CS4516: Project Proposal

Tag-Based IP Spoofing Prevention Under Low Deployment Scenarios

Author:
Daniel Robertson
Michael Calder

Supervisor:
Dr. Craig Shue

February 27, 2014

1 Executive Summary

The current approach to routing over the Internet Protocol enables an unfortunate opportunity for serious exploit. Packets are routed based off of destination prefix alone, and leave the source address field to only be considered by end hosts. Inherently, this means that an arbitrary source address could be used in place of a sender's true source address. Denial of Service attacks leverage this weakness by issuing dozens of requests using forged source addresses, so that a defending server can't know which requests can be ignored, and which are legitimate. Numerous solutions to mitigating DoS attacks exist, but they fail to guarantee absolute prevention. An absolute preventative solution would require the underlying Internet to conspire against IP spoofing, rather then source or end hosts. Shue et. al. present an elegant solution to the underlying problem, and show that it prevents a significant amount of spoofing attempts near the source, even under partial deployment [5].

In their approach, implementing routers within a prefix range "mark" packets originating from within their domains with a 48 bit tag. This acts as the router's signature and verification that the packet's source address is indeed within the router's prefix range. Upstream implementing routers verify these tags, strip them, and add their own.

2 Need

Under full deployment, their analysis shows that this approach is effective in curbing IP spoofing attacks. However, there are notable pitfalls under partial and low deployment scenarios. As noted by Shue et al., both data and control plane modifications must be made to implementing routers. While no hardware modifications are required, the cost of changing router functionality, especially in core routers, is likely to deter deployment speeds. Tag theft can compromise the effectiveness of IP spoofing prevention.

The implementation detailed in [5] uses randomly generated (unhashed and unencrypted) fixed length tags. The verification process was designed to be fast, but introduces security concerns when tags are stolen. These security concerns bring about the need our work will address. Partial deployment scenarios (espe-

cially 10 to 30 percent) can have up to 65 percent of stolen tags be exploitable. The risk may cause hesitation in adoption. A way to prevent this issue may be to treat the tags as a secret, and use them to generate a safer token that can be sent over the wire and verified with the tag on the receiving end.

3 Approach

We present two possible avenues for research and analysis that could improve security for the packet tagging protocol presented by Shue et al., and help further incentivize early adoption. We aim to provide a stronger mechanism for protecting the underlying tags, such that the possibility and impact of tag theft are short reached and effectively negligible.

Our first solution is to render stolen tags ineffective for further use. In this case, "stolen" refers to a tag that has not been leaked by a compromised router, but rather one that was intercepted during transit, or by an end host. We continue the assumption made by Shue et al. where any number of hosts may be malicious, but implementing routers are generally trustworthy.

We propose a mechanism inspired by the Time-based One Time Password (TOTP) algorithm, detailed in [4]. TOTP uses fixed length intervals of unix time (usually 30 seconds) to generate a key that only remains valid for the duration of the interval. We aim to apply this time-based mechanism to the creation and verification of tokens, derived from a router's base tag. The interval used controls the duration of time that an intercepted tag-token could be exploited. Necessarily, it should be difficult to derive the base tag from an intercepted token. We will use a variety of cryptographic and non-cryptographic hashing algorithms in order to generate these tokens, and analyze their overhead and effectiveness. We will measure the speed at which the hashing can be performed and the space needed to store the hashes.

Second, we plan to further secure the underlying tags using an evolving hash chain. Shue et al. suggest this as an appropriate means for changing a router's tag over time, but give no empirical or practical analysis of the effectiveness of such an alteration. We will follow the hash-chain generating approach given by Hu et al. [3] in constructing the chains, and test its resilience when compromised. We also aim to qualify differing time intervals to advance the chain by. Finally, the

TOTP-inspired approach will be compared to the chaining method by discussing what would be required to exploit a stolen hash under each method. The two are very different because the latter involves used precomputed hashes of hashes one day at a time, while the former computes hashes on the fly based on the (rounded off) current time.

Our implementation and analysis of this extended protocol will focus more heavily on tag exchange within a low deployment topology, where the assumption is made that tags have already been distributed and learned by deploying routers. As our proposal aims to add security mechanisms to an initialized network, we will not implement any of the tag discovery/exchange mechanisms detailed in [5]. Instead, our implementation will emulate an arbitrary exchange between two end hosts across a network whose deploying routers have fully built tag tables. The focus is thus on the overhead of differing token generating mechanism to the existing protocol, and the effectiveness of the added security. We will design a set of evaluation metrics to use for testing the strength of generated tokens, should they be stolen. This will likely include pre-image and collision resistance.

We will use NS-3 to create and test our protocol. We plan to spend more time in its native C++ library, using the python bindings for high level simulations. NS-3 provides an Ipv4RoutingProtocol interface which exposes two primary methods, RouteInput and RouteOutput. We will extend these functions to preform the necessary tag inspection logic, passing or dropping the packets accordingly. Conveniently, NS-3's packet implementation provides an API for attaching attributes to packets that persist through a packets lifespan, called Tags. Implementing our protocols tags as a Tag allows for an easy implementation of the protocol.

4 Benefits

The IP spoofing prevention methods provided by Shue et al. present a very strong case for adoption. Unfortunately, the security of their protocol performs poorly under low deployment, giving network operators less of an incentive to roll out their implementation. This lack of performance is discussed in section IV-C of [5] and the authors explain how (especially under deployments under 30 percent) up to 65 percent of tags can be both stolen, making 40 percent vulnerable to exploitation.

Our proposed extension aims to curb the possibility of tag spoofing in par-

tial deployment topologies. Added security assurances would further incentivize adoption, assuming that the resulting time and space overhead doesn't affect the performance of the original protocol. Increasing the out-of-the-box resilience of the protocol, especially while it is being rolled out, makes a better case for integration into current networks.

5 Competition

Many approaches to preventing inter-domain IP spoofing (where attackers attempt to spoof IP addresses outside of their personal domain), already exist. Most prominent are those which attempt to trace the route of the attacking packet back to the (true) source, such as [1]. Shue et. al's approach follows more along the lines of packet filtering, ie. dropping packets near or around the source. Research such as [2] perform ingress filtering, which drop spoofed packets from within the originating network, but fail to cover packets that escape into the wild.

But the main competition to our proposed extended protocol is the existing tag-based protocol. While our enhancements provide better security guarantees, especially under partial deployment (an area where these guarantees are lacking), if the overhead proves to be too significant then the original protocol may be favored. In addition, given the case where the entire Internet conspires to implement the original protocol, the issues coming form low deployment would be bypassed, and our extended protocol would not be necessary.

References

[1] S. Bellovin, M. Leech, and T. Taylor. Icmp traceback messages. *IETF Draft*, October 2001.

[2] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice), May 2000. Updated by RFC 3704.

[3] Yih-Chun Hu, Markus Jakobsson, and Adrian Perrig. Efficient constructions for one-way hash chains. In John Ioannidis, Angelos Keromytis, and Moti Yung,

editors, *Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 423–441. Springer Berlin Heidelberg, 2005.

[4] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. TOTP: Time-Based One-Time Password Algorithm. RFC 6238 (Informational), May 2011.

[5] Craig A. Shue, Minaxi Gupta, and Matthew P. Davy. Packet forwarding with source verification. *Computer Networks*, 52(8):1567 – 1582, 2008.