Network Working Group
Request for Comments: 1919
Category: Informational

M. Chatel
Consultant

March 1996

Classical versus Transparent IP Proxies

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

Many modern IP security systems (also called "firewalls" in the trade) make use of proxy technology to achieve access control. This document explains "classical" and "transparent" proxy techniques and attempts to provide rules to help determine when each proxy system may be used without causing problems.

Table of Contents

1. Background

An increasing number of organizations use IP security systems to provide specific access control when crossing network security perimeters. These systems are often deployed at the network boundary between two organizations (which may be part of the same "official" entity), or between an organization's network and a large public internetwork such as the Internet.

Some people believe that IP firewalls will become commodity products. Others believe that the introduction of IPv6 and of its improved security capabilities will gradually make firewalls look like stopgap solutions, and therefore irrelevant to the computer networking scene. In any case, it is currently important to examine the impact of inserting (and removing) a firewall at a network boundary, and to verify whether specific types of firewall technologies may have different effects on typical small and large IP networks.

Current firewall designs usually rely on packet filtering, proxy technology, or a combination of both. Packet filtering (although hard to configure correctly in a security sense) is now a well documented technology whose strengths and weaknesses are reasonably understood. Proxy technology, on the other hand, has been deployed a lot but studied little. Furthermore, many recent firewall products support a capability called "transparent proxying". This type of feature has been subject to much more marketing attention than actual technical analysis by the networking community.

It must be remembered that the Internet's growth and success is strongly related to its "open" nature. An Internet which would have been segmented from the start with firewalls, packet filters, and proxies may not have become what it is today. This type of discussion is, however, outside the scope of this document, which just attempts to provide an understandable description of what are network proxies, and of what are the differences, strengths, and weaknesses of "classical" and "transparent" network proxies. Within the context of this document, a "classical" proxy is the older (some would say old-fashioned) type of proxy of the two.

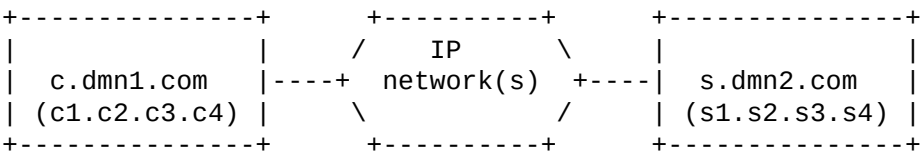
Also note that in this document, the word "connection" is used for an application session that uses TCP, while the word "session" refers to an application dialog that may use UDP or TCP.

2. Direct communication (without a proxy)

In the "normal" Internet world, systems do not use proxies and simply use normal TCP/IP to communicate with each other. It is important (for readers who may not be familiar with this) to take a quick look at the operations involved, in order to better understand what is the exact use of a proxy.

2.1 Direct connection example

Let's take a familiar network session and describe some details of its operation. We will look at what happens when a user on a client system "c.dmn1.com" sets up an FTP connection to the server system "s.dmn2.com". The client system's IP address is c1.c2.c3.c4, the server's IP address is s1.s2.s3.s4.



The user starts an instance of an FTP client program on the client system "c.dmn1.com", and specifies that the target system is "s.dmn2.com". On command-line systems, the user typically types:

```
ftp s.dmn2.com
```

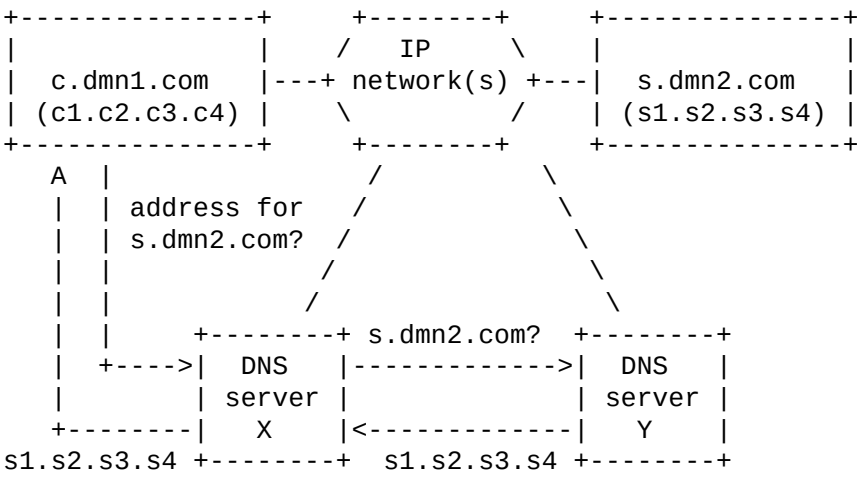
The client system needs to convert the server's name to an IP address (if the user directly specified the server by address, this step is not needed).

Converting the server name to an IP address requires work to be performed which ranges between two extremes:

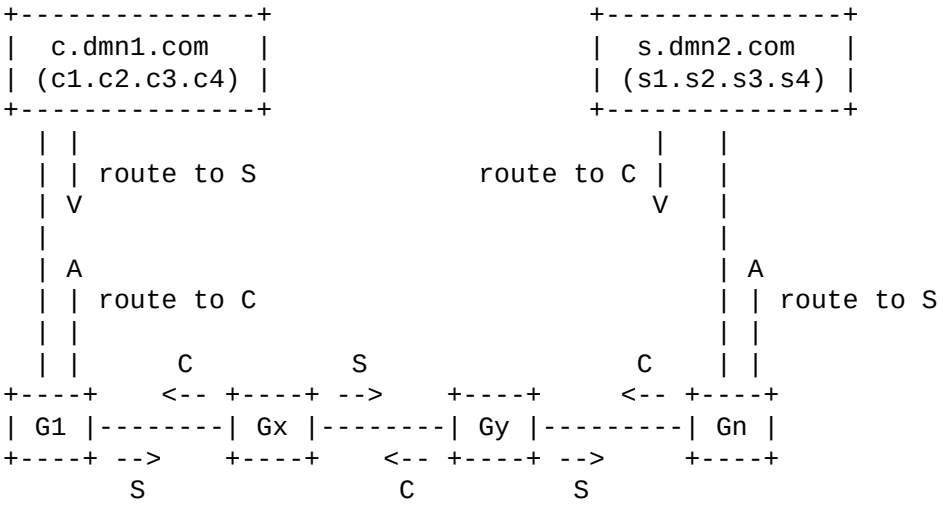
a) the client system has this name in its hosts file, or has local DNS caching capability and successfully retrieves the name of the server system in its cache. No network activity is performed to convert the name to an IP address.

b) the client system, in combination with DNS name servers, generate DNS queries that eventually propagate close to the root of the DNS tree and back down the server's DNS branch. Eventually, a DNS server which is authoritative for the server system's domain is queried and returns the IP address associated with "s.dmn2.com" (depending on the case, it may return this to the client system directly or to an

intermediate name server). Ultimately, the client system obtains a valid IP address for s.dmn2.com. For simplicity, we assume the server has only one IP address.



Once the client system knows the IP address of the server system, it attempts to establish a connection to the standard FTP "control" TCP port on the server (port 21). For this to work, the client system must have a valid route to the server's IP address, and the server system must have a valid route to the client's IP address. All intermediate devices that behave like IP gateways must have valid routes for both the client and the server. If these devices perform packet filtering, they must ALL allow the specific type of traffic required between C and S for this specific application.



The actual application work for the FTP session between the client and server is done with a bidirectional flow of TCP packets between the client's and server's IP addresses.

The FTP protocol uses a slightly complex protocol and TCP connection model which is, luckily, not important to the present discussion. This allows slightly shortening this document...

2.2 Requirements of direct communication

Based on the preceding discussion, it is possible to say that the following is required for a direct session between a client and server to be successful:

a) If the client uses the NAME of the server to reference it, the client must either have a hardcoded name-to-address binding for the server, or it must be able to resolve the server name (typically using DNS). In the case of DNS, this implies that the client and server must be part of the same DNS architecture or tree.

b) The client and server must be part of the same internetwork: the client must have a valid IP route towards the server, the server must have a valid IP route towards the client, and all intermediate IP gateways must have valid routes towards the client and server ("IP gateway" is the RFC standard terminology; people often use the term "IP router" in computer rooms).

c) If there are devices on the path between the client and server that perform packet filtering, all these devices must permit the forwarding of packets between the IP address of the client and the IP address of the server, at least for packets that fit the protocol model of the FTP application (TCP ports used, etc.).

3. Classical application proxies

A classical application proxy is a special program that knows one (or more) specific application protocols. Most application protocols are not symetric; one end is considered to be a "client", one end is a "server".

A classical application proxy implements both the "client" and "server" parts of an application protocol. In practice, it only needs to implement enough of the client and server protocols to accomplish the following:

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.