Network Working Group M. Blanchet
Internet-Draft Viagenie

Implications of Blocking Outgoing Ports Except Ports 80 and 443
draft-blanchet-iab-internetoverport443-02.txt

Abstract

Users are often connected to Internet with very few outgoing ports available, such as only port 80 and 443 over TCP. This situation has many implications on designing, deploying and using IETF protocols, such as encaspulating protocols within HTTP, difficulty to do traffic engineering, quality of service, peer-to-peer, multi-channel protocols or deploying new transport protocols. This document describes the situation and its implications.

Table of Contents

1. Introduction

A trend started many years ago has been to provide Internet access to end-users with limited outgoing ports. The most constraint but common case is to only have outgoing TCP port 80 and port 443 opened. Port 80 is expected to carry HTTP and some middleboxes in the network may block non-HTTP traffic on that port. Port 443 is often less policed than port 80 based on the assumption that it is carrying encrypted traffic. However, enterprise firewalls sometimes verify the use of TLS/SSL on port 443.

A consequence of this trend is that Internet statistics show [Labovitz] that a majority of the Internet traffic is over port 80 and 443. And the concentration on these ports are further increasing every year.

While the purpose of this document is not to find or judge the reasons why providers (in the large sense of providing) are blocking all outgoing ports except very few, a few known reasons can be listed, while no opinion on the validity is expressed:

- Users only need HTTP anyway. Now email and chat is over HTTP.
- Less number of ports means easier control over shadow traffic.
- Provider wants to control, verify, police all outgoing traffic.

A consequence for the enterprise or non-HTTP application service provider is that there are very few ways to offer a service to its end-users. For example, an application (VoIP, ssh, jabber, ftp, ...) provider need to use an additional IP address and to bind its application server to the port 443 to make sure its users can reach it whatever the characteristics of the access network the nomadic users are attaching to. The other way is to build a tunnel such as VPN to the service infrastructure and then tunnel all application traffic to that tunnel. Obviously for the same reason, the tunnel server itself has to be bound on port 443.

From the application developer point of view, HTTP framework is often chosenfor its own benefits with or without the limited outgoing ports deployment considerations, as discussed in [I-D.tschofenig-post-standardization].

2. Terminology

This document uses the term provider in a large sense of some organization offering the Internet access to users. For example, a provider in this document includes coffee shop wifi access, guest access in various public places and networks, hotel networks, enterprise guest access networks, as well as traditional providers such as broadband, mobile and wifi network established large providers.

3. Implications

3.1. IETF Guidance

IETF provided guidance about the use of HTTP and port 80. For example, [RFC3205] recommended to use different ports than 80 for new services, even when HTTP encapsulation was used. This guidance may need to be revisited.

This situation further complicates the Internet transparency, end-to-end and hourglass model, as discussed in [RFC2775],[RFC3234] and [RFC4924].

3.2. Trafic Policing

If all traffic goes over one or two ports, then it is more difficult to differentiate delay sensitive traffic to bulk traffic while applying policies on forwarding engines at the transport level. The policing nodes on the network haveto open the application payload. For example, for Motion-JPEG over http, parsing the HTTP headers is needed to discover that this data is streaming.

3.3. Deploying New Protocols

If port 80 and 443 are the only ports opened, then given that middleboxes in networks are inspecting packets and validate HTTP traffic, then a new protocol not based on HTTP and requiring a different transport port or protocol is difficult, while impossible, to deploy as is.

3.4. Overloading HTTP

Another consequence of this situation is that protocols and data go over HTTP. HTTP is defined with a specific set of requirements and is implemented in a solution set that is far from the IP layer. It uses TCP transport, has multiple ascii headers in the payload to be parsed, has state, etc. However, the HTTP protocol is being revised [RFC6455][httpbis] related to some of these new requirements.

3.5. Increasing the rate of usage of IP addresses

If an organization has N different services where each one takes a different port, then, in the context of its users only able to use outgoing port 80/443, the organization has to use N IP addresses, one for each service and bind the service on port 443 (or 80) on that IP address. Therefore, the organization increases the rate of its usage of IP addresses. Since IPv4 addresses are almost exhausted, this situation adds pain to the IPv4 address exhaustion. IPv6 addresses are almost limitless to this issue, but having too many IPv6

addresses on the same server to support the services add complexity to the operations.

3.6. More Complex Operations

As a network operator likes to monitor traffic to engineer and troubleshoot the network, it cannot do anymore by only looking at the ports used by the traffic. For example, a peak traffic from a source node that always uses a single outgoing port for all its traffic, may be a video call or video streaming or file copy or a virus related traffic or torrent or ... Therefore, the network operations is blind to what the traffic is, unless the monitoring is at done within the application payload.

3.7. Inability to Deploy Applications and Protocols

A good example of limitations to deploy applications and protocols are IP cameras. These devices send video streams to outside. While a typical protocol stack would use RTP/RTSP for this purpose, often the only way to successfully send the stream in all cases is to encapsulate it over HTTP using Motion JPEG or other coding over HTTP. Similar issues also happen for interactive applications. The constraint of the transport protocol to use may have an important impact on the application design and behavior.

3.8. Applications Become Only HTTP-based

From the application developer point of view, the most garanteed way to get its outgoing traffic from the client host to the Internet (servers) is to carry its application data and protocols over HTTP over port 80 and/or 443. This is, whatever the type of traffic, such as gaming, voice, video, file transfer, augmented reality, 3D, ..., with a wide set of different characteristics. Within the HTTP framework, the Websocket Protocol[RFC6455] is one way to support the variety of applications over HTTP.

3.9. Applications Need to Become Very Smart for Opening Connection

Skype is a good illustration of a deployable application that works in most cases. Analysis of Skype behavior [ColumbiaSkype] shows Skype is trying to open outgoing ports, and when not possible, defaults to port 80 or 443 as last resort. Therefore, this illustrates that a successful deployable application should use similar techniques with the last resort being port 80 or 443. That also means the other peer of the communication must be bound to the same 80 or 443 port. This application behavior may require to have standardized ways of handling encapsulation over 80/443 for realtime applications.

3.10. Internet Transport

Written differently, this situation can be described as the Internet can only run with a single transport protocol(TCP) and two transport ports(80,443). Given that some deployments have HTTP-aware middleboxes on those ports, then the Internet can only run "reliably" over a single transport protocol (HTTP) and a single transport port (443).

3.11. Should IETF Protocols Only Use HTTP Encapsulation

Given above, should the IETF only design protocols over HTTP? Should all current protocols be redesigned to be carried over HTTP? (more a question to debate than an affirmation...)

For example, 3GPP and MPEG produced the Dynamic Adaptive Streaming over HTTP(DASH) protocol[DASH] where one of the reasons is related to firewalls and NAT traversal. This new protocol is intended to replace the RTSP [RFC2326] protocol.

Websockets[RFC6455] is a standardized way to encapsulate subprotocols within HTTP and therefore multiplexing the various application protocols within HTTP.

[I-D.tschofenig-post-standardization] also discuss about this issue.

4. Mitigation

IPv6 could be seen as a way to mitigate that problem. As discussed above, the reasons why access providers or enterprises are limiting outgoing ports are not related to IPv4 address exhaustion or IPv4 itself.

However, on the server side of the connections, given the large IPv6 address space available per server, IPv6 could be used to partly mitigate the problem by having, on a single server, each service bound to a different IPv6 address while using the same transport port 80.

It should be noted that some IPv6 access providers are not blocking any port, helping restoring the Internet transparency.

5. Recommendations

A new network protocol that would likely be used and deployed in an environment described above, should:

- consider the issues listed and identify how the protocol specification will mitigate the issues. For example, what happens if only port 80 and/or 443 over TCP are available for the end user to start its connection with that protocol? What happens if HTTP protocol inspection is done on those ports by an intermediate node?

- consider, for the "transport" of the protocol, using the HTTP protocol, or enhancements of HTTP such as the RESTFUL or Websockets[RFC6455] methods.

The access network providers, including small organisations such as Internet cafes, should consider opening their outbound ports to mitigate the issues raised above and to enable a full Internet user experience. There is an opportunity to implement these no-outgoing-ports blocking policies for the new IPv6 deployments.

6. Security Considerations

This document does not specify a new protocol. However, it does highlight security impacts of the current Internet access.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

Dave Thaler, Hannes Tschofenig, Brian Carpenter, Bernard Adoba, Joel Halpern, Cameron Byrne have provided input and suggestions to this document.

9. Informative References

[ColumbiaSkype] Baset, S. and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", , <http://www.cs.columbia.edu/~salman/publications/skype1_4.pdf>.

[DASH] , "ISO/IEC 23009-1:2012 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", , <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623>.

[I-D.tschofenig-post-standardization] Tschofenig, H., Aboba, B., Peterson, J., and D. McPherson, "Trends in Web Applications and the Implications on Standardization", draft-tschofenig-post-standardization-02 (work in progress), May 2012.

[Labovitz] Labovitz, C., "Internet Traffic and Content Consolidation", March 2010, <http://www.ietf.org/proceedings/77/slides/plenaryt-4.pdf>.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, February 2000.

[RFC3205] Moore, K., "On the use of HTTP as a Substrate", BCP 56, RFC 3205, February 2002.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

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

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

[httpbis] , "Hypertext Transfer Protocol Bis (httpbis)", , <http://datatracker.ietf.org/wg/httpbis/charter>.

Author's Address

Marc Blanchet
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1

Canada

Email: Marc.Blanchet@viagenie.ca
URI: http://viagenie.ca