

# Modified Reverse Proxy Website Vulnerability Test Results

**Vincent Berk and  
Marion Bates**

Institute for Security Technology Studies  
*Dartmouth College*

September 10, 2001

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Preparation</b>	
2.1	RPS Summary . . . . .	1
2.2	Lab Test Setup . . . . .	1
<b>3</b>	<b>Using the Reverse Proxy</b>	<b>2</b>
3.1	Configuration . . . . .	2
3.1	Vulnerability Tests . . . . .	3
<b>4</b>	<b>Conclusions</b>	<b>4</b>
	Bibliography	

# 1 Introduction

The implementation and function of a reverse proxy server, as described in [Ber01], would in theory render a vulnerable webserver (such as Microsoft IIS 4.0 unpatched) immune to exploits such as Unicode directory traversals and remote code execution. In this paper, we will document how we tested this theory in a “real world” environment.

## 2 Preparation

See [Ber01] for a detailed explanation of this method of implementing a reverse proxy.

### 2.1 RPS Summary

In a nutshell, the reverse proxy server (RPS) improves webserver security by comparing incoming requests for webpages to a preordained list of allowable URLs, stored on the proxy. Since most exploits against web servers involve sending malformed URL strings to access non-public portions of the webserver’s content, the RPS effectively stops these sorts of attacks from succeeding.

### 2.2 Lab Test Setup

We used a PC running Windows NT 4.0 with Microsoft IIS 4.0 and no security patches. We created a simple website with four html pages and some images in the webroot directory.

We then tested the site by pointing a browser to the host:

```
http://lab4.ists.dartmouth.edu
```

It returned the default page, index.html.

We then entered a Unicode string into the browser, of the form:

```
http://lab4.ists.dartmouth.edu/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+d:\
```

This URL, adapted from an example in [Jia00], encodes a directory traversal command (“double-dot”) as a string of Unicode characters, which the webserver doesn’t interpret as a directory climb. The rest of the URL is a call to cmd.exe, the DOS shell, along with the command to list the contents of the D: drive.

This did indeed elicit a directory listing, thus proving conclusively that this site is severely vulnerable to attack. (The DOS shell can be handed any number of dangerous commands, allowing the attacker to download private data and upload malicious code.) This was the directory listing returned to the web browser:

```
Directory of d:\

11/04/00  10: 57p                591 AUTOEXEC. BAT
03/13/00  11: 08a                <DIR>      i 386
11/29/00  10: 58a                <DIR>      Inetpub
03/16/00  10: 46a                512 LINUX. BIN
03/13/00  10: 48a                <DIR>      Multimedia Files
11/15/00  04: 13p                <DIR>      NTPackSetup
09/06/01  06: 06a                134, 217, 728 pagefile. sys
11/29/00  10: 54a                <DIR>      Patches
11/29/00  10: 57a                <DIR>      Program Files
08/08/96  08: 00p                219 system. ini
12/20/00  12: 32p                <DIR>      TEMP
09/06/01  06: 06a                <DIR>      WINNT
          12 File(s)      134, 219, 050 bytes
          1, 544, 031, 232 bytes free
```

We were also able to telnet to port 80, issue a bogus request, and obtain information about the webserver software type and version:

```
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/4.0
Date: Thu, 06 Sep 2001 19:08:10 GMT
Content-Type: text/html
Content-Length: 87
```

This banner would give an attacker a great deal of information about where to start in order to run an exploit against the server.

Vulnerability scans of the server (using SARA 3.0.2, a derivative of the SATAN tool) revealed several known-vulnerable CGI scripts, admin password weaknesses, and null session capability -- all standard fare for an unpatched WindowsNT 4.0 server running IIS 4.

## 3 Using the Reverse Proxy

### 3.1 Configuration

We then configured the Squid proxy server, along with the RPS Jeanne plugin, to point to

lab4, our horrendously vulnerable webserver. We edited the /etc/hosts file on the client browsing machine such that requests to “lab4” or “lab4.ists.dartmouth.edu” were locally resolved to the proxy.

In a real production environment, the name of the proxy would be “www.thedomain.com” -- in other words, the proxy would appear to the rest of the world to be the company’s webserver. The actual machine hosting the website could be named anything. The only machine that communicates directly with it is the proxy. See [Ber01].

As per [Ber01], we compiled a list of accessible URLs for the proxy, which included the four HTML pages and their associated images. We also created a reject.html page on the webserver, to be returned whenever an attempt was made to access a URL not explicitly allowed in the URLs list. Even if reject.html did not exist on the webserver, the configuration would still work correctly -- a user would receive the generic “Error 404 Not Found” message from the server if the URL was not valid. However, the advantage of the RPS’s ability to return a defined reject.html page is that an administrator could set up reject.html to be a “page not found, please report errors to webmaster” CGI form page, or have reject.html simply dump the user back to the home page, or some other “nicer” (and more professional) option besides the dead-end 404 error.

### **3.2 Vulnerability Tests**

We then directed a client browser to lab4 and made normal requests for the webpages, which were returned successfully. To the client, the presence of the proxy is unknown. The URL in the browser window still says “lab4” even though the data transfer takes place between the proxy and the client. (This was verified by running a sniffer on the client machine and watching for any traffic directly to or from the webserver. There was none. The client only “talked” to the proxy.)

We also tried the same Unicode request, detailed above. Instead of a directory listing, we got back the reject.html page. Other requests for nonexistent pages, as well as pages which were present on the actual webserver but NOT present in the RPS’s list of allowed URLs, were also blocked and resulted in the reject page.

Telnet to port 80 reveals only the RPS’s banner information -- nothing about the actual webserver behind the proxy is returned. Clearly, if a vulnerability in Squid is discovered, and if an attacker is able to gather version information about the proxy in this manner, then the proxy may become the target of an attack. Thus it’s always advisable to alter banner information to deter some (but not all!) potential attackers.

One other security by-product of this setup is that if the proxy is compromised and goes offline, the website will be invisible to the rest of the world, but at least it will not become wide-open to attack (since the firewall will block access directly to the website from the Internet, if we use the rulebase detailed in [Ber01].) This setup is “fail-closed” -- a failure of the proxy will cause the entire site to become inaccessible to everyone, which could be bad business for the company in question, but it helps ensure that a compromise in one portion of the network doesn't lead to a massive security hole into the rest of the network. As usual, business need must be weighed against security.

Vulnerability scanners found no problems when run against the proxied website.

## **4 Conclusions**

The use of Squid with the Jeanne plugin allows an unsecured webserver to serve pages through a secure filter. Since Squid only allows access to explicitly-listed webpages, scripts, images, etc., malformed requests and requests for nonexistent files are stopped before they can reach the server.

# References

- [Ber01] Vincent Berk. *Improving Security, Accessibility and Maintainability of Websites using Modified Reverse Proxy servers: A Design and Implementation*. Institute for Security Technology Studies, Dartmouth College, and Leiden Institute for Advanced Computer Science, Leiden University. August 7, 2001.
- [Jia00] Guofei Jiang. *Microsoft IIS 4.0/5.0 Extended Unicode Directory Traversal Vulnerability*. Institute for Security Technology Studies, Dartmouth College. November 16, 2000.

Typeset in ClarisWorks 5.0 for Macintosh, using Times New Roman and Monaco.