POSTER: A Secure, Practical & Safe Packet Spoofing Service

Will Scott
NYUAD
willscott@nyu.edu

ABSTRACT

SP^3 presents the design and implementation of a service to allow clients to send themselves a limited amount of network traffic from an arbitrary source IP address. Packet Spoofing is a powerful tool, although often misused, and has the potential to establish TCP connections between clients located behind NATs, to learn about network firewall policies, and to obscure communication patterns by separating source and destination. SP^3 is the first system to offer this capability as a service, while implementing safeguards to prevent malicious users from attacking others. This poster presents the design of SP^3.

1. INTRODUCTION

Internet traffic is sent with a "from" address, the source IP used to let the recipient know where to respond and where the message is coming from. Today, a variety of on line DDOS attacks continue to occur when the assumption that the source address is truthful are violated. Packet spoofing, while most closely associated with these attacks, has legitimate uses as well. Many of these remain under-explored due to routing restrictions limiting and discouraging the practice. Today, under a quarter of IPv4 source addresses, and a smaller fraction of end hosts, are able to transmit packets with forged sender addresses. [Beverly and Bauer 2005]

To enable more exploration of valid uses of Packet spoofing, we introduce SP^3, a system designed to allow any host to opt-in to receiving spoofed packets. To provide this service, we focus on ensuring that SP^3 cannot be misused as an additional source of DDOS or malicious traffic, while still providing a general purpose protocol that can be used for more than just our own research. To this end, SP^3 uses proof-of-ownership verification where clients must prove their liveness and ability to send and receive packets at an IP address before spoofed packets will be sent to them.

Designing a system offering to spoof packets for others is tough for two major reasons. First, limiting the potential for misuse is at odds with running an open proxy of any sort, and it is difficult to define a line for what type of traffic is beneficial versus detrimental. Second, it is unclear how such a system could realistically scale, and how available bandwidth can be effectively shared between clients. Many network administrators are inherently opposed to packet spoofing as a technique in general, making it important to design a protocol where no individual can cause the service to loose effectiveness for others.

Asia CCS '17 April 02–06, 2017, Abu Dhabi, United Arab Emirates

ACM ISBN 978-1-4503-4944-4/17/04.

DOI: 10.1145/3052973.3055155

The contributions described in the rest of this paper are:

- The design and prototype implementation of SP^3, a system allowing any host to *receive* spoofed packets.
- A protocol for remote attestation of IP ownership - proving to a third party that your host is accessible at a given IP address without direct communication.
- A preliminary exploration of novel protocols that can be designed with the ability to spoof packets on a limited basis.

2. BACKGROUND

Packet spoofing is not a new technique, though it has received relatively little study in recent years. A significant amount of research on spoofing occurs in the context of malicious traffic, where spoofing can be used as a mechanism to reflect amplified amounts of traffic towards a victim. Research continues to look at how to detect if incoming traffic is spoofed, so that these forms of attacks can be mitigated [Jin et al. 2003, Duan et al. 2008]. These techniques rely on the network not wanting spoofed traffic, and become much less effective if the spoofed traffic is created in-tandem with a host within the network. When traffic is requested by a host on the network, the IP fields can be filled such that TTL and sequence number values appear much more in-line with an existing communication stream and become much harder for the network to observe as anomalous.

There are a couple exciting legitimate uses of packet spoofing that have been described and prototyped, but none of them have received widespread adoption or supported general use an experimentation by normal users. NUTSS [Guha et al. 2004] presents a design for establishing a TCP connection between two participating hosts both trapped behind NAT devices, using packet spoofing to trick the routers into each believing that their host initiated the connection. Spooky Scan [Ensafi et al. 2014] shows how packet spoofing can be used to learn connectivity between two remote computers without needing to control them, providing an alternative Internet measurement technique.

The technique we describe for asserting control of an IP address in 3.1 has also been approached before. The "Accountable Internet Protocol" presents a network level design where addresses are self certifying, making it easy for a host to prove indirectly who and where it was [Andersen et al. 2008]. The need for proving IP address is also used in variants of bitcoin which provide routing, or proof-of-bandwidth [Ghosh et al. 2014]. We believe our approach is unique in providing a transitive proof-of-presence without revealing a network-noticeable signature.

3. SYSTEM DESIGN

The SP^3system consists of three participants: a server, client, and sender. The server is the host which can send

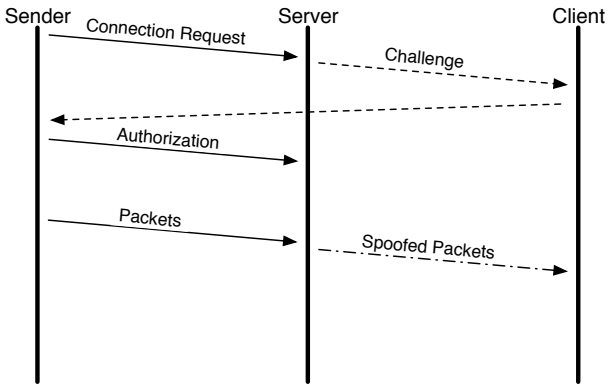

Figure 1: The SP^3 protocol. Dashed lines indicate the consent process, and represent the logical communication of a challenge, rather than explicit packets sent between those entities.

spoofed packets. It acts as a relay, accepting encapsulated IP packets from the sender and sending them to the client, even when their source address is spoofed. The client is the destination that receives the packets. The sender is the host that generates the packets. While the client and sender may be the same host, they don't have to be, and some applications are based on them being logically distinct.

A primary consideration in the design of SP^3 is safety. In order to provide a service that makes a reasonable trade-off between enabling valid use cases for packet spoofing while not opening itself up to abuse and attacks, the SP^3server enforces a policy on packets it is willing to spoof. The primary property we choose to guarantee in SP^3 is that the server will only send packets to a client that wants to receive them. In 3.1, we describe how we define "want", and the protocol the server uses to establish consent from the client.

The design of the SP^3 protocol is shown in Figure 1. The protocol is structured similarly to other proxy protocols, like SOCKS or HTTP connect, with an initial negotiation preamble which concludes with the connection becoming a channel over which to relay packets. Rather than operating at a TCP level, SP^3 messages are sent over a Web Socket connection, allowing the sender to be implemented within a web page, and taking advantage of an existing standard for message integrity and confidentiality. Because there is no response expected to spoofed packets, the response channel from the SP^3 server to the sender can safely continue be used to send status messages about the status of the connection, remaining quota, and errors.

3.1 Client Consent

The SP^3 protocol includes the notion of client consent. We use this term to mean that the client wishes and can receive spoofed packets, and guarantee it by ensuring the the client is able to prove that it can receive a piece of data that can only be accessed by the IP address it claims to be at. This form of authentication defines a *proof-of-ownership*, that the client really is at the IP address it claims to be, therefor mitigating the potential for denial of service or other attacks on unwary hosts. The specific properties we enforce in protocols to prove ownership are:

- **Active engagement.** The client should be running at the time of transfer. It should not be possible to launch replay attacks, and traffic should not be able to continue once the client has disconnected.
- **DDOS resistance.** The server should not be usable to overwhelm the remote network.
- **Locality.** The Client must be able to prove it can receive traffic from the destination address.

There are a number of mechanisms by which the client can provide this consent. The simplest is that the client establishes a connection to the server, and directly communicates to the server that it would like to receive traffic. This approach works for many use cases, such as when performing NAT hole-punching, or when a host attempts to learn routing policies of its public interfaces. One use case we want to enable which is not addressed by direct connection is when SP^3 is used as part of a protocol for circumventing censorship, since a direct connection to a SP^3 server would become a single point of failure that could disrupt such a protocol. To this end, we also support indirect forms of confirmation that do not require easily discriminable communication on behalf of the client.

We next describe two mechanisms which meet these criteria without a direct connection between the client and SP^3 server.

3.1.1 STUN Candidate injection

One existing protocol that can be used to transmit a challenge code indirectly is the original STUN protocol. STUN, Session Traversal Utilities for NAT, is a standardized UDP protocol providing a mechanism for clients to automatically learn their publicly facing IP address, and facilitate session initiations. In the original specification of STUN, clients may ask that the response to their query be sent to an alternative IP address, in order to learn more about the NAT device they are behind. In SP^3 , we can use this same mechanism to send a STUN message from the subset of STUN servers which support the first version of the protocol to the IP address and port specified by the client.

In subsequent version of the STUN protocol, the server will only respond to the IP address it receives packets from. SP^3 can also use these servers for candidate injection by sending a spoofed packet to the STUN server claiming to be from the client IP address.

STUN is an ideal protocol for IP address authentication for three reasons. First, STUN is a session initiation protocol for learning IP addresses, which is the purpose of our mechanism, and causes the protocol to easily adapt to our use case. Second, the protocol is structured such that a client query may result in multiple responses in the process of session initiation, meaning that an additional response triggered by SP^3 will not appear abnormal to a network monitor. Third, the mechanism doesn't involve either need to race packets, or a custom client, and allows the authorization challenge to be learned indirectly by a web browser client.

3.1.2 HTTP Reflection

Triggering additional data within an HTTP or HTTPS stream offers an alternative mechanism for authorizing clients. While the use of STUN has many attractive properties, it

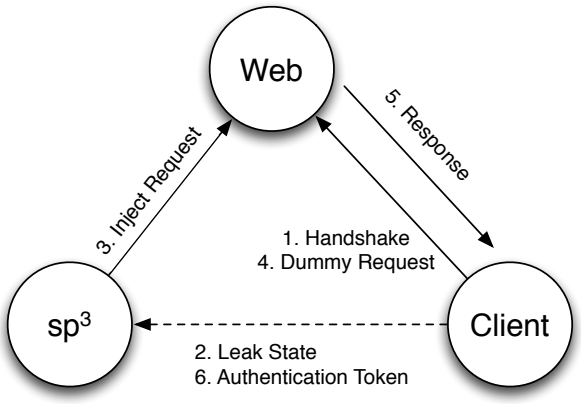

Figure 2: Consent protocol using HTTP reflection. Communication between the client and SP^3 server are indirected through the SP^3 sender. The challenge token used to verify the client is passed in steps 3, 5, and 6. Step 4 is optional, and is sent unilaterally by the client so that the response is not unexpected by the network following the outline in NUTTs [Guha et al. 2004].

may be undesirable because the protocol stands out[1], or because the messages cannot be encrypted. While the challenge is transmitted as a random nonce, it leaves the method open to active probing attacks. the use of Web protocols can overcome this limitations.

The design of our basic HTTP authorization mechanism is shown in Figure 2. In this design, the client picks a trusted, single-homed web server, and initiates a connection to it. It then uses the out-of-band protocol it has with the packet sender to relay the connection state, the TCP sequence number, and potentially the SSL key agreed upon. The SP^3 server uses this information to send an HTTP request to the web server, spoofed to appear to be on the existing connection. The HTTP request will have the authorization challenge in the request path, structured such that the response, sent to the client, will also include the challenge code.

This mechanism is more complex. It requires agreement on an appropriate candidate web server, which can be learned from [Scott et al. 2016]. It requires knowledge of TCP header information, which means that the client will need to have root permissions on the machine. It also requires packet spoofing by the server, which must be strictly rate-limited to prevent abuse towards web servers. However, encrypted web requests are ubiquitous, and are very difficult to discriminate against.

4. APPLICATIONS

The goal of SP^3 is to democratize access to a limited form of packet spoofing, so that general clients can automatically participate in protocols which require spoofed packets. The premise of such a system is that there are interesting or novel protocols of that type. We finish this paper with a description of an initial set of protocols which are both valuable, require the use of spoofed packets, and fit within the security framework described in the previous section.

NAT hole-punching Currently, NAT hole-punching only works for UDP, partially because even when the clients are controlled, it generally requires root permissions to send packets with a specific sequence number. Having a source of packet injection can provide a mechanism to synchronize sequence numbers and create TCP connections between two NAT'ed machines.

Firewall characterization It's often difficult to test how your network will respond to packets sent from black-holed or unadvertised prefixes. A source of spoofed packets allows you to validate firewall rules and routing policy.

Communication stenography The ability to send packets from arbitrary sources can help to mask traffic by adding a layer of cover traffic and IP diversity that makes surveillance much more difficult.

5. CONCLUSION

This paper has described the SP^3 system, a protocol and design for allowing more clients on-line to participate in communication using spoofed packets. Packet spoofing remains an important tool for digital attackers, and for challenging security assumptions in protocol designs and network security systems. Code for a working prototype of the system is available at github.com/willscott/sp3

6. REFERENCES

[Andersen et al. 2008] David G Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. 2008. Accountable internet protocol (aip). In *SIGCOMM CCR*. ACM.

[Beverly and Bauer 2005] Robert Beverly and Steven Bauer. 2005. The Spoofer project: Inferring the extent of source address filtering on the Internet. In *SRUTI*, Vol. 5. USENIX.

[Duan et al. 2008] Zhenhai Duan, Xin Yuan, and Jaideep Chandrashekar. 2008. Controlling IP spoofing through interdomain packet filters. *Dependable and Secure computing* 5, 1 (2008).

[Ensafi et al. 2014] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R Crandall. 2014. Detecting intentional packet drops on the Internet via TCP/IP side channels. In *PAM*. Springer.

[Ghosh et al. 2014] Mainak Ghosh, Miles Richardson, Bryan Ford, and Rob Jansen. 2014. *A TorPath to TorCoin: proof-of-bandwidth altcoins for compensating relays*. Technical Report. DTIC Document.

[Guha et al. 2004] Saikat Guha, Yutaka Takeda, and Paul Francis. 2004. NUTSS: A SIP-based approach to UDP and TCP network connectivity. In *SIGCOMM FDNA*. ACM.

[Jin et al. 2003] Cheng Jin, Haining Wang, and Kang G Shin. 2003. Hop-count filtering: an effective defense against spoofed DDoS traffic. In *CCS*. ACM.

[Scott et al. 2016] Will Scott, Thomas Anderson, Tadayoshi Kohno, and Arvind Krishnamurthy. 2016. Satellite: Joint analysis of CDNs and network-level interference. In *USENIX ATC*.

[1]there have been reports of networks where all UDP traffic is blocked.