

Improving Security, Accessibility and
Maintainability of Websites using Modified
Reverse Proxy servers
A Design and Implementation

Vincent Berk

Institute for Security Technology Studies
Dartmouth College
Leiden Institute for Advanced Computer Science
Leiden University

May 1, 2001

Contents

1	Introduction	1
2	Technical Background	1
2.1	Network Setup	1
2.2	Proxy and Reverse Proxy Servers	3
3	Reverse Proxy Design	4
3.1	Revised Network Setup	4
3.2	Security	5
3.3	An Example	6
4	Reverse Proxy Implementation	6
4.1	Squid Configuration	6
4.2	Redirector	8
5	Test Case: Bugtraq ID 1806 Microsoft ISS Unicode Double Dot Attack	8
5.1	Description of the Attack	8
5.2	Attacking the webserver	9
6	Conclusion	9
A	Source Code for Redirector Program	11
	Bibliography	

1 Introduction

Modern day businesses rely on the Internet as a one of their primary ways to distribute information. Because of its accessibility and its ease of use, the HTTP protocol has become the standard for this kind of communication. Customers and relations can easily obtain information from a well maintained website.

Because of its popularity a whole range of webservers has become available. Each of these is itself a huge and complex software package, usually with numerous versions and patches. Since these servers are accessible from all over the world, they pose a big security risk. If a maintainer is unable to keep up with the rapid flow of updates and patches the server can easily become vulnerable to attack. Also, due to the increased complexity, configuration of modern day webservers can be confusing, leading to vulnerabilities due to misconfiguration.

This paper proposes a new way of securing websites through *Reverse Proxy Servers*. The concept is explained by using an example design and actually implementing and testing this design. Using a different network setup, in which the webserver is actually placed behind the firewall, a more secure website can be created. The only machines reachable from the Internet will be the specially configured Reverse Proxy servers, which are easier to maintain and less vulnerable to attacks. Also, since multiple reverse proxy servers can be used to mirror just one actual webserver, access time and maintainability will greatly improve.

2 Technical Background

2.1 Network Setup

In the figure below (**Figure 1**) an overview is given of how networks are setup in general followed by a simplified version of the firewall rulebase (**Table 1**). The connection with the Internet comes in through one or more routers, usually called *Border Routers*. Immediately behind the border router follows a *Firewall*. Behind the firewall is the local network and on a third

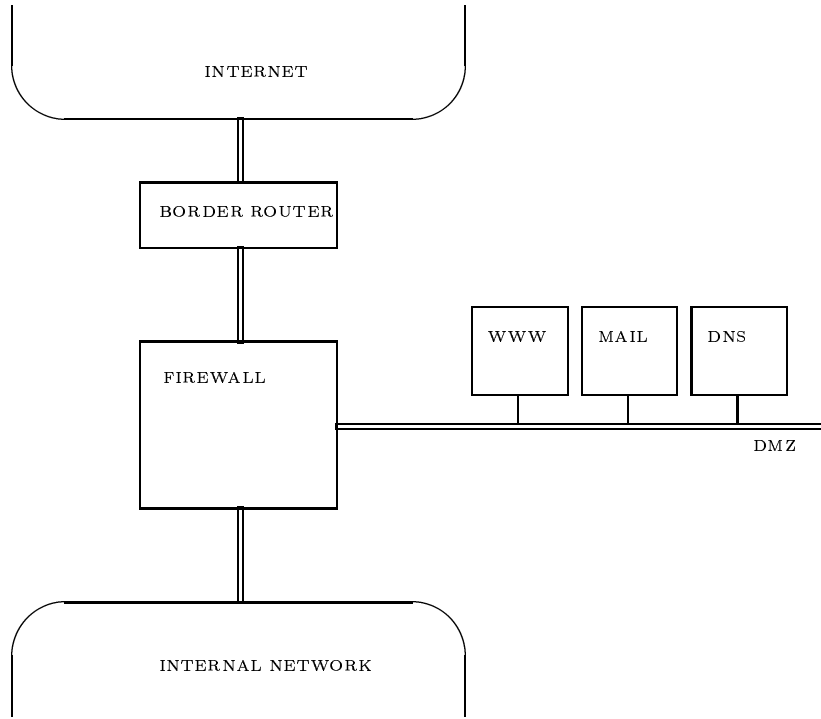


Figure 1: Typical Network Layout

segment of the firewall are the public servers. The public servers usually are web (HTTP), mail (SMTP,IMAP,POP) and domain name resolution (DNS). If an FTP server is used, it is also commonly located in the public servers section of the network. This public servers section of the network is generally called the *DeMilitairized Zone* or *DMZ*. (See [NSB00] and [Bre99])

This setup may differ from place to place. It is, for example, possible to move the public servers to the segment between the border router and the firewall, thus eliminating the need for a third firewall segment. Although this will limit the firewalls ability to control and log traffic for these servers.

In the given setup the firewall will be able to filter the incoming connections to the public servers. For the webserver this would be port 80 TCP and UDP. (see [NSB00] and [Bre99]). Although this will greatly improve the security of the website the server is still vulnerable to attacks directed through the HTTP protocol. Examples of these are the well known double-dot attack and buffer overflows (see [SFo] and [NSB00]).

SOURCE	DESTINATION	SERVICE	ACTION
Internet	WWW	HTTP	Accept
Internet	Mail	SMTP/IMAP/POP	Accept
Internet	DNS	DNS	Accept
Internet	Local Net	ANY	Deny
DMZ	Local Net	ANY	Deny
Local Net	Internet	ANY	Accept
Local Net	DMZ	ANY	Accept
ANY	ANY	ANY	Deny

Table 1: Typical Firewall Rulebase

2.2 Proxy and Reverse Proxy Servers

To increase both the security and the response time of web request *Proxy Servers* were introduced. A proxy server is generally located on the inside of a network and will process the web requests for the local clients. Instead of connecting and making their request to the remote web servers directly, local clients connect and make the request to the local proxy server. The local proxy server then looks in its cache to see if the page has recently been retrieved. If so, it will return the cached page, if not, the page will be requested from the remote web server and returned to the local client by the proxy server.

A *Reverse Proxy Server* will do the exact opposite. It will be located in the DMZ section of a network and accept requests for the local web server from the Internet. In contrary to normal proxy servers, which accept requests from a local client group and fetch webpages from the entire Internet, a reverse proxy server will accept webrequests from the Internet for pages of the local web server. This will speed up the operation of the local web server, since the reverse proxy will cache the most frequently requested pages as well as adding an extra layer of security to the website.

The reverse proxy server will act like a webserver.

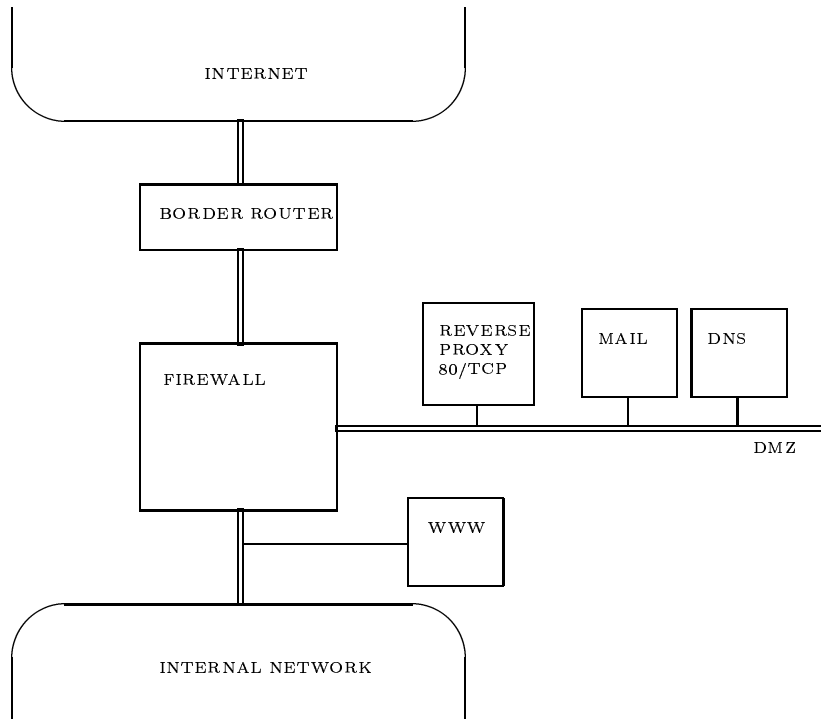


Figure 2: Revised Network Layout

3 Reverse Proxy Design

3.1 Revised Network Setup

In the setup below (**Figure 2**) the webserver is placed behind the firewall. Also a revised firewall rulebase is given (**Table 2**). A reverse proxy server is placed in the DMZ. Normal proxy servers generally listen on port 8080/TCP but because the reverse proxy server will be acting like a webserver it should be listening for requests on the regular HTTP port; 80/TCP. This way the outside world will see the reverse proxy server as the webserver of the site. The firewall will block all traffic from the outside world to the real webserver on the inside and will only allow HTTP connections to the reverse proxy server in the DMZ. Finally, the firewall will allow HTTP connections from the reverse proxy server to the actual webserver in order for the actual data to be retrieved.

Also, it is possible to install multiple reverse proxy servers in the DMZ, each sourcing from a single webserver behind the firewall. Since all the reverse proxy servers will cache frequently retrieved pages, the load on the

SOURCE	DESTINATION	SERVICE	ACTION
Internet	Reverse Proxy	HTTP	Accept
Reverse Proxy	WWW	HTTP	Accept
Internet	Mail	SMTP/IMAP/POP	Accept
Internet	DNS	DNS	Accept
Internet	Local Net	ANY	Deny
DMZ	Local Net	ANY	Deny
Local Net	Internet	ANY	Accept
Local Net	DMZ	ANY	Accept
ANY	ANY	ANY	Deny

Table 2: Revised Firewall Rulebase

actual webserver will be relatively low. This way a busy Internet site will be able to offer multiple access points with reverse proxy servers while only having to maintain one actual webserver.

3.2 Security

In order to improve security, each incoming request will be processed and rewritten by the reverse proxy server. This means that if the request arrives in multiple packet fragments it will be reassembled and evaluated before any action is taken. The requested file and/or method is abstracted from the request and matched against an *Access Control List* or *ACL*. The ACL determines if the client making the request is allowed to do so, based on its IP address. Next it will see if the protocol (HTTP) and the method (GET/PUT/etc.) are allowed for that client. If a file is requested it will be matched against a list of allowed files. This way no files can be retrieved from the webserver which are not verified and explicitly stated in the listing of the reverse proxy server. If a match is found a new request will be constructed with the entry from the allowed files list. If the request cannot be handled by the reverse proxy server (ie. the files are not cached locally) the newly constructed request will be sent on to the actual webserver behind the firewall.

Since most of the operating software of the reverse proxy server can be read-only (except for certain OS-specific files and the cache directories), it can be written onto a CDROM. The system can run from the CDROM while the cache is stored partly in memory and partly on local disks. In case an

attack on the reverse proxy succeeds no mayor damage can be done.

To increase redundancy, multiple operating systems can be used to run multiple reverse proxy servers on (eg. Linux/Solaris/Irix). In this way not all servers will be vulnerable to the same operating system specific attack. Thus not allowing all reverse proxy servers to be taken down by the same attack.

3.3 An Example

A HTTP request comes in and is allowed through to the reverse proxy server by the firewall. The reverse proxy server will process the request and verify the IP address of the requesting client. Next the request itself is verified on both protocol (is this really an HTTP request) and on HTTP method (GET/PUT/etc.). If the checks are succesful and a file (eg. webpage) was requested, it will be checked against a list of allowed files. If there is a match, a new HTTP request will be constructed with the matching entry from the file list. This request will now be matched against the cache of the reverse proxy server. If the file is present it will be returned to the requesting client. If not, the request is send on to the actual webserver, which in its turn will return the file to the reverse proxy server where it will be both stored in the cache and returned to the requesting client.

4 Reverse Proxy Implementation

4.1 Squid Configuration

For the implementation products from the open source community are used. Squid (see [squ]) is a proxy server package that can be configured as a reverse proxy server. It is designed to run on Unix systems under the *GNU General Public License* or *GPL*. A Unix system should be configured to be a *Bastion Host* (see [SH95] pages 283-294) before putting it in the DMZ where it will be open to attacks.

This paper is not meant to give description of how to install the Squid package. Instead see the documentation that comes with the package. Below is a description of the most important options in the *squid.conf* configuration file to make Squid function as a reverse proxy server.

```
# squid.conf - basic httpd reverse proxy server configuration
#

# Make reverse proxy listen on port 80 like normal webserver.
# Hostname will be www.
http_port 80
visible_hostname www.domain.com

# The IP address and port number of the real webserver located behind
# the firewall.
httpd_accel_host 129.170.248.226
httpd_accel_port 80

# Turn off the original proxy server.
httpd_accel_with_proxy off

# Configure the cache to be valid for 30 seconds.
refresh_pattern . 0 0% 30

# The Access Control Lists
# Everyone is allowed to use the GET method on the HTTP port 80.
# All other methods and ports are denied.
# For a more accurate description see the Squid documentation.
# Also deny SNMP control access!
acl all src 0.0.0.0/0.0.0.0
acl localhost src 127.0.0.1/255.255.255.255
acl localnet src 10.20.30.0/255.255.255.0
acl safeports port 80
acl safemethods method GET
http_access allow all
http_access deny !safeports
http_access deny !safemethods
snmp_access deny all

# We'll get to this later on.
redirect_program /revproxy/redirector

# End of squid.conf reverse proxy server configuration
```

This is not an exhaustive configuration for Squid. Both the ACLs and

the cache options should be configured for specific websites.

4.2 Redirector

The redirector is a program that will be run by Squid in the background. When a request comes in, Squid will forward it to the redirector and expect a valid request back. The redirector should be a continuous program that never exits. This can be conveniently used to check for the requested files. If a request comes in for a file that the webserver is not allowed to export (ie. it is not in the verified file list of the reverse proxy server) the redirector will return an HTTP request to Squid for an *Access Denied* page. This way a the requesting client will never get to the webserver.

The source code for an example of such a redirector can be found in Appendix A. In the *squid.conf* configuration file:

```
redirect_program /revproxy/redirector
```

5 Test Case: Bugtraq ID 1806 Microsoft ISS Unicode Double Dot Attack

5.1 Description of the Attack

Bugtraq ID 1806 (<http://www.securityfocus.com/bid/1806/>, see [SFo]) describes this attack as: “Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability”. This is a common *Double Dot* attack. Web-servers store webpages in a directory sturcture resembling the website. A double dot attack is aimed at traversing out of that directory tree so that system files can be accessed on the webserver. The general form of such an attack is an HTTP reques as shown below:

```
GET ../../../../etc/passwd HTTP/1.0\n\n
```

This specific attack targets Microsoft IIS (Internet Information Server) 5.0 for Windows 2000, Microsoft ISS 4.0 for Windows NT 4.0, Microsoft PWS (Personal Web Server) 4.0 for Windows 98 and NT. The directories are traversed by using *Unicode* characters:

```
http://target/scripts/..%c1%c1../path/file.ext
```

Where ‘target’ is the host being under attack and ‘path/file.ext’ the path and the file that will be accessed. Yet a more serious vulnerability results from this. It is possible to execute any arbitrary command on the webserver through:

```
http://target/scripts/..%c1%c1../winnt/system32/cmd.exe?/c+dir
```

A more elaborate discussion on this vulnerability can be found on the Securityfocus website. ([SFo])

5.2 Attacking the webserver

An unknowing attacker will see the reverse proxy server (probably with hostname ‘www’) in the DMZ as the actual webserver. The attack will thus be launched at the reverse proxy server. First the IP address, port and the method (GET) will be verified by Squid. Obviously this will pass. Now the attack, which is essentially a normal HTTP request, will be forwarded for processing by the redirector. The redirector does not recognize this URL as a valid file for the webserver and return the ‘reject’-page to the attacker.

Even if the attacker is aware of the IP address of the real webserver the attack will still be unsuccessful since the firewall will block all incoming traffic from outside to machines other than those in the DMZ.

6 Conclusion

In this whitepaper a design was proposed to protect any webserver from direct attack by using a modified reverse proxy server. By placing the real webserver behind the firewall and placing a reverse proxy server in the DMZ, all webrequests are first handled by that reverse proxy server. No direct connections to the webserver are allowed. The reverse proxy is modified to perform a range of security checks on the HTTP request. This extra layer of security will, for example, protect the webserver from most common *Input Validation Error* attacks like double-dot and webserver buffer overflows.

Also the response time of the website will improve since the reverse proxy server will cache the most frequently made requests, thus speeding them up. A website can have multiple access points using multiple reverse proxy servers while maintaining only one actual webserver behind the firewall. These features both speed up operation and increase maintainability of the entire website, as well as greatly improving its security.

A Source Code for Redirector Program

```
/*
 * Squid Reverse Proxy Redirector to Verify URLs
 *
 * Vincent Berk    vberk@ists.dartmouth.edu
 * ISTS/Dartmouth College
 * LIACS/Leiden University
 *
 * Copyright (C) 2001 Vincent Berk    GNU/GPL
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NOERR 0
#define SOMEWEIRDERR 1

#define BUFSIZE 1500
#define VALIDS 100
#define LOGFILE "/revproxy/logs/pukepile.log"
#define URLFILE "/revproxy/redirect/verified.url"

#define REJECT "http://www.domain.com/reject.html localhost/- - GET"

int main ( int argc, char** arv )
{
    FILE* I;
    FILE* O;
    FILE* L;
    FILE* V;
    int error, i, maxurls;
    char* buffer;
    char* url;
    char** valid;

    I = stdin;
```

```

O = stdout;
L = fopen ( LOGFILE, "a" );
V = fopen ( URLFILE, "r" );
error = NOERR;

buffer = malloc ( sizeof (char) * BUFSIZE );
valid = malloc ( sizeof (char*) * VALIDS );
url = malloc ( sizeof (char) * BUFSIZE );

if ( !( V && L ) )
{
    fprintf ( stderr, "Mandatory file %s or %s could not
                    be opened\n", LOGFILE, URLFILE ),
    error = SOMEWEIRDERR;
}

/* Get the verified URL list ... */
i = 0;
fprintf ( L, "Allowed URLs:\n", i, valid [i] );
while ( !error && fgets_unlocked ( buffer, BUFSIZE, V ) && i < VALIDS )
{
    /* Make sure that we don't get buffer overflows. */
    buffer [BUFSIZE - 1] = '\0';

    if ( buffer [0] != '#' && buffer [0] != '\n' &&
        buffer [0] != '\0' && buffer [0] != ' ' )
    {
        valid [i] = malloc ( sizeof (char) * BUFSIZE );
        /* Killing the endline with some magic... ;- ) */
        *( strchr ( buffer, '\n' ) ) = '\0';
        strncpy ( valid [i], buffer, BUFSIZE );
        fprintf ( L, "%i: %s\n", i, valid [i] );
        i ++;
    }
}
fflush ( L );
maxurls = i;

/* Forever and ever ... */
while ( !error && fgets ( buffer, BUFSIZE, I ) )

```

```

{
    /* Make sure that we don't get buffer overflows. */
    buffer [BUFSIZE - 1] = '\0';

    fprintf ( L, "%s", buffer );

    /* Find first space in string. That'll give the url. Magic? ;-) */
    *( strchr ( buffer, ' ' ) ) = '\0';
    strcpy ( url, buffer );

    fprintf ( L, "%s\n", url );
    fflush ( L );

    i = 0;
    while ( i < maxurls && strcmp ( valid [i], url ) )
        i ++;

    if ( i < maxurls )
    {
        fprintf ( L, "Match Found: %i\n", i );
        fprintf ( 0, "%s\n", valid [i] );
    }
    else
    {
        fprintf ( L, "No match.\n" );
        fprintf ( 0, "%s\n", REJECT );
    }
    fflush ( L );
    fflush ( 0 );

}

fprintf ( L, "Redirector exited with code: %i\n", error );
fflush ( L );
return error;
}

```

References

- [Bre99] Chris Brenton. *Mastering Network Security*. Sybex Network Press, 1999.
- [Nor99] Stephen Northcutt. *Network Intrusion Detection*. New Riders, 1999.
- [NSB00] Stephen Northcutt, Lance Spitzner, and Chris Brenton. Sans track 2: Firewalls, perimeter protection and virtual private networks. Course Material, october 2000. SANS Network Security 2000, Monterey California.
- [SFo] www.securityfocus.com. Internet publication. Bugtraq vulnerability archive.
- [SH95] Karanjit Siyan and Chris Hare. *Internet Firewalls and Network Security*. New Riders, 1995.
- [squ] www.squid-cache.org. Internet webpage. Squid Web Proxy Cache.
- [Tan96] Andrew Tanenbaum. *Computer Networks*. Prentice Hall PTR, 1996.

This paper was typeset using L^AT_EX 2_ε and B^IB^TE_X and a lot of tasty Ben&Jerry's!!!