Rating of application layer proxies
Document number: AT-0008
Revision 2

Michael C. Richardson
mcr@milkyway.com

November 13, 1996

Abstract

This document defines a rating system for application layer proxies. The purpose of these definitions is inject some sanity into the proxy debate by defining 5 levels of proxies. Vendors are called upon to rate their proxies based on these definitions so that customers can understand what is meant by the word *supported*.

1 Introduction

In the firewall vendor wars a great deal of energy seems to be spent in the marketing department(s) trying to differentiate products from each other. One typical way of rating a firewall is the number and/or type of proxies supported. This is often presented as a series of checkboxes or points similar to this:

- telnet
- ftp
- http
- shttp
- nntp
- etc..

So vendor with the longest list wins! Often vendors say they support a long list of protocols via a *generic* proxy. These generic proxies are more properly called *circuit layer* proxies, which are the most basic type of proxy. Security is generally better when the proxy knows more about the protocol. In the case of support for UDP-based RealAudio, RPC services, and FTP services, this protocol knowledge is required for the protocol be used securely.

What isn't clear from the bulleted list is how the firewall vendor intends to support the protocol. Do they have a proxy that is knowledgable about the protocol, or is it just a circuit layer proxy? The customer may be not be making a fair comparison.

2 The rating system

The rating system defines five levels of proxies: *circuit-layer, traffic-aware, command-aware, content type-aware* and *policy-aware* proxies. These definitions are pretty loose. This is intentional. Vendors are expected to provide some argument as to why they their support for a particular protocol is at a given level. A person reviewing firewalls should read the vendor's arguments, not their marketing glossies and make their own decisions. As such, this should perhaps be called a glossary.

2.1 Circuit layer proxy

This kind of proxy does no special processing of *any* octets in the data stream. Circuit layer proxies are only aware of information from the TCP/UDP headers: source/destination addresses and ports. Circuit layer proxies may also keep track of how many bytes are sent/received. In-protocol challenges are not possible with this proxy, since this proxy does not speak the application protocol. No TCP urgent pointer need be done.

2.2 Traffic-aware proxies

This kind of proxy does some special processing on some bytes of the data stream. Traffic-aware proxies may propagate TCP urgent pointers properly, and may flush buffers at appropriate points in the data stream. In-protocol challenges are possible, but no logging or filtering of the data is done once the end-to-end session is established.

2.3 Command-aware proxies

This kind of proxy understands all the commands that can pass through the proxy. Command-aware proxies allow legitimate commands and reject commands that are not legitimate. Commands are logged and the proxy may filter the commands. The filters may be controlled by the administrator. In-protocol challenges are possible. Note: the results and/or data transfers associated with the commands are not logged or filtered.

2.4 Content type-aware proxies

This kind of proxy understands not only the commands, but also the general format of the content. Content type-aware proxies can verify that the correct content type is present. It may filter out certain content types or scan the packaging of the content for known security problems. For example: scaning email for header exploits, FTP downloads for self-extracting archives, http transfers of type `application/java`.

2.5 Policy-aware proxies

Note: This kind of proxy is largely an ideal. Implementations will fall short of this goal. AI research may have to catch up a bit first for all these ideas to work.

This kind of proxy is aware of the content type as before, but also possess abilities to decode and reject the content of a transmission for reasons of local policy. This would include not only security policy, but other things, perhaps including local sexual harassement policy.

The most probable case is scanning for viruses. A less ambitious HTTP proxy might search the HTML content of the download for Java code and remove it if the local policy says no Java from the net.

This case is distinguished from the previous case in that the content of the transmission is examined, while the previous case examines only the packaging. This case might recognize a MS-Word document that appears to need an X rating, while the previous case would simply remove any MSWord virus macros.