MiddleAuth: Authenticating the Sources in the Middle of the Internet

Zhuotao Liu (zliu48@illinois.edu)

Abstract

The Internet-wide source spoofing may cause a variety of security problems such as the lack of accountability and DDoS attacks. Despite over a decades of research, little progress has been made on real world deployment. This project proposes MiddleAuth, an immediately deployable source validation approach for the HTTP/HTTPs traffic in the Internet. The following two features ensure MiddleAuth's deployability. First, to perform source validation, MiddleAuth relies only on deployment from commercially related parties (*e.g.,* cloud providers) without demanding further deployment at unrelated Autonomous Systems (ASes). Second, no upgrades at clients are ever required. The project discusses the security properties that MiddleAuth can offer. Finally, we include an implementation prototype of MiddleAuth.

1 Motivation

The current Internet does not implement source validation mechanisms, allowing a source to claim arbitrary addresses. Despite over a decade of research, none of the proposed approaches have been fully deployed, making the Internet-wide spoofing still possible [5]. The effectiveness of approaches such as Ingress Filter (*i.e.,* BCP 38) [7] depends on *universal deployment* from the Autonomous Systems (ASes) in the Internet. Consequently, although we have witnessed significant fraction of deployment in the Internet [1], few misbehaviors can still break the entire system [5].

Passport [8] performs source authentication on the basis of AS: each AS border router stamps non-forgeable signatures for all its sources to prevent malicious ASes from spoofing its source addresses. Although provably correct, Passport is hard to deploy in the Internet. To begin with, it requires cooperation and coordination (*e.g.,* key setup and management) among different ASes. Although inter-domain coordination is possible based on the existing inter-domain relation model (*e.g.,* provider, transit and customer), Passport's inter-domain coordination requires a completely new model for inter-domain relations. For instance, two non-neighbor ASes may need to perform the handshake for key establishment. Further, a new Passport packet header needs to be added, requiring global router upgrades to avoid the comparability problem.

Approaches such as AIP [3] and SCION [12] propose to re-design the Internet so as to improve its security properties, which, arguably, would impose even larger deployment efforts.

2 Design Goals

The major design goal of MiddleAuth is to be deployable in the Internet. We focus on two primary requirements to ensure deployability. (i) Strong properties from limited deployment. The end-to-end basis of Internet communications, combined with large numbers of end points, is what gives rise to its tremendous utility. This scale and diversity of administrative domains, including end points, edge-ASes, and small transit ASes, give rise to a wide variation in the technological sophistication and cooperativeness of various in-network entities. However, some ASes can be expected to help with deployment; many ISPs already provide some sort of DDoS-protection services [2], so we can expect that such providers would be willing to deploy a protocol under commercially reasonable terms. Our goal in MiddleAuth was to design a system that requires only limited deployment, from willing and related parties, and yet achieve strong properties. (ii) No network stack modification at client. Because clients run on a wide diversity of operating systems, including embedded hardware that may be difficult to upgrade, modifying the standard network stack may be difficult to deploy at some clients.

3 Design Scope and Assumptions

Design Scope. MiddleAuth is designed for validating the source IP address of HTTP/HTTPs traffic in the Internet, the majority of user-facing Internet traffic. MiddleAuth's source validation is to prove that one sender is *on-path* of its claimed source address towards a service provider (SP hereafter). The adversaries that are on-path, however, would be able to spoof the sender's address. Note that on-path adversaries could do much more damage than source spoofing, *e.g.,* they could simply drop all packets from the sender. Selecting Internet paths to avoid on-path adversaries is out of the project's scope.

Assumptions. MiddleAuth depends on a set of middleboxes (mboxes hereafter) to perform the source authentication in the middle of the Internet. To only involves commercially related parties for deploying mboxes, the victim can outsource mboxes to the cloud providers [11]. We assume that the set of mboxes are well-connected with the Internet backbone such that denial of service against all of them is prohibited. Such an assumption is the standard assumption when building CDNs. Performing source validation in the cloud allows the mboxes to only forward the requests with valid sources to the SP, saving the SP's downstream bandwidth.

4 System Design

Figure 2 illustrates the architecture of MiddleAuth. A SP that is interested to validate its clients' sources redirects its traffic to mboxes, for instance, by modifying its DNS entries (similar to the approached adopted by CloudFlare). Each mbox performs the source validation on behalf of the SP. Note that different from CloudFlare that all connections are terminated at CloudFlare, mboxes only interpret HTTP connections during the initialization stage (see more details below). Once the source is validated, mboxes will simply serve as relays for the future traffic.

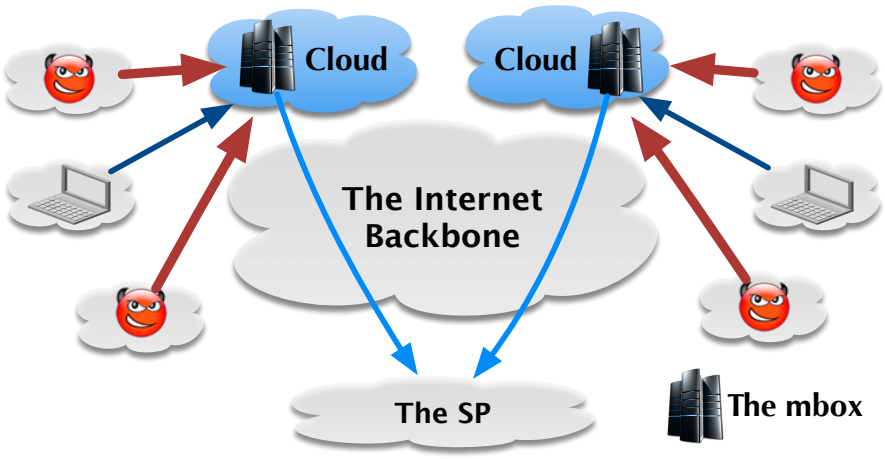


Figure 1. The architecture of MiddleAuth.

5 System Design

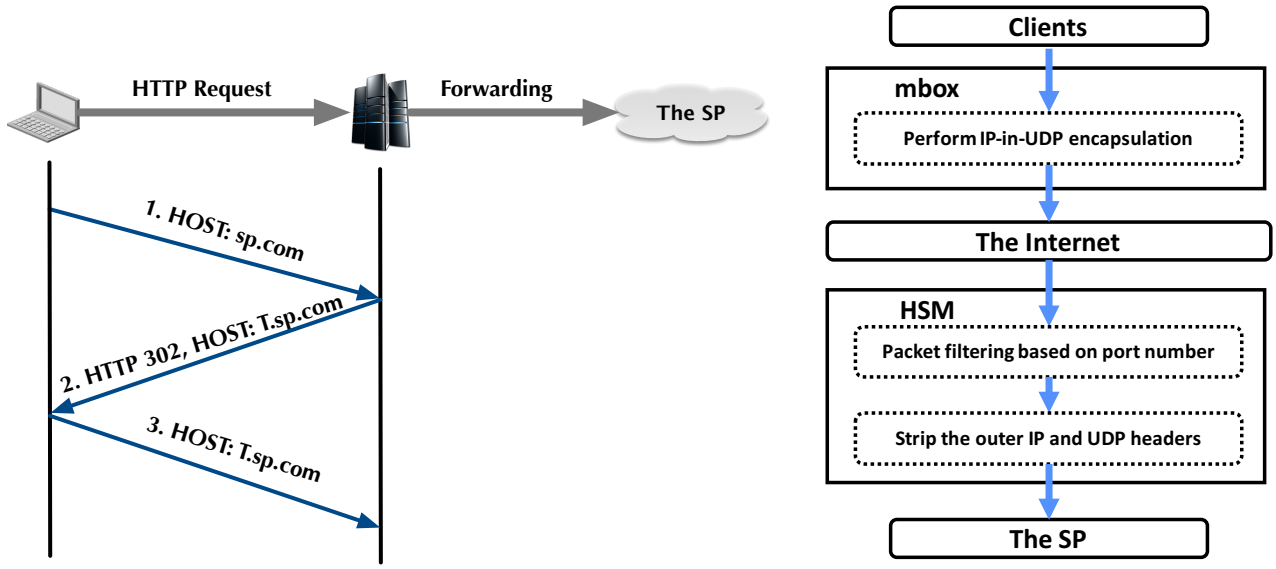
Figure 2(a) illustrates the the design of MiddleAuth. The key insight of the design is that the HTTP Host header is in the first few packets of each HTTP connection. As a result, each mbox, employed by the SP, monitors the TCP requests toward the SP. Upon receiving the new connection, the mbox reads the Host header. If the Host header reflects a generic (not sender-specific) hostname (*e.g.,* sp.com), the mbox intercepts this flow, and redirects the connection (HTTP 302) to a Host name containing a token cryptographically generated based on the sender's claimed source address, *e.g.,* T.sp.com, where T is the token. In particular, the host-specific is computed as follows.

$$T = n \mid\mid t_s \mid\mid H_{K_s}(n \mid\mid t_s \mid\mid s), \tag{1}$$

where n is a 64-bit (or even longer) cryptographic random nonce, t_s is a timestamp, s is the claimed source address, and $H_{K_s}()$ is a keyed cryptographic hash function based on a secure key K_s. As T incorperates a message authentication code, adversaries cannot generate valid tokens without the security key K_s.

If the sender is on-path to its claimed addresses, it will receive the redirection, and its further connection will use the sender-specific hostname generated by the mbox. When an mbox receives a request with a sender-specific Host, it verifies that the Host is proper for the claimed IP source address (if not, the mbox initiates a new redirection), and forwards the request (and any keys established by HTTPS) to the SP. Note that for HTTPs traffic, the mbox needs to obtain valid certificates from the SP.

Token Validation. To begin with, the mbox validates a received token by computing hash and checking whether the computed hash value matches the hash carried in the token. More importantly, the token validation mechanism needs to prevent reuse: *i.e.,* one token can be utilized once for verifying a specific source address. The combination of n and t_s can achieve such a goal. Specifically, a token is not expired only if $t_a - t_s < \mathcal{T}$, where t_a is the arrival time of the token and $\mathcal{T}$ is a pre-defined period. Further, the nonce used in the token is stored by the mbox, associating with its source s and the expiration time point $t_s + \mathcal{T}$, where t_s is the timestamp when the nonce is generated. To bound the storage usage, the mbox can erase



(a) Source validation design.

(b) The design of MiddleAuth+.

Figure 2. Figure 2(a) illustrates the source validation mechanism of MiddleAuth. The mbox redirects the http/https request to a hostname cryptographically generated based on the source's claimed source address. The sender can only obtain the hostname if it is on-path to its claimed source address. Figure 2(b) illustrates the design of MiddleAuth+. The mbox encapsulates all traversing packets with UDP headers and use the UDP source and destination port to carry the secret for packet filtering.

all expired nonces. A nonce is fresh if (i) it is not expired and (ii) it has not been utilized by its associated sender for source validation. To prevent token reuse, the mbox accepts a token T only if (i) T is not expired and (ii) T carries a fresh nonce. Therefore, as a nonce becomes non-fresh once its associated source sends it to the mbox, token reuse by any entities (including the associated source) is prevented.

6 Security Properties enabled by MiddleAuth

Deny spoofed traffic. The most straightforward property enabled by MiddleAuth is to allow the SP to deny traffic with invalid source addresses. Note that even without MiddleAuth, an adversary has to be on-path to its claimed source in order to establish TCP connections. However, the adversary could send "garbage" TCP packets to the SP to consume SP's bandwidth, which will be escalated to network-oriented DDoS attacks once the volume of attack traffic is large. With MiddleAuth, the upstream mboxes, however, are able to throttle the spoofed traffic to save more downstream bandwidth for senders with valid sources.

SP-defined bandwidth sharing. MiddleAuth can enforce self-defined bandwidth sharing policies at mboxes. For instance, with the source validation, the mboxes can enforce per-source fair share of SP's resources, a sharing model proposed in [9]. Argumented by the IP-AS-Mapping [6], the mboxes can further enforce per-AS fair share (a sharing model pro-

posed in [4]) to prevent bot-compromised ASes from taking bandwidth from legitimate ASes. Finally, MiddleAuth can support bandwidth reservation at the mboxes for premium clients or critical Internet traffic, achieving the goals of ARROW [10] without any source and Internet modification.

7 MiddleAuth+

Preventing mbox bypassing. The design of MiddleAuth is subject to mbox bypassing. In particular, adversaries knowing the IP address of the SP can directly send traffic to the SP, bypassing the mboxes. Thus, MiddleAuth+ is proposed to resolve such issue. To ensure immediately deployablibility, MiddleAuth+ invents a mechanism based on the existing ACL configurations on commodity routers. Specifically, each mbox encapsulates its traversing packets into UDP packets (similar techniques have been applied in VXLAN and [**?**]), and uses the UDP source and destination ports (a total of 32 bits) to carry an authenticator, a shared secret between the mbox and the SP. Although a 32-bit secret is not long enough to prevent attackers from generating the secret, a 400Gbps attack (the largest attack viewed by Arbor Networks [**?**]) that uses random port numbers will be reduced to ∼90bps as the chance of a correct guess is 1 over 2^{32}. The shared secret can be negotiated periodically based on a cryptographically secure pseudo-random number generator.

Packet Filtering Points. Once the volume of bypassing traffic is large, simply filtering packets at the SP may not be enough since the upstream network link may be congested. Thus, The filtering mechanism should be deployed at, or upstream of, each bottleneck link. Deployed filtering points should have sufficient bandwidth so that the bypassing attack traffic cannot cause packet losses prior to the filtering. To ensure this, the SP can either purchase sufficient bandwidth from large ISPs and deploy the filtering mechanism at the inbound points of its network. Or it can work with its ISP to deploy the filtering deeper in the ISP's network. Working with the victim's ISP does not violate the deployment model preferred in §2 as MiddleAuth+ never requires deployment from unrelated ASes.

Header Stripping Module. To deliver the original data context to the SP, MiddleAuth+ needs to strip the extra headers added by the mbox. Towards this end, MiddleAuth+ implements a header stripping module (HSM) at the packet filtering points. Figure 2(b) illustrates the design of MiddleAuth+.

8 Implementation

I implement a prototype for both MiddleAuth and MiddleAuth+. For simplicity, in the prototype, the mbox and SP are co-located in the same server. However, in reliability, the mboxes should be outsourced to the cloud. Further, the prototype does not include the HTTPS traffic. All the source code is submitted to the github repository maintained by the course staff.

MiddleAuth's prototype. The implementation of MiddleAuth is straightforward. Whenever the mbox receives a HTTP request carrying generic hostname, it computes the host-specific hostname based on Equation (1) and sends the HTTP 302 redirection with the

generated token.

MiddleAuth+'s prototype. MiddleAuth+'s implementation is based on the NetFilter Linux kernel module. All inbound traffic from clients to an mbox is subject to UDP encapsulation. The HSM is responsible to trim the extra headers to deliver the original payload to the SP's applications. To avoid packet fragmentation due to the additional 28 bytes added by the mbox (20 bytes for outer IP header and 8 bytes for the outer UDP header), the mbox needs to be a priori aware of the MTU M_d on its path to the SP. Then the mbox sets its MSS as no more than $M_d - 28 - 40$ (the MSS is 40 less than the MTU). Note that MiddleAuth does not directly set the MTU of the mbox's NIC to rely on the path MTU discovery to find the right packet size because ISPs may block ICMP packets. In the prototype, as $M_d = 1500$, MiddleAuth sets the mbox's MSS as 1400.

9 Further Direction

Dear Professor:

Given the detailed design and implementation, I hope that it is possible to further take this project into a research paper or something even bigger. Towards that, I really appreciate that if you can help me figure out some practical use cases for MiddleAuth and MiddleAuth+. Further, for any specific case, we can further add some features and implementation to MiddleAuth. For instance, a lot of research papers have proposed to add *network capabilities* into the packets. that allow the SP (even other network entities) to make routing/forwarding decisions. For instance, an SP can defend itself from DDoS attacks by only accepting packets with valid network capabilities. Adding network capabilities into MiddleAuth is actually very straightforward as NetFilter allows us to manipulate the packets. And I have implemented the function to append additional data (will be the network capabilities) into the packets.

I am very thankful if you could come up some practical use cases for MiddleAuth and MiddleAuth+ based on your outstanding security expertise. I am looking forward to your reply.

References

[1] Everyone should be deploying BCP 38! Wait, they are . http://www.senki.org/everyone-should-be-deploying-bcp-38-wait-they-are/, 2011.

[2] AT&T Denial of Service Protection. http://soc.att.com/1IIlUec, Accessed on 2016.

[3] Andersen, D. G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., and Shenker, S. Accountable Internet Protocol (AIP). In *ACM SIGCOMM* (2008).

[4] Basescu, C., Reischuk, R. M., Szalachowski, P., Perrig, A., Zhang, Y., Hsiao, H.-C., Kubota, A., and Urakawa, J. SIBRA: Scalable Internet Bandwidth Reservation Architecture. *NDSS* (2016).

[5] Beverly, R., Koga, R., and Claffy, K. Initial Longitudinal Analysis of IP Source Spoofing Capability on the Internet.

[6] Cymru, T. Ip to asn mapping. http://www.team-cymru.org/IP-ASN-mapping.html, Acccess on 2016.

[7] Ferguson, P., and Senie, D. Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. RFC2827.

[8] Liu, X., Li, A., Yang, X., and Wetherall, D. Passport: Secure and Adoptable Source Authentication. In *NSDI* (2008).

[9] Liu, X., Yang, X., and Xia, Y. NetFence: Preventing Internet Denial of Service from Inside Out. *ACM SIGCOMM* (2011).

[10] Peter, S., Javed, U., Zhang, Q., Woos, D., Anderson, T., and Krishnamurthy, A. One Tunnel is (often) Enough. In *ACM SIGCOMM* (2014).

[11] Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. Making Middleboxes Someone else's Problem: Network Processing as a Cloud Service. *ACM SIGCOMM CCR* (2012).

[12] Zhang, X., Hsiao, H.-C., Hasker, G., Chan, H., Perrig, A., and Andersen, D. G. SCION: Scalability, Control, and Isolation on Next-generation Networks. In *IEEE Symposium on Security and Privacy* (2011).