Proposal:
Phase II Testing of Jeanne, Modified Reverse Proxy

Vincent Berk and
Marion Bates

Institute for Security Technology Studies
Dartmouth College

September 20, 2001
## Contents

1 **Introduction**
   1.1 Overview .............................................. 1
   1.2 Why We Are Doing This ............................... 1

2 **Implementation Details** .............................. 1
   2.1 Hardware Requirements ............................... 1
   2.2 Configuration Changes ............................... 2
   2.3 Backup Plan ........................................... 2

3 **What We Will Do Next** ............................... 3

Bibliography

Appendix A: jeanne.c source code

Appendix B: makeurls & getlist scripts
1 Introduction

Given the success of the lab tests using Jeanne, the reverse proxy server, and a known-vulnerable webserver, the Jeanne project team would like to perform further testing of the RPS. We would like to place the ISTS website behind an instantiation of the reverse proxy and evaluate its performance in that environment.

1.1 Overview

The reverse proxy server (RPS) will be placed in front of the ISTS webserver, and the RPS will be renamed “www.” DNS and firewall alterations will be made such that requests for “www.ists.dartmouth.edu” are directed to the RPS, and the RPS will process the request by comparing the requested URL to its list of allowed URLs. If the request is valid, the proxy asks for the page from the webserver (or returns the page from its own cache, if the page has already been requested within a time frame). If the request is not valid, the server redirects the user to a “reject” page, which could display anything -- an error message, a copy of the home page, etc. Users’ web requests will never go directly to the webserver; they will always pass through the reverse proxy’s filter first. See [Ber01] and [BB01] for details.

1.2 Why We Are Doing This

This environment would provide a real-world set of circumstances for testing the reverse proxy, and would also add another layer of security to our website. See appendices to this document.

2 Implementation Details

2.1 Hardware Requirements

The proxy server hardware will need to consist of an Intel Pentium® III or IV PC capable of running some flavor of UNIX/Linux (most likely RedHat 7.1, which is free). It will require a 100Mbit-compatible Ethernet network card, either included with the computer or installed later by our project team. It may also need a hard disk RAID (Redundant Array of Inexpensive Disks), consisting of a controller card and several additional hard disks (as well as a fast SCSI bus). This additional disk storage is needed for Squid’s local cache, which minimizes delay during HTTP requests (recently-requested pages are cached on the proxy, so it does not need to communicate with the webserver for every single request). The estimated cost of this setup is roughly $3000.
2.2 Configuration Changes

On the webserver itself, a script called “makeurls” will be added to root’s daily crontab. The makeurls script will search for webpages in the web root directory and all accessible subdirectories thereof, using the “find” function, and will update as needed a textfile called “urls” by adding URLs for new webpages and removing URLs for missing webpages. This url list file is stored at the web root, readable as a regular webpage, and it will be used to generate the allowed-URLs list for the reverse proxy. Note that this URLs file does NOT contain the URL to itself; in other words, no one (except the proxy itself, see below) will be allowed to browse to this list through the www.

On the proxy server, another script called “getlist” will be run daily. This script consists of a GET command which retrieves the URLs file from the webserver and writes it to a textfile. After retrieving the urls list, Squid is restarted and so it can process the updated URL list. Again, Squid can initiate this connection to the webserver for its own purposes, but normal users cannot request the list themselves because the request would have to be allowed by the reverse proxy, and that URL is deliberately excluded from the url list. (Even if the outside world could view the list, it wouldn’t matter much. But the Principle of Least Privilege dictates that users should only be able to see what they need to see.)

The proxy server will point to the webserver, and the firewall rules will be modified to deny HTTP access directly to the webserver from any source except the proxy. DNS records will need to be altered to direct www.ists.dartmouth.edu requests to the proxy, and the webserver itself will be renamed something else, such as “wwwfiles.”

Maintenance duties would be split among the current webmaster and the authors listed above. In short, the webmaster will need to report to us any new webpages (and therefore new URLs) which have been added, so we can make sure that the list of allowed URLs is up to date. We will implement detailed logging in conjunction with an automatic alerting method to tell us right away if a vital service fails on either the proxy or the webserver (this alerting method is already in use on our network, via Swatch, and would simply be expanded to include monitoring the proxy as well).

2.3 Backup Plan

The firewall rule changes can be made in such a way that it is trivial to revert them to the current status (i.e., direct connections to the webserver, no proxy involved). Reverting DNS is equally trivial. (In both cases, we would “comment out” the relevant config lines; reverting everything back to normal operation would involve editing one character at the beginning of a few lines and restarting the service.)
So, in the event that a problem occurs with the proxy, we can fall back to straight webserver access in a matter of minutes. The only potential problem with this revert scheme is that DNS propagation times to other DNS servers could result in slow response time for first-time visitors to the website, but we can configure our DNS servers to actively advertise the webserver address if and when we make a change. Thus propagation times will be minimized. At worst, we could post a simple HTML redirector on the proxy server, bypassing Squid and the reverse proxy plugin, just to keep our website fully viable at all times. This is a bit extreme.

3 What We Will Do Next

The next step for this testing period will be to accumulate more beta-testers outside of ISTS. These would be system administrators running production websites who are willing to install the RPS in front of their websites, and who would either give us remote access for configuring and evaluating its performance ourselves or agree to report to us about how the server performs and what if any problems they encounter.
References
