Network Working Group Ned Freed, Innosoft
Kevin Carosso, Innosoft
Internet Draft <draft-freed-firewall-req-02.txt>

An Internet Firewall Transparency Requirement

December 1997

Status of this Memo

1. Abstract

This memo defines a basic transparency requirement for Internet firewalls. While such a requirement may seem obvious, the fact of the matter is that firewall behavior is currently either unspecified or underspecified, and this lack of specificity often causes problems in practice. This requirement is intended to be a necessary first step in making the behavior of firewalls more consistent and correct.

2. Introduction

The Internet is now being used for an increasing number of mission critical applications. Because of this many sites find isolated secure intranets insufficient for their needs, even when those intranets are based on and use Internet protocols. Instead they find it necessary to provide direct communications paths between the unsecured and sometimes hostile Internet and systems or networks which either deal with valuable data, provide vital services, or both.

The security concerns that inevitably arise from such setups are often dealt with by inserting one or more "firewalls" on the path between the Internet and the internal network. A "firewall" is an agent which screens network traffic in some way, blocking traffic it believes to inappropriate, dangerous, or both.

More specifically, firewalls either act as a protocol end point (e.g. a SMTP client/server or a Web proxy agent), as a packet filter, or some combination of both.

When a firewall acts a protocol end point it may

(1) implement a "safe" subset of the protocol,

(2) perform extensive protocol validity checks,

(3) use an implementation methodology designed to minimize the liklihood of bugs,

(4) run in an insolated, "safe" environment, or

(5) use some combination of these techniques in tandem.

In the case of a packet filter the firewall isn't visible as a protocol end point. Each packet is examined and the firewall may then

(1) pass the packet through to the other side unchanged,

(2) drop the packet entirely, or

(3) handle the packet itself in some way.

Firewalls typically base some of their decisions on IP source and destination addresses and port numbers. For example, firewalls may

(1) block packets from the Internet side that claim a source address of a system on the internal network,

(2) block TELNET or RLOGIN connections from the Internet to the internal network,

(3) block SMTP and FTP connections to the Internet from internal systems not authorized to send email or move files,

(4) act as an intermediate server in handling SMTP and HTTP connections in either direction, or

(5) require the use of an access negotiation and encapsulation protocol like SOCKS [1] to gain access to the Internet, to the internal network, or both.

(This list is only intended to illustrate the sorts of things firewalls often do; it is by no means exhaustive, nor are all firewall products able to perform all the operations on this list.)

Unfortunately, the development and deployment of firewalls has for the most part been ignored by the Internet standards community. As a result of this inattention the use of firewalls has solved some old problems, but not without generating lots of new ones in the process.

This memo is intended to address the new problems firewalls can cause by establishing a basic transparency requirement for firewalls.

2.1. Requirements notation

This document occasionally uses terms that appear in capital letters. When the terms "MUST", "SHOULD", "MUST NOT", "SHOULD NOT", and "MAY" appear capitalized, they are being used to indicate particular requirements of this specification. A discussion of the meanings of these terms appears in RFC 2119 [3].

3. The Transparency Requirement

The basic transparency requirement for firewalls is quite simple:

> The introduction of a firewall and any associated tunneling or access negotiation facilities MUST NOT cause the gratuitous failure of legitimate and standards-compliant usage that would work were the firewall not present.

A necessary corollary to this requirement is that when such failures do occur it is incumbent on the firewall and associated software to address the problem: Changes to either implementations of existing standard protocols or the protocols themselves MUST NOT be necessary.

Note that this requirement only applies to legitimate protocol usage and gratuitous faiures -- a firewall is entitled to block any sort of access that is seen as illegitimate, regardless of whether or not it is standards-compliant. This is, after all, the primary reason to have a firewall in the first place.

4. Security Considerations

The transparency rule impacts security to the extent that it precludes certain simpleminded firewall implementation techniques. Firewall implementors must therefore work a little harder to achieve a given level of security. However, the transparency rule in no way prevents an implementor from achieving whatever level of security is necessary. Moreover, a little more work up front results in better security in the long run because techniques that do not interfere with existing services will almost certainly be more widely deployed than ones that do interfere and prevent people from performing useful work.

Now, some firewall implementors will inevitably claim that the burden of total transparency is overly onerous and that adequate security cannot be achieved in the face of such a requirement. And there's no question that meeting the transparency requirement is more difficult than not doing so.

Nevertheless, it is important to remember that the only perfectly secure network is one that doesn't allow any data through at all, and that the only problem with such a network is that it is unusable. Anything less is necessarily a tradeoff between useability and security. And the simple fact of the matter is that at present firewalls are being circumvented in ad hoc ways, necessarily weakening security dramatically, simply because they don't meet this transparency requirement. In other words, the only reason that some firewalls remain in use is because they have essentially been disabled. As such, one reason to have a transparency requirement is to IMPROVE security.

Good security may occasionallly result in interoperability failures between components. This is understood. However, this doesn't mean that gratiutous interoperability failures caused by security components are acceptable. They aren't.

5. Example Violations of the Transparency Rule

This document will conclude with a (long) list of existing firewall behavior that violates the transparency requirement.

(1) In an effort to enhance security by hiding internal system names that might otherwise be revealed in email headers, some firewalls either strip information from or completely delete certain header fields. There are even known cases of certain text strings (e.g. names of internal hosts systems) being deleted from message bodies.

Blindly deleting information from message body text is simply not acceptable. Consider what happens when a string is deleted from a binary part encoded in base64 simply because it matches some string pattern somewhere, or what happens when someone has given their own name to their personal system.

Deleting "Received:" fields from message headers is also problematic, as it interferes with message loop detection. In addition, some firewalls delete "Received:" fields in messages passing from the Internet to the local network in addition to messages going the other way, and this actually compromises

security as it eliminates trace information vital in determining a hostile message's possible origin.

The solution is simple for messages passing from the external Internet to internal hosts: Don't delete Received: headers because this compromises, rather than enhances, security. As for Received: headers going in the other direction, one approach is to obscure host name information using name replacement, hash functions, or encryption, rather than removing the field entirely. Simple encryption schemes lead to host names are meaningless to outsiders but if need be can be analyzed by a security manager to determine the actual underlying name.

(2) Firewalls implementing Web proxies often trash URLs which are very long, contain odd (but legal) characters, or contain many separator characters. The result is that Web applications which employ such URLs, such as directory applications, tend not to work properly through many firewall products, even though the URLs being used are completely legal, safe, and correct.

(3) Many firewalls which act as SMTP proxy agents implement only the most rudimentary form of SMTP service. The result is that the ability to use many useful SMTP facilities (DSNs, size negotiation, authentication, pipelining, etc.) is eliminated. In the case of DSNs and authentication once again this action lessens security rather than strengthening it.

(4) DNS service behind many firewalls works very poorly. Firewalls often implement the concept of a split between the part of a domain the outside world can see and the part the inside world can see. And firewalls are often called upon to create DNS setups of this sort. This is often done poorly -- perhaps it is just too difficult to configure such things properly. The net result often is that such restrictions end up getting summarily dumped, which again may compromise security more than it strengthens it.

Note that this also makes it hard to deploy a good mailer on the inside even if the firewall lets the SMTP

traffic through to/from the mail hub. A mail hub inside such a setup cannot get a true picture of the outside world, and this once again may end up compromising security rather than strengthening it.

(5) Many firewalls handle TCP connections in a way which lies somewhere between acting as a full-fledged application protocol proxy and a transpart connection path. Specifically, they intercept all attempts to open TCP connections across the firewall and respond to them immediately, making the connection initiator think the open has succeeded. They then make a connection of their own to the actual destination host. If that connection succeeds subsequent data is then forwarded from one side to the other, possibly with some additional examination nd/or modification occurring at the firewall, or possibly not.

The first problem with this technique arises when the connection from the firewall to the actual destination cannot be opened. There is no way to properly convey the semantics of such a failure to the connection originator since the firewall has already completed the connection to the originator. The firewall must then either content itself with simply closing the connection or else send some protocol-specific response, which applications may interpret quite differently than a transport level connection failure. (In fact such differing interpretation is required in some protocols.)

In the case of an SMTP transfer to a destination with multiple MX and A records, for example, existing clients may interpret a successful connection open as constituting a delivery attempt requiring no subsequent connection attempts to other MX or A records. This in turn can lead to delivery failures when one or more hosts in an MX or A record list aren't available for a prolonged period of time.

The second problem with this technique arises when modifications are made to packets transferred from one side to the other without full knowledge of the underlying protocol. For example, situations have arisen where firewalls attempt to "censor" an SMTP data

stream and end up removing data from that stream, operating under the assumption that since SMTP uses dot-stuffing rather than counted data such editing is acceptable. Unfortunately, the SMTP protocol has changed in recent years and now includes the BDAT command [2] for transferring counted data, among other things. Removing octets from a BDAT data stream inevitably leads to protocol desynchronization, timeouts,transfer failures, and message delivery failures.

A similar problem can occur with SMTP's VRFY command, which is a mandatory part of the SMTP protocol. Firewalls like to intercept VRFY because of the perception that VRFY exposes internal information unnecessarily. (This perception is false as comparable exposure occurs with SMTP's RCPT TO, and disabling or interfering with RCPT TO breaks the protocol completely.) Unfortunately, while it is possible to intercept and effectively disable VRFY properly [4], several firewall don't do it correctly. Ways of disabling VRFY incorrectly include returning a 5XX SMTP error unconditionally (which can lead to delivery failures when working with clients that do VRFYs prior to each RCPT TO) and failing to return a properly formatted 2XX SMTP success code (RFC 821 [5] requires that the response to a VRFY include an RFC 822 [6] address enclosed in angle brackets).

6. References

[1] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones, "SOCKS Protocol Version 5", RFC 1928, April, 1996.

[2] Vaudreuil, G., "SMTP Service Extensions for Transmission of Large and Binary MIME Messages", RFC1830, August, 1995.

[3] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[4] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, USC/Information Sciences Institute, October 1989.

[5] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, USC/Information Sciences Institute, August 1982.

[6] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, August 1982.

7. Authors' Addresses

Ned Freed
Innosoft International, Inc.
1050 Lakes Drive
West Covina, CA 91790
USA
tel: +1 626 919 3600 fax: +1 626 919 3614
email: ned.freed@innosoft.com

Kevin Carosso
Innosoft International, Inc.
1050 Lakes Drive
West Covina, CA 91790
USA
tel: +1 626 919 3600 fax: +1 626 919 3614
email: kevin.carosso@innosoft.com

8. Full Copyright Statement