Reflections on Host Firewalls

Abstract

   In today's Internet, the need for firewalls is generally accepted in
   the industry, and indeed firewalls are widely deployed in practice.
   Unlike traditional firewalls that protect network links, host
   firewalls run in end-user systems. Often the result is that software
   may be running and potentially consuming resources, but then
   communication is blocked by a host 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 document is not an Internet Standards Track specification; it is
   published for informational purposes.

   This document is a product of the Internet Architecture Board (IAB)
   and represents information that the IAB has deemed valuable to
   provide for permanent record. It represents the consensus of the
   Internet Architecture Board (IAB). Documents approved for
   publication by the IAB are not a candidate for any level of Internet
   Standard; see Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7288.

Table of Contents

1. Introduction

   [BLOCK-FILTER] discusses the issue of blocking or filtering abusive
   or objectionable content and communications, and the effects on the
   overall Internet architecture. This document complements that
   discussion by focusing on the architectural effects of host firewalls
   on hosts and applications.

   "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.

   Many firewalls today do not follow that guidance, such as by blocking
   traffic containing IP options or IPv6 extension headers (see
   [RFC7045] for more discussion).

   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 bugs in 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.

   While the Internet Security Glossary [RFC4949] contains an extended
   definition of a firewall, informally, most people would tend to think
   of a firewall as simply "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.

A Network Address Translator (NAT) is also often viewed, or even
marketed, as a type of network firewall; Section 2.2 of [RFC4864]
addresses this misconception, but nevertheless some of the same
observations in the present document may also apply to NATs.

Sandboxed environments, such as those provided by browsers, can also
be thought of as a type of host firewall in the more general sense.
For example, a cross-site check in a browser can be thought of as a
mechanism 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.

The term "application" is used in this document 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 Scenario Host Firewall	Consumer Scenario Network Firewall	Enterprise Scenario Host Firewall	Enterprise Scenario 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

                 Figure 1: Common Sources of Firewall Rules

   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 that
   affect other users, unless the end user is a host or network
   administrator. Application developers can, however, provide such
   rules to some firewalls, such as providing rules at installation
   time. They can do this, 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 Universal Plug and Play (UPnP) [UPNPWANIP] or the
   Port Control Protocol (PCP) [RFC6887] 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, focusing specifically on host firewalls
(although some points might be equally applicable to network
firewalls).

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, at present. One reason stems from
mitigating the threat of vulnerability exploitation 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 Section 5 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. 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.

3.1.1. 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. Such mechanisms should allow scheduling updates
at appropriate times, taking into account operational considerations
such as dependencies, compatibility, testing and maintenance windows.

3.1.2. 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.

3.1.3. Run the Software behind a Host 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. A poorly designed application
   might not cope well and consume even more resources (e.g., retrying
   an operation that continually fails).

   A third disadvantage is that it is common for a firewall rule to
   block more that is appropriate for attack surface reduction,
   impacting protocol operation and even having adverse effects on other
   endpoints. For example, some firewalls that cannot perform full deep
   packet inspection at line speed have adopted a block by default
   approach to anything they don't understand from the first few bytes;
   this is very harmful to innovation as it interferes with the ability
   to deploy new protocols and features.

   As another 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). This can
   happen due to lack of understanding all the details of application
   behavior, or due to accidental misconfiguration. Section 2.1 of
   [RFC5505] states, "Anything that can be configured can be
   misconfigured," and discusses this in more detail.

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

4. Category 2: Security Policy

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

   One might ask whether this category of rules is typically empty, and
   the answer varies somewhat. For enterprise-scenario firewalls, it is
   almost never empty, and hence the problems discussed in Section 3.1.3
   can be common here too. Similarly, for consumer-scenario firewalls,
   it is generally not empty but there are some notable exceptions. For
   example, for the host firewall in some operation systems, this
   category always starts empty and most users never change this.

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 the 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 3.1.3,
   this approach is impractical for a number of reasons. First, the
   complexity makes it difficult to implement and is 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 Host Firewalls

   Putting security policies in firewalls without explicit interaction
   with the applications results in the problems discussed in
   Section 3.1.3. 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 an 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 administrator desires leads to an arms race between firewalls and
   deep packet inspection on the one hand, vs. the use of obfuscation or
   tunnels on the other.

   Although such arms races are often thought of in the context of
   network firewalls, they also occur with host firewalls. It is,
   however, generally easier for a host firewall to overcome, since it
   may be more practical for a host firewall to establish some form of
   trust between the policy-desiring entities, and have a 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 3.1.3, 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-vendor-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 firewall-vendor-agnostic 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 some host
operating system firewalls, and it is also the approach used by PCP
in the IETF. As such, we encourage the deployment and use of this
model.

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/service to learn complex
policies from the network. Further work in this area is needed.

5. 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.

6. Security Considerations

There is a common misconception that firewalls protect users from
malware on their computer, 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] and [BLOCK-FILTER].

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 ensuing
discussion, and subsequent IAB tech chat in November 2011, led to
this document. Dan Simon, Stephen Bensley, Gerardo Diaz Cuellar,
Brian Carpenter, and Paul Hoffman also provided helpful suggestions.

8. IAB Members at the Time of Approval

   Bernard Aboba
   Jari Arkko
   Marc Blanchet
   Ross Callon
   Alissa Cooper
   Joel Halpern
   Russ Housley
   Eliot Lear
   Xing Li
   Erik Nordmark
   Andrew Sullivan
   Dave Thaler
   Hannes Tschofenig

9. Informative References

   [BLOCK-FILTER]
            Barnes, R., Cooper, A., and O. Kolkman, "Technical
            Considerations for Internet Service Blocking and
            Filtering", Work in Progress, January 2014.

   [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.

   [RFC4864] Van de Velde, G., Hain, T., Droms, R., Carpenter, B., and
            E. Klein, "Local Network Protection for IPv6", RFC 4864,
            May 2007.

   [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.

   [RFC5505] Aboba, B., Thaler, D., Andersson, L., and S. Cheshire,
              "Principles of Internet Host Configuration", RFC 5505, May
              2009.

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

   [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing
              of IPv6 Extension Headers", RFC 7045, December 2013.

   [UPNPWANIP]
              UPnP Forum, "WANIPConnection:2 Service", September 2010,
              <http://upnp.org/specs/gw/
              UPnP-gw-WANIPConnection-v2-Service.pdf>.

Author's Address

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

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