The Definitive Guide to Life behind Squid

Artagnon

Compiled with LaTeX on February 7, 2013

0.1 Audience

Moo

0.2 Author's note

It's frustrating to constantly live behind a HTTP proxy, here at IIT Kharagpur. No IRC, no email, no torrents, no GIT, and probably the worst of all- no SSH. This book initially started out as a small document describing the steps I'd taken to circumbent the system legitimately. However, because I don't like cookbooks, I started explaining the logic behind every step I was taking. The draft eventually evolved into a book for dealing with various types of proxies.

0.3 Introduction

to enable people to do more than just web browsing and DC++ in a restrictive environment. However, I don't write cookbooks- I'll explain every decision I've taken along the way to break every program out legitimately. Note, however, that I'm no networking expert. I might therefore have made glaring mistakes- The book is written purely from an engineeer-hacker's perspective and should not be used for academic purposes.

The Squid setup at IIT Kharagpur is especially restrictive and frustrating, which is the reason tools like Corkscrew can't be used to break out[1]. I've written this book for the worst Squid setup, and should hence be adaptable to any setup.

0.4 Networking 101

0.4.1 The server

In principle, a server is similar to a shopkeeper attending to a number of customers (clients) simultaneously. The webserver is one of the most widely cited examples of a server. Google runs a webserver, a computer application that sits and "listens" for connections, to server its clients. Client applications like web browsers make requests to Google's webserver situated at google.com. The client-server model principally pivots on "sockets". A socket is an endpoint of bidirectional communication flow, and is characterized by "protocol" (or language of communication) and "address" (

However, the client-server model is used in several non-networking related applications as well. For example, the graphical user interface in modern GNU/Linux systems is often provided by the X Window System. The X server listens for connections, while serveral user applications constantly give it commands to draw/ move/ resize various windows on the screen.

[1] Yes, I will explain what the setup is and why it's particularly restrictive

Two additional pieces of information is required for client-server communication: The medium of communication and the “protocol” (language) they’ll be using to communicate with each other. Broadly, there are two mediums of communication, TCP/UDP “ports” and “sockets”.

In modern computers, several servers are constantly running on different computer “ports” to facilitate simple actions like remote login.

0.4.2 Secure Shell

On *nix systems, the most popular way to login to a remote system is called Secure Shell (SSH). As usual, an SSH server runs on one of the machines while an SSH client polls it for connections from the client system.

0.5 The “proxy” primer

A proxy server is

0.6 Tests

0.6.1 How to use Telnet

0.6.2 Executions

In fact, I repeated this several times trying different ports, with the same result. Except port 443. Otherwise, I wouldn’t be able to visit sites like gmail.com from my web browser.