

NTALG – TCP NAT Traversal with Application-Level Gateways

Matthäus Wander*, Sebastian Holzapfel*, Arno Wacker†, Torben Weis*

*Universität Duisburg-Essen, Bismarckstraße 90, 47057 Duisburg, Germany
{matthaeus.wander|sebastian.holzapfel|torben.weis}@uni-due.de

†Universität Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
arno.wacker@uni-kassel.de

Abstract—Consumer computers or home communication devices are usually connected to the Internet via a Network Address Translation (NAT) router. This imposes restrictions for networking applications that require inbound connections. Existing solutions for NAT traversal can remedy the restrictions, but still there is a fraction of home users which lack support of it, especially when it comes to TCP. We present a framework for traversing NAT routers by exploiting their built-in FTP and IRC application-level gateways (ALG) for arbitrary TCP-based applications. While this does not work in every scenario, it significantly improves the success chance without requiring any user interaction at all. To demonstrate the framework, we show a small test setup with laptop computers and home NAT routers.

I. INTRODUCTION

Network Address Translation (NAT) works well for outbound connections for client-server applications, but it hinders inbound connections for peer-to-peer or voice-over-ip applications. The de-facto standard for traversing a NAT is *Universal Plug and Play* (UPnP). It is supported by many NAT routers, but not as widely available to solely rely on UPnP. In a test out of 40 Internet home users in Germany 4 users had UPnP enabled and working for NAT traversal [1]. A test of 2700 Internet BitTorrent users showed that despite the existence of UPnP and manual port forwarding about 40% of the users suffered from NAT connection restrictions [2].

Hole punching is a NAT traversal technique that works without user configuration. It has proven to work well for UDP-based applications [3], and there are also some TCP-based approaches, e.g. [3], [4], [5], [6]. The known TCP hole punching mechanisms, however, impose requirements tough to be met in practice, e.g. capture raw packets or send packets with forged headers. Our goal is a simple NAT traversal mechanism for TCP that (1) does not require user configuration, (2) uses plain sockets as offered by the operating system, and (3) does not rely on spoofing, sniffing or administrator privileges.

II. APPLICATION-LEVEL GATEWAYS

Our approach is based on application-level gateways (ALG) that are built-in to basically all home NAT routers. An ALG is a NAT router component that looks for IP addresses and port numbers in particular types of application traffic and creates public port mappings. A prominent example is the *File Transfer Protocol* (FTP), a legacy protocol created before

NATs emerged. FTP in *active mode* would not work with a client behind a NAT router, because the FTP server needs to create inbound connections to the client. An FTP ALG solves this by looking for the *PORT* command which contains the users' private IP address and port number, opens a port mapping and rewrites the *PORT* command to contain the publicly reachable IP endpoint. The FTP server connects back to the public IP endpoint which is forwarded by the NAT router to the users' device. While an ALG is made to support a specific type of application, we will show that it is possible to exploit specific types of ALGs for generic TCP NAT traversal.

III. NTALG FRAMEWORK

Our *NAT Traversal with Application-Level Gateways* framework (NTALG) encapsulates the TCP connection establishment and is meant to be run as part of a peer-to-peer application. In a peer-to-peer network there is either a server or there are peers with unrestricted connectivity that serve as an entry point into the network. Such a server or peer does not only aid in bootstrapping the peer-to-peer network, it can also act as message relay between two peers. We call it a *rendezvous node* and use it in our framework as mediator of two NAT peers trying to connect to each other.

After a peer has joined the network, its public IP address and port number is known to the rendezvous node and can be distributed in the network. In case port forwarding is configured correctly, this peer is fully accessible at this point and further NAT traversal techniques are not necessary. This is similar to when the peer's NAT router uses an endpoint independent mapping and filtering behavior (classification according to [7]) respectively a full cone NAT behavior (STUN classification [8]): other peers can now establish inbound connections without further actions. When a direct connection to a peer fails, NTALG attempts NAT traversal by using an FTP ALG and an IRC ALG.

A. FTP ALG

NTALG connects to an FTP server emulation on a rendezvous node and mimics a simple FTP control session. The peer opens a new local listening TCP socket and sends the *PORT* command with the local port number. The FTP ALG on the peer's NAT router opens a mapping and forwards the

altered *PORT* command. The emulated FTP server reads the *PORT* command, but instead of connecting back it sends the peer's public IP address and port number back via the FTP control connection. Now any other peer can attempt to connect to the public IP endpoint opened by the FTP ALG. This does not work in all cases because some FTP ALGs expect the FTP server to connect back and filter connection attempts from other IP addresses. Thus, NTALG in parallel attempts NAT traversal with an IRC ALG.

B. IRC ALG

Similar to the above, the framework connects to an IRC server emulation on a rendezvous node and mimics the IRC protocol. It opens a new local listening TCP socket and sends a *Direct Client-to-Client (DCC SEND)* request. A *DCC SEND* request is an IRC chat message with a particular data format called *Client-To-Client Protocol (CTCP)*. It contains, besides an arbitrary file name and size, the local IP address and port number which are rewritten by an IRC ALG. The port mapping is now open and can be connected to by another peer. IRC ALGs are not as common as FTP ALGs, but unlike FTP, an IRC ALG does not know the expected remote IP address and thus can not filter inbound connection attempts.

IV. DEMONSTRATION

We have evaluated the NTALG framework in our lab with 15 off-the-shelf NAT routers in their factory default configurations. 4 out of 15 NAT routers allow inbound TCP connections after setting up an outbound connection on that port number, i.e. with trivial NAT traversal. Given the fact that peers can switch roles during connection establishment (*reversal*), only one of both needs to allow an inbound connection. Thus the success probability that *not* both NAT routers block inbound connections is $1 - \frac{11}{15} \cdot \frac{10}{14} \approx 48\%$. With NTALG we can establish inbound connections on 8 out of 15 NAT routers which corresponds to an overall success probability of $1 - \frac{7}{15} \cdot \frac{6}{14} = 80\%$ in our setup. We will demonstrate the NTALG framework along with a GUI that visualizes the underlying connection procedures (Fig. 1). Our test setup will consist of three laptop computers and select NAT router devices (cf. our lab setup in Fig. 2). Two hosts behind NAT routers attempt to connect to each other by using a rendezvous node.

V. CONCLUSION

NTALG is a framework for TCP NAT traversal which exploits ALG capabilities of NAT routers. We use FTP ALG and IRC ALG in parallel, as this proved to be most effective. The software is implemented in C# .NET, though it basically works in any programming language that supports socket. It does not require user configuration, administrator privileges or a priori knowledge of the involved NAT routers. As plain sockets are used, an established connection provides the service quality of the operating system's TCP implementation, unlike e.g. custom reliable UDP or application-level tunneling implementations. NTALG does not work with all NAT routers but yields a decent success probability with no user constraints.

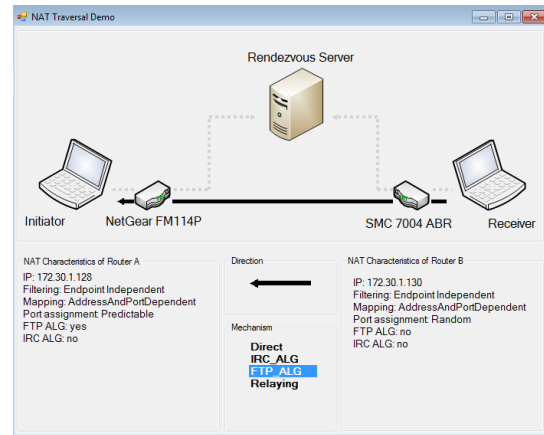


Fig. 1. NTALG framework visualization



Fig. 2. Experimental lab setup

REFERENCES

- [1] S. Holzappel, M. Wander, A. Wacker, L. Schwittmann, and T. Weis, "A New Protocol to Determine the NAT Characteristics of a Host," in *Proceedings of 25th IEEE International Parallel Distributed Processing Symposium, International Workshop on Hot Topics in Peer-to-Peer Systems (HOT2P)*, Anchorage, Alaska, USA, May 2011.
- [2] K. Jünemann, P. Andelfinger, and H. Hartenstein, "Towards a Basic DHT Service: Analyzing Network Characteristics of a Widely Deployed DHT," in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, 31 2011-aug. 4 2011, pp. 1–7.
- [3] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators," in *USENIX Annual Technical Conference*, 2005, pp. 179–192.
- [4] S. Guha, Y. Takeda, and P. Francis, "NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity," in *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. New York, NY, USA: ACM, 2004, pp. 43–48.
- [5] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig, "NATBLASTER: Establishing TCP connections between hosts behind NATs," in *Proceedings of ACM SIGCOMM ASIA Workshop*, Apr. 2005.
- [6] S. Holzappel, M. Wander, A. Wacker, and T. Weis, "SYNI – TCP Hole Punching Based on SYN Injection," in *Proceedings of the The 10th IEEE International Symposium on Network Computing and Applications, IEEE NCA11*, Cambridge, MA, USA, August 2011, (to appear).
- [7] S. Guha and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls," *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005.
- [8] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," IETF, RFC 3489, 2003. [Online]. Available: <http://tools.ietf.org/html/rfc3489>