rtcweb
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

C. Jennings
S. Nandakumar
J. Rosenberg
Cisco

July 06, 2015

Firewall Traversal for WebRTC
draft-jennings-behave-rtcweb-firewall-00

Abstract

Traversal of RTP through corporate firewalls has traditionally been complex, requiring the deployment of Session Border Controllers (SBCs) or wide open pinholes. This draft proposes a simple technique that allows WebRTC based RTP traffic to traverse firewalls without complex firewall configuration and without deployment of SBCs or other middleboxes.

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 January 7, 2016.

Copyright Notice

Table of Contents

1. Problem Statement

WebRTC [I-D.ietf-rtcweb-overview] based voice and video communications systems are becoming far more common inside enterprises, which often need voice and video media to traverse the enterprise firewall. This can happen when a device inside the firewall such as a web browser or phone is exchanging media with a conference bridge or gateway outside the firewall, or it can happen when a device inside the firewall is talking to a device in another enterprise or behind a different firewall.

This problem is not unique to WebRTC media of course. It is common practice for enterprise administrators to block outbound UDP through the corporate firewall. This is done for several reasons:

1. The lack of any kind of return messages means that there is no way to know that the recipient of the UDP traffic really wants it. Infected computers within the enterprise could utilize UDP

as the source of a DDoS attack. If the firewall permitted such outbound traffic, the enterprise could in effect be a contributing source to such an attack. By blocking UDP, the enterprise IT admin ensures that this cannot happen - at least not to external targets.

2. There have been prior attacks that have utilized UDP as a command and control channel for orchestrating DDoS attacks. At the time, UDP had little usage within enterprises (most VoIP was internal to the enterprise when it existed at all). Consequently, infosec departments have deemed it safer to block UDP outright in order to prevent such further incidents.

3. Many IT administrators enable various packet inspection operations on traffic flowing through the firewall. High volume UDP traffic - such as voice or video - can be costly to inspect. As such, in cases where there is a need for traversal of such traffic, IT has preferred to deploy an SBC that, in essence, verifies that the traffic is VoIP and authorizes its egress. The IT administrator then enables traffic to/from the SBC through the firewall. In other words, VoIP authorization is delegated to an outsourced SBC.

As more and more IP communications services move to the cloud, there is an increased need for VoIP traffic to traverse the enterprise firewall. At the same time, the entire point of a cloud service is that it does not require the deployment of on premises infrastructure, making SBC-based solutions less desirable. An alternative solution that has been historically used is to enable outbound UDP in the firewall to specific IP addresses, corresponding to the external service (TURN servers or conference servers) that the enterprise wishes to authorize. With more applications running on virtual machines within cloud compute platforms like Amazon EC2, IP addresses are decreasingly usable as identifiers for a service. VMs running TURN servers or conferencing servers may be established and torn down by the day, hour or even minute, with continuously changing IP addresses. Given the multitenant nature of such providers, IT departments are unwilling to whitelist the IP addresses for the entire block used by such providers.

Consequently, there is a growing need for solutions that allow VoIP traversal through the corporate firewall that alleviate the concerns above. This issue is further exacerbated by the growing adoption of WebRTC by enterprise applications, which provide a ready source of RTP traffic which often needs to traverse the firewall.

2. Solution Requirements

We believe the solution must meet the following requirements:

REQ-1: The solution must enable traversal of real-time media without requiring deployment of additional media intermediaries on premise (e.g., no SBC required)

REQ-2: The solution must not require the whitelisting of specific external IP addresses

REQ-3: The solution must enable the enterprise to be sure that the receiving party of the traffic desires the traffic

REQ-4: The solution must work with P2P calls between users in different enterprises without requiring a TURN server

REQ-5: The solution must work with cloud services external to the enterprise which terminate media on servers, such as conference servers, voicemail servers, recording servers, and so on.

REQ-6: The solution must not require decryption of either signalling or media traffic at the firewall or at any other intermediary

REQ-7: The solution must allow the IT department to easily make policy decisions about which applications are allowed, or not allowed, to traverse the firewall

REQ-8: The solution must not require DPI of every single UDP packet that traverses the firewall

REQ-9: The solution must provide a minimum level of proof that the traffic is RTP and not something else

REQ-10: The solution must work with WebRTC traffic. Note that solving this for non-WebRTC is a non-requirement.

3. Solution Overview

Many of the reasons for blocking UDP at the corporate firewall have their origins in the lack of a three-way handshake for UDP traffic. TCP’s three-way handshake ensures that the receiving party of the connection desires the traffic. Similarly, HTTP traffic easily traverses the firewall since it provides application identification information in the URL.

Consequently, the solution proposed here relies on the ICE connectivity checks, which provide a similar handshake and ensure

consent of the remote party. When a firewall sees an outbound UDP packet on a 5-tuple which is not yet authorized, it begins looking for STUN packets (as identified by the STUN magic cookie). Any outbound packet that is not a STUN packet is discarded. Once an outbound STUN packet is identified, the 5-tuple is put in a pending state, and the firewall begins looking for STUN packets among inbound UDP packets. When it sees one, it matches the transaction IDs to ensure that they are correlated. Once matched, the 5-tuple is placed into an authorized state, and UDP traffic is allowed to freely traverse.

In addition, the initial outbound STUN packets contain the STUN ORIGIN field which the firewall can use to make an authorization decision on the application.

4. Firewall Processing

The firewall processing is broken into three stages: recognizing STUN packets, making a policy decision as to whether each STUN packet should trigger a pinhole to be created, and managing the lifetime of any pinholes that are created.

4.1. Recognizing STUN packets

STUN messages all have a magic cookie value of 0x2112A442 in the 4th to 8th byte. This can be used to quickly filter nearly all UDP packets that are not STUN packets. Many firewalls are capable of doing this in hardware. STUN supports an optional FINGERPRINT attribute that provides a 32 bit CRC over the message.

Firewalls SHOULD look at outbound UDP packets and if they have the correct magic cookie they can classify them as STUN packets. Firewalls that desire fewer false positives MAY also check that the FINGERPRINT attribute is correct.

4.2. Policy decision

Once the firewall has received a STUN packet from inside the firewall, it needs to decide if the packet is acceptable. For most situations the firewall SHOULD accept all outbound STUN packets. This is similar to allowing all outbound TCP flows. Some firewalls may choose to look at other factors including the outside UDP port and the ORIGIN attribute in the STUN packet.

In general WebRTC media can be sent on a wide range of UDP ports but the two ports that are commonly used are the the RTP port (5004) and TURN port (3478). Some firewalls MAY choose to only allow flows

where the destination port on the outside of the firewall is one of these.

The STUN ORIGIN attribute [I-D.ietf-tram-stun-origin] carries the origin of the web page that caused the various STUN requests. So for example, if a browser was on a page such as example.com and that page used the WebRTC calls to set up a connection, the STUN request's ORIGIN attribute would include example.com. This allows the firewall to see the web application (in this case, example.com) that is requesting the pinhole be opened. The firewall MAY have a white list or black list for domains in STUN ORIGIN.

4.3. Creating the pinhole rules

Once a STUN packet is accepted, the firewall MUST create a temporary rule that allows incoming and outgoing packets for that 5-tuple for at least 5 seconds. If in that 5 seconds, a response is received to the STUN request, the lifetime of the rule must be extended to at least 30 seconds from last accepted STUN packet from inside the firewall. Once the rule has been extended to 30 seconds the first time, any additional UDP packets from inside the firewall MUST extend the lifetime of the rule by at least 30 seconds from the time that packet was received. The procedures in [I-D.ietf-rtcweb-stun-consent-freshness] will ensure that an outbound packet is sent at least every 30 seconds.

4.4. Tracking media vs data

WebRTC can send audio and video as well as carry a data channel. Confidential data could leave an enterprise by a video camera being pointed at a document, but IT departments are often more concerned about the data channel. It is easy for the firewall to separately track the amount of RTP media and non-media data for each WebRTC flow. If the first byte of the UDP message is 23, it is non-media data; if it is in the range 127 to 192 it is audio or video data. More information about this can be found in [I-D.ietf-avtcore-rfc5764-mux-fixes]. Network management systems on the firewall can track these two separately which can help identify unusual usage.

5. WebRTC Browsers

This specification would require browsers to include the FINGERPRINT and ORIGIN attributes in STUN for this to work correctly.

Open Issue: Does adding the ORIGIN reduce user privacy? Consider the following case. The user goes to https://facebook.com and initiates a call with another Facebook user. The domain facebook.com will

appear (unencrypted) in the STUN packets sent from the browser to Facebook's TURN server. Anyone along the network path could tell that the user is using Facebook's TURN server. However, when the original TLS connection for the HTTP was made, the Server Name Indication (SNI) in the TLS of the HTTPS connection also revealed facebook.com, largely for the same reasons - so that the firewall would be able to see which applications are using the network.

6. Blocking Media Hiding in HTTP

The IETF is designing systems to send interactive audio and video such that it looks like HTTPS and HTTP to the proxies and firewalls. The reasons for doing this is that sometimes the proxies and firewalls allow this to work when the mechanisms and channels designed for sending audio and video data have been explicitly disabled by the firewall administrators. Many firewall administrators feel this circumvents the policy they are trying to enforce and desire a way to prevent this. Any scheme for preventing this has some risk of impacting normal HTTP traffic, so there is a desire to provide guidance around ways to do that in this draft.

Any HTTP or HTTPS connection that sends more than 10 requests per second for longer than 10 seconds should be paused for 1 second, and any HTTP/S requests from that client's IP address in the 1 second pause time should be buffered or simply dropped. This strategy ensures there is no impact to clients other than the one exceeding the rate limit and minimizes the impact to other applications on the device while still reducing the incentive to try and run calls this way.

7. Deployment Advice

7.1. WebRTC Servers

WebRTC media servers and TURN servers with public IP address(es) that can receive incoming packets from anywhere on the Internet are suggested to listen for UDP on ports 53 (DNS), 123 (NTP), and 5004 for RTP media servers and 3478 for TURN servers. UDP destined for port 53 or 123 if often allowed by firewalls that otherwise block UDP.

7.2. Firewall Admins

Often the approach has been to lock down everything, so that all UDP is blocked. This simply causes applications to do things like embed the data in normal looking HTTP or HTTPS requests. Malware and viruses use similar approaches. Just turning off all UDP results in a poor user experience some of the time, which results in users

moving to applications and devices outside the firewall. The IT department loses the visibility into what is going on and can no longer protect its users when their computers become compromised. Allowing things that users want to use to work and monitoring them to detect when things have gone wrong is very valuable.

8. Design Consideration

8.1. Why not just use TCP?

TODO

9. Security Concerns

Enterprises have a range of concerns around WebRTC traffic traversal of the firewall. The major concerns that are raised include:

1. Unlike TCP, UDP does not have a connection where a device inside the firewall has confirmed that it wants to talk to the thing outside.

2. Incoming UDP pinholes allow out of band packets to be spoofed into connecting as there is no equivalent of a TCP sequence number to check.

3. UDP has been used by malware command and control protocols so we block it.

4. We do not want enable ways for data to be exfiltrated outside the firewall with no monitoring.

5. An encrypted data channel in WebRTC can be used to bring malware into the company.

6. An encrypted media or data channel in WebRTC can be used as a command and control channel for malware inside the firewall.

7. An encrypted data channel in WebRTC can be used by an outside attacker to exfiltrate private files from inside the firewall.

10. Alternate Approaches

10.1. Firewall Auth Tokens

[I-D.reddy-rtcweb-stun-auth-fw-traversal] attempts to solve a similar problem by proposing a new comprehension-optional FW-FLOWDATA STUN attribute as part of ICE Connectivity checks enabling the firewall to permit outgoing UDP flows across the firewall. FW-FLOWDATA STUN

provides necessary information, such as lifetime, and candidate information, enabling a firewall to apply the required policy rules. However, [I-D.reddy-rtcweb-stun-auth-fw-traversal] requires establishing shared keys across the firewall(s) and the WebRTC server for successfully verifying the authenticity of the FW-FLOWDATA information. In summary, we believe [I-D.reddy-rtcweb-stun-auth-fw-traversal] to have following short-comings

1. Requiring a tight coupling between the application server (WebRTC server) and firewall(s)

2. Requiring additional efforts for Firewall Admins within an enterprise to distribute and maintain the shared authentication keys needed to generate authentication tags for the FW-FLOWDATA attribute.

3. [I-D.reddy-rtcweb-stun-auth-fw-traversal] doesn’t apply for distributing keys across firewalls in different administrative domains.

10.2. Any Cast Whitelist

Deploying media or TURN servers on a single any-cast IP address also makes it easier for firewall administrators to whitelist the address. Concerns have been raised that two packets sent from the same host to a given any-cast address may get delivered to different servers. This is certainly possible in theory but in practice it does not seem be happen in limited experiments done so far.

11. Acknowledgements

Many thanks to Shaun Cooley and Alissa Cooper.

12. References

12.1. Normative References

[I-D.ietf-avtcore-rfc5764-mux-fixes]
Petit-Huguenin, M. and G. Salgueiro, "Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)", draft-ietf-avtcore-rfc5764-mux-fixes-02 (work in progress), March 2015.

[I-D.ietf-tram-stun-origin]
Johnston, A., Uberti, J., Yoakum, J., and K. Singh, "An Origin Attribute for the STUN Protocol", draft-ietf-tram-stun-origin-05 (work in progress), February 2015.

12.2. Informative References

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-14 (work in progress), June 2015.

[I-D.ietf-rtcweb-stun-consent-freshness]
Perumal, M., Wing, D., R, R., Reddy, T., and M. Thomson, "STUN Usage for Consent Freshness", draft-ietf-rtcweb-stun-consent-freshness-15 (work in progress), June 2015.

[I-D.reddy-rtcweb-stun-auth-fw-traversal]
Reddy, T., Perumal, M., and D. Wing, "STUN Extensions for Authenticated Firewall Traversal", draft-reddy-rtcweb-stun-auth-fw-traversal-00 (work in progress), July 2012.

Authors' Addresses

Cullen Jennings
Cisco

Email: fluffy@iii.ca

Suhas Nandakumar
Cisco

Email: snandaku@cisco.com

Jonathan Rosenberg
Cisco

Email: jdrosen@cisco.com