text[9] = '\0';
p_WriteFile = (int)GetProcAddress(hKernel32, Text);
```

II. XORing the first 4 bytes before using, and setting the terminator character

```
*(_DWORD *)szKernel32Dll = (unsigned int)&unk_21FFAF5 ^ 0x6C6D9FBE;
*(_DWORD *)&szKernel32Dll[4] = '23le';
*(_DWORD *)&szKernel32Dll[8] = 'lld.';
szKernel32Dll[12] = '\0';
```

III. Moving each character one by one

```
szAdvapi32Dll[0] = 'A';
szAdvapi32Dll[1] = 'd';
szAdvapi32Dll[2] = 'v';
szAdvapi32Dll[3] = 'a';
szAdvapi32Dll[4] = 'p';
szAdvapi32Dll[5] = 'i';
szAdvapi32Dll[6] = '3';
szAdvapi32Dll[7] = '2';
szAdvapi32Dll[8] = '.';
szAdvapi32Dll[9] = 'd';
szAdvapi32Dll[10] = 'l';
szAdvapi32Dll[11] = 'l';
szAdvapi32Dll[12] = '\0';
```

IV. Setting 4-byte values first, and XORing the whole buffer in a loop

```
*(_DWORD *)ProcName = 0x40767443u;
*(_DWORD *)&ProcName[4] = 0x68637464u;
*(_DWORD *)&ProcName[8] = 0x647D7047u;
*(_DWORD *)&ProcName[12] = 0x50695474u;
*(_DWORD *)&ProcName[16] = 0x11111111u;
v4 = 0;
do
{
    ProcName[v4] ^= 0x11u;
    ++v4;
}
while ( v4 < 0x10 );
ProcName[16] = '\0';// finally "RegQueryValueExA"
```

V. Using strcpy and strcat to combine strings; some junk calls may appear between normal calls

```
memset((void *)&Dst, 0, 0x20u);
strcpy((char *)&Dst, "Inte");
strcpy((char *)&Dest, "kie");// junk
strcat((char *)&Dst, "rnetOpenA");
v100 = GetProcAddress(v4, &Dst);
if ( v100 )
{
    memset((void *)&Dst, 0, 0x20u);
    strcpy((char *)&Dst, "InternetO");
    strcpy((char *)&Dest, "dfee");// junk
    strcat((char *)&Dst, "penUrlA");
    v77 = GetProcAddress(v4, &Dst);
```

VI. Enfal 2015's new API resolution trick: Partial-matching obfuscation

In the 2015 version of Enfal, we observed a new API analysis technique that customizes its own GetProcAddress functions. As shown in Fig. 6 below, apiInfo is a data structure that allows attackers to send three characters of the API name and its location in order to access the API address. This shelters the malware from static AV engines, which only alert when detecting a certain API. Fig. 7 below shows that the program searches for APIs named p, O, and A in the 3rd, 4th, and 15th characters, respectively. Upon finding HttpOpenRequestA that corresponds to the aforementioned requirements, it then searches through the directory to find its corresponding address.

```
apiInfo.moduleHandle = LoadLibraryA("wininet");
apiInfo.numOfListEntry = 3;
apiInfo.apiCharList    = "p,O,A";
apiInfo.apiCharPosList = "3,4,15";
apiAddress = CustomGetProcAddress(&apiInfo);
// no traditional Win32 GetProcAddress(handle, apiName)
```

Figure 6. Self-defined GetProcAddress API

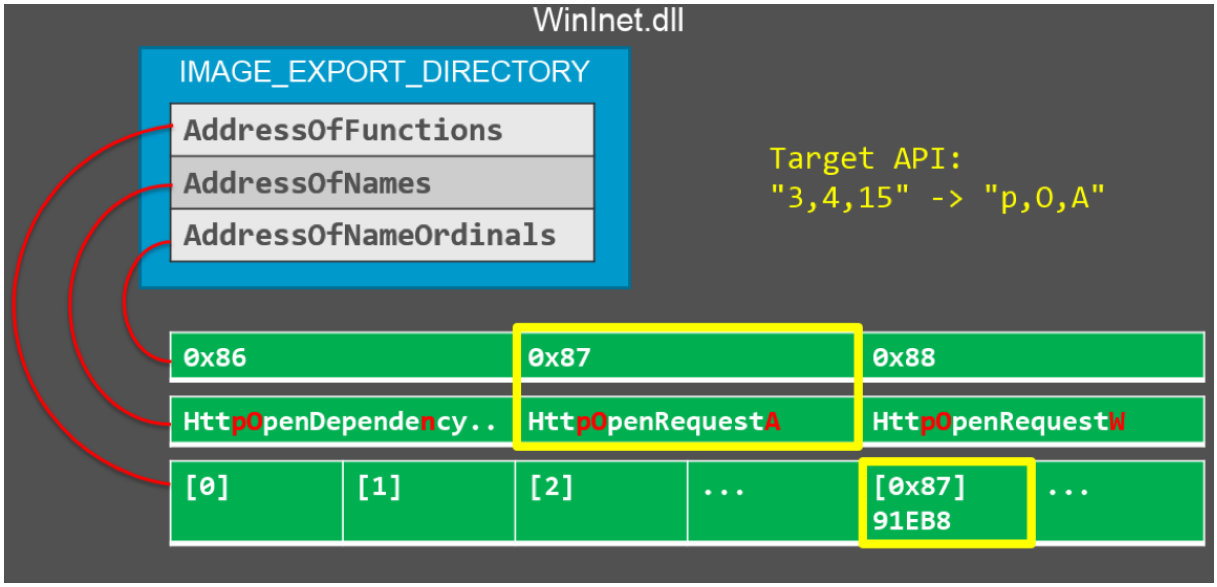


Figure 7. Enfal locates APIs by searching for API characters in their respective locations

Configuration Block Encryption

In the samples we managed to obtain, we saw that some use encryption to put the config block and API name into the executable files. The whole configuration block can be decrypted all at once, or one column at a time. Enfal uses three different decryption methods:

I. Directly XORing each byte with a Constant Byte Value (e.g. 0x55)

```
char *__cdecl DecodeXorStrings()
{
    char *bufBase; // eax@1
    unsigned int bufSize; // ecx@1

    bufBase = "}ff1;0\"x777&|UUUUUUUUUUUUUUUUUU\rWUUyTUU777{\"81&='{6:8
    bufSize = 0x138u;
    do
    {
        *bufBase ^= 0x55u;
        ++bufBase;
        --bufSize;
    }
    while ( bufSize );
    return bufBase;
}
```

```
}ffl;0\"x777&|UUUUUUUUUUUUUUUUUUUU
777{\"81&='{6:8UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU...
84%61{0-0UUUUUUUUUUUUUUUUUUUUUUUUU:90%&{0-0UUUUUUUUUUUUUUUUUUUUUUUUU
&0'#<60&{0-0UUUUUUUUUUUUUUUUUUUUUUU\\
```

```
(339new-bbbs)
bbb.wmdshr.com
3+2=5
mapcd.exe
oleps.exe
services.exe
```

II. XORing Each Byte Using Multiple Fixed Byte Values (equivalent to Single Byte Value)

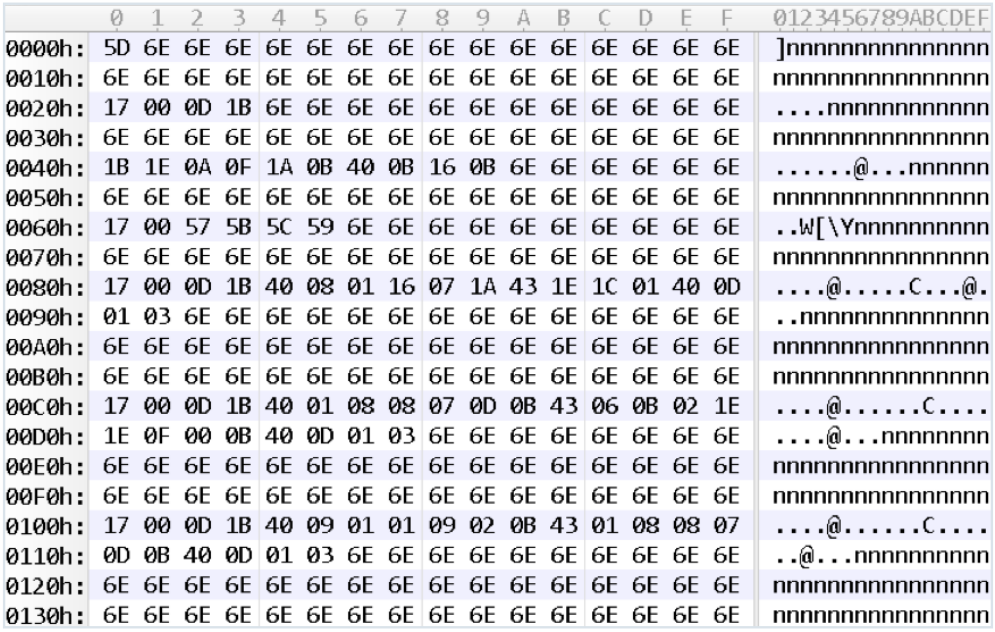
```
void __cdecl DecodeString(BYTE *data, int dataSize)
{
    int i; // ecx@1
    signed int xorSeriesLen; // esi@2

    for ( i = 0; i < dataSize; ++i )
    {
        xorSeriesLen = 10;
        do
            data[i] ^= xorSeries[xorSeriesLen--];// "iloudearmao", strlen==11
        while ( xorSeriesLen >= 0 );
    }
}
```

This method loops byte values and the string “iloudearmao,” then calculates the numerical value of each character using XOR calculations. The numerical value of each character in “iloudearmao” is displayed in hexadecimals as 69 6C 6F 75 64 65 61 72 6D 61 6F, and we are able to calculate the following:

0x69 ^ 0x6C ^ 0x6F ^ 0x75 ^ 0x64 ^ 0x65 ^ 0x61 ^ 0x72 ^ 0x6D ^ 0x61 ^ 0x6F

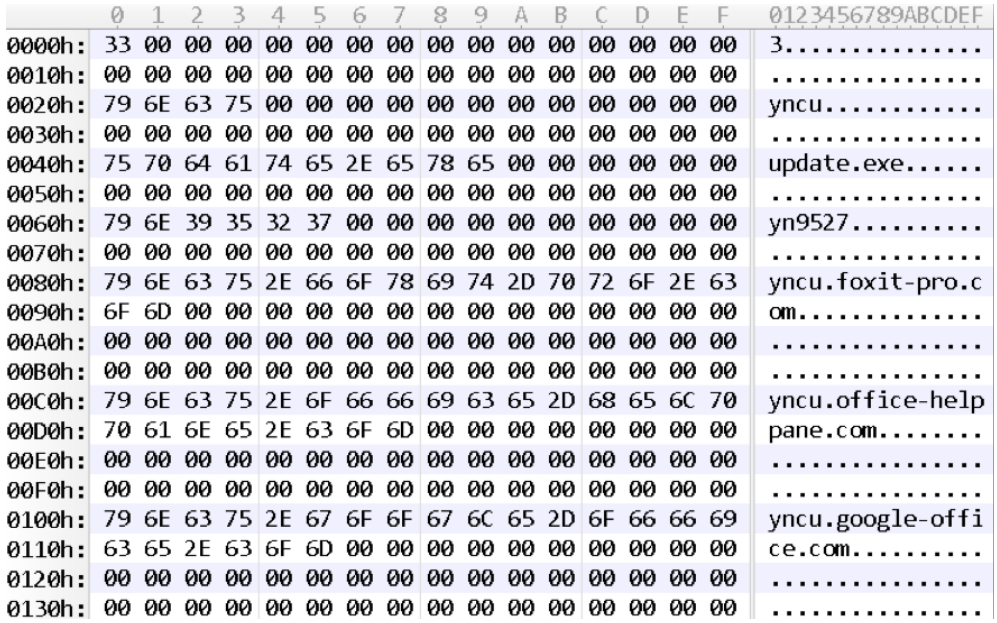
== 110 == ‘n’

The effect of the whole algorithm, in a nutshell, is equivalent to calculating each byte in the data with 110, using XOR calculations. See below for the data images before and after decryption:



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	5D	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	]nnnnnnnnnnnnnnn
0010h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
0020h:	17	00	0D	1B	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	....nnnnnnnnnnnn
0030h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
0040h:	1B	1E	0A	0F	1A	0B	40	0B	16	0B	6E	6E	6E	6E	6E	6E	......@...nnnnnn
0050h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
0060h:	17	00	57	5B	5C	59	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	..W[\Ynnnnnnnnnn
0070h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
0080h:	17	00	0D	1B	40	08	01	16	07	1A	43	1E	1C	01	40	0D	....@.....C...@.
0090h:	01	03	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	..nnnnnnnnnnnnnn
00A0h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
00B0h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
00C0h:	17	00	0D	1B	40	01	08	08	07	0D	0B	43	06	0B	02	1E	....@......C....
00D0h:	1E	0F	00	0B	40	0D	01	03	6E	6E	6E	6E	6E	6E	6E	6E	....@...nnnnnnnn
00E0h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
00F0h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
0100h:	17	00	0D	1B	40	09	01	01	09	02	0B	43	01	08	08	07	....@......C....
0110h:	0D	0B	40	0D	01	03	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	..@...nnnnnnnnnn
0120h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn
0130h:	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	6E	nnnnnnnnnnnnnnnn

Figure 8. Before decryption: Numerous "n" characters, indicating "n" and XOR encryption



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	33	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	3...............
0010h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0020h:	79	6E	63	75	00	00	00	00	00	00	00	00	00	00	00	00	yncu............
0030h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0040h:	75	70	64	61	74	65	2E	65	78	65	00	00	00	00	00	00	update.exe......
0050h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0060h:	79	6E	39	35	32	37	00	00	00	00	00	00	00	00	00	00	yn9527..........
0070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0080h:	79	6E	63	75	2E	66	6F	78	69	74	2D	70	72	6F	2E	63	yncu.foxit-pro.c
0090h:	6F	6D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	om..............
00A0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00B0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00C0h:	79	6E	63	75	2E	6F	66	66	69	63	65	2D	68	65	6C	70	yncu.office-help
00D0h:	70	61	6E	65	2E	63	6F	6D	00	00	00	00	00	00	00	00	pane.com........
00E0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00F0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0100h:	79	6E	63	75	2E	67	6F	6F	67	6C	65	2D	6F	66	66	69	yncu.google-offi
0110h:	63	65	2E	63	6F	6D	00	00	00	00	00	00	00	00	00	00	ce.com..........
0120h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0130h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Figure 9. After decryption: "n" characters are revealed after XOR calculations

Fig. 9 reveals the Enfal C2 location following decryption:

- yncu.foxit-pro.com (95.211.172.143)
- yncu.office-helppane.com (NO LONGER EXISTS)
- ync.google-office.com (NO LONGER EXISTS)

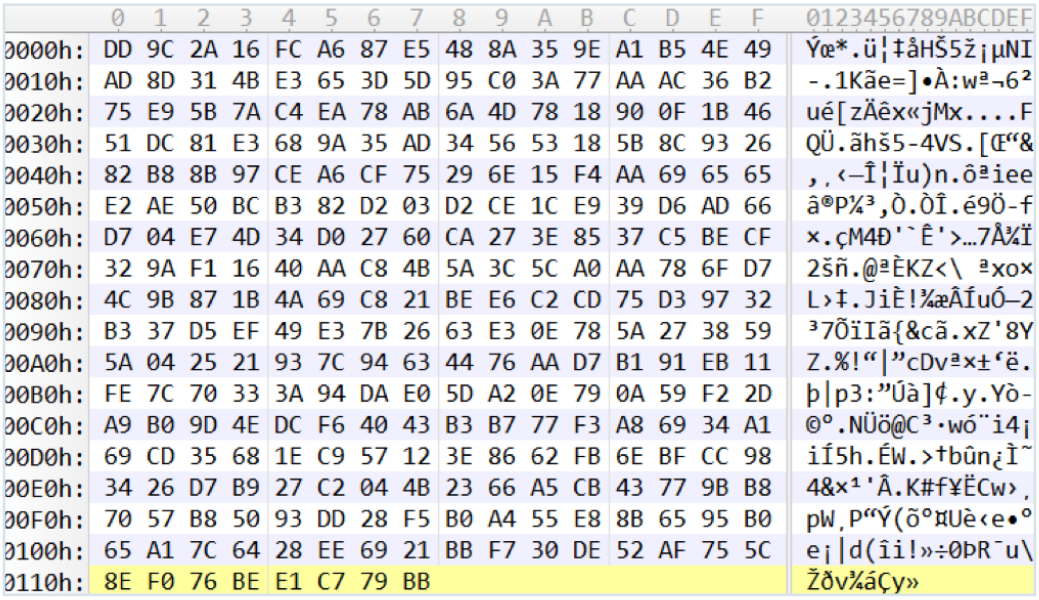
III. Using Standard RC4 Algorithm with MD5 Result of a Predefined String as Key

We also discovered that some recent Enfal malwares are using RC as the encryption algorithm. This type of key has a predefined string, which can be found in the executable files. The MM RAT utilizes this method to encrypt and decrypt using standard RC4 library:

Encryption and Decryption with RC4 Method

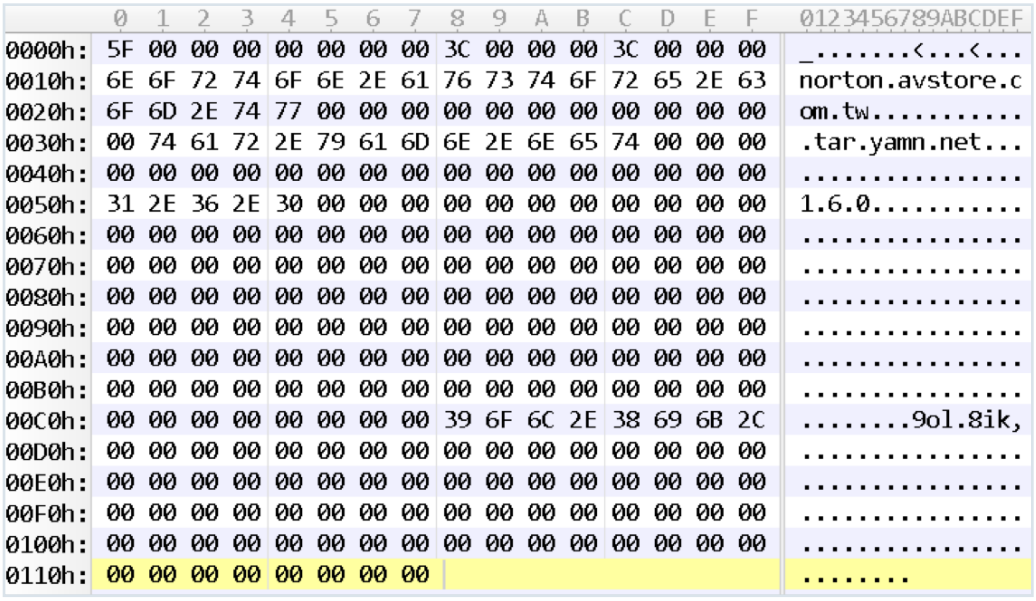
```
from hashlib import md5
from M2Crypto import RC4

def enfal_rc4_req_convert(data, key_str):
    key = md5(key_str).digest()
    rc4 = RC4.RC4(key)
    return rc4.update(data)
```

Filename	c_000.exe
MD5	B0E5FEA80334BA81B8D00B4DC53061E2
SHA1	2C9B8414DAB50B54EEF46DD1197892AB1DBFD634
RC4 Key	MD5(“rEdstArs”) = 00 65 A4 F5 64 2C CC BE 90 5C FA D1 C7 C0 20 5E
Filename	acctress.dll
MD5	5CF9C43CC1D39399F2310B7E72912F86
SHA1	CBAA817171D25B8147A728B63EE0080BC5B9F0B5
RC4 Key	MD5("CbqTsjvv") = 9C 87 DF FE 50 97 89 A2 B4 E8 CE 80 79 33 A7 72



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	DD	9C	2A	16	FC	A6	87	E5	48	8A	35	9E	A1	B5	4E	49	Ýœ*.ü¦‡åHŠ5ž¡µNI
0010h:	AD	8D	31	4B	E3	65	3D	5D	95	C0	3A	77	AA	AC	36	B2	-.1Kãe=]•À:wª¬6²
0020h:	75	E9	5B	7A	C4	EA	78	AB	6A	4D	78	18	90	0F	1B	46	ué[zÄêx«jMx....F
0030h:	51	DC	81	E3	68	9A	35	AD	34	56	53	18	5B	8C	93	26	QÜ.ãhš5-4VS.[Œ“&
0040h:	82	B8	8B	97	CE	A6	CF	75	29	6E	15	F4	AA	69	65	65	‚¸‹—Î¦Ïu)n.ôªiee
0050h:	E2	AE	50	BC	B3	82	D2	03	D2	CE	1C	E9	39	D6	AD	66	â®P¼³‚Ò.ÒÎ.é9Ö-f
0060h:	D7	04	E7	4D	34	D0	27	60	CA	27	3E	85	37	C5	BE	CF	×.çM4Ð'`Ê'>…7Å¾Ï
0070h:	32	9A	F1	16	40	AA	C8	4B	5A	3C	5C	A0	AA	78	6F	D7	2šñ.@ªÈKZ<\ ªxo×
0080h:	4C	9B	87	1B	4A	69	C8	21	BE	E6	C2	CD	75	D3	97	32	L›‡.JiÈ!¾æÂÍuÓ—2
0090h:	B3	37	D5	EF	49	E3	7B	26	63	E3	0E	78	5A	27	38	59	³7ÕïIã{&cã.xZ'8Y
00A0h:	5A	04	25	21	93	7C	94	63	44	76	AA	D7	B1	91	EB	11	Z.%!“|”cDvª×±‘ë.
00B0h:	FE	7C	70	33	3A	94	DA	E0	5D	A2	0E	79	0A	59	F2	2D	þ|p3:”Úà]¢.y.Yò-
00C0h:	A9	B0	9D	4E	DC	F6	40	43	B3	B7	77	F3	A8	69	34	A1	©°.NÜö@C³·wó¨i4¡
00D0h:	69	CD	35	68	1E	C9	57	12	3E	86	62	FB	6E	BF	CC	98	iÍ5h.ÉW.>†bûn¿Ì˜
00E0h:	34	26	D7	B9	27	C2	04	4B	23	66	A5	CB	43	77	9B	B8	4&×¹'Â.K#f¥ËCw›¸
00F0h:	70	57	B8	50	93	DD	28	F5	B0	A4	55	E8	8B	65	95	B0	pW¸P“Ý(õ°¤Uè‹e•°
0100h:	65	A1	7C	64	28	EE	69	21	BB	F7	30	DE	52	AF	75	5C	e¡|d(îi!»÷0ÞR¯u\
0110h:	8E	F0	76	BE	E1	C7	79	BB									Žðv¾áÇy»

Figure 10. Before RC4 encryption



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	5F	00	00	00	00	00	00	00	3C	00	00	00	3C	00	00	00	_.......<...<...
0010h:	6E	6F	72	74	6F	6E	2E	61	76	73	74	6F	72	65	2E	63	norton.avstore.c
0020h:	6F	6D	2E	74	77	00	00	00	00	00	00	00	00	00	00	00	om.tw..........
0030h:	00	74	61	72	2E	79	61	6D	6E	2E	6E	65	74	00	00	00	.tar.yamn.net...
0040h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0050h:	31	2E	36	2E	30	00	00	00	00	00	00	00	00	00	00	00	1.6.0...........
0060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0080h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0090h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00B0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00C0h:	00	00	00	00	00	00	00	00	39	6F	6C	2E	38	69	6B	2C	........9ol.8ik,
00D0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00E0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00F0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0100h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0110h:	00	00	00	00	00	00	00	00									....

Figure 11. After decryption

Fig. 11 shows that following decryption, the Enfal C2 locations are revealed:

- norton.avstore.com.tw (127.0.0.1)
- tar.yamn.net (tar.yamn.net NO LONGER EXISTS; yamn.net = 184.168.221.41)

Communications with the C2 Server

After Enfal connects to a C2 server, it uses HTTP GET request to access commands. When transmitting execution results or registering information of a compromised computer, it uses POST requests to send data. The URL, computer information format and data encryption method used may differ according to various Enfal malware mutations.

I. GET Requests

Client-end programs may use the GET method to access data in the C2 server, such as command files. Possible paths are listed below:

Possible Path of GET Requests
/httpdocs/mm/MYNAME(345-Ci)00-23-21-12-19-1A/ComMand.sec (In the 2016 variant, it became command.sec)
/httpdocs/mm/MYNAME(345-Ci)00-23-21-12-19-1A/Error.sec
/httpdocs/mm/MYNAME(345-Ci)00-23-21-12-19-1A/Reply.sec
/httpdocs/mm/MYNAME_00-23-21-12-19-1A/ComMand.sec (no bot campaign ID "(345-Ci)")
/httpdocs/mm/MYNAME_00-23-21-12-19-1A/CmDsHell.sec (no bot campaign ID "(345-Ci)")
/httpdocs/prx.sec
/httpdocs/update/update.ini

Some samples also use "trandocs/mm/" instead of "httpdocs/mm/". These are the two most commonly seen types.
Samples of retrieving commands reference the command of known sinkhole "czozNjA=". This is a base64 command, which decrypts into "s:360" for sleeping 360 seconds.

II. POST Requests

The URL pattern of POST requests can be grouped into different types. "cgi-bin" is most commonly seen, but "cgl-bin" and "cgm-bin" may also appear at times.

Group Specific Signature	URL String Examples for POST Request
/cg*-bin/CMS_*+	/cgi-bin/CMS_ClearAll.cgi /cgi-bin/CMS_ListImg.cgi /cgi-bin/CMS_SubitAll.cgi
/cg*-bin/[A-Z][a-z0-9]+.cgi	/cgi-bin/Clnpp5.cgi /cgi-bin/Crpq2.cgi /cgi-bin/Dwpq3.cgi /cgi-bin/Owpq4.cgi /cgi-bin/Rwpq1.cgi
/iupw82 /ienlg	Crpq2.cgi /iupw82/xcup/ /ienlg/Dwpq3ll.cgi dieosn83.cgi Rwpq1.cgi Clnpp5.cgi
/cg*-bin/ (CReply|CErr|ClrF|CFile|CNor).cgi	/cgi-bin/Owpp4.cgi /cgi-bin/CReply.cgi /cgi-bin/CErr.cgi /cgi-bin/ClrF.cgi /cgi-bin/CFile.cgi /cgi-bin/vip.cgi

Using Owpp4.cgi to register online hosts

To register the information of compromised computers onto the C2 server, Enfal typically sends unencrypted information to "/cgi-bin/Owpp4.cgi" file. There are distinct differences between the two formats.

General	Newer(MM RAT)
WORK-DD656867(339new-bbbs)00-0C-33-BB-11-22/ I=192.168.107.128 O=XP Professional (Service Pack 3) L=Chinese(TW) V=3.3.9 M=mapcd.exe B=oleps.exe sen=N Prxy=N Lg=n	WORK-DD656868_00-0C-33-BB-11-33/IpAddress=192.168.107.129 OsVersion=Windows XP/2003 Logined=n Language=English (US) Version=1.6.0 MainFilename=C:\WINDOWS\system32\

POST Request Encryption and Decryption

With the exception of Owpp4.cgi, Enfal transmits encrypted data to other cgi files. It uses two encryption methods - the simple XOR method and RC4 encryption.

I. XOR Method (e.g. Using 0x55 as XOR key)

Encryption	Decryption
Data[0] ^= 0x55 for all i = 1 to length-1 Data[i] ^= Data[i-1]	for all i=length-1 to 1 Data[i] ^= Data[i-1] Data[0] ^= 0x55

def enfal_req_encode(s, xor_key=0x55):
if not s:
return ''
 	
orig_values = map(ord, s)
ret_values = [orig_values[0] ^ xor_key]
for i in range(1, len(s)):
ret_values.append(
orig_values[i] ^ ret_values[i-1])
 	
return ''.join(map(chr, ret_values))
	def **enfal_req_decode**(s, xor_key=0x55):
	if not s:
	return ''
	
	enc_values = map(ord, s)
	ret_values = []
	for i in range(len(s)-1, 0, -1):
	ret_values.insert(0,
	enc_values[i] ^ enc_values[i-1])
	
	ret_values.insert(
	0, enc_values[0] ^ xor_key)
	return ''.join(map(chr, ret_values))

Sending POST requests to CReply.cgi is one way of transmitting encrypted data. The complete dataset begins with computer names, bot campaign ID, and MAC address, and the encrypted data is placed behind "/". The following is an example of the encryption process:

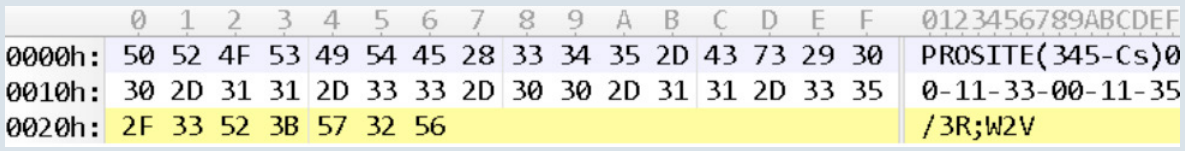
POST Request Data of "OK"

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	50	52	4F	53	49	54	45	28	33	34	35	2D	43	73	29	30	PROSITE(345-Cs)0
0010h:	30	2D	31	31	2D	33	33	2D	30	30	2D	31	31	2D	33	35	0-11-33-00-11-35
0020h:	2F	1A	51														/.Q

Apply **enfal_req_encode**() with key 0x55 to encrypt "OK":

4F 4B | OK
1A 4B | .K
1A 51 | .Q

POST Request Data of "OK"



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	50	52	4F	53	49	54	45	28	33	34	35	2D	43	73	29	30	PROSITE(345-Cs)0
0010h:	30	2D	31	31	2D	33	33	2D	30	30	2D	31	31	2D	33	35	0-11-33-00-11-35
0020h:	2F	33	52	3B	57	32	56										/3R;W2V

66 61 69 6C 65 64 | failed
33 61 69 6C 65 64 | 3ailed
33 52 69 6C 65 64 | 3Riled
33 52 3B 6C 65 64 | 3R;led
33 52 3B 57 65 64 | 3R;Wed
33 52 3B 57 32 64 | 3R;W2d
33 52 3B 57 32 56 | 3R;W2V

II. RC4 Method

Encryption methods are the same as those used for the configuration block. Please refer to enfal_rc4_req_convert function.

Revealing the Inner Workings of the Enfal Backend

In 2005, we discovered an Enfal backend malware called sens32.dll. It uses Winsock functions to implement HTTP web server functions, and is able to handle requests from compromised computers. Upon executing the program, a directory is created to store logs and information from the compromised computers, as shown below:

Directory Created by sens32.dll
%SystemRoot%\system32\myweb
%SystemRoot%\system32\Logs
%SystemRoot%\system32\httpdocs
%SystemRoot%\system32\httpdocs\mm
%SystemRoot%\system32\httpdocs\update

The server (backend) binds port 80 to provide a multi-functional .cgi path, which is responsible for receiving client-side GET requests to access commands, or make client-side POST requests to /cgi-bin/<CGINAME>.cgi. It then changes <CGINAME> to lower-case letters, to determine which function to execute. As shown in the graph below, Enfal uses a small number of simple functions, such as uploading and downloading files, executing, listing, and so on. This is one of the reasons why Enfal is often used as a frontline APT backdoor. Commands and their descriptions are as follows:

Served CGI Names in Lowercase	Description (CLIENT_HOST_ID is like "TIGER-5A7A9A(345-CC)00-30-11-33-77-66")
test	Test if service is on.
cerr	Report error to server. Server will store the message to myweb\httpdocs\mm\<CLIENT_HOST_ID>\Error.sec
cfile	
clrf	Delete files on server. After accessing to a command file (ComMand.sec) on client-end, it uses this CGI to delete the file.
cnor (management)	Write command into the command file of its specific compromised computer (myweb\httpdocs\mm\<CLIENT_HOST_ID>\ComMand.sec). The compromised computer accesses the file and executes the commands. If ComMand.sec already exists, the content is rewritten.
creply	Report reply to server. Server will store the message to myweb\httpdocs\mm\<CLIENT_HOST_ID>\Reply.sec
mlist (management)	Access all information on the compromised list, and directly send to client.
mlist2 (management)	Access all information on the compromised list, and store to myweb\httpdocs\Online.dat The information can be accessed through a browser on other computers: http://<SERVER_HOST>/httpdocs/Online.dat
owpp4	Report information of compromised computer to server. Server will store the message to myweb\httpdocs\mm\<CLIENT_HOST_ID>\Online.txt Client periodically (for example once every 10 minutes) reports back computer data using this function. When reporting for the first time, myweb\httpdocs\mm\<CLIENT_HOST_ID>\Ctime.txt is established to record the time of the first report.
vip	Test if service is on.
shutdown (management)	Shutdown service. Clean up.

The server stores the unencrypted data sent by the client to CNor.cgi, CReply.cgi, CErr.cgi, and Owpp4.cgi. This characteristic allows server-end programs to support various versions of Enfal clients. Whether using XOR encryption or RC4 encryption, what to do with the data is decided by attackers and compromised client-end programs.

Based on previously analyzed samples, the client-end may not use all .cgi functions. For example, management functions may be operated by hidden actors rather than the compromised client.

I. Command File Format

The process of a client executing a specific function and reporting back to a server is as follows. The compromised computer periodically downloads its own specific ComMand.sec. Download intervals vary, for example once every five minutes. These intervals and the Owpp4.cgi report intervals can be set separately. The contents are then decrypted and restored to the original commands, and the results are sent to the server's CReply.cgi or CErr.cgi, depending on the success of the actions below.

Command Format	Action
02 07 <PATH> 00	Copy kernel32.dll's file time to specified file
02 01 <PATH_PATTERN_LENGTH:BYTE> <PATH_PATTERN> 0	List files with metadata
04 02 <FILE_OFFSET:DWORD> <FILENAME> 00	Read file. File content will be send to CReply.cgi if successfully
04 01 <FILE_OFFSET:DWORD> <SIZE_TO_WRITE:DWORD> <FILENAME> 00 <DATA_TO_WRITE	Write file
02 04 <DIR> 00 <COMMAND_LINE> 00	Set current directory to specified path, and execute command
02 03 <FILENAME> 00	Delete file
02 07 <SOURCE_FILENAME> 00 <DESTINATION_FILENAME> 00	Copy file

II. Backend DLL File

Filename	sens32.dll
MD5	1ABA3CBE9A1937D938D96DB2C478C4F7
SHA1	3D0164FC9CF1C67FA9CEF9D3C36734645F76EE53

We ran a search for sens32.dll on VirusTotal, and discovered that the executable file is not regarded by any AV as malware. Despite the fact that it was first uploaded in 2010, the detection rate is still zero to date.

Figure 12. sens32.dll on VirusTotal

III. Backend Online.dat

This screenshot is a sample of an Enfal attack in 2008. To understand its compromised area, we must access the compromised data in Online.dat of a backend server. At the time, the domain of the computer was czc.lookbar8.com. Information related to clients that connected to this computer is shown below (Fig. 13), including the version, system version, country, bot version, and bot executable file names. We grouped the compromised clients to find that some computers were clearly located in the same organization, and most attacks were concentrated in Taiwan, China, and the US.

mac	campaignid	createtime	refreshtime	webipaddress	webhostname	asn	asncountrycode	asnregistry	hostip	os
00-16-76-18-B8-E0	344-Cs	Mon Aug 25 15:33:03 2008	Fri Sep 12 13:27:07 2008	203.198.107.21	awork117021.netvigator.com	4760	HK	apnic	192.168.1.9	XP Home Edition (Service Pac
00-0D-61-04-33-80	351-Cs	Mon Sep 01 08:54:00 2008	Fri Sep 12 15:00:38 2008	203.204.103.225	host-203-204-103-225.static.kbtelecom.net	9416	TW	apnic	0.0.0.0	XP Professional (Service Pac
00-13-D3-A2-AD-CF	345-Cs	Fri Aug 22 09:31:12 2008	Fri Sep 12 14:59:32 2008	203.70.63.113		4780	TW	apnic	192.168.0.172	XP Professional (Service Pac
00-16-E6-62-FC-F8	351-Cs	Fri Aug 22 11:18:43 2008	Fri Aug 22 15:28:46 2008	203.73.49.62	203-73-49-62.adsl.dynamic.seed.net.tw	4780	TW	apnic	192.168.1.120	XP Professional (Service Pac
00-1E-58-AB-AD-EC	351-Cs	Mon Aug 25 08:46:23 2008	Fri Sep 12 14:56:50 2008	203.92.163.114		NA	CN	apnic	192.168.88.148	XP Professional (Service Pac
00-E0-18-F7-17-28	344-Ce	Fri Aug 22 09:01:43 2008	Fri Sep 12 14:58:16 2008	206.205.114.226	ext-dc01.embassyofindonesia.org	2828	US	arin	10.1.1.156	XP Home Edition (Service Pac
00-08-74-AB-1C-29	345-Cs	Fri Aug 22 08:59:48 2008	Fri Sep 12 14:54:21 2008	209.11.41.25		4136	US	arin	10.1.1.4	XP Professional (Service Pac
00-07-40-09-E5-71	345-Cs	Sat Aug 23 10:51:29 2008	Thu Sep 11 19:10:25 2008	210.245.199.90		17444	HK	apnic	192.168.1.102	XP Home Edition (Service Pac
00-0C-6E-E4-F5-58	351-Cs	Fri Aug 22 09:01:54 2008	Fri Aug 29 18:41:10 2008	210.64.152.14	sw64-152-14.adsl.dynamic.seed.net.tw	4780	TW	apnic	192.168.0.1	XP Home Edition (Service Pac
00-C0-9F-28-A5-C8	345-Cs	Thu Aug 28 18:57:38 2008	Sat Aug 30 09:47:36 2008	211.95.62.5		17621	CN	apnic	192.168.110.1	2003 server Standard Edition
00-1A-92-6E-4A-42	344-Cs	Fri Aug 22 11:00:26 2008	Thu Sep 11 23:31:43 2008	218.102.221.72	pcd689072.netvigator.com	4760	HK	apnic	192.168.2.2	XP Home Edition (Service Pac
00-04-23-63-32-6D	351-Cs	Fri Aug 22 14:37:32 2008	Tue Sep 09 02:27:32 2008	218.166.100.20	218-166-100-20.dynamic.hinet.net	3462	TW	apnic	0.0.0.0	XP Home Edition (Service Pac
00-80-AD-72-AA-2E	351-Cs	Mon Aug 25 10:51:14 2008	Mon Aug 25 10:51:14 2008	218.173.99.106	218-173-99-106.dynamic.hinet.net	3462	TW	apnic	192.168.1.172	XP Professional (Service Pac
00-50-FC-6B-F8-34	351-Cs	Fri Aug 22 09:03:53 2008	Tue Sep 02 12:07:59 2008	220.130.162.154	220-130-162-154.HINET-IP.hinet.net	3462	TW	apnic	192.168.0.2	2k Advanced Server(Service P
00-01-80-5C-2E-0C	351-Cs	Sun Aug 31 10:14:41 2008	Mon Sep 08 22:03:52 2008	220.131.213.160	220-131-213-160.dynamic.hinet.net	3462	TW	apnic	192.168.0.31	XP Home Edition (Service Pac
00-18-FE-68-CF-45	343-Cs	Fri Aug 22 10:56:30 2008	Fri Sep 12 14:59:17 2008	220.163.80.227	227.80.163.220.broad.km.yn.dynamic.163data.com.cn	4134	CN	apnic	192.168.1.11	XP Professional (Service Pac
00-11-43-AD-69-F8	351-Cs	Thu Aug 28 11:14:17 2008	Mon Sep 08 19:11:05 2008	222.128.24.186		4808	CN	apnic	192.168.2.89	XP Professional (Service Pac
00-19-D2-BD-49-62	351-Cs	Fri Aug 22 09:19:05 2008	Fri Sep 12 14:55:01 2008	222.128.24.186		4808	CN	apnic	0.0.0.0	XP Professional (Service Pac
00-0B-CD-B9-70-3C	345-Cs	Mon Aug 25 15:17:37 2008	Thu Sep 11 17:55:59 2008	222.128.24.186		4808	CN	apnic	192.168.2.141	XP Professional (Service Pac
00-19-D2-3C-1C-2C	344	Tue Aug 26 15:17:34 2008	Fri Sep 12 14:41:25 2008	222.128.24.186		4808	CN	apnic	0.0.0.0	XP Professional (Service Pac
00-13-72-70-22-59	345-Ce	Fri Aug 22 09:22:51 2008	Thu Sep 04 15:37:47 2008	222.128.24.186		4808	CN	apnic	192.168.2.85	XP Professional (Service Pac
00-19-D2-BD-46-E9	351-Cs	Mon Sep 08 10:02:52 2008	Fri Sep 12 14:54:37 2008	222.128.24.186		4808	CN	apnic	0.0.0.0	XP Professional (Service Pac
00-13-72-70-25-D8	351-Cs	Fri Aug 22 09:13:36 2008	Thu Sep 11 18:02:01 2008	222.128.24.186		4808	CN	apnic	192.168.2.144	XP Professional (Service Pac
00-06-5B-3C-6E-E4	344-Ci			61.135.139.162		4808	CN	apnic	61.135.139.162	2k Server(Service Pack 4)
00-11-D8-CA-A5-94	351-Cs	Fri Aug 29 20:45:11 2008	Thu Sep 11 06:22:03 2008	61.230.51.45	61-230-51-45.dynamic.hinet.net	3462	TW	apnic	0.0.0.0	XP Professional (Service Pac
00-30-1B-3D-D7-6E	345-Cs	Fri Aug 22 21:28:12 2008	Thu Sep 11 21:57:35 2008	61.230.90.101	61-230-90-101.dynamic.hinet.net	3462	TW	apnic	0.0.0.0	XP Professional (Service Pac
00-15-F2-56-F2-D4	351-Cs	Tue Aug 26 14:28:39 2008	Thu Sep 11 17:11:12 2008	61.231.245.23	61-231-245-23.dynamic.hinet.net	3462	TW	apnic	0.0.0.0	XP Home Edition (Service Pac
00-1C-BF-BD-14-88	351-Cs	Fri Aug 22 16:08:19 2008	Fri Aug 22 16:42:17 2008	61.64.47.96		NA	TW	apnic	0.0.0.0	XP Professional (Service Pac
00-04-75-B0-6E-F3	344-Cs	Mon Aug 25 14:07:00 2008	Thu Sep 11 22:03:30 2008	62.215.248.49		21050	KW	ripencc	192.168.2.13	XP Professional (Service Pac

Figure 13. Online.dat of a backend server

Some computers revealed report records of different bot versions. We even discovered, in one of the computers, that after its system update, its bot followed suit and also upgraded to the newest version:

Old	New
{'B': 'taskmgre.exe', 'I': '0.0.0.0', 'L': 'Chinese(CN)', 'Lg': 'n', 'M': 'rsmss.exe', 'O': 'XP Home Edition', 'Prxy': 'N', 'V': '3.4.4', 'campaignid': '344-Cs', 'hostid': 'YOUR-2828F87C55(344-Cs)00-15-F2-56-F2-D4', 'mac': '00-15-F2-56-F2-D4', 'pcname': 'YOUR-2828F87C55', 'sen': 'N'}	'B': 'safrcmd.exe', 'I': '0.0.0.0', 'L': 'Chinese(TW)', 'Lg': 'n', 'M': 'chipset.exe', 'O': 'XP Home Edition (Service Pack 2)', 'Prxy': 'N', 'V': '3.5.1', 'campaignid': '351-Cs', 'hostid': 'YOUR-2828F87C55(351-Cs)00-15-F2-56-F2-D4', 'mac': '00-15-F2-56-F2-D4', 'pcname': 'YOUR-2828F87C55', 'sen': 'N'}

Conclusions

The longevity of the Enfal malware is legendary - its APT backdoors have been active since 2004. Yet, unlike other APT backdoors that were only active during certain periods or attacks, revisions of this malware continue to appear on a consistent basis from 2008 until 2016. Based on our attack target analysis, we can see the global dispersion of Enfal's operations. While the malware is still most active in South East Asia and the US, where its targets include NGOs, and diplomatic organizations, it has also spread as far as Ethiopia and Brazil. Because of its simple yet precise characteristics, and its unsynchronized command downloads from servers, Enfal's C2 communications cannot be easily detected on compromised computers using standard traffic anomaly-based techniques. As a result, even if many compromised devices have undergone system upgrades, they are still being attacked year after year.

Unlike previously published reports on Enfal, which only cover partial malware samples and data of a single period or specific attack, this report presents a complete observation and analysis of Enfal over an extended period of time. Data disclosures of such scope and depth are unprecedented. This is the first time that Enfal's communications protocols have been fully revealed, as well as the encryption and obfuscation methods of its API and configuration block. Using reverse engineering and other methods, our research team was able to connect back to their C2 backend, decrypt their backend protocols and gain a full picture of Enfal's attack campaigns, including a complete list of victims and their locations.

By enabling a deeper understanding of the tools, tactics and procedures used by this sophisticated and evasive malware, we believe that security organizations worldwide will be able to improve their defense posture and prepare more effective countermeasures against the likes of Enfal and other complex cyber attacks.

APPENDIX

Sample File Lists (ordered by build time)

MD5	File Build Time
8ec4fc310915b6db5d62ff476d95fc87	Sat Oct 02 05:13:09 2004
46679d05a02e065a5f082d86d7635488	Wed Feb 22 01:06:32 2006
712a9e2f68590c5194fbf6dacf7e59cc	Mon Feb 26 05:48:06 2007
923d3c72026a56bb9bc54843a6016854	Mon Feb 26 05:49:42 2007
33c9c65a0b68821577da2813db0adab1	Mon Mar 12 15:09:03 2007
9d923db236608136c39f29062e71b70f	Mon Mar 12 15:48:04 2007
bdfb76d4dd25ee3b4d36172a3c3cd98e	Fri Aug 17 03:47:13 2007
8314534e82f50fc258b867b094083826	Thu Dec 10 08:12:12 2009
e61b128e97a39fe869cf89be571fe021	Fri Apr 23 03:20:04 2010
06572d93d87a8d0fb7e070be79692c87	Tue May 11 09:20:48 2010
7bd2cf1a96fe27c301111785799233ea	Mon May 24 09:38:14 2010
bf0ad3625fd7cd2c8a7ba3fa74bf1605	Thu Sep 30 00:29:32 2010
2bfd304e3433cb0de9c2f284e9417409	Wed Oct 12 07:08:08 2011
0ecd791525cc30ced610e81ef67290b0	Wed Oct 19 03:10:07 2011
f36007400f0c85784fd374ad4ed23c6e	Thu Dec 08 00:24:35 2011
d05f012c9c1a7fb669a07070be821072	Tue Mar 27 00:01:02 2012
b0e5fea80334ba81b8d00b4dc53061e2	Tue Dec 10 00:28:13 2013
347ad5f8a3e6a1512398cb8d56989273	Thu May 22 03:07:47 2014
5cf9c43cc1d39399f2310b7e72912f86	Tue Aug 19 01:42:39 2014
3ac75af442414ea7529b2460c8b17d90	Tue Aug 19 01:43:47 2014
8f42d1f21e9d7a3d4f33c6886a010056	Tue Mar 01 03:51:48 2016

Yara Rule

```
import "pe"

rule Enfal
{
    meta:
        description = "Enfal Series"
        author = "Verint Systems"

    strings:
        $cgxbin_cgx_withnumber_strict = /\/cg.-bin\/[A-Z][a-zA-Z]{2,7}[0-
9]\.?(cgi)?/
        $cgxbin_cgi_known_series = /\/cg.-
bin\/(ClrF|CReply|CErr|CNor|CFile|Mlist|vip|shutdown)\.?(cgi)?/
        $xdocs_mm = /(http|tran)?docs\/mm\//
        $command_sec = "ComMand.sec"
        $cmdshell_sec = "CmDsHell.sec"
        $goldsun = "GoLdSuN"
        $ienlg = "/ienlg/"
        $white = /\/[A-Z][a-zA-Z]{,2}white/   // "/Cmwhite" "/Dfwhite"
"/Ufwhite" "/Ccmwhite"
        $rc4keys = /rEdstArs|CbqTsjvv/

    condition:
        (pe.machine == pe.MACHINE_I386 or pe.machine == pe.MACHINE_AMD64)
        and any of them
}
```

About Verint Systems Inc.

Verint® (Nasdaq: VRNT) is a global leader in Actionable Intelligence® solutions with a focus on customer engagement optimization, security intelligence, and fraud, risk and compliance. Today, more than 10,000 organizations in 180 countries — including over 80 percent of the Fortune 100 — count on intelligence from Verint solutions to make more informed, effective and timely decisions.

www.verint.com/cyber | Info.cyber@verint.com

VERINT®