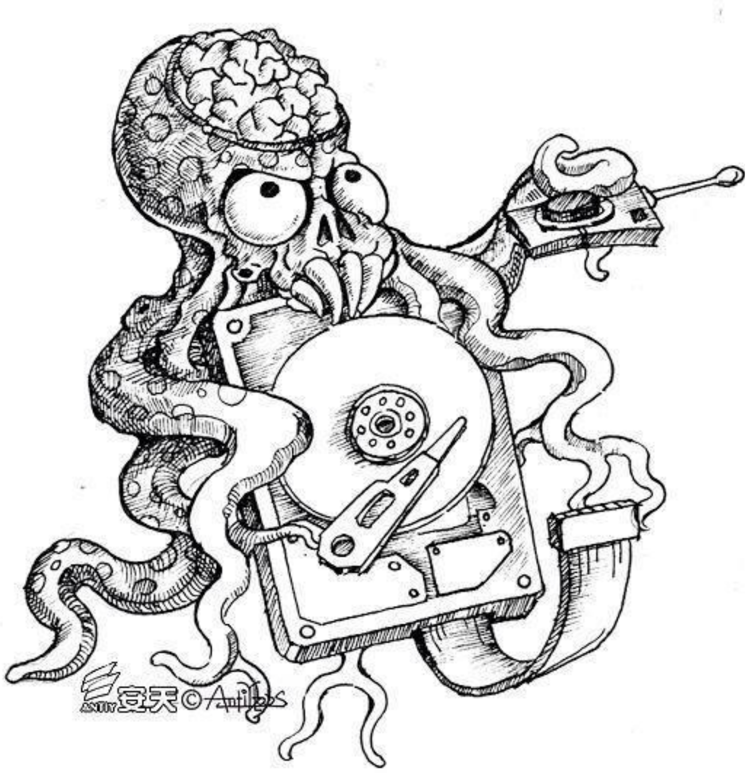
FROM EQUATION TO EQUATIONS

Revealing the multi-platform operational capability of Equation Group

Antiy CERT



Draft: Jan. 15, 2014 16:43 (UTC+8)

Published: Nov. 4, 2016 10:00 (UTC+8)

Updated: Nov. 4, 2016 13:00 (UTC+8)

Contents

1 Background

From February 2015, Antiy has published two reports about Equation Group, both of which analyzed the code components for Windows platform, the persistent ability in disks and the use of encryption algorithms. In this report, Antiy will publish the analysis of Equation components for Solaris and Linux platforms for the first time. We can also be proud to say that this is the first report to prove the existence of these kinds of “Evils”. We actually finished the analysis several years ago, and Antiy has concerned with the Great Attack Group since 2012, and trying to analyze its operational in all invasion and persistence scenarios, where the core targets are the server operating systems, such as Linux, Solaris and FreeBSD. These loads are not usual script Trojans, but are **binary components** with an **encrypted communication**. These components act as **Rootkits, and have strict encryption anti-analysis technique and trick**. Therefore, Antiy has named attacks performed by super-attack organizations as A^{2}PT, and make the capabilities to attack all platforms indicators.

Based on long-term experience in tracking and analyzing capability of advanced threats and malicious code, Antiy’s product PTD (Persistent Threat Detection System) can help users capture the payload delivery and lateral movement; IEP (Intelligent Endpoint Protection System) provide protection for traditional Windows hosts and Chinese Operating system, and assists PTA (Persistent Threat Analysis System) to analyze malware for various platforms. The deployment of these products also enables customers to support Antiy to get more threat indicators. Meanwhile, we have been paying attention on open-source intelligence and public information, also the information and development trends of relevant organizations.

After Kaspersky and Antiy released reports about Equation (called the Group for short) last year, the Group still launched a series of attacks. In August 2016, the malware used by Equation was disclosed in *Equation Group Cyber Weapons Auction – Invitation* [1], and this group was connected with attack weapon system named ANT for the first time. Based on this, we can also find its ability to inject and persist in products of Cisco, Juniper, Fortinet and other firewalls. On October 31, 2016, an article called *Shadow Brokers reveals list of servers Hacked by the NSA* [2] was published in The Hacker News, which contained more documents revealed by Shadow Brokers, including some of the foreign server list compromised by the Group. The related documents claimed that most of the infected servers are running Solaris, Oracle-owned Unix operating system, and some are running FreeBSD or Linux. With the mutual prove of public information and Antiy’s analysis conclusion of the samples, we can clearly figure out the **powerful full-platforms attacking capability** of this organization.

Our analysis work is continuously validated by ongoing information. During the past years, the analysis of this attack was sophisticated and challenging; whose analysis is more difficult than Stuxnet or Flame. The malware with this kind of highly complex and hidden capability is a huge challenge for both victims and analysts. Especially when the scope of its combat coverage of almost All computer architecture and operating systems, the traditional security team relatively good at analyzing the malware for Windows, Linux and Android and other mainstream operating system platform will feel much more pressure and challenges. If you use the name of Equation to do a parable about the difficulty of analysis, **what we need to conquer is not an "Equation" but more complex "Equations".**

Antiy Labs released the Chinese version of this report on November 4th, 2016. Due to the lack of translation ability and experience, the English version was not released synchronously. Many colleagues in international cybersecurity field may read this report with the help of Google Translator. This version got finished until November 8th, and we welcome your advices and suggestions.

2 The multi - platform operational capability of Equation

Equation employs industrial-grade standard arm arsenal attack weapons arsenals, including six components: EquationLaser, EquationDrug, DoubleFantasy, TripleFantasy, Fanny and GrayFish. Antiy has found samples of EquationDrug and DoubleFantasy attacking on other platforms. The arsenal information is shown in the following table:

Component	Platform	Description	Period
Equation-Laser	**Not found**	An early implant from the EQUATION group, used around 2001-2003. Compatible with Windows 95/98.	2001-2003
Equation-Drug	**Some plugins found**	A very complex attacking platform used by Equation. It supports a module plugin system, which can be dynamically uploaded and unloaded by attackers. May be the upgraded version of EquationLaser.	2003-2013
Double-Fantasy	**Proved**	A validator-style Trojan, which is designed to confirm the target, is the intended one. If the target is confirmed, they get upgraded to a more sophisticated platform such as EQUATIONDRUG or GRAYFISH.	2004-2012
Triple-	**Maybe**	A full-featured backdoor sometimes used in tandem with	2012-now

Fantasy	**existing**	GRAYFISH. It looks like an upgrade of DOUBLEFANTASY, and is possibly a more recent validator-style plugin.	
Fanny	**Not found**	A computer worm created in 2008 and used to collect information about targets in the Middle East and Asia. Some victims appear to have been upgraded first to DoubleFantasy, and then to EQUATIONDRUG. Fanny used exploits for two 0day vulnerabilities which were later discovered with Stuxnet.	2008-2011
GrayFish	**Not found**	The most sophisticated attack platform from Equation. It completely resides in the registry, relying on a Bootkit to gain execution at OS startup.	2008-now

Based on the following table, readers can put together jigsaw puzzles of Equation attack.

Information	Windows	Linux	Solaris	Oracle-owned Unix	FreeBSD	Mac OS
Antiy The Trojan modifying firmware Exploration in attack components of Equation Group[3]	Analysis of sample load and hard disk persistence					
Antiy Analysis of encryption skills used in Equation Group attack components[4]	Encryption algorithm analysis					
Antiy Revealing the multi-platform loading capability of Equation Group（this report）		Found Analysis of related loads	Analysis of related loads			
The Hacker News：Shadow Brokers reveals list of Servers Hacked by the NSA			Existed	Existed	Existed	
Kaspersky Equation: The Death Star of Malware Galaxy[5]	Revealing Equation					
Kaspersky A Fanny Equation: "I am your father, Stuxnet"[6]	Fanny analysis					

Kaspersky Equation Group: from Houston with love[[7]	Doublefantasy analysis					
Kaspersky EQUATION GROUP: QUESTIONS AND ANSWERS[8]	Equation Group Questions and Answers					Speculation based on network features

Note: Antiy has found User Agent with Solaris logo during the analysis of samples, and Kaspersky released a series of reports, revealing the Equation Group; one named as "Equation Group Questions and Answers"[8] publishes the capture information of the Mac OS X Agent. So far, both Antiy and Kaspersky have not yet found Mac OS X samples, but the payload for Mac OS X does exist.

3 Analysis of Partial Load for Linux Linux(x86)

We have captured the samples on Linux platform and confirmed that the sample is DoubleFantasy component after analyzing. The component is used to perform incipient detection on targets with Linux platform. It is a sample of the Linux platform, so it has different features with others.

3.1 Preceding Module——DoubleFantasy

3.1.1 File Tag

Name	Trojan/Linux.DoubleFantasy
Original File Name	[redacted]
MD5	[redacted]
Processor Architecture	X86（32-bit）
Size	[redacted]
Format	BinExecute/ELF
Timestamp	N/A
Signature	None
Shell	None

Compiler language	C

3.1.2 Running Process

On Linux platform, samples execution is divided into two cases, with parameters or no parameters. If the parameter '-c' engages in, only system information can be obtained and it can be regarded as scene detection. The process can be shown by the following flow chart:

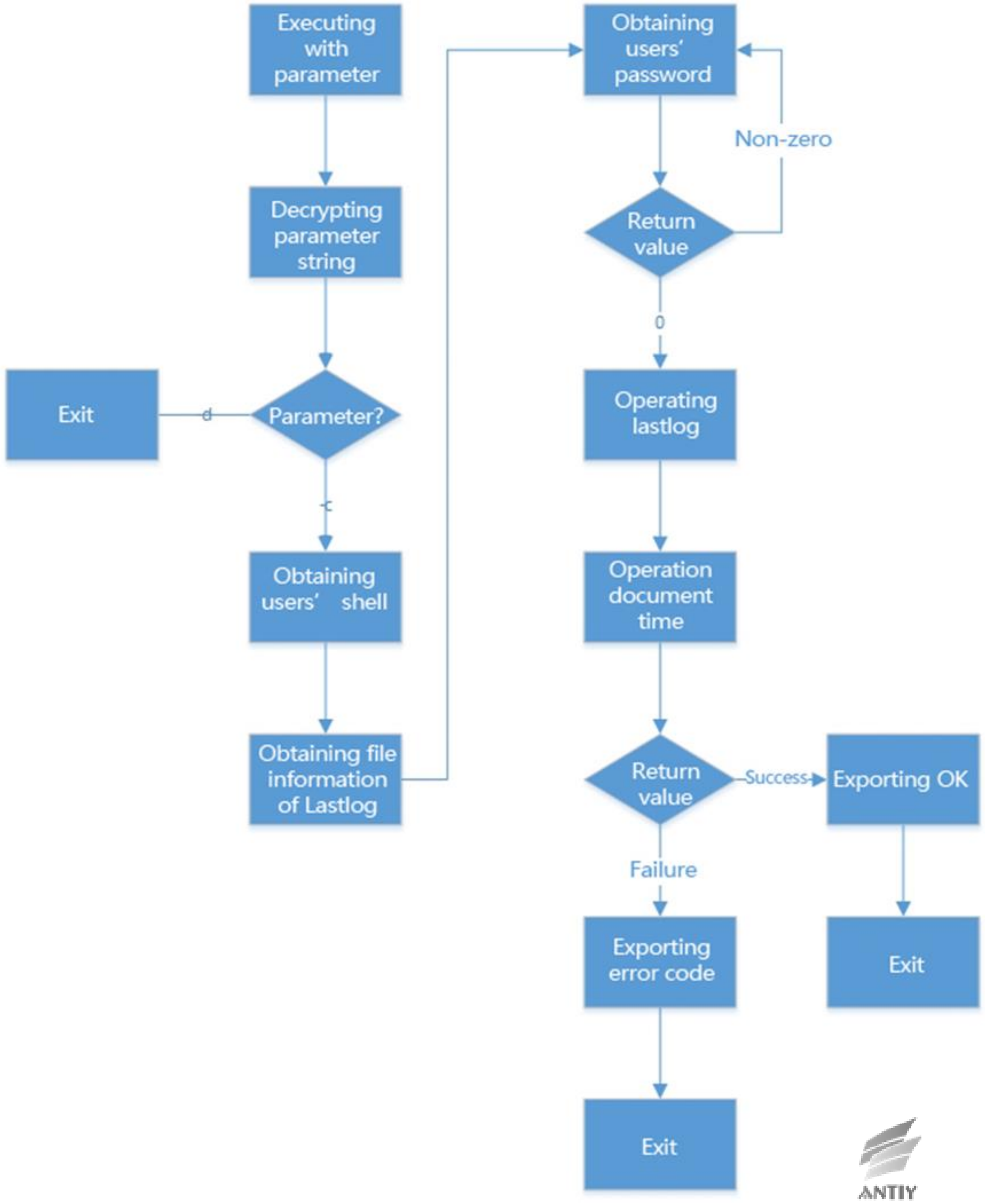


Figure 1 Running process with parameter–c

In the case without parameter '-c', the process can be shown as:

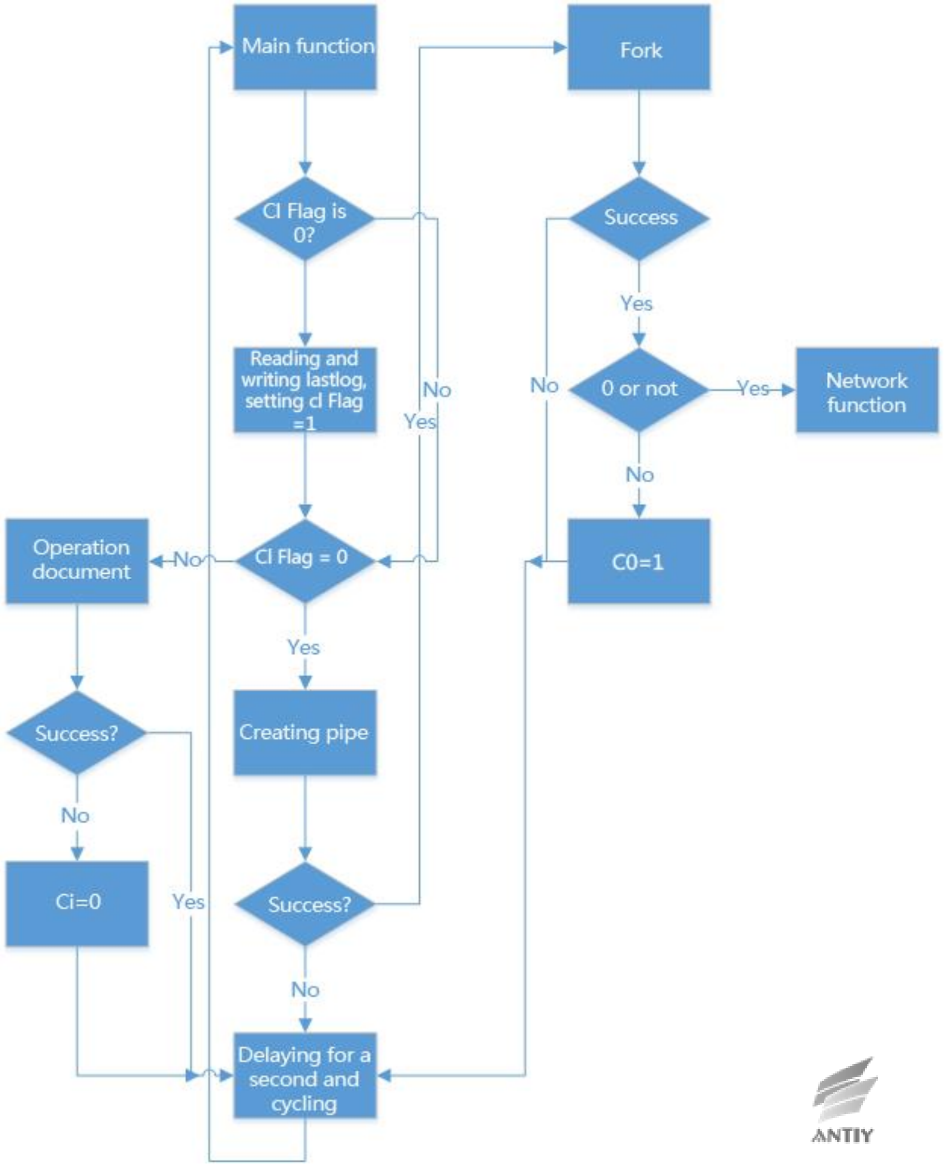


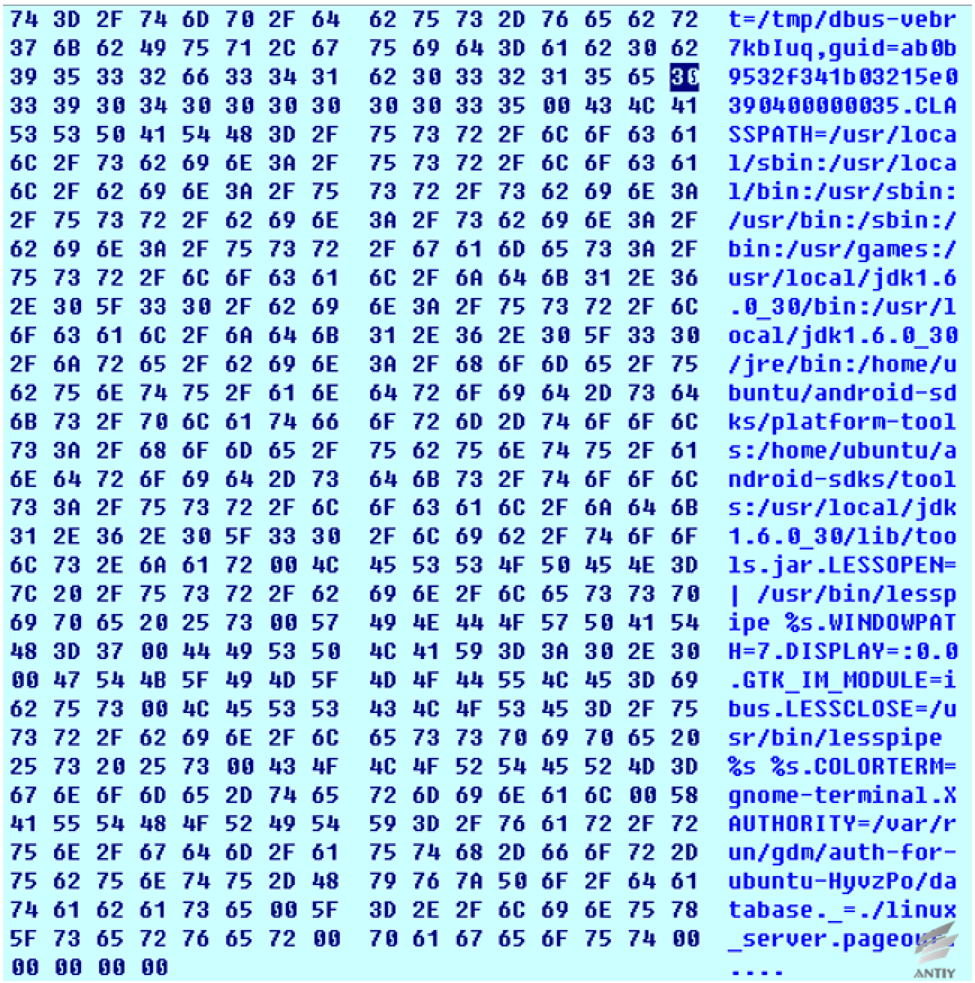
Figure 2 Running process without parameter–c

3.1.3 Basic functions

- Traversing system files, clearing / var / log / lastlog records, obtaining system account password information.
- Connecting Google to determine network connectivity.
- Connecting remote sever and making different operations based on remote control instructions.
- Many encryption algorithms used in communication and information.
- Starting itself with a linked file, and the proc /% d / exe file pointing to the files of the sample.

- Opening three PID threads (two of them are consecutive) after running.

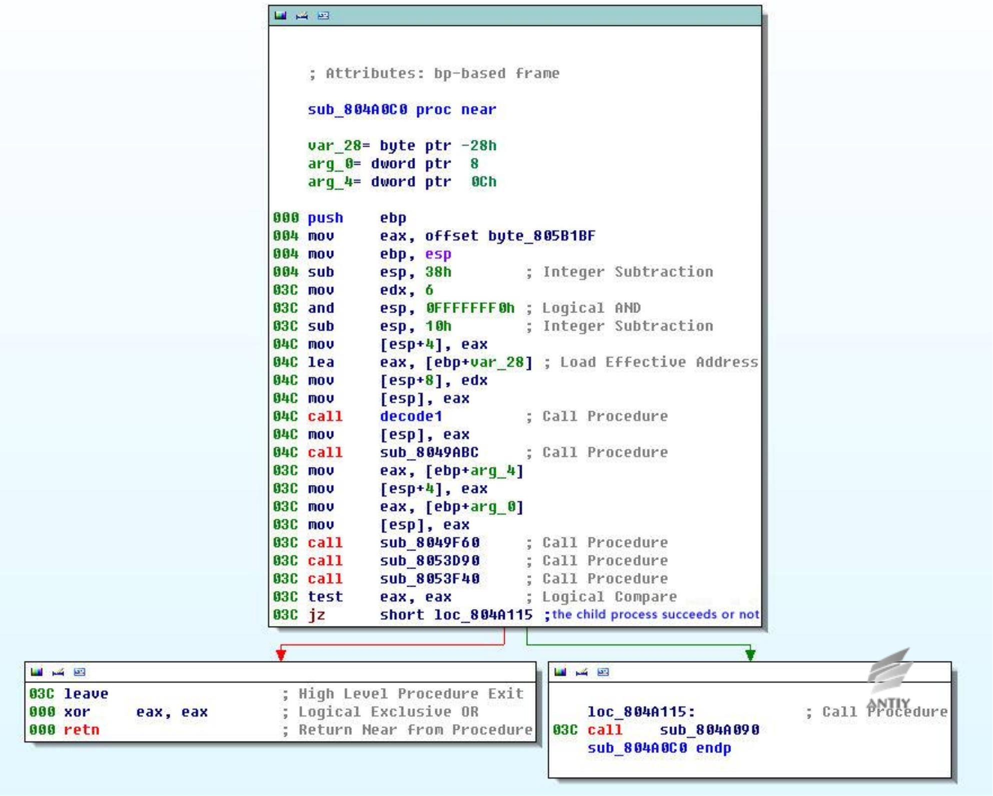
- Collecting information about infected computers, including system directory, file extension, and other information. As shown below:



```
74 3D 2F 74 6D 70 2F 64  62 75 73 2D 76 65 62 72  t=/tmp/dbus-vebr
37 6B 62 49 75 71 2C 67  75 69 64 3D 61 62 30 62  7kbIuq,guid=ab0b
39 35 33 32 66 33 34 31  62 30 33 32 31 35 65 30  9532f341b03215e0
33 39 30 34 30 30 30 30  30 30 33 35 00 43 4C 41  390400000035.CLA
53 53 50 41 54 48 3D 2F  75 73 72 2F 6C 6F 63 61  SSPATH=/usr/loca
6C 2F 73 62 69 6E 3A 2F  75 73 72 2F 6C 6F 63 61  l/sbin:/usr/loca
6C 2F 62 69 6E 3A 2F 75  73 72 2F 73 62 69 6E 3A  l/bin:/usr/sbin:
2F 75 73 72 2F 62 69 6E  3A 2F 73 62 69 6E 3A 2F  /usr/bin:/sbin:/
62 69 6E 3A 2F 75 73 72  2F 67 61 6D 65 73 3A 2F  bin:/usr/games:/
75 73 72 2F 6C 6F 63 61  6C 2F 6A 64 6B 31 2E 36  usr/local/jdk1.6
2E 30 5F 33 30 2F 62 69  6E 3A 2F 75 73 72 2F 6C  .0_30/bin:/usr/l
6F 63 61 6C 2F 6A 64 6B  31 2E 36 2E 30 5F 33 30  ocal/jdk1.6.0_30
2F 6A 72 65 2F 62 69 6E  3A 2F 68 6F 6D 65 2F 75  /jre/bin:/home/u
62 75 6E 74 75 2F 61 6E  64 72 6F 69 64 2D 73 64  buntu/android-sd
6B 73 2F 70 6C 61 74 66  6F 72 6D 2D 74 6F 6F 6C  ks/platform-tool
73 3A 2F 68 6F 6D 65 2F  75 62 75 6E 74 75 2F 61  s:/home/ubuntu/a
6E 64 72 6F 69 64 2D 73  64 6B 73 2F 74 6F 6F 6C  ndroid-sdks/tool
73 3A 2F 75 73 72 2F 6C  6F 63 61 6C 2F 6A 64 6B  s:/usr/local/jdk
31 2E 36 2E 30 5F 33 30  2F 6C 69 62 2F 74 6F 6F  1.6.0_30/lib/too
6C 73 2E 6A 61 72 00 4C  45 53 53 4F 50 45 4E 3D  ls.jar.LESSOPEN=
7C 20 2F 75 73 72 2F 62  69 6E 2F 6C 65 73 73 70  | /usr/bin/lessp
69 70 65 20 25 73 00 57  49 4E 44 4F 57 50 41 54  ipe %s.WINDOWPAT
48 3D 37 00 44 49 53 50  4C 41 59 3D 3A 30 2E 30  H=7.DISPLAY=:0.0
00 47 54 4B 5F 49 4D 5F  4D 4F 44 55 4C 45 3D 69  .GTK_IM_MODULE=i
62 75 73 00 4C 45 53 53  43 4C 4F 53 45 3D 2F 75  bus.LESSCLOSE=/u
73 72 2F 62 69 6E 2F 6C  65 73 73 70 69 70 65 20  sr/bin/lesspipe 
25 73 20 25 73 00 43 4F  4C 4F 52 54 45 52 4D 3D  %s %s.COLORTERM=
67 6E 6F 6D 65 2D 74 65  72 6D 69 6E 61 6C 00 58  gnome-terminal.X
41 55 54 48 4F 52 49 54  59 3D 2F 76 61 72 2F 72  AUTHORITY=/var/r
75 6E 2F 67 64 6D 2F 61  75 74 68 2D 66 6F 72 2D  un/gdm/auth-for-
75 62 75 6E 74 75 2D 48  79 76 7A 50 6F 2F 64 61  ubuntu-HyvzPo/da
74 61 62 61 73 65 00 5F  3D 2E 2F 6C 69 6E 75 78  tabase._=./linux
5F 73 65 72 76 65 72 00  70 61 67 65 6F 75 74 00  _server.pageout.
00 00 00 00                                        ....
```

Figure 3 Collecting basic system information

- The malware starts process fork() and determine the PID number of its child process. If the execution succeeds, then the main function will exit and cannot debug. Debugging process is shown as below:



```
    ; Attributes: bp-based frame

    sub_804A0C0 proc near

    var_28= byte ptr -28h
    arg_0= dword ptr  8
    arg_4= dword ptr  0Ch

000 push    ebp
004 mov     eax, offset byte_805B1BF
004 mov     ebp, esp
004 sub     esp, 38h        ; Integer Subtraction
03C mov     edx, 6
03C and     esp, 0FFFFFFF0h ; Logical AND
03C sub     esp, 10h        ; Integer Subtraction
04C mov     [esp+4], eax
04C lea     eax, [ebp+var_28] ; Load Effective Address
04C mov     [esp+8], edx
04C mov     [esp], eax
04C call    decode1         ; Call Procedure
04C mov     [esp], eax
04C call    sub_8049ABC     ; Call Procedure
03C mov     eax, [ebp+arg_4]
03C mov     [esp+4], eax
03C mov     eax, [ebp+arg_0]
03C mov     [esp], eax
03C call    sub_8049F60     ; Call Procedure
03C call    sub_8053D90     ; Call Procedure
03C call    sub_8053F40     ; Call Procedure
03C test    eax, eax        ; Logical Compare
03C jz      short loc_804A115 ;the child process succeeds or not
```

```
03C leave                   ; High Level Procedure Exit
000 xor     eax, eax        ; Logical Exclusive OR
000 retn                    ; Return Near from Procedure
```

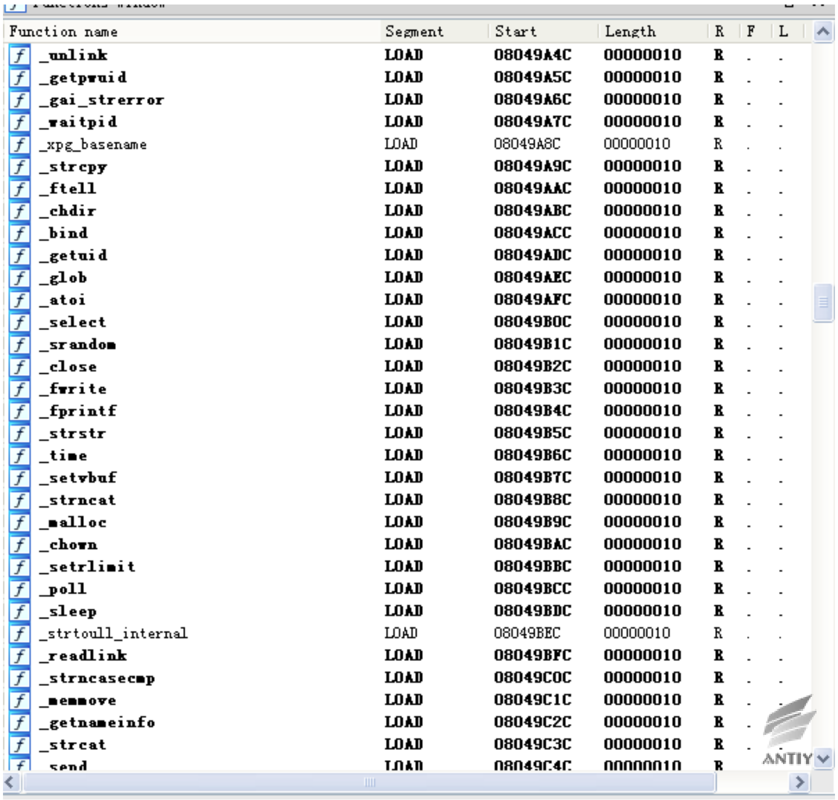
```
    loc_804A115:            ; Call Procedure
03C call    sub_804A090
    sub_804A0C0 endp
```

Figure 4 Child-process judgment

- Decrypting various strings, obtaining user’s information including the system version
- Obtaining user’s login information getpwnam
- Viewing file /bin/fast /sbin/login /usr/sbin/nologin
- Getting user’s login password getpwuid
- Read user’s log var/log /lastlog

3.1.4 Dynamic Loading of Function and Data

The function and data called by this sample is dynamically loaded and debugged, dynamic debugging is wanted in the analysis. We explain the calling addresses through dynamic analysis decryption, and the details are shown as bellows:

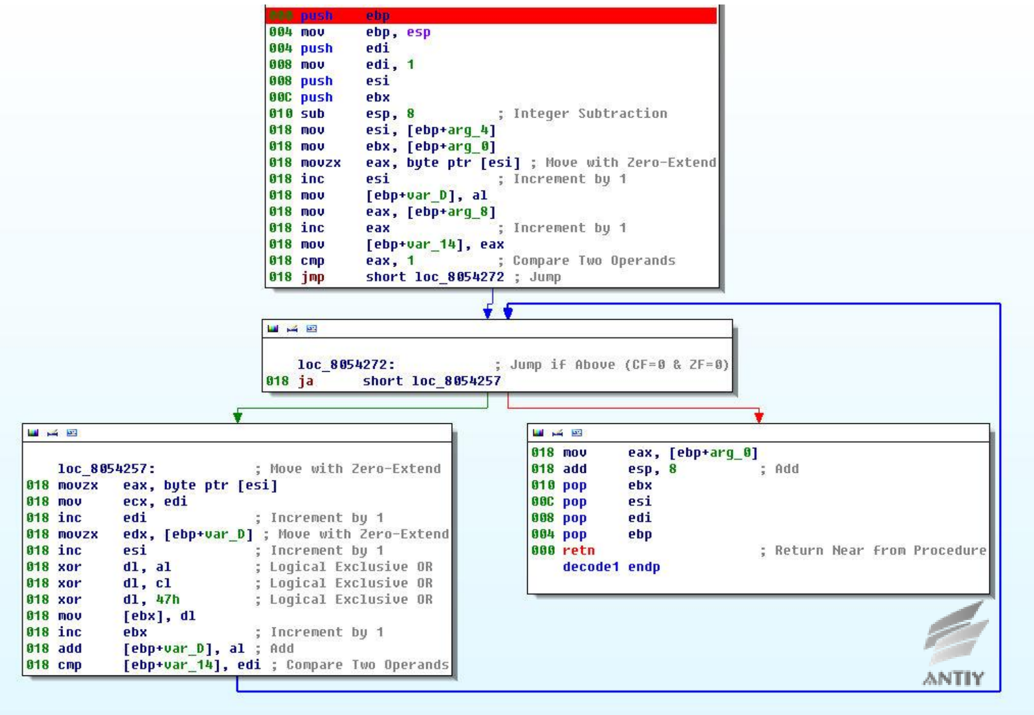


Function name	Segment	Start	Length	R	F	L
_unlink	LOAD	08049A4C	00000010	R	.	.
_getpwuid	LOAD	08049A5C	00000010	R	.	.
_gai_strerror	LOAD	08049A6C	00000010	R	.	.
_waitpid	LOAD	08049A7C	00000010	R	.	.
_xpg_basename	LOAD	08049A8C	00000010	R	.	.
_strcpy	LOAD	08049A9C	00000010	R	.	.
_ftell	LOAD	08049AAC	00000010	R	.	.
_chdir	LOAD	08049ABC	00000010	R	.	.
_bind	LOAD	08049ACC	00000010	R	.	.
_getuid	LOAD	08049ADC	00000010	R	.	.
_glob	LOAD	08049AEC	00000010	R	.	.
_atoi	LOAD	08049AFC	00000010	R	.	.
_select	LOAD	08049B0C	00000010	R	.	.
_srandom	LOAD	08049B1C	00000010	R	.	.
_close	LOAD	08049B2C	00000010	R	.	.
_fwrite	LOAD	08049B3C	00000010	R	.	.
_fprintf	LOAD	08049B4C	00000010	R	.	.
_strstr	LOAD	08049B5C	00000010	R	.	.
_time	LOAD	08049B6C	00000010	R	.	.
_setvbuf	LOAD	08049B7C	00000010	R	.	.
_strncat	LOAD	08049B8C	00000010	R	.	.
_malloc	LOAD	08049B9C	00000010	R	.	.
_chown	LOAD	08049BAC	00000010	R	.	.
_setrlimit	LOAD	08049BBC	00000010	R	.	.
_poll	LOAD	08049BCC	00000010	R	.	.
_sleep	LOAD	08049BDC	00000010	R	.	.
_strtoull_internal	LOAD	08049BEC	00000010	R	.	.
_readlink	LOAD	08049BFC	00000010	R	.	.
_strncasecmp	LOAD	08049C0C	00000010	R	.	.
_memmove	LOAD	08049C1C	00000010	R	.	.
_getnameinfo	LOAD	08049C2C	00000010	R	.	.
_strcat	LOAD	08049C3C	00000010	R	.	.
_send	LOAD	08049C4C	00000010	R		

Figure 5 Function call address

3.1.5 Decryption and Analysis of String

In the sample, a self-defined encryption algorithm is used to encrypt the internal string information. The algorithm is called 115 times. The encryption algorithm is as follows:



```
000 push    ebp
004 mov     ebp, esp
004 push    edi
008 mov     edi, 1
008 push    esi
00C push    ebx
010 sub     esp, 8            ; Integer Subtraction
018 mov     esi, [ebp+arg_4]
018 mov     ebx, [ebp+arg_0]
018 movzx   eax, byte ptr [esi] ; Move with Zero-Extend
018 inc     esi               ; Increment by 1
018 mov     [ebp+var_D], al
018 mov     eax, [ebp+arg_8]
018 inc     eax               ; Increment by 1
018 mov     [ebp+var_14], eax
018 cmp     eax, 1            ; Compare Two Operands
018 jmp     short loc_8054272 ; Jump
```

```
    loc_8054272:              ; Jump if Above (CF=0 & ZF=0)
018 ja      short loc_8054257
```

```
    loc_8054257:              ; Move with Zero-Extend
018 movzx   eax, byte ptr [esi]
018 mov     ecx, edi
018 inc     edi               ; Increment by 1
018 movzx   edx, [ebp+var_D] ; Move with Zero-Extend
018 inc     esi               ; Increment by 1
018 xor     dl, al            ; Logical Exclusive OR
018 xor     dl, cl            ; Logical Exclusive OR
018 xor     dl, 47h           ; Logical Exclusive OR
018 mov     [ebx], dl
018 inc     ebx               ; Increment by 1
018 add     [ebp+var_D], al ; Add
018 cmp     [ebp+var_14], edi ; Compare Two Operands
```

```
018 mov     eax, [ebp+arg_0]
018 add     esp, 8            ; Add
010 pop     ebx
00C pop     esi
008 pop     edi
004 pop     ebp
000 retn                      ; Return Near from Procedure
    decode1 endp
```

Figure 6 String encryption algorithm used in Linux samples

3.1.6 Network Communication Encryption

During the period of network communication performed by Linux samples, the 16-bit key hardcoded in the sample is the same as the 16-bit key in the Windows platform DoubleFantasy sample that encrypts the registry:

66 39 71 3c 0f 85 99 81 20 19 35 43 fe 9a 84 11

The calculated subkeys are:

E9 BE CD E0 A8 9F 4D DB C3 42 AC 2B 24 77 AB CB 5A C1 52 F8 5B 3E F0 78 CB 01 0A 69 29 8F 85 8C

03 9C 7C EF 5E 36 0E 8B C0 40 76 28 9C 9C F2 24 81 9D 02 72 4F 6A BB B5 5B 42 73 14 88 F2 73 75

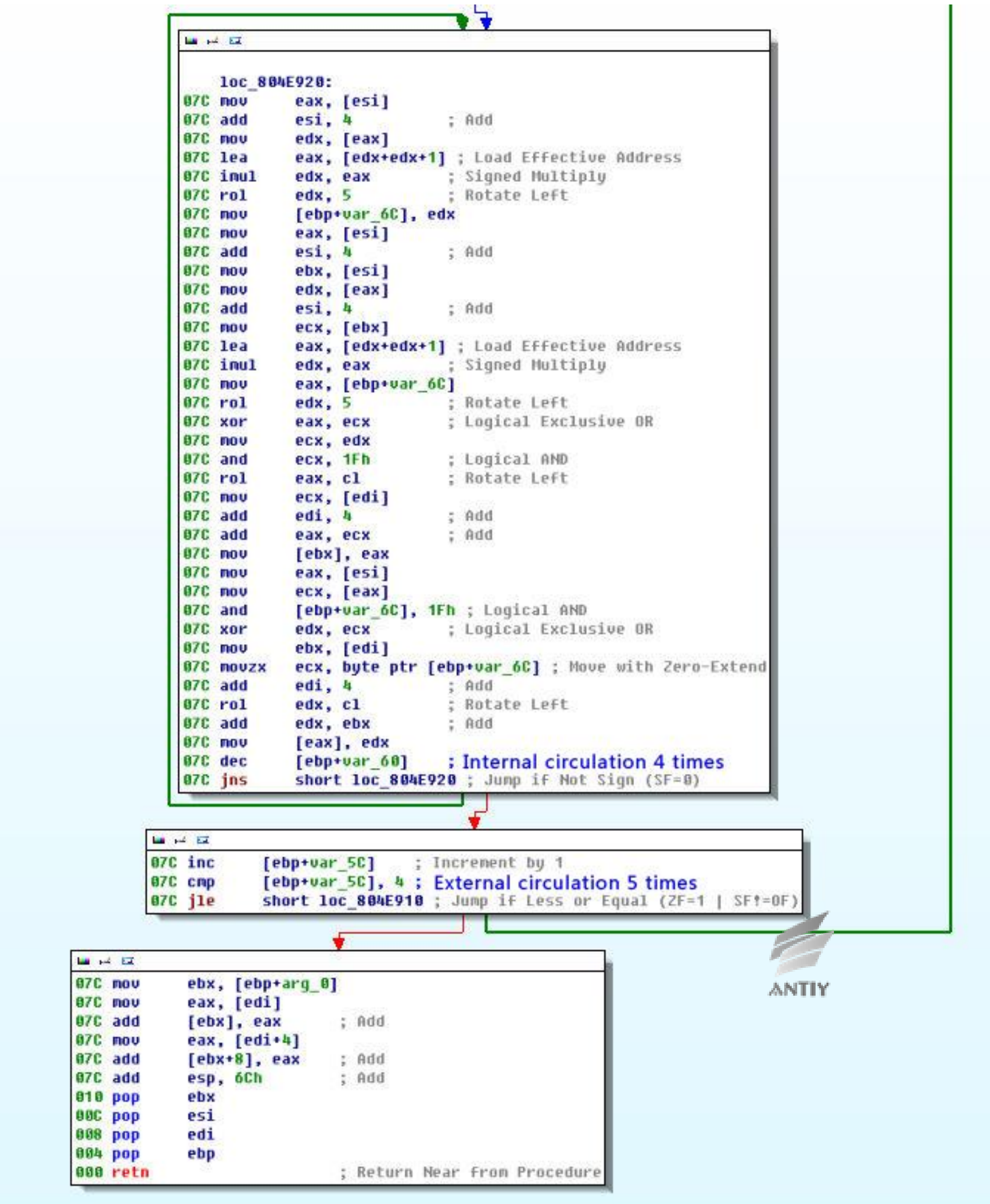
8B F9 37 98 3B 9F 64 2B A3 C4 FF C7 8A 40 67 C1 25 9F 65 54 45 36 48 FF E2 86 05 1A F4 94 AC 2B

08 D5 E5 83 BE 2C AD EE D0 A6 98 CB 8D 35 ED EE C4 F0 8C F2 CD BA 87 03 54 27 3D 13 A7 9B 6A 05

C7 02 30 21 05 67 58 3B E6 A1 44 0A 37 16 3C 86 E9 BC 8B 20 1A 98 7E 28 E6 7F F7 CA F7 9E 38 31

7F F0 2F 93 11 2B 28 F0 FF 11 B7 FC 1C 63 86 CB

The custom algorithm for Linux samples is the same as for Windows, and there is only one encryption key to use (Because the Linux system does not have a registry, there is no registry encryption function). It uses the Windows platform Key for encryption and decryption, and we can see that both platforms use the same secondary key transformation algorithm (Specific details can be seen in the Windows encryption algorithm analysis part).



```
    loc_804E920:
07C mov     eax, [esi]
07C add     esi, 4             ; Add
07C mov     edx, [eax]
07C lea     eax, [edx+edx+1] ; Load Effective Address
07C imul    edx, eax           ; Signed Multiply
07C rol     edx, 5             ; Rotate Left
07C mov     [ebp+var_6C], edx
07C mov     eax, [esi]
07C add     esi, 4             ; Add
07C mov     ebx, [esi]
07C mov     edx, [eax]
07C add     esi, 4             ; Add
07C mov     ecx, [ebx]
07C lea     eax, [edx+edx+1] ; Load Effective Address
07C imul    edx, eax           ; Signed Multiply
07C mov     eax, [ebp+var_6C]
07C rol     edx, 5             ; Rotate Left
07C xor     eax, ecx           ; Logical Exclusive OR
07C mov     ecx, edx
07C and     ecx, 1Fh           ; Logical AND
07C rol     eax, cl            ; Rotate Left
07C mov     ecx, [edi]
07C add     edi, 4             ; Add
07C add     eax, ecx           ; Add
07C mov     [ebx], eax
07C mov     eax, [esi]
07C mov     ecx, [eax]
07C and     [ebp+var_6C], 1Fh ; Logical AND
07C xor     edx, ecx           ; Logical Exclusive OR
07C mov     ebx, [edi]
07C movzx   ecx, byte ptr [ebp+var_6C] ; Move with Zero-Extend
07C add     edi, 4             ; Add
07C rol     edx, cl            ; Rotate Left
07C add     edx, ebx           ; Add
07C mov     [eax], edx
07C dec     [ebp+var_60]       ; Internal circulation 4 times
07C jns     short loc_804E920 ; Jump if Not Sign (SF=0)
```

```
07C inc     [ebp+var_5C]      ; Increment by 1
07C cmp     [ebp+var_5C], 4 ; External circulation 5 times
07C jle     short loc_804E910 ; Jump if Less or Equal (ZF=1 | SF!=OF)
```

```
07C mov     ebx, [ebp+arg_0]
07C mov     eax, [edi]
07C add     [ebx], eax        ; Add
07C mov     eax, [edi+4]
07C add     [ebx+8], eax      ; Add
07C add     esp, 6Ch          ; Add
010 pop     ebx
00C pop     esi
008 pop     edi
004 pop     ebp
000 retn                      ; Return Near from Procedure
```

Figure 7 Secondary key transformation algorithm

3.1.7 Network control instruction

Instruction branch of Linux sample is basically the same as Windows. There are a total of nine instruction branches, and the function is also roughly the same. The instruction codes are: 0x4A, 0x4B, 0x60, 0x70, 0x75, 0x76, 0x78, 0x79, 0x80.

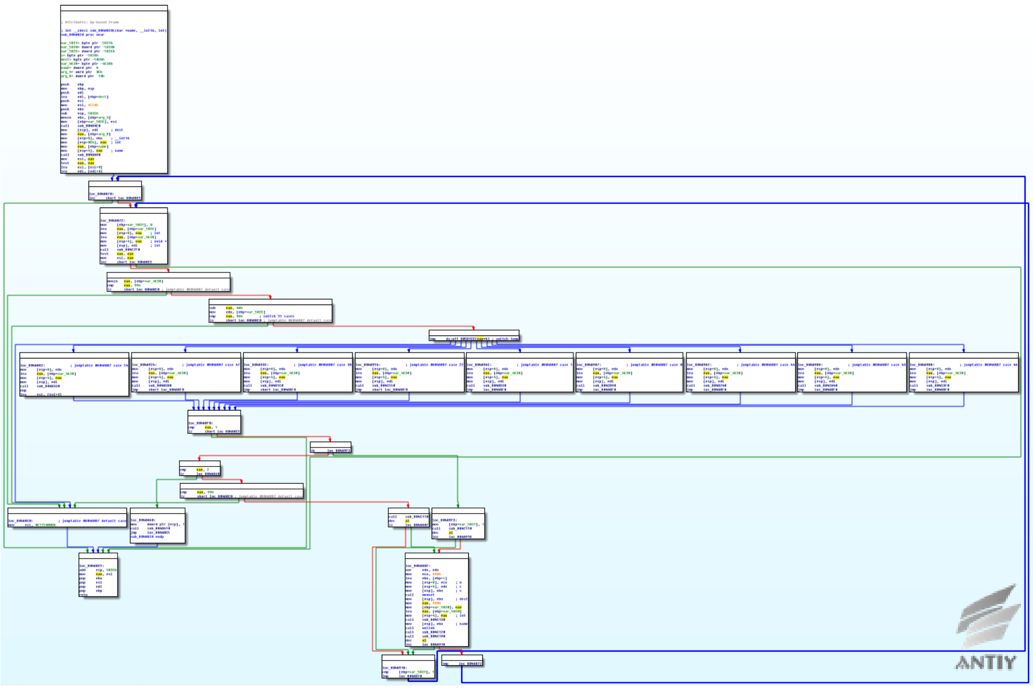
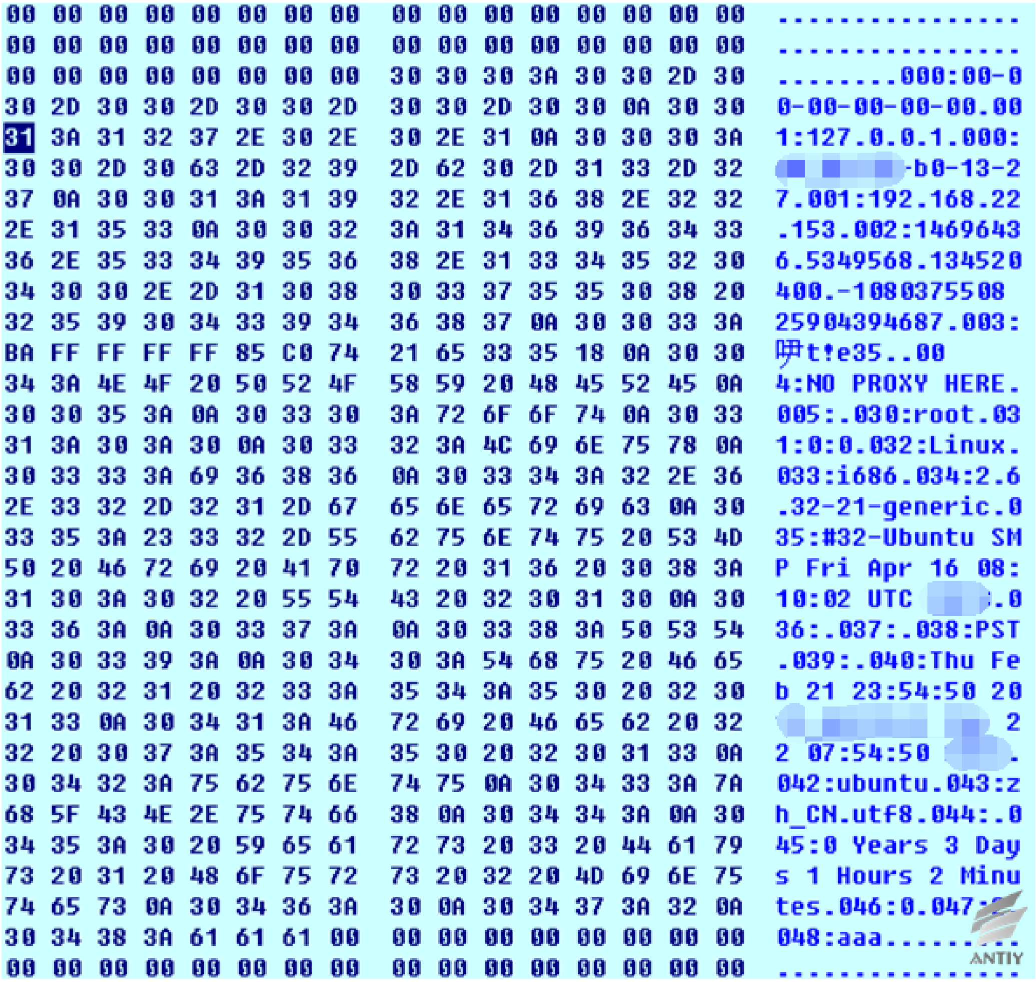


Figure 8 Instruction branch code for Linux samples

The function of instructions for Linux system is the same as the Windows sample function, only with the difference between obtaining system information. The following shows Linux sample accessing to information format:



```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  30 30 30 3A 30 30 2D 30  ........000:00-0
30 2D 30 30 2D 30 30 2D  30 30 2D 30 30 0A 30 30  0-00-00-00-00.00
31 3A 31 32 37 2E 30 2E  30 2E 31 0A 30 30 30 3A  1:127.0.0.1.000:
30 30 2D 30 63 2D 32 39  2D 62 30 2D 31 33 2D 32  [redacted]-b0-13-2
37 0A 30 30 31 3A 31 39  32 2E 31 36 38 2E 32 32  7.001:192.168.22
2E 31 35 33 0A 30 30 32  3A 31 34 36 39 36 34 33  .153.002:1469643
36 2E 35 33 34 39 35 36  38 2E 31 33 34 35 32 30  6.5349568.134520
34 30 30 2E 2D 31 30 38  30 33 37 35 35 30 38 20  400.-1080375508
32 35 39 30 34 33 39 34  36 38 37 0A 30 30 33 3A  25904394687.003:
BA FF FF FF FF 85 C0 74  21 65 33 35 18 0A 30 30  甲t!e35..00
34 3A 4E 4F 20 50 52 4F  58 59 20 48 45 52 45 0A  4:NO PROXY HERE.
30 30 35 3A 0A 30 33 30  3A 72 6F 6F 74 0A 30 33  005:.030:root.03
31 3A 30 3A 30 0A 30 33  32 3A 4C 69 6E 75 78 0A  1:0:0.032:Linux.
30 33 33 3A 69 36 38 36  0A 30 33 34 3A 32 2E 36  033:i686.034:2.6
2E 33 32 2D 32 31 2D 67  65 6E 65 72 69 63 0A 30  .32-21-generic.0
33 35 3A 23 33 32 2D 55  62 75 6E 74 75 20 53 4D  35:#32-Ubuntu SM
50 20 46 72 69 20 41 70  72 20 31 36 20 30 38 3A  P Fri Apr 16 08:
31 30 3A 30 32 20 55 54  43 20 32 30 31 30 0A 30  10:02 UTC [redacted].0
33 36 3A 0A 30 33 37 3A  0A 30 33 38 3A 50 53 54  36:.037:.038:PST
0A 30 33 39 3A 0A 30 34  30 3A 54 68 75 20 46 65  .039:.040:Thu Fe
62 20 32 31 20 32 33 3A  35 34 3A 35 30 20 32 30  b 21 23:54:50 20
31 33 0A 30 34 31 3A 46  72 69 20 46 65 62 20 32  [redacted] 2
32 20 30 37 3A 35 34 3A  35 30 20 32 30 31 33 0A  2 07:54:50 [redacted].
30 34 32 3A 75 62 75 6E  74 75 0A 30 34 33 3A 7A  042:ubuntu.043:z
68 5F 43 4E 2E 75 74 66  38 0A 30 34 34 3A 0A 30  h_CN.utf8.044:.0
34 35 3A 30 20 59 65 61  72 73 20 33 20 44 61 79  45:0 Years 3 Day
73 20 31 20 48 6F 75 72  73 20 32 20 4D 69 6E 75  s 1 Hours 2 Minu
74 65 73 0A 30 34 36 3A  30 0A 30 34 37 3A 32 0A  tes.046:0.047:2.
30 34 38 3A 61 61 61 00  00 00 00 00 00 00 00 00  048:aaa.........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Figure 9 Linux sample accessing to information format

The description of obtaining information format:

NO.	Description	NO.	Description	NO.	Description
000	MAC	**033**	Platform type（i386\i686）	**042**	OS（ubuntu）
001	IP	**034**	System kernel version	**043**	Regional language（zh_cn.utf8）
002	Version	**035**	OS type time	**044**	Unknown
003	Clsid	**036**	Unknown	**045**	System uptime
004	Settings information of proxy	**037**	Unknown	**046**	Unknown
005	Unknown	**038**	PST	**047**	Unknown
030	Username	**039**	Unknown	**048**	Sample name
031	Password	**040**	Time		
032	OS type(eg. Linux)	**041**	Time		

4 Attack Payload for Solaris SPARC

Equation may have created the first malware with Rootkit features in SPARC architecture, and provide cover for DoubleFantasy targeting Solaris.

4.1 Solaris and SPARC

Solaris is the computer operating system which is developed by Sun Microsystems. It adopts SPARC or X86, and is mainly used for workstations, servers, operating systems. The malicious code on Solaris platform is rare. From Antiy's statistic, there are no more than 60 kinds of malicious code variants which are binary compiled form even in the period of SUN OS. They are almost based on X86 platform.

The full name of SPARC is Scalable Processor Architecture, one of microprocessor architecture. Its instruction set is significantly differ from X86 and has its own unique window, delay slot, the process call features.

The computer which has SPARC is generally used for industrial, space-related areas. It is seldom used in similar IDC and general IT scenario.

4.2 Hidden Rootkits

This is a rootkit program on the Solaris platform of the SPARC architecture. It is primarily responsible for hiding the main function sample files like other rootkit programs, as well as the associated derived files and itself, including process, file, and service information. It runs on the target computer firstly, investigates system environment, configuration information, network status of the target computer, and hides the specified files and processes.

4.2.1 File Tags

Name	Trojan[Rootkit]/Solaris. Equation
Original file name	[redacted]
MD5	[redacted]
Processor architecture	SPARC-32
Size	[redacted]
Format	BinExecute/ELF

Timestamp	n/a
Signature	None
Shell	None
Language	Linux C

4.2.2 Main function

The samples have 249 functions, as shown in the sample main function flow. Some of the functions are relatively complex. There are a variety of encrypted data in the samples.

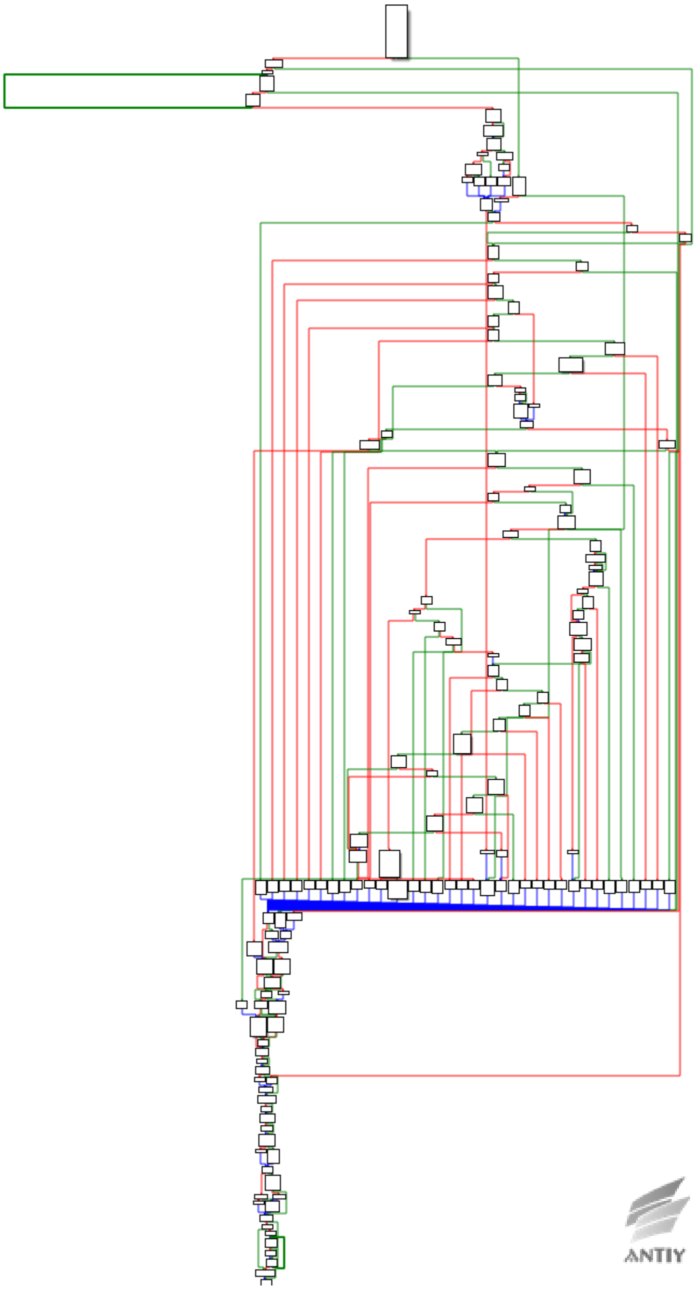


Figure 10 Main functions of the samples

4.2.3 Derived file name and path

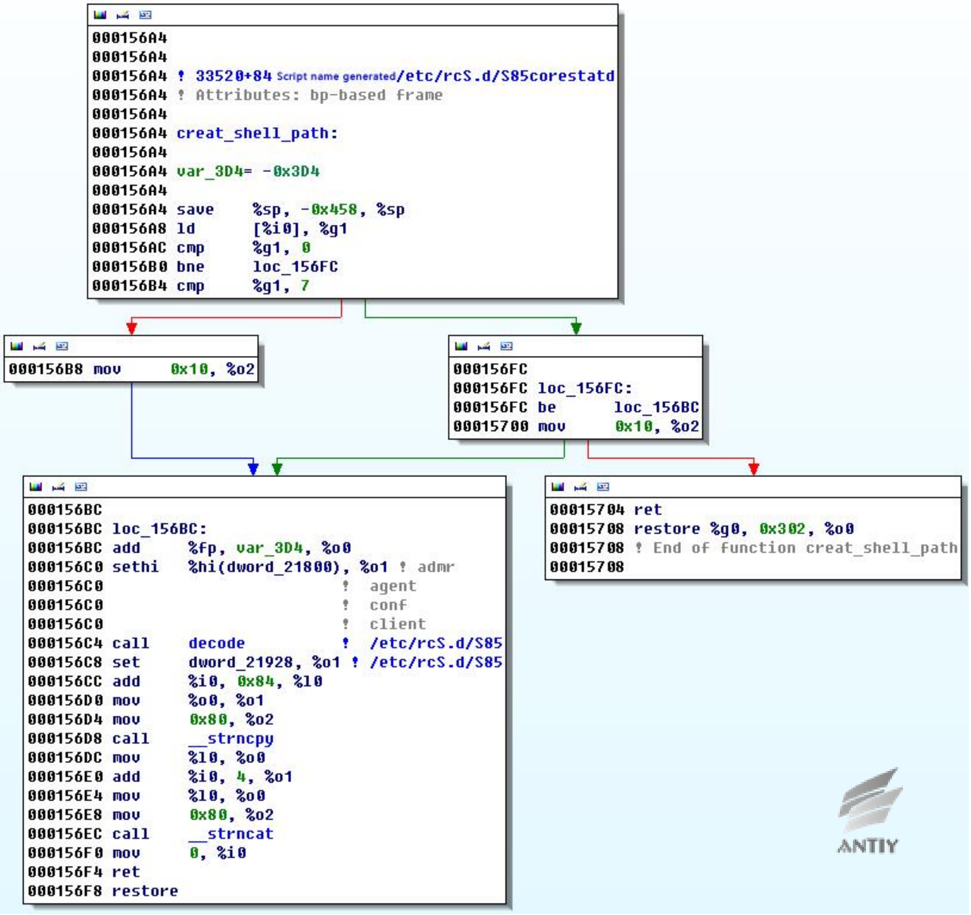
After running, it can combine two sets of strings which are according to the internal configuration to generate file name as its own new file name, and copy itself to the / sbin / directory.

String 1	String 2
audit	admr
boot	agent
cache	conf
core	client
cron	info
init	mgr
inet	statd
filesys	serv

We can find that these words are highly frequency words or suffix used in system files and system command. Thus, the file name of the sample is carefully structured and confusing. The general administrator is also difficult to detect abnormal situation in the system files.

4.2.4 Starting script

The script realizes the startup by using the service. It creates the script in the etc / rc.d / directory (S85s%). This script will run with the start parameter as the service which is executed when open the computer.



```
000156A4
000156A4
000156A4 ! 33520+84 Script name generated/etc/rcS.d/S85corestatd
000156A4 ! Attributes: bp-based frame
000156A4
000156A4 creat_shell_path:
000156A4
000156A4 var_3D4= -0x3D4
000156A4
000156A4 save     %sp, -0x458, %sp
000156A8 ld       [%i0], %g1
000156AC cmp      %g1, 0
000156B0 bne      loc_156FC
000156B4 cmp      %g1, 7
```

```
000156B8 mov      0x10, %o2
```

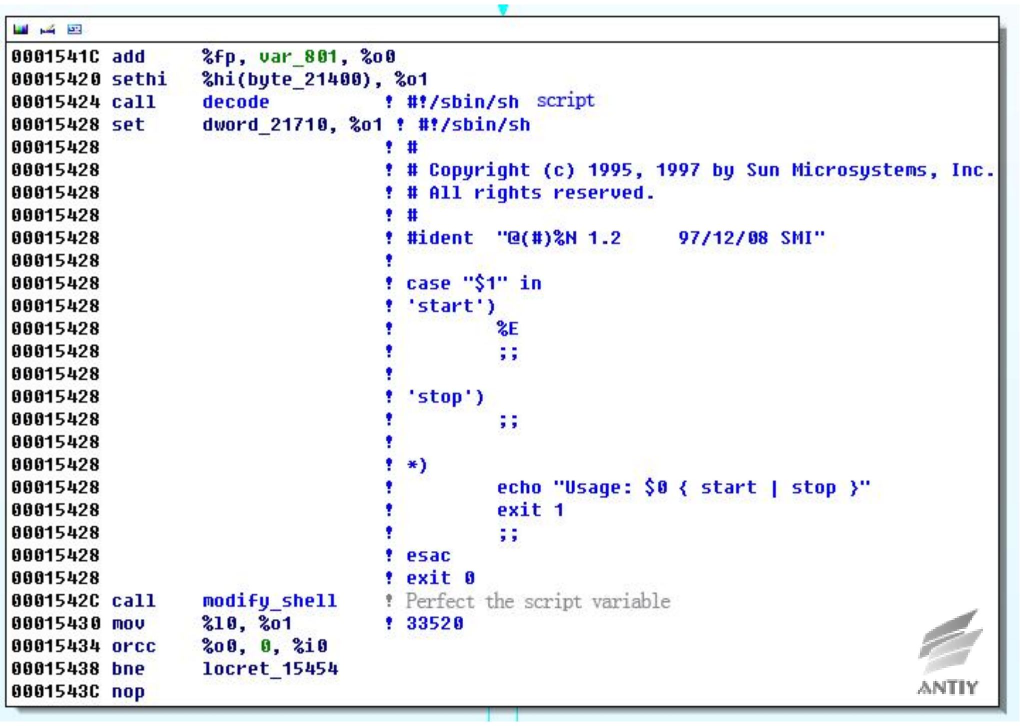
```
000156FC
000156FC loc_156FC:
000156FC be       loc_156BC
00015700 mov      0x10, %o2
```

```
000156BC
000156BC loc_156BC:
000156BC add      %fp, var_3D4, %o0
000156C0 sethi    %hi(dword_21800), %o1 ! admr
000156C0                               !  agent
000156C0                               !  conf
000156C0                               !  client
000156C4 call     decode               !  /etc/rcS.d/S85
000156C8 set      dword_21928, %o1 ! /etc/rcS.d/S85
000156CC add      %i0, 0x84, %l0
000156D0 mov      %o0, %o1
000156D4 mov      0x80, %o2
000156D8 call     __strncpy
000156DC mov      %l0, %o0
000156E0 add      %i0, 4, %o1
000156E4 mov      %l0, %o0
000156E8 mov      0x80, %o2
000156EC call     __strncat
000156F0 mov      0, %i0
000156F4 ret
000156F8 restore
```

```
00015704 ret
00015708 restore %g0, 0x302, %o0
00015708 ! End of function creat_shell_path
00015708
```

Figure 11 Service script

The content of S85s% document is encrypted. After running, it can call its own function to decrypt, and modify the variable of the file name. Then it can write into /etc/rc.d/ directory (It will be modified to the path of the sample in figure below % E).

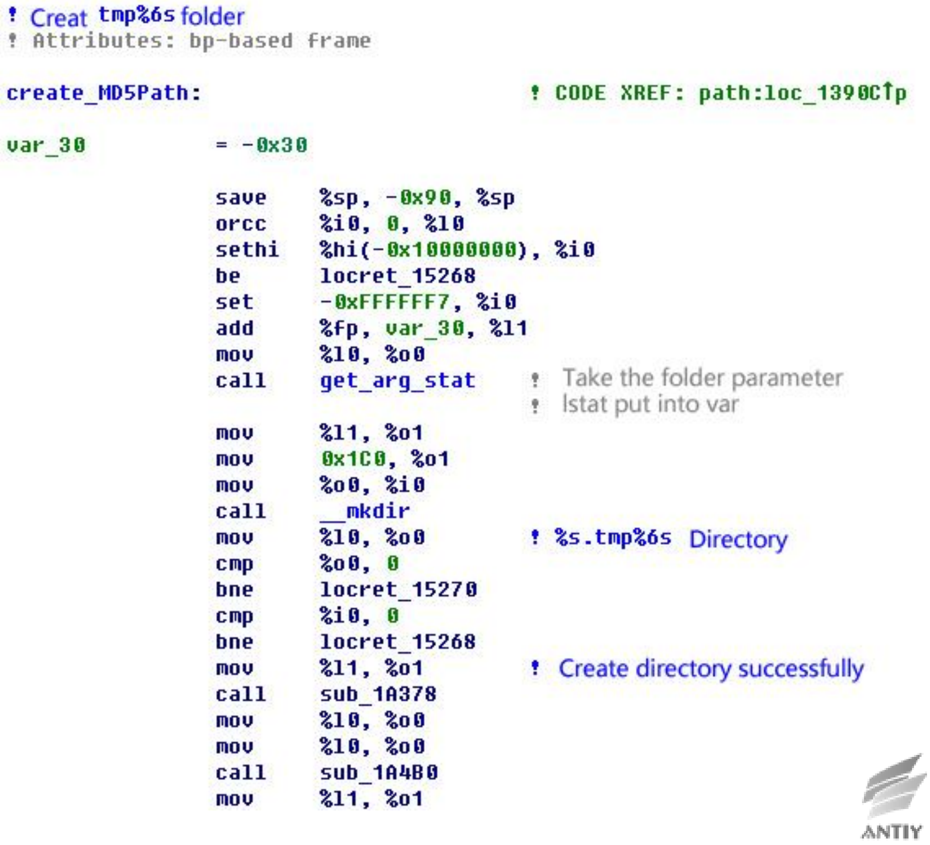


```
0001541C add     %fp, var_801, %o0
00015420 sethi   %hi(byte_21400), %o1
00015424 call    decode          ! #!/sbin/sh  script
00015428 set     dword_21710, %o1 ! #!/sbin/sh
00015428                         ! #
00015428                         ! # Copyright (c) 1995, 1997 by Sun Microsystems, Inc.
00015428                         ! # All rights reserved.
00015428                         ! #
00015428                         ! #ident  "@(#)%N 1.2     97/12/08 SMI"
00015428                         !
00015428                         ! case "$1" in
00015428                         ! 'start')
00015428                         !         %E
00015428                         !         ;;
00015428                         !
00015428                         ! 'stop')
00015428                         !         ;;
00015428                         !
00015428                         ! *)
00015428                         !         echo "Usage: $0 { start | stop }"
00015428                         !         exit 1
00015428                         !         ;;
00015428                         ! esac
00015428                         ! exit 0
0001542C call    modify_shell    ! Perfect the script variable
00015430 mov     %l0, %o1        ! 33520
00015434 orcc    %o0, 0, %i0
00015438 bne     locret_15454
0001543C nop
```

Figure 12 The content of the script after decryption

4.2.5 Hidden directory and files

The sample will generate MD5 based on HOSTID of the target computer, then calculate class base64 algorithm, take the first six bits finally. It can splice .tmp with the first six bits into a folder name and create the folder.



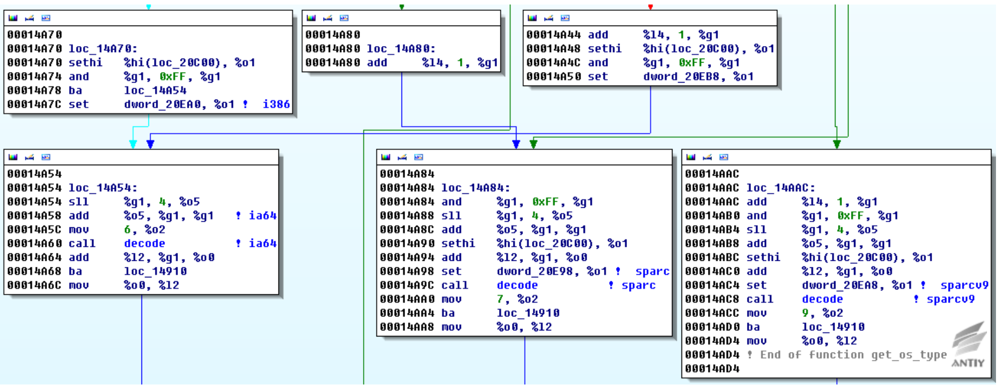
```
! Creat tmp%6s folder
! Attributes: bp-based frame

create_MD5Path:                         ! CODE XREF: path:loc_1390C↑p

var_30          = -0x30

                save    %sp, -0x90, %sp
                orcc    %i0, 0, %l0
                sethi   %hi(-0x10000000), %i0
                be      locret_15268
                set     -0xFFFFFF7, %i0
                add     %fp, var_30, %l1
                mov     %l0, %o0
                call    get_arg_stat    ! Take the folder parameter
                                        ! lstat put into var
                mov     %l1, %o1
                mov     0x1C0, %o1
                mov     %o0, %i0
                call    __mkdir
                mov     %l0, %o0        ! %s.tmp%6s Directory
                cmp     %o0, 0
                bne     locret_15270
                cmp     %i0, 0
                bne     locret_15268
                mov     %l1, %o1        ! Create directory successfully
                call    sub_1A378
                mov     %l0, %o0
                mov     %l0, %o0
                call    sub_1A4B0
                mov     %l1, %o1
```

Figure 13 Folder name created by samples

The sample will also copy other files to execute according to the running parameters. It is responsible for hiding all the files in this folder.

4.2.6 Version judgement

The sample can determine that the system is not sun4m, sun4d version through the uname function. And it can determine the system architecture by reading / dev / ksyms files: i386, ia64, sparc, sparcv9. To make sure the SPARC architecture and the release version is 5.1.

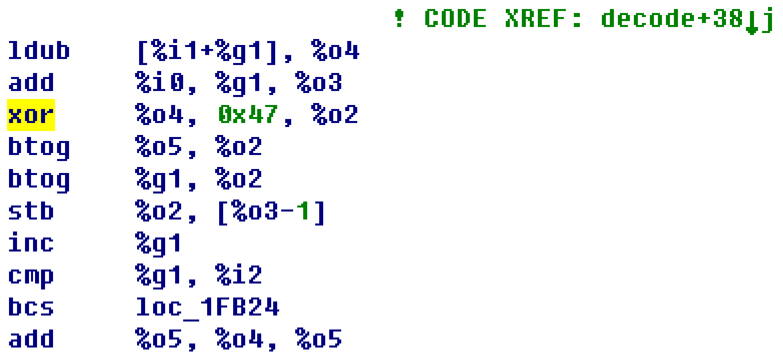


```
00014A70
00014A70 loc_14A70:
00014A70 sethi   %hi(loc_20C00), %o1
00014A74 and     %g1, 0xFF, %g1
00014A78 ba      loc_14A54
00014A7C set     dword_20EA0, %o1 !  i386

00014A80
00014A80 loc_14A80:
00014A80 add     %l4, 1, %g1

00014A44 add     %l4, 1, %g1
00014A48 sethi   %hi(loc_20C00), %o1
00014A4C and     %g1, 0xFF, %g1
00014A50 set     dword_20EB8, %o1

00014A54
00014A54 loc_14A54:
00014A54 sll     %g1, 4, %o5
00014A58 add     %o5, %g1, %g1    ! ia64
00014A5C mov     6, %o2
00014A60 call    decode           ! ia64
00014A64 add     %l2, %g1, %o0
00014A68 ba      loc_14910
00014A6C mov     %o0, %l2

00014A84
00014A84 loc_14A84:
00014A84 and     %g1, 0xFF, %g1
00014A88 sll     %g1, 4, %o5
00014A8C add     %o5, %g1, %g1
00014A90 sethi   %hi(loc_20C00), %o1
00014A94 add     %l2, %g1, %o0
00014A98 set     dword_20E98, %o1 !  sparc
00014A9C call    decode           ! sparc
00014AA0 mov     7, %o2
00014AA4 ba      loc_14910
00014AA8 mov     %o0, %l2

00014AAC
00014AAC loc_14AAC:
00014AAC add     %l4, 1, %g1
00014AB0 and     %g1, 0xFF, %g1
00014AB4 sll     %g1, 4, %o5
00014AB8 add     %o5, %g1, %g1
00014ABC sethi   %hi(loc_20C00), %o1
00014AC0 add     %l2, %g1, %o0
00014AC4 set     dword_20EA8, %o1 !  sparcv9
00014AC8 call    decode           ! sparcv9
00014ACC mov     9, %o2
00014AD0 ba      loc_14910
00014AD4 mov     %o0, %l2
00014AD4 ! End of function get_os_type
00014AD4
```

Figure 10 Version judgement

4.2.7 Encrypting the configuration data

There are multiple encryption algorithms inside the sample. One of the encryption algorithms was called multiple times. We analyzed and decrypted the data.



```
                                      ! CODE XREF: decode+38↓j
ldub       [%i1+%g1], %o4
add        %i0, %g1, %o3
xor        %o4, 0x47, %o2
btog       %o5, %o2
btog       %g1, %o2
stb        %o2, [%o3-1]
inc        %g1
cmp        %g1, %i2
bcs        loc_1FB24
add        %o5, %o4, %o5
```

Figure 15 Encryption algorithm

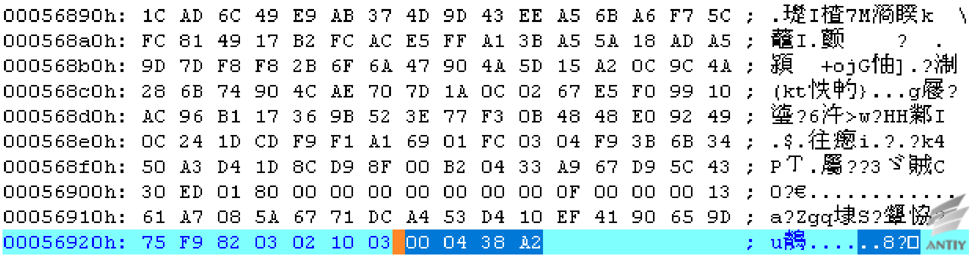
Decrypting encrypted data as follows：

Offset	Plaintext	Offset	Plaintext
0x10cd0	/platform/%s/kernel/sparcv9/unix	0x11830	mgr
0x10cf8	/var/sadm/i	0x11838	statd
0x10d28	SUNW	0x11840	serv
0x10d30	/var/sadm/patch/%s/README.%s	0x11848	svcd
0x10d50	var/sadm/pkg/%s/pkginfo	0x11851	\
0x10d70	PATCHLIST	0x11855	\W
0x10d80	/var/sadm/pkg/%s/pkginfo	0x11859	\O
0x10da0	PATCH_INFO	0x1185d	\G
0x10db8	Requires:	0x11861	\w
0x10dc8	Ob	0x11865	\o
0x10dcc	!8I 秨;	0x11869	\g
0x10dd8	Incompatibles:	0x1186d	\
0x10df0	module_main	0x11871	\
0x10e10	%s/%s	0x11878	audit
0x10e18	date	0x11880	boot
0x10e20	/etc/mnttab	0x11888	cache

0x10e38	swap	0x11890	core
0x10e40	tmpfs	0x11898	cron
0x10e48	ro	0x118a0	init
0x10e50	noexec	0x118a8	inet
0x10e60	D	0x118b0	filesys
0x10e68	sun4m	0x118c0	key
0x10e70	sun4d	0x118c8	ntp
0x10e78	sparc	0x118d0	root
0x10e80	/dev/ksyms	0x118d8	sys
0x10e98	sparc	0x118e0	rpcd
0x10ea0	i386	0x118e8	vol
0x10ea8	sparcv	0x11940	/
0x10eb8	ia64	0x11948	/usr/bin/
0x10ec0	sparc	0x11958	/bin/
0x10ec8	SunOS	0x11960	/sbin/
0x10ed0	Generic	0x11970	var/tmp/faipprep001
0x10ee0	boothowto	0x11990	init
0x10ef8	/dev/ksyms	0x11998	fini
0x10f08	/dev/kmem	0x119a0	minit
0x116d0	/var/tmp/	0x119a8	fini
0x116e0	/lib/	0x119b0	mdata
0x116e8	/dev/	0x119b8	priocntlsys
0x116f0	/etc/	0x119c8	/dev/ksyms
0x116f8	/	0x119d8	init
0x11700	%s.tmp%6s	0x119e0	/dev/kmem
0x11710	#!/sbin/sh	0x119f0	/dev/mem
0x11800	admr	0x11a00	/proc/self
0x1180a	NCk	0x11a10	.got
0x11810	conf	0x11a18	.got
0x11818	client	0x11a28	.got
0x11828	info	0x11a30	GLOBAL_OFFSET_TABLE_

4.2.8 Decrypting and executing other codes / samples

The sample adds the encrypted data at the end of the file. After running, it can decide the size of the encrypted data through the end data, parse and read the data through the defined format. It may load and execute after decrypting the data.



```
00056890h: 1C AD 6C 49 E9 AB 37 4D 9D 43 EE A5 6B A6 F7 5C ; .瑗I楂7M滵睽k  \
000568a0h: FC 81 49 17 B2 FC AC E5 FF A1 3B A5 5A 18 AD A5 ; 麎I.颤    ?  .
000568b0h: 9D 7D F8 F8 2B 6F 6A 47 90 4A 5D 15 A2 0C 9C 4A ; 頴  +ojG愐J].?淛
000568c0h: 28 6B 74 90 4C AE 70 7D 1A 0C 02 67 E5 F0 99 10 ; (kt怢畃}...g屦?
000568d0h: AC 96 B1 17 36 9B 52 3E 77 F3 0B 48 48 E0 92 49 ; 瑬?6汻>w?HH鄵I
000568e0h: 0C 24 1D CD F9 F1 A1 69 01 FC 03 04 F9 3B 6B 34 ; .$.往瘛i.?.?k4
000568f0h: 50 A3 D4 1D 8C D9 8F 00 B2 04 33 A9 67 D9 5C 43 ; P T.屬??3 g賊C
00056900h: 30 ED 01 80 00 00 00 00 00 00 00 0F 00 00 00 13 ; 0?€.............
00056910h: 61 A7 08 5A 67 71 DC A4 53 D4 10 EF 41 90 65 9D ; a?Zgq埭S?颽恊?
00056920h: 75 F9 82 03 02 10 03 00 04 38 A2                ; u鶄.......8?
```

4.3 DoubleFantasy modules for SPARC

The function of the sample is the same with the one on Windows and Linux platform. The main differences are CPU architecture, assembly instructions, storage location of the configuration information and obtained system information.

4.3.1 File Tags

Name	Trojan/Solaris.DoubleFantasy
Original File Name	[redacted]
MD5	[redacted]
Processor Architecture	SPARC-32
Size	[redacted]
Format	BinExecute/ELF
Timestamp	n/a
Signature	None
Shell	None
Language	Linux C

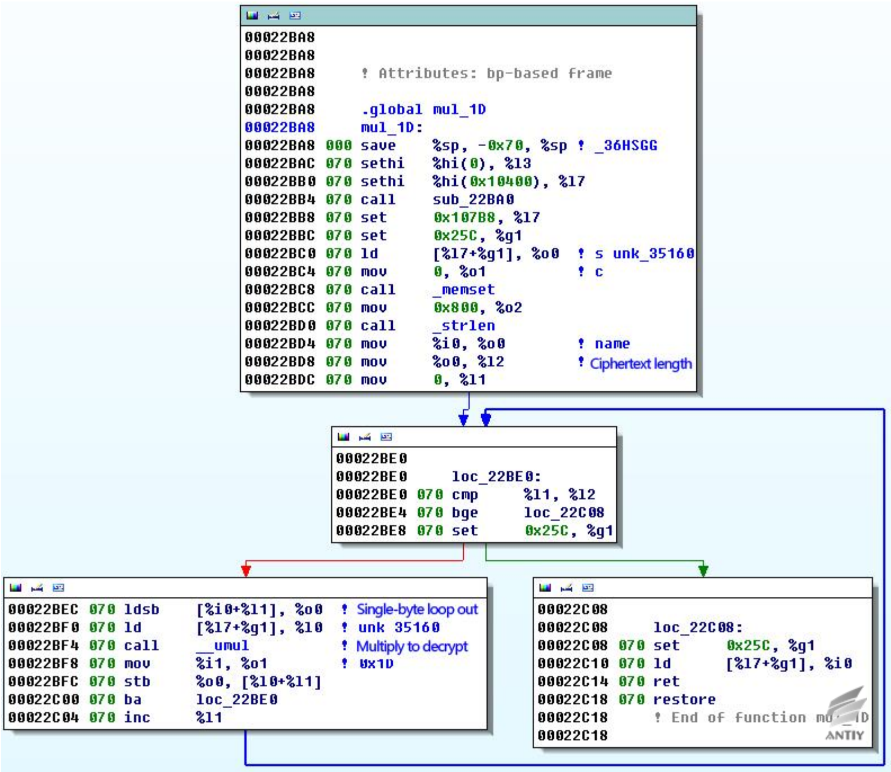
4.3.2 Basic functions

- Initializing the string, dynamic array, decrypt the internal configuration information.
- Connecting Google or Yahoo URL to determine network connectivity.
- Connecting remote URL address. Its remote C & C server address is xxxech.com. The corresponding IP is xxx.xxx.235.237 (One IP of Windows:xxx.xxx.235.235. xxx.xxx.235.235. It belongs to the same network segment. It's basically determined as the same attack source). It will collect the host information, back to the above address, and wait for the remote host to send instructions.

- Reading the system account password file, retrieve user information and password.
- Run with the daemon mode in the sample, achieve the ability of self-protection.
- Use a variety of encryption algorithms to encrypt string information.
- Collecting detailed system information and sending back to the server (such as computer name, IP address, process information, account information, etc., the details can be seen in detailed analysis later in this chapter).
- Have 7 network instructions, same functions with Windows version, execute the corresponding instruction operation. The detailed functions of the corresponding command can be seen in detailed analysis later in this chapter.

4.3.3 Configuring Information Encryption

Due to Solaris system does not have the Windows registry, the configuration data will be directly used after decryption. We can see one of the decryption algorithms as follows. The decryption function is called for 63 times.



```
00022BA8
00022BA8
00022BA8      ! Attributes: bp-based frame
00022BA8
00022BA8      .global mul_1D
00022BA8      mul_1D:
00022BA8 000 save    %sp, -0x70, %sp ! _36HSGG
00022BAC 070 sethi   %hi(0), %l3
00022BB0 070 sethi   %hi(0x10400), %l7
00022BB4 070 call    sub_22BA0
00022BB8 070 set     0x107B8, %l7
00022BBC 070 set     0x25C, %g1
00022BC0 070 ld      [%l7+%g1], %o0  ! s unk_35160
00022BC4 070 mov     0, %o1          ! c
00022BC8 070 call    _memset
00022BCC 070 mov     0x800, %o2
00022BD0 070 call    _strlen
00022BD4 070 mov     %i0, %o0        ! name
00022BD8 070 mov     %o0, %l2        ! Ciphertext length
00022BDC 070 mov     0, %l1
```

```
00022BE0
00022BE0      loc_22BE0:
00022BE0 070 cmp     %l1, %l2
00022BE4 070 bge     loc_22C08
00022BE8 070 set     0x25C, %g1
```

```
00022BEC 070 ldsb    [%i0+%l1], %o0  ! Single-byte loop out
00022BF0 070 ld      [%l7+%g1], %l0  ! unk 35160
00022BF4 070 call    __umul          ! Multiply to decrypt
00022BF8 070 mov     %i1, %o1        ! 0x1D
00022BFC 070 stb     %o0, [%l0+%l1]
00022C00 070 ba      loc_22BE0
00022C04 070 inc     %l1
```

```
00022C08
00022C08      loc_22C08:
00022C08 070 set     0x25C, %g1
00022C10 070 ld      [%l7+%g1], %i0
00022C14 070 ret
00022C18 070 restore
00022C18      ! End of function mul_1D
00022C18
```

Figure 16 Strings decryption

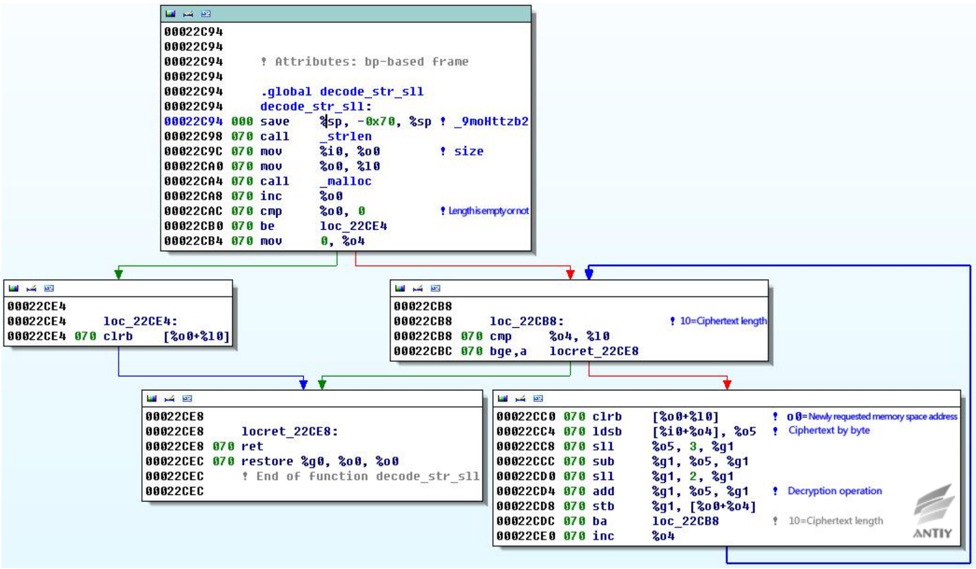
Decrypted string information can be seen in the table below:

Offset	Plaintext	Offset	Plaintext
0x1346c	' 200 Connection established'	0x13560	'Content-Length:'

0x13470	' 200 OK'	0x13458	'Content-Length: %d'
0x133fc	' days '	0x13460	'Content-length: %d'
0x13400	' hrs '	0x13488	'Cookie: %s'
0x13540	' HTTP/1.1\r\n'	0x13554	'GET '
0x1340c	' logged in'	0x13544	'Host: '
0x13404	' mins '	0x13474	'HTTP/'
0x13408	' total'	0x13478	'HTTP/1.0 200 OK'
0x133f4	' yrs '	0x13534	'http://'
0x133d8	'""'	0x13384	'I_MASK'
0x13584	'%02x-%02x-%02x-%02x-%02x-%02x	0x133ec	'LANG'
0x13594	'%u.%u.%u.%u'	0x133f0	'LANGUAGE'
0x134dc	'/.mozilla/'	0x133c0	'LD_PRELOAD='
0x134e0	'/.mozilla/firefox/'	0x13418	'M_MASK'
0x134c4	'/.netscape'	0x133e4	'MACHTYPE'
0x13438	'/bin/false'	0x134b4	'network.proxy.http'
0x13520	'/bin/false'	0x134b8	'network.proxy.http_port'
0x1338c	'/dev/null'	0x134bc	'network.proxy.ssl'
0x133b0	'/dev/null'	0x134c0	'network.proxy.ssl_port'
0x134c8	'/preferences.js'	0x1348c	'p'
0x134cc	'/prefs.js'	0x133b8	'PATH'
0x13428	'/proc'	0x133bc	'PATH='
0x1343c	'/sbin/nologin'	0x1355c	'POST '
0x13524	'/sbin/nologin'	0x13410	'process info: '
0x13390	'/tmp/'	0x1354c	'Proxy-Connection: close\r\n'
0x133b4	'/tmp/'	0x134d8	'S'
0x135a0	'@C\xe3\xc0'	0x1353c	'S'
0x13558	'\r\n'	0x133c4	'sendmail'
0x13464	'\r\n\r\n'	0x13484	'SESSID="0%x%s%x:eac:%lu:%lu"\r\n'
0x13588	'0x%02x%02x%02x%02x%02x%02x'	0x1349c	'user_pref("'
0x1341c	'0xA857'	0x134a0	'user_pref("%s"
0x13388	'0xAA%llu'	0x134a8	'user_pref("%s%s'

0x13568	'CONNECT '	0x134f8	'v'
0x13550	'Connection: close\r\n'	0x13548	'y"'
0x13454	'Content-Length:'	0x1347c	'y"y"'
0x1345c	'Content-length:'		

Another encrypted string algorithm is as the required configuration information when encrypted sample is running. The decryption algorithm is as follows:



```
00022C94
00022C94
00022C94         ! Attributes: bp-based frame
00022C94
00022C94         .global decode_str_sll
00022C94         decode_str_sll:
00022C94 000 save     %sp, -0x70, %sp ! _9moHttzb2
00022C98 070 call     _strlen
00022C9C 070 mov      %i0, %o0        ! size
00022CA0 070 mov      %o0, %l0
00022CA4 070 call     _malloc
00022CA8 070 inc      %o0
00022CAC 070 cmp      %o0, 0          ! Lengthisemptyornot
00022CB0 070 be       loc_22CE4
00022CB4 070 mov      0, %o4
```

```
00022CE4
00022CE4         loc_22CE4:
00022CE4 070 clrb     [%o0+%l0]
```

```
00022CB8
00022CB8         loc_22CB8:                  ! 10=Ciphertext length
00022CB8 070 cmp      %o4, %l0
00022CBC 070 bge,a    locret_22CE8
```

```
00022CE8
00022CE8         locret_22CE8:
00022CE8 070 ret
00022CEC 070 restore %g0, %o0, %o0
00022CEC         ! End of function decode_str_sll
00022CEC
```

```
00022CC0 070 clrb     [%o0+%l0]       ! o0=Newly requested memory space address
00022CC4 070 ldsb     [%i0+%o4], %o5  ! Ciphertext by byte
00022CC8 070 sll      %o5, 3, %g1
00022CCC 070 sub      %g1, %o5, %g1
00022CD0 070 sll      %g1, 2, %g1
00022CD4 070 add      %g1, %o5, %g1   ! Decryption operation
00022CD8 070 stb      %g1, [%o0+%o4]
00022CDC 070 ba       loc_22CB8       ! 10=Ciphertext length
00022CE0 070 inc      %o4
```

Figure 17 Another decryption algorithm

The content of decryption is shown in the table below:

Offset	Plaintext	Offset	Plaintext
0x13c12	'www.google.com'	0x1439b	'ntp'
0x13d11	'www.yahoo.com'	0x143ac	'mail'
0x13e32	'\x91 xxx atech.com'	0x143bd	'mysql'
0x14034	'\\X'	0x143cd	'named'
0x1406c	'\\'	0x143db	'sys'
0x140e7	'\x91puX;\xc7;\xc7Xupp\x8dTq\x01{User-Agent: Mozilla/5.0 (X11; U; Solaris; en-US; rv:1.7.5) Gecko/20041111 Firefox/1.0\r\n'	0x143ec	'smtp'
0x1419d	'Accept: image/png	0x143fe	'nobody'
0x14314	""	0x1440c	'auth'
0x1437e	'daemon'	0x1441a	'LP'
0x1438b	'adm'	0x1442c	'UUCP'

4.3.4 Network communication encryption

The custom algorithm for the Solaris samples is the same as the one on Windows. There is only one encrypted key (Solaris system does not have a registry. There is no registry encryption function). The key is the same as the one of registry encryption data on Windows platform. The custom encryption algorithms of the two platforms are the same (the specific algorithm can participate in 3.1.6 encryption algorithm analysis).

After analysis, the original 16-bit key of samples on Solaris is :

```
66 39 71 3c 0f 85 99 81 20 19 35 43 fe 9a 84 11
```

Address of the original 16-bit key in file is the same length as the original 16-bit key of Windows.

Due to that Solaris and Windows samples generate the same algorithm of network communication sub-key, it can generate a sub key:

```
E9 BE CD E0 A8 9F 4D DB C3 42 AC 2B 24 77 AB CB 5A C1 52 F8 5B 3E F0 78 CB 01 0A 69 29 8F 85 8C
03 9C 7C EF 5E 36 0E 8B C0 40 76 28 9C 9C F2 24 81 9D 02 72 4F 6A BB B5 5B 42 73 14 88 F2 73 75
8B F9 37 98 3B 9F 64 2B A3 C4 FF C7 8A 40 67 C1 25 9F 65 54 45 36 48 FF E2 86 05 1A F4 94 AC 2B
08 D5 E5 83 BE 2C AD EE D0 A6 98 CB 8D 35 ED EE C4 F0 8C F2 CD BA 87 03 54 27 3D 13 A7 9B 6A 05
C7 02 30 21 05 67 58 3B E6 A1 44 0A 37 16 3C 86 E9 BC 8B 20 1A 98 7E 28 E6 7F F7 CA F7 9E 38 31
7F F0 2F 93 11 2B 28 F0 FF 11 B7 FC 1C 63 86 CB
```

This sub key is used for encrypting and decrypting to send and receive data.

4.3.5 Network control instruction

In the analysis of Solaris samples, we found its function is less than Windows sample orders. There are only seven instructions on Solaris whose function is roughly the same as Windows. Here is the comparison of IDA on two platforms. It can be seen that the instructions of the samples on Solaris is much less and easier than that on Windows.

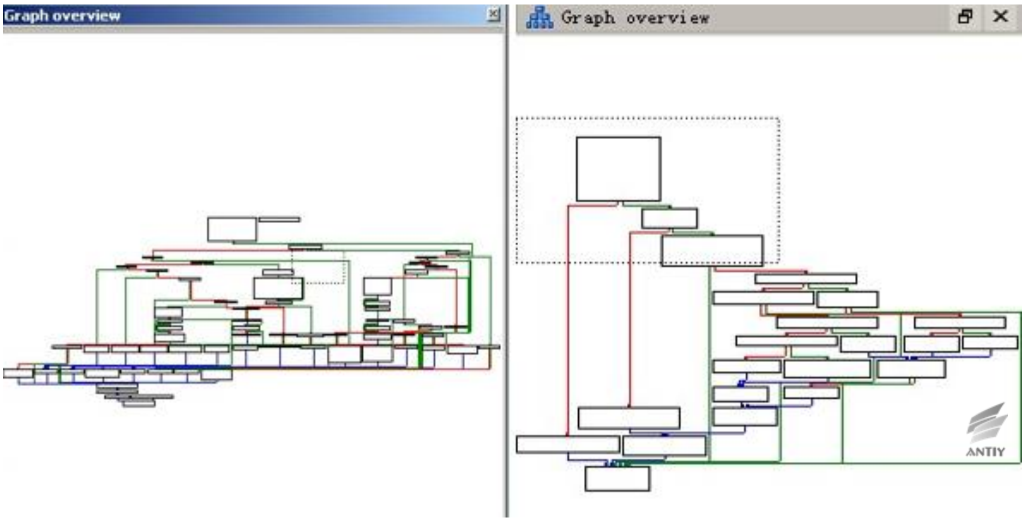


Figure 18 Comparison of network instruction on Windows and Solaris

After analysis, we found that the Solaris sample instruction function is not implemented above. At first, we thought that the instruction function of Solaris sample has not yet been completed, but after further analysis, we found that Solaris samples use a special kind of dynamic calculation to jump to a different branch instruction code, the red part below is the jump instruction after dynamic calculation.

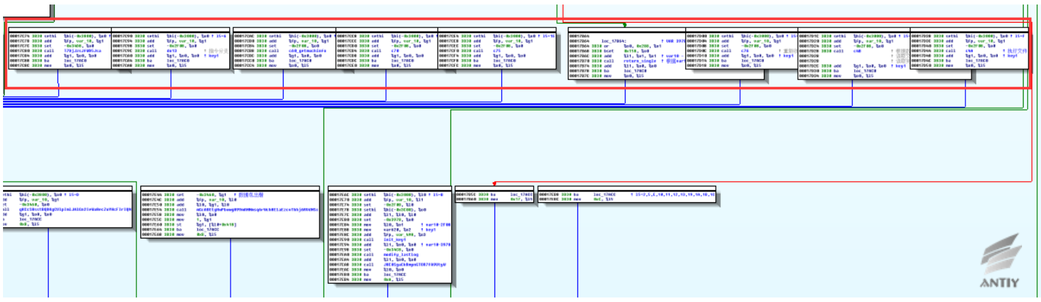


Figure 19 Solaris Branch instruction function

The functions of Solaris instructions are described as below, which is generally the same as Windows instructions:

Hexadecimal instruction code	Command function
0x42	Clear traces of infection, delete itself
0x4A	Create a file
0x44	Written in file
0x56	Execute file

0x4B	Return read file
0x60	Collect various information and return (specific format see chart below)
0x70	Update sample configuration information
0x75	Update sample SLEEP time and collect information and return
0x76	Update C&C server address

The download executable samples are the same as Windows, use the same instruction tag through three steps (create, write, execute) to complete the download and execution function, which is different only in code structure. Solaris integrates three instructions to a function.

When executing file, escalate privilege of file first, then use the excele function with parameter executable files,

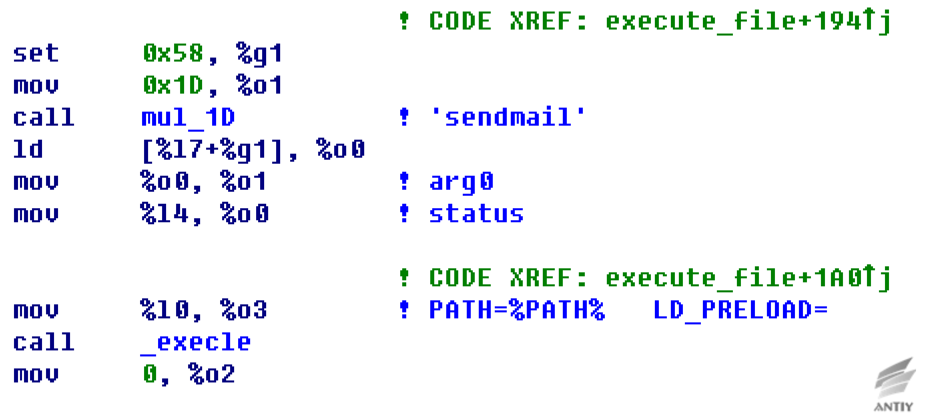
Parameter 1:file B path

Parameter 2:file B or "sendmail"(relates to mails suspected)

Parameter 3:0

Parameter 4:PATH=%PATH% (environment variable)

For instance: execle("/usr/bin/sample","sample", NULL, %envp%);



```
                                    ! CODE XREF: execute_file+194↑j
set     0x58, %g1
mov     0x1D, %o1
call    mul_1D                      ! 'sendmail'
ld      [%l7+%g1], %o0
mov     %o0, %o1                    ! arg0
mov     %l4, %o0                    ! status

                                    ! CODE XREF: execute_file+1A0↑j
mov     %l0, %o3                    ! PATH=%PATH%   LD_PRELOAD=
call    _execle
mov     0, %o2
```

Figure 20 Executable file parameter

The instruction function and packet format of Solaris samples is the same as Windows samples, the full explanation of instruction function and packet format are in section 3.5.6: analysis of instructions of Windows samples.

The collected system information of Solaris samples is slightly different from that of Windows, as follows:

Computer name	HostID	MAC address	IP address	User name	Typically user's

					full name
User UID/GID	System hardware structure information	System detailed time	Default Language types	Current running path	System process information

5 Summary

5.1 Improvement based on real threats

Our disclosures of great attack organizations' capabilities that cover all functional platforms proved to be a real threat not fictitious.

The complicated techniques of attack load, precise design depth and comprehensive environment covering platform have shown the technical capabilities of Equation attacks. The persistent attacks targeting at various certain goals also embody the attackers’ firmly attacking intention. In previous studies, Antiy defined the organizations that equipped with this kind of ability as A^2PT and summarized many characteristics of similar attacks from malware payload perspective. These standards are conforming to the behavior and ability of Equations.

A^2PT characteristic by Antiy	**Equations implementation and working characteristic**
Sufficient 0day reserve	Fanny exploits LNK 0day, MS09-025
Highly complex and modular load	Highly complex, modular EquationDrug and GragFish attack component
Local encryption block analysis, strictly encrypt communication and camouflage	Configuration data resources encryption, Registry, network communication encryption
Multiple implant ways	Network intrusion Logistics hijack (maybe) Personnel on-site implant(maybe)
Basically complete the carrier technology Without file and memory segment block analysis	Bootkit start Registry stores samples, Segmented decryption
Persist to expand depth (firmware), breadth (firewall, mail gateway, lateral movement in Lan)	Hard drive firmware changes Firewall and other network security devices implant Persistence targeted at mail server
Completely cover all operating system platforms (including mobile)	Windows, Linux, Solaris and OS X samples

As we have previously outlined, the related attack organizations own " **organic network attack teams and huge supporting engineering system and structured attack arsenal,**

powerful vulnerability collection and analysis and exploration capacity, and associated resources reserves, as well as systematic operation procedures and manuals, with features as equipment system covering the whole situation, exploitation tools and malicious code that covers the whole platform, persistent ability covering the whole link. In face of such systematic, industrial-strength and highly targeted attacks, perpetual motion must be stop and silver bullet misfiring. Only a clear strategy, full cost investment, defense against systematic [11] attacks, through long-term, solid hard work and ability construction can gradually achieve the initiative.

In some domestic reports about Equation, they read the persistent implant targeting at firmware in high-value targets as that all the current hard disk owns backdoor, which is a misunderstanding. However, when an organization's ability is big enough and only can be speculated and imaged, it can cause panic, which results in the query of "abuse of supply chain and information chain advantage".

5.2 Antiy's efforts

Starting from 2010, Antiy successively analyzed the advanced attacks or attack organizations as " Stuxnet", "Duqu", "Flame", "APT - TOCS (Lotus) ", "White Elephant", " Ukraine Power Outage", etc., and release hundreds of pages analysis reports accumulatively. There is no doubt that the ability of advanced threat detection products is relying on solid and effective analysis with continuously improvement. Antiy released product systems for advanced threat detection and situational awareness: **PTD** (Persistent Threat Detection System) can help users capture the network load and lateral movement; IEP (Intelligent Endpoint Protection System) provide multiple defense strategies including "Whitelist + Security baseline", **PTA** (Persistent Threat Analysis System) provides the ability to deeply analyze threat payload through dynamic and static methods. Antiy also plays an important role in **situational awareness** and **early warning platforms** of multiple industries and departments with overall design support, development and key detection ability.

Antiy focuses on the **next generation threat detection engine, highly customized in-depth analysis, interactive visual analysis and knowledge and intelligence support targeted at assets and threats.**

5.3 Future work

The great attack organizations' coverage ability has triggered a concern of security that "All cannot be trusted" for all global users. Last year, some domestic reports on Equation interpreted the actions that attackers inject and achieve persistence in hard drive firmware of high-value target, and concluded hard drive with backdoors is the current mainstream. It is of course a misunderstanding, but we must say that when the ability of a super attack organization is o strong that we can only imagine and speculate it. This situation must lead to the mass panic. Therefore, the question on superpower to "abuse of supply chain and information flow advantage" comes out.

The recent leakage of Equations code and exposure of ANT equipped system enable us to believe that relevant reserves of exploits and attack mentality have flowed into network crime, and even terrorist organizations. Due to the low reproduction cost of existing network attack technology, there exist more serious cyber arms proliferation risks. Therefore, if superpowers can reasonably control their arms development speed and scale of network and effectively prevent and control network arms proliferation that caused by lack of responsibility are the key factors to reach a more secure network.

We are looking forward to a more secure network world!

Appendix 1: References

[1] Equation Group Cyber Weapons Auction - Invitation

https://github.com/theshadowbrokers/EQGRP-AUCTION

[2] Shadow Brokers reveals list of Servers Hacked by the NSA

http://thehackernews.com/2016/10/nsa-shadow-brokers-hacking.html

[3] Antiy The Trojan modifying firmware Exploration in attack components of Equation Group

http://www.antiy.com/response/EQUATION_ANTIY_REPORT.html

[4] Antiy Analysis of encryption skills used in Equation Group attack components

http://www.antiy.com/response/Equation_part_of_the_component_analysis_of_cryptographic_techniques.html

[5] Kaspersky：Equation: The Death Star of Malware Galaxy

http://securelist.com/blog/research/68750/equation-the-death-star-of-malware-galaxy/

[6] Kaspersky：A Fanny Equation: "I am your father, Stuxnet"

http://securelist.com/blog/research/68787/a-fanny-equation-i-am-your-father-stuxnet/

[7] Kaspersky：Equation Group: from Houston with love

http://securelist.com/blog/research/68877/equation-group-from-houston-with-love/

[8] Kaspersky：Equation group questions and answers

https://securelist.com/files/2015/02/Equation_group_questions_and_answers.pdf

[9] The SPARC Architecture

https://en.wikipedia.org/wiki/SPARC

[10] The Solaris System

https://en.wikipedia.org/wiki/Solaris_(operating_system)

[11] Huang Sheng: Thoughts on Network Defense in Depth

http://www.antiy.com/wtc/2015/02_Joe.pdf

Appendix 2：About Antiy

Starting from antivirus engine research and development team, Antiy now has developed into a group level security enterprise with Antiy Labs as headquarters and both enterprise security company and mobile security company as two wings. Antiy always adheres to the belief of securing and protecting user value and advocates independent research and innovation, forming the layout of the capacity of the whole chain in the following aspects: security detection engine, mobile security, network protocol reduction analysis, dynamic analysis, terminal protection, and virtualization security and so on. Antiy has fostered nationwide detection and monitoring capability with our products and services covering multiple countries. With effective combination of techniques and products of both big data analysis and security visualization, Antiy expands the group work competence of engineers and shortens the product response cycle by massive automation sample analysis platform. With years' continual accumulation of massive security threat knowledge library, Antiy promotes the solution of situational awareness and monitoring and early warning that targets against APT and at scale network and critical infrastructure, combining with the experience of integrated application of big data analysis and security visualization.

More than 30 famous security vendors and IT vendors select Antiy as their partner of detection capability. The antivirus engine of Antiy has provided security protection for nearly a hundred thousand network devices and security devices and nearly two hundred million mobile phones. The mobile detection engine of Antiy was the first Chinses product that won AV-TEST reward in the world. The technical strength of Antiy has been recognized by industry management organizations, customers and partners. Antiy has consecutively been awarded the qualification of national security emergency support unit four times and one of the six of CNNVD first-level support units. Antiy is the significant enterprise node of China emergency response system, which has provided alarms, in-depth analysis or systematic solution in a few severe security incidents, such as Code Red, Dvldr, Stuxnet, Bash Shellcode, Sandworm, and Equation and so on.

More information about Antiy Labs:	*http://www.antiy.com（Chinese）*
	http://www.antiy.net （English）
More information about enterprise security company:	*http://www.antiy.cn*
More information about Antiy AVL TEAM:	*http://www.avlsec.com*