USING TURN SERVERS AS PROXIES

A PREPRINT

Sean Liao
sean.liao@os3.nl

June 5, 2020

1 Introduction

SOCKS is a widely supported proxy protocol with a client-server model, providing a simple interface useful for NAT and firewall traversal. SOCKS5 [2] brings support for proxying both TCP and UDP.

Traversal Using Relays around NAT (TURN) [3] is also a proxy protocol, extending Session Traversal Utilities for NAT (STUN) [4]. While STUN provides utilities for clients to establish peer to peer connections through NAT and firewalls, this is not always successful, in which case TURN relays can be used to relay data between the clients. Given its primary usecase in audio/video communications such as in WebRTC [1], TURN uses UDP for peer connections. RFC 6062 [6] specifies an extension to TURN to use TCP connections.

Given the nature of proxies, operators of TURN relays need to be careful in the design of their network and in the security policies enforced by the proxy itself. Failure to do so could result in connections being made into internal networks.

For users of other networks, TURN relays run by public entities, in particular those used by videoconferencing software, stand in a privileged position as connections to them are often allowed to pass through both NAT and firewall due to business needs. Using these TURN relays as generic proxies could punch through firewalls for a wider class of applications.

2 Research Question

The goal of this research is to produce a translation layer acting as a SOCKS server and TURN client, handling protocol translations and the TURN protocol's permissions system, offering a way to leverage TURN relays as proxies for a wide range of existing tools, many of which only understand SOCKS.

- How can the translation layer be implemented?
- What are the applications and limitations of the translation layer? As a forwarding proxy and using the TURN relay to make arbitrary connections, and/or as a reverse shell using the TURN relay as a known whitelisted endpoint.
- What can be done by firewalls and TURN relays to prevent abuse of proxying? For operators of TURN relays this means its use as an "open" proxy beyond the original planned scope, typically SIP or WebRTC, and for firewall operators this means its use in bypassing any controls the firewall was intended to enforce.

3 Related Work

From early in its design [5] STUN/TURN was recognized to stand at a critical juncture between networks. The latest RFC [3] expands on the security considerations when running TURN relays. Additionally, both an authentication and permissions system is built into the protocol, as well as recommendations in configuration.

The only notable publicised instance of using TURN relays as a proxy is an April 2020 report by Enable Security outlining misconfiguration of Slack's TURN relays and their internal proxying tool [7].

4 Methodology

coturn is the most popular and widely deployed TURN relay. It is also the most feature complete, supporting both TCP and UDP connections. Testing will be run against a locally hosted instance of coturn.

A cursory search did not find any publicly available libraries in any programming language that implemented TURN-TCP client side code. github.com/pion/turn has been identified as a suitable candidate for implementing TCP support if necessary.

Once a translation layer has been complete, end to end tests can be run to identify issues and try out ideas for detection and control.

5 Ethical Considerations

Development and testing will be run primarily against against a self hosted instance of coturn. For validation, running a proxy relayed through a third party TURN relay should not consume outsized resources compared to their usual load of audio/video traffic. Should any misconfiguration or other vulnerabilities be discovered during validation, responsible disclosure will be handled by the supervisor (Cedric van Bockhaven <CvanBockhaven@deloitte.nl>) at Deloitte.

References

[1] C. Holmberg, S. Hakansson, and G. Eriksson. *Web Real-Time Communication Use Cases and Requirements*. Tech. rep. RFC 7478. Mar. 2015.

[2] M. Leech et al. *SOCKS Protocol Version 5*. Tech. rep. RFC 1928. Mar. 1996.

[3] R. Mahy et al. *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*. Tech. rep. RFC 5766. Apr. 2010.

[4] J. Rosenberg et al. *Session Traversal Utilities for NAT (STUN)*. Tech. rep. RFC 5389. Oct. 2008.

[5] J. Rosenberg et al. *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*. RFC 3489. Mar. 2003.

[6] Ed. S. Perreault and J. Rosenberg. *Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations*. Tech. rep. RFC 6062. Nov. 2010.

[7] Enable Security. *How we abused Slack's TURN servers to gain access to internal services*. 2020. URL: `https://www.rtcsec.com/2020/04/01-slack-webrtc-turn-compromise/` (visited on 06/04/2020).