Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 24, 2013

D. Thaler
Microsoft

June 22, 2013

Reflections On Host Firewalls
draft-iab-host-firewalls-00.txt

Abstract

In today's Internet, the need for firewalls is generally accepted in the industry and indeed firewalls are widely deployed in practice. Often the result is that software may be running and potentially consuming resources, but then communication is blocked by a firewall. It's taken for granted that this end state is either desirable or the best that can be achieved in practice, rather than (for example) an end state where the relevant software is not running or is running in a way that would not result in unwanted communication. In this document, we explore the issues behind these assumptions and provide suggestions on improving the architecture going forward.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 24, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

"Behavior of and Requirements for Internet Firewalls" [RFC2979] provides an introduction to firewalls and the requirement for transparency in particular, stating:

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

In Section 2.1 of "Reflections on Internet Transparency" [RFC4924], the IAB provided additional thoughts on firewalls and their impact on the Internet architecture, including issues around disclosure obligations and complexity as applications evolve to circumvent firewalls. This document extends that discussion with additional considerations.

Traditionally, firewalls have been about arming customers to protect against badly written applications and services. This document discusses a number of fundamental questions such as who a firewall is meant to protect from what. It does so primarily, though not exclusively, from an end system perspective with a focus on host firewalls in particular.

The Internet Security Glossary [RFC4949] defines a firewall as follows.

> 1. (I) An internetwork gateway that restricts data communication traffic to and from one of the connected networks (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall). (See: guard, security gateway.)
>
> 2. (O) A device or system that controls the flow of traffic between networks using differing security postures. [SP41]
>
> Tutorial: A firewall typically protects a smaller, secure network (such as a corporate LAN, or even just one host) from a larger network (such as the Internet). The firewall is installed at the point where the networks connect, and the firewall applies policy rules to control traffic that flows in and out of the protected network.
>
> A firewall is not always a single computer. For example, a firewall may consist of a pair of filtering routers and one or more proxy servers running on one or more bastion hosts, all connected to a small, dedicated LAN (see: buffer zone) between the two routers. The external router blocks attacks that use IP to break security (IP address spoofing, source routing, packet fragments), while proxy servers block attacks that would exploit a vulnerability in a higher-layer protocol or service. The internal router blocks traffic from leaving the protected network except through the proxy servers. The difficult part is defining criteria by which packets are denied passage through the firewall, because a firewall not only needs to keep unauthorized traffic (i.e., intruders) out, but usually also needs to let authorized traffic pass both in and out.

Informally, most people would tend to think of a firewall as "something that blocks unwanted traffic" (see [RFC4948] for a discussion on many types of unwanted traffic). A fundamental question is, however: "unwanted by whom?"

Possible answers include end users, application developers, network administrators, host administrators, firewall vendors, and content providers. We will exclude by definition the sender of the traffic in question, since the sender would generally want such traffic to be delivered. Still, the other entities have different, and often conflicting, desires which means that a type of traffic might be wanted by one entity and unwanted by another entity. Thus, not surprisingly, there exist various types of firewalls, and various types of "arms race" as we will discuss in Section 4.1.2.

1.1. Terminology

In this document we distinguish between a "host firewall" which simply intends to protect the single computer on which it runs, and a "network firewall" which is located in the network and intends to protect the network and any hosts behind it.

The term "application" is used generically to apply to any component that can receive traffic. In this sense, it could refer to a process running on a computer (including a system service) or even to a portion of a TCP/IP stack itself, such as a component that responds to pings.

2. Firewall Rules

Desires for wanted or unwanted traffic can be expressed in terms of "allow" vs. "block" rules, with some way to resolve conflicting rules. Many firewalls are actually implemented in terms of such rules. Figure 1 shows some typical sources of such rules.

Source	Consumer Grade Host Firewall	Consumer Grade Network Firewall	Enterprise Grade Host Firewall	Enterprise Grade Network Firewall
End user	Sometimes (as host admin)	Sometimes (as network admin)		
App developer	Yes	Sometimes (via protocols)		
Network admin		Sometimes		Yes
Host admin	Sometimes		Yes	

Firewall vendor	Yes	Yes	Yes	Yes

Common sources of firewall rules

Figure 1

Figure 1 assumes that network firewalls are administered by network administrators (if any), and host firewalls are administered by host administrators (if any). A firewall may also have rules provided by the firewall vendor itself.

End users typically cannot directly provide rules to firewalls, except when acting as host or network administrators. Application developers can, however, provide such rules to some firewalls, such as providing rules at installation time, for example by invoking an API provided by a host firewall included with the operating system, or by providing metadata to the operating system for use by firewalls, or by using a protocol such as UPnP-IGD or PCP to communicate with a network firewall (see Section 4.1.3 for a longer discussion).

Firewall rules generally fall into two categories:

1. Attack surface reduction: Rules intended to prevent an application from doing things the developer does not want it to do.

2. Security policy: Rules intended to prevent an application from doing things the developer might want it to do, but an administrator does not.

A firewall is unnecessary if both categories are empty. We will now treat each category in turn.

3. Category 1: Attack Surface Reduction

As noted above, this category of firewall rule typically attempts to prevent applications from doing things the developer did not intend.

One might ask whether this category of rules is typically empty, and the answer is that it is not today. One reason stems from mitigating code injection threats by putting a security barrier in a separate process isolated from the potentially compromised process. Furthermore, there is also some desire for a "stealth mode" (see below).

Hence, typically a firewall will have rules to block everything by default. A one-time, privileged, application install step adds one or more Allow rules, and then normal (unprivileged) application execution is then constrained by the resulting rules.

A second reason this category of rules is non-empty is where they are used as workarounds for cases the application developer found too onerous to implement. These cases include:

1. Simple policies that the developer would want, but that are difficult to implement. One example might be a policy that an application should communicate only within the local network (e.g., a home or enterprise) but not be reachable from the global Internet or while the device is moved to some public network such as a hotspot. A second example might be the reverse, i.e., a policy to communicate over the Internet but not with local entities. The need for this category would be reduced by better platform support for such policies, making them easier for developers to implement and use.

2. Complex policies where the developer cannot possibly be aware of specifics. One example might be a policy to communicate only during, or only outside of, normal business hours, where the exact hours may vary by location and time of year. Another example might be a policy to avoid communication over links that cost too much, where the definition of "too much" may vary by customer, and indeed the end host and application might not even be aware of the costs. The need for this category would be reduced by better platform support for such policies, allowing the application to communicate through some simple API with some other library or service that can deal with the specifics.

3.1. Stealth Mode

There is often a desire to hide from address and port scans on a public network. However, compliance to many RFCs requires responding to various messages. For example, TCP [RFC0793] compliance requires sending a RST in response to a SYN when there is no listener, and ICMPv6 [RFC4443] compliance requires sending an Echo Reply in response to an Echo Request.

Firewall rules can allow such stealth without changing the statement of compliance of the basic protocols. However, stealth mode could instead be implemented as a configurable option used by the applications themselves. For example, rather than a firewall rule to drop a certain outbound message after an application generates it, fewer resources would be consumed if the application knew not to generate it in the first place.

3.2. Discussion of Approaches

When running an application would result in unwanted behavior, customers have three choices, which we will discuss in turn:

a. fix (or get the developer to fix) the software,

b. not use the software, or

c. let the software run, but then use a firewall to thwart it and prevent it from working in unwanted ways.

3.2.1. Fix the Software

Firewall vendors point out that one can more quickly and reliably update firewall rules than application software. Indeed some applications might have no way to update them, and support for other applications might no longer be available (e.g., if the developers are no longer around). Most modern operating systems (and any applications that come with them) have automatic updates, as do some independent applications. But until all applications have automatic updates, and automatic updates are actually used, it will still be the case that firewall rules can be updated more quickly than software patches. Furthermore, in some contexts (e.g., within some enterprises), a possibly lengthy retesting and recertification process must be employed before applications can be updated. In short, mechanisms to encourage and ease the use of secure automatic software updates are important and would greatly reduce overall complexity.

3.2.2. Don't Use the Software

A key question to ask is whether the application could still do something useful when firewalled. If the answer is yes, then not using the software is probably unrealistic. For example, a game with both single-player and multi-player capabilities could still be useful in single-player mode when firewalled. If instead the answer is no, it is better to not allow the application to run in the first place (and some host firewalls can indeed block applications from running).

3.2.3. Run the Software Behind a Firewall

As noted earlier, one disadvantage of this approach is that resources still get consumed. For example, the application can still consume memory, CPU, bandwidth (up to the point of blockage), ports in the transport layer protocol, and possibly other resources depending on the application, for operations that provide no benefit while firewalled.

A second important disadvantage of this approach is the bad user experience. Typically the application and the end-user won't know why the application doesn't work. In addition, a poorly designed application might not cope well and consume even more resources (e.g., retrying an operation that continually fails). Finally, using a firewall without changing the application could impact protocol operation and hence have adverse effects on other endpoints; for example blocking ICMP adversely affects path MTU discovery which can cause problems for other entities (see [RFC4890] and Section 3.1.1 of [RFC2979] for further discussion).

Sandboxed environments, such as those provided by browsers, can be thought of as a type of firewall in the more general sense. For example, a cross-site check in a browser can be thought of as a rule specified by the "firewall" vendor to block unwanted outbound traffic per a "same origin policy" where a script can only communicate with the "site" from which the script was obtained, for some definition of site such as the scheme and authority in a URI.

In short, it is important to make applications more aware of the constraints of their environment and hence better able to behave well when contrained.

4. Category 2: Security Policy

As noted in Section 2, this category of firewall rule typically attempts to prevent applications from doing things an administrator does not want, even if the application developer does intend them to do.

One might ask whether this category of rules is typically empty, and the answer varies somewhat. For enterprise-grade firewalls, it is almost never empty, and hence the problems discussed in Section 3.2.3 can be common here too. Similarly, for consumer-grade firewalls, it is generally not empty but there are some notable exceptions. For example, for the host firewall in the Windows operation system, this category always starts empty and most users never change this. [[Any other major firewalls that also contain no such rules by default?]]

4.1. Discussion of Approaches

Security policy can be implemented in any of three places, which we will discuss in turn: the application, a firewall, or a library/ service that that application explicitly uses.

4.1.1. Security Policies in Applications

In this option, each application must implement support for potentially complex security policies, along with ways for administrators to configure them. Although the explicit interaction with applications avoids the problems discussed in Section 3.2.3, this approach is impractical for a number of reasons. First, the complexity makes it difficult to implement and error-prone, especially for application developers whose primary expertise is not networking. Second, the potentially large number of applications (and application developers) results in an inconsistent experience that makes it difficult for an administrator to manage common policies across applications, thus driving up training and operational costs.

4.1.2. Security Policies in Firewalls

Putting security policies in firewalls without explicit interaction with the applications results in the problems discussed in Section 3.2.3. In addition, this leads to "arms races" where the applications are incented to evolve to get around the security policies, since the desires of the end user or developer can conflict with the desires of the host or network administrator. As stated in Section 2.1 of [RFC4924]:

> In practice, filtering intended to block or restrict application usage is difficult to successfully implement without customer consent, since over time developers will tend to re-engineer filtered protocols so as to avoid the filters. Thus over time, filtering is likely to result in interoperability issues or unnecessary complexity. These costs come without the benefit of effective filtering since many application protocols began to use HTTP as a transport protocol after application developers observed that firewalls allow HTTP traffic while dropping packets for unknown protocols.

Such arms races stem from inherent tussles between the desires of different entities. For example, the tussle between end user desires and network administrator desires led to the arms race between network firewalls and deep packet inspection on the one hand, vs. the use of tunnels and obfuscation on the other. Similarly, the tussle between application developer desires and network administrator desires contributed to the use of HTTP as a transport in order to work from within the widest possible set of networks. Section 4 of [RFC2979] states:

> Wrapping a new protocol around HTTP and using port 80 because it is likely to be open isn't a good idea, since it will eventually result in added complexity in firewall handling of port 80.

However the practice is now widespread and even standardized ([RFC6455]). See Section 3.3.1 of [RFC6250] and [I-D.blanchet-iab-internetoverport443] for more discussion.

Such arms races most typically, though not exclusively, occur with network (not host) firewalls. This is because it is more likely for network firewalls to have lack of trust between the policy-desiring entities, and less likely that there is any trusted arbiter.

4.1.3. Security Policies in a Service

In this approach, applications use a library or other external service whereby the applications have explicit knowledge of the impact of the security policies in order to avoid the problems in Section 3.2.3 (and in a sandboxed environment this might be the application's only way to interact with the network).

Thus in this opt-in approach, applications provide a description of the network access requested, and the library/service can ensure that applications and/or users are informed in a way they can understand, and that administrators can craft policy that affects the applications.

This approach is very difficult to do in a firewall-implementation-specific library/service when there can be multiple firewall implementations (including ones in the middle of the network), since it is usually impractical for an application developer to know about and develop for many different firewall APIs. It is, however, possible to employ this approach with a generic library/service that can communicate with both applications and firewalls. Thus application developers and firewall developers can use a common platform.

We observe that this approach is very different from the classic firewall approach. It is, however the approach used by the aforementioned Windows firewall, and it is also the approach used by the Port Control Protocol (PCP) [RFC6887] in the IETF. As such, we encourage the deployment and use of this model. [[Any other major platforms besides Windows that already use this approach?]]

Furthermore, while this approach lessens the incentive for arms races as discussed above, one important issue still remains. Namely, there is no standard mechanism today for a library to learn complex policies from the network. Further work in this area is needed.

5. Security Considerations

There is a common misconception that firewalls protect users from malware, when in fact firewalls protect users from buggy software. There is some concern that firewalls give users a false sense of security; firewalls are not invulnerable and will not prevent malware from running if the user allows it.

This document has focused primarily on host firewalls. For additional discussion (focused more on network firewalls), see [RFC2979], [I-D.iab-filtering-considerations], and [I-D.baker-opsawg-firewalls].

6. IANA Considerations

This document requires no actions by the IANA.

7. Acknowledgements

Stuart Cheshire provided the motivation for this document by asking the thought-provoking question of why one would want to firewall an application rather than simply stop running it. The ensuring discussion, and subsequent IAB tech chat, led to this document. Stephen Bensley and Gerardo Diaz Cuellar also provided helpful suggestions.

8. IAB Members at the Time of This Writing

Bernard Aboba
Jari Arkko
Marc Blanchet
Ross Callon
Alissa Cooper
Joel Halpern
Russ Housley
Eliot Lear

Xing Li
Andrew Sullivan
Dave Thaler
Hannes Tschofenig

9. Informative References

[I-D.baker-opsawg-firewalls]
Baker, F., "On Firewalls in Internet Security", draft-baker-opsawg-firewalls-00 (work in progress), January 2012.

[I-D.blanchet-iab-internetoverport443]
Blanchet, M., "Implications of Blocking Outgoing Ports Except Ports 80 and 443", draft-blanchet-iab-internetoverport443-01 (work in progress), October 2012.

[I-D.iab-filtering-considerations]
Barnes, R., Cooper, A., and O. Kolkman, "Technical Considerations for Internet Service Filtering", draft-iab-filtering-considerations-01 (work in progress), October 2012.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2979] Freed, N., "Behavior of and Requirements for Internet Firewalls", RFC 2979, October 2000.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, May 2007.

[RFC4924] Aboba, B. and E. Davies, "Reflections on Internet Transparency", RFC 4924, July 2007.

[RFC4948] Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, August 2007.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC6250] Thaler, D., "Evolution of the IP Model", RFC 6250, May 2011.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

Author's Address

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 703 8835
Email: dthaler@microsoft.com