AN IP-LAYER ANONYMIZING INFRASTRUCTURE

H.T. Kung
Scott Bradner
K-S Tan

Harvard University
Cambridge, MA

ABSTRACT

We describe an IP-layer anonymizing infrastructure, called ANON, which allows server addresses to be hidden from clients and vice versa. ANON uses a network resident set of IP-layer anonymizing forwarders that can forward IP packets with encryption and decryption applied to their source and destination addresses. Using ANON, a client can send and receive packets to and from application servers without knowing their IP addresses. We have designed and implemented a laboratory testbed for this anonymizing infrastructure. This paper gives an overview of the ANON architecture and its implementation, and describes its security threat models and our countermeasures.

INTRODUCTION

Over the current Internet, when a client acquires services from an application server, packets sent and received by the client reveal server IP addresses in the packet headers. There are a number of situations where it would be useful for an application to be able to send traffic to a destination without revealing the IP address of the destination to the source, the IP address of the source to the destination, or both. For example, a Web site may want to hide its IP addresses to reduce the risk of denial of service (DoS) attacks aimed at these addresses. See [9] for discussion on supporting anonymity at the IP layer.

One way to achieve this anonymity, as described in this paper, is to use a network resident set of IP-layer servers that can forward IP packets, with encryption and decryption applied to their source and destination addresses when appropriate. We will call these network resident IP-layer servers anonymizing forwarders, or simply forwarders, and an IP anonymizing infrastructure based on these anonymizing forwarders ANON.

Using ANON, a client can send and receive packets to and from application servers without knowing their IP addresses. This is analogous to a user sending and receiving U.S. mail using the P.O. Box number of an organization without using its street address. In this way, the organization can receive mails while not revealing its street address to the public.

ANON incorporates countermeasures to provide protection against various security threats such as unauthorized monitoring of links in the infrastructure and launching of DoS attacks through the infrastructure. The countermeasures include previously known techniques such as link encryption, link padding, traffic mixing, multi-hop packet encryption/decryption and protocol camouflaging, as well as new techniques such as on-demand link padding and per-destination rate-limiting.

The design of ANON assumes that it will be used mainly for low-bandwidth signaling and data applications, not data transfer that may require high bandwidth. As described later in the paper, this assumption will increase the effectiveness of our countermeasures. There are many applications that fit the model defined here, that is, they only need medium bandwidth to function properly. These include signaling applications such as connection setup and termination, user authentication, user authorization, service registration, and service discovery.

Consider, for example, the use of ANON to protect authentication servers against DoS attacks. By definition, an authentication server needs to process requests from unknown users. An adversary can exploit this fact to launch DoS attacks on the authentication server. This means that the adversary can swamp the authentication server by sending a large number of fake authentication requests to it. The risk of this type of DoS attack increases when sophisticated authentication that requires increased processing is used. ANON provides a solution to this problem by hiding the IP address of the authentication server from the public and thus from adversaries. When ANON is used, an adversary can reach the authentication server only through the ANON infrastructure. Using rate-limiting, ANON can mitigate DoS attacks through the infrastructure itself.

We have implemented a laboratory testbed for ANON at Harvard. The testbed incorporates the countermeasures mentioned above. For ease of use, the testbed also includes

gateway servers that allow existing clients and application servers to use ANON as is, without modification.

This paper gives an overview of ANON and describes several important aspects related to the systems. These include usage examples, threat models and countermeasures, rate-limiting schemes, and implementation. In the next section, we first compare ANON with previous approaches in related areas.

COMPARISON WITH RELATED WORK

ANON differs from previous anonymizing approaches in two ways. First, ANON works at the IP layer. This is in contrast with other MIX-based [2] approaches such as onion routing [10] that use layer-4 or higher-layer protocols, and Mixmaster [3] that uses application-layer protocols. Each anonymizing forwarder in ANON is a stateless packet forwarder at the IP layer. The forwarder does not keep any connection or session state or any mapping tables that could be used to generate translated IP addresses and port numbers. This means that the forwarder is oblivious to the number of connections or sessions, and does not have translation tables to manage. However, the forwarders can keep statistics on traffic to specific destinations or from specific sources so that per-source or per-destination rate-limiting can be implemented. Since ANON is an infrastructure working at the IP layer, it does not modify packet content in the upper layers, and thus it is able to avoid complexities related to such modification [1]. In [4] a recent work on IP-layer anonymizing was reported. Unlike ANON, it uses a P2P networking infrastructure instead of a network resident set of forwarders managed by trusted third parties.

Second, ANON attempts to hide a target application server in a large candidate set of possible servers unknown to users and possible adversaries. By using a sufficiently large candidate set, ANON can ensure that DoS attacks on the application server based on indiscriminating DoS attacks on all these candidate servers will be ineffective. In contrast, traditional MIX-based anonymizing systems attempt to hide a target server with which a user communicatea. In such a system, the target server is in a candidate set of servers that are known to the public including adversaries. The anonymizing system in this case must make sure that traffic related to the user's communication with the target server is indistinguishable from that with any other server in the candidate set, so that adversaries cannot identify the target server easily. For this, artificial load must be injected into the network in order to disguise the user's traffic in the presence of traffic analysis [2]. To keep the artificial load under a modest level, the candidate set in these approaches needs to be small.

OVERVIEW OF THE ANON INFRASTRUCTURE

The ANON infrastructure consists of a set of anonymizing forwarders and some number of initialization servers, as depicted in Figure 1. Forwarders will encrypt and decrypt IP addresses, whereas initialization servers will provide clients with encrypted addresses of application servers. (In the figures of this paper, an application server is denoted as "server" for simplicity.) Forwarders themselves form an overlay IP network. That is, a pair of forwarders may be connected through a route involving multiple IP routers. Request packets from a client to a server will be forwarded over a forwarding path consisting of a subset set of these forwarders. Reply packets from the server to the client will use the same path in the reverse direction. For different reply-request sessions, different forwarding paths may be used.

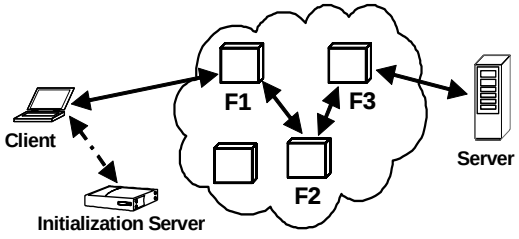

Figure 1. The ANON infrastructure, where Fs are anonymizing forwarders and solid arrows indicate an instance of a packet's forwarding path.

Some trusted third parties will operate forwarders. Since forwarders may decrypt IP addresses and thereby have access to the IP addresses that ANON intends to hide, it is important that forwarders are properly protected from being compromised or being monitored. To increase availability forwarders may use anycast-style addressing [6, 8], so that any of a number of forwarders using the same anycast address may forward a packet sent to it.

The role of initialization servers is to provide clients with encrypted addresses of application servers. Thus, initialization servers and the encrypted addresses provided by them need to be properly authenticated, possibly using digital certificates, to ensure that these encrypted addresses will be trustworthy. In addition, initialization servers may need to be replicated in various locations to ensure their high availability.

Consider, for example, the case of hiding the IP address of an application server from clients. In this case, the use of ANON will involve three usage steps:

- Server registration. An application server whose IP address needs to be hidden will invoke a process that selects a sequence of forwarders, computes the encrypted IP address for the application server, and registers the results to initialization servers. The sequence of forwarders can be selected manually or automatically, and statically or dynamically.
- Client initialization. Given an application server to which a client wishes to access, the client obtains the encrypted address for the application server, the address of the first decrypting forwarder, and other information required for packet forwarding.
- Packet forwarding. According to the information obtained from the client initialization, ANON forwards packets to and from the application server over the selected sequence of forwarders.

NOTATIONS AND ASSUMPTIONS

C: Client

S: Application Server
- S has generated an asymmetric key pair of public and private keys. S holds the private key.

I: Initialization Server

F: Anonymizing Forwarder
- F is assumed to be outside firewalls or NATs of C, S and I, should these firewalls or NATs exist.
- F holds a symmetric key for all its forwarding operations. Each F has its own symmetric key not known to others. Fs using the same anycast address share the same symmetric key.

[X]: IP address of X
- If X is a client behind a firewall or NAT, [X] is the IP address as seen from the outside of the organization.
- If X is a forwarder, [X] may be its unicast or anycast address.

[X]{payload}[Y]
- A packet with its source and destination IP addresses being [X] and [Y], respectively.

(z)r, where r is a symmetric key
- It is the content z encrypted in r.
- When r is the lower case letter of the name of a forwarder or server, r denotes the symmetric key of the forwarder or server. For example, (z)f means z encrypted in the symmetric key of forwarder F.

(z)A, where A is the name of a forwarder or a server
- It is the content z signed in A's private key or encrypted in A's public key. In the former case, A did the signing, whereas in the latter case another entity did the encryption.

X->Y: [X]{payload}[Y]
- X sends packet, [X]{payload}[Y], to Y.

X: operation
- X performs operation.

FORWARDING OPERATIONS

Depending on the application, a forwarder may perform one of the forwarding operations listed below. Subsequent usage examples will illustrate the use of these operations.

FWD-INC (“forward and include”):
Input packet: [X]{msg, [Y]}[F]
Output packet: [F]{msg, [X]}[Y]

FWD-CLR (“forward and clear”):
Input packet: [X]{msg, [Y]}[F]
Output packet: [F]{msg}[Y]

FWD-ENC (“forward and encrypt”):
Input packet: [X]{msg, [Y]}[F]
Output packet: [F]{msg, ([X])f}[Y]

DEC-FWD-INC (“decrypt, forward and include”):
Input packet: [X]{msg, ([Y])f}[F]
Output packet: [F]{msg, [X]}[Y]

DEC-FWD-CLR (“decrypt, forward and clear”):
Input packet: [X]{msg, ([Y])f}[F]
Output packet: [F]{msg}[Y]

DEC-FWD-ENC (“decrypt, forward and encrypt”):
Input packet: [X]{msg, ([Y])f}[F]
Output packet: [F]{msg, ([X])f}[Y]

In addition to these forwarding operations, a forwarder may also support management operations such as application servers' registration.

BASELINE USAGE EXAMPLE B1: HIDE SERVER'S ADDRESS

This example illustrates the use of ANON to achieve the following two objectives:

- A client C sends a request to an application server S without knowing S's address.
- C receives a reply from S without knowing S's address.

As described earlier, the client C first interacts with an initialization server I. In its message to I, C expresses its wish to access an application server S. Then the initialization server I securely sends C a message, e.g., via SSL, containing the following two items:

- [F], the unicast address of a forwarder F, or the anycast address of a set of forwarders, also denoted by F.
- ([S])f

When the client wishes to send a request to S, it builds a request packet containing the following contents and sends it to [F]:

- (req, ck)S, where req is C's request to S, and ck is a cookie associated with the packet. (req, ck)S is (req, ck) encrypted by S's public key. The purpose of ck is to identify the request. After the packet is sent, C will keep ck around, so that it can be used later to associate the reply received from S with the request.
- ([S])f

Upon receiving the request packet, F decrypts the packet, forwards it to [S], with the source address of the original packet, [C], as seen by F included in the packet payload. That is, F performs the operation DEC-FWD-INC.

When S receives the request packet, it builds a reply packet containing the following contents and sends it to [F]:

- (rep, ck)S: reply and cookie signed by S with its private key.
- [C]: the source address of the original packet as seen by F.

Upon receiving the reply packet, F forwards it to [C] without including [S] in the packet payload, so [S] will not be revealed. That is, F performs the operation FWD-CLR. When C receives the packet, it decrypts the reply and cookie using S's public key. By comparing the decrypted cookie with the original cookie stored at C, C checks whether the received reply is the one corresponding to its original request.

We summarize usage B1 for hiding [S] as follows:

C -> F: [C]{(req, ck)S, ([S])f}[F]
F: DEC-FWD-INC
F -> S: [F]{(req, ck)S, [C]}[S]
S: reply
S -> F: [S]{(rep, ck)S, [C]}[F]
F: FWD-CLR
F -> C: [F]{(rep, ck)S}[C]
C: decrypt reply and cookie, and verify the reply

BASELINE USAGE EXAMPLE B2: HIDE CLIENT'S ADDRESS

This example illustrates the use of ANON to achieve the following two objectives:

- S receives request from C without knowing C's address.
- S sends reply to C without knowing C's address.

In this case, C will obtain [F] and [S] from an initialization server. We summarize usage B2 for hiding [C] as follows:

C -> F: [C]{req, [S]}[F]
F: FWD-ENC
F -> S: [F]{req, ([C])f}[S]
S: reply
S -> F: [S]{rep, ([C])f}[F]
F: DEC-FWD-INC
F -> C: [F]{rep, [S]}[C]
C: receive reply and [S]

Note that usage B1 and B2 can be combined to yield a scheme that will hide both [C] and [S].

ENHANCED TWO-HOP USAGE EXAMPLE: HIDE SERVER'S ADDRESS

This example is an enhanced version of baseline usage example B1 above. It is designed to defend against a type of replay attack. Suppose that an adversary repetitively submits C's request:

C->F: [C]{(req, ck)S, ([S])f}[F]

while monitoring packet contents on links that could be on the path from F to S and vice versa. If those packets on the links that result from these repeated request submissions are distinguishable, then the adversary will be able to learn [S] by examining destination or source addresses of these packets. It is thus important to avoid invariant bit strings in packet load, such as [C], (req, ck)S and (rep, ck)S in usage example B1 above, that could be used to identify these packets.

ANON has provided mechanisms to protect itself against this type of attack. In particular, the infrastructure satisfies the “semantically secure packet encryption” property. That is, *N* repeated submissions of the same packet will yield *N* different encrypted packet payloads on a link. This property is depicted in Figure 2.

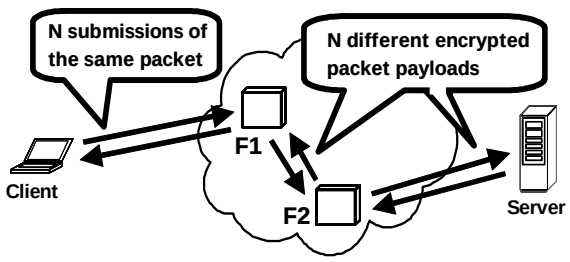

Figure 2. Semantically secure packet encryption.

Below is an enhanced version of usage example B1 satisfying this semantically secure packet encryption" property. It is a two-hop example, involving forwarders F1 and F2. Client C gets (([S], s)f2, [F2])f1 and [F1] from an initialization server.

C->F1: [C]{req, ck, (([S], s)f2, [F2])f1}[F1]

F1->F2: [F1]{req, ck, ([S], s)f2, [C]}[F2]

F2->S: [F2]{(req, ck, [C])r1, (r1)s, {[C], [F1]}r2, (r1, r2)f2}[S]

S->F2: [S]{(rep, ck)r1, ([C], [F1])r2, (r1, r2)f2}[F2]

F2->F1: [F2]{rep, ck, [C]}[F1]

F1-> C: [F1]{rep, ck}[C]

In this method, the IPsec Encapsulating Security Payload (ESP) [7] is used for the encryption of packet transport between forwarders F1 and F2. Note that IPsec ESP has built-in support for semantically secure packet encryption. For packet transport between F2 and S, F2 randomly selects symmetrical session keys r1 and r2 to implement semantically secure packet encryption. One can verify that *N* submissions of the same packet by C will yield *N* different encrypted packet payloads on path segments from F2 to S and S to F2. For packet transport between C and F1, there is no need to apply packet encryption, since [F1] need not be hidden from the client or the public.

It is straightforward to extend this two-hop enhanced scheme to hide [C], or both [C] and [S], and to allow additional hops.

THREAT MODELS AND COUNTERMEASURES

We consider the following three types of security-related threats for the ANON infrastructure:

- Type 1 threat. The forwarding infrastructure may leak address information that it is supposed to hide. This is a case where the physical links in and out of a forwarder might be monitored by an unauthorized party, or the forwarder itself is compromised.
- Type 2 threat. The forwarding infrastructure may itself be subject to DoS attacks.
- Type 3 threat. The forwarding infrastructure may be used as a conduit to launch DoS attacks.

To defend against Type 1 threats, ANON uses multi-hop forwarding, as illustrated in the enhanced two-hop usage example earlier. Note that the adversary's objective is to identify the *exit forwarder*, the one that will have access to the decrypted IP address of the target application server whose address ANON attempts to hide. Starting from an *entry forwarder*, the adversary would need to follow the forwarding chain in order to discover the exit forwarder. (For the illustrative scenario in Figure 1, F1 is the entry forwarder and F2 is the exit forwarder.) The longer the forwarding chain, the harder must the adversary work. This is especially true if the forwarders are under different administrative authorities, since in this case the attacker will need to compromise all the authorities in order to succeed.

ANON provides countermeasures that can hide forwarding traffic from the adversary. To prevent packet contents from revealing forwarding information, ANON uses techniques such as semantically secure packet encryption described earlier.

To defend against traffic analysis, ANON employs previously known techniques such as traffic mixing, link padding and protocol camouflaging. For example, ANON camouflages UDP packets forwarded between forwarders as normal TCP packets. ANON also employs new techniques such as on-demand link padding and per-destination rate-liming that allow artificial padding traffic to be inserted only when there is real traffic to hide. In addition, ANON supports dynamic re-selection of forwarders. A target application server can dynamically register forwarding paths so that a new forwarding path can be used before the old one is cracked. That is, application servers may change the forwarders they use from time to time through the registration process.

To defend against Type 2 threats, ANON can have entry forwarders perform high-volume, lightweight filtering of packets. Operating at wire speed these forwarders could filter packet addressees and send challenges to packet sources, so that packets with illegitimate IP addresses will be discarded. DoS attacks at forwarders other than the entry forwarders will not be possible unless their addresses become known to the attacker. This leaking should not happen if countermeasures against Type 1 threats work.

To defend against Type 3 threats, ANON can reject packets with spoofed source IPs, and rate limit on a per-link, per-source or per-destination basis.

To implement rate-limiting, each forwarder alternates between two phases, *equalization* and *relaxation*. The forwarder can use the "push-back" technology [5] to send control signals to its upstream nodes to regulate the rate of traffic from the upstream nodes to the current forwarder. In the equalization phase, if its current total rate is above certain threshold HI, a forwarder will increase its push-

back signal to sources that have relatively larger traffic usage at present. In contrast, in the relaxation phase, if its current total rate is below certain threshold LOW, a forwarder will reduce its push-back signal to sources that have relatively small traffic usage at present. This rate-limiting scheme can keep the utilization of the ANON network at a reasonably high level and its packet loss rate due to congestion at a reasonably low level, while blocking large users, including DoS attackers, from taking away bandwidths that other small users may need.

TESTBED IMPLEMENTATION

We have implemented a laboratory testbed for ANON at Harvard. The testbed, depicted in Figure 3, consists of a half-dozen forwarders. Nodes in the testbed are implemented on top of FreeBSD 4.5-Stable. The divert socket available from FreeBSD is used to implement various header processing tricks at the user-level. For the symmetric key algorithm, the testbed uses the AES reference implementation from NIST.

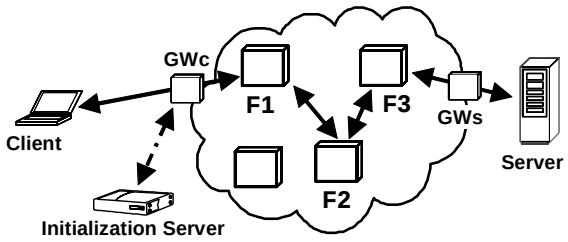

Figure 3. Experimental testbed that has been implemented. GWc and GWs are gateways that allow existing clients and servers to use ANON without modifications.

The testbed includes NAT-like gateways, GWc for clients and GWs for servers. These gateways allow existing clients and servers to use the testbed without modifications. Clients may connect to their GWc directly or via a VPN connection. In a similar way, servers connect to their GWs.

The current testbed implementation can achieve a throughput of 5 Mbps. This performance, which is more than adequate for signaling applications, is made possible mainly because we have managed to avoid using public key encryption and decryption in packet forwarding, as illustrated in usage examples earlier.

SUMMARY AND CONCLUDING REMARKS

In this paper, we have described the architecture of an anonymizing infrastructure at the IP layer. The infrastructure is specially designed for low-bandwidth applications such as authentication and authorization. We have shown usage examples of hiding addresses of servers, clients, etc. We have described security threat models for the anonymizing infrastructure, and a suite of countermeasures, such as semantically secure packet encryption and rate-limiting schemes. Finally, we have described a laboratory testbed implementation of the infrastructure.

This work has shown that it is both feasible and natural to provide an anonymizing infrastructure at the IP layer. For example, rate-limiting and link padding can be conveniently implemented at the IP layer. If one would attempt to implement these functions at higher layers, it would incur a great deal of efforts if possible at all.

ACKNOWLEDGMENTS

This work was supported in part by DARPA through AFRL/SNZW under contract F33615-01-C-1983.

REFERENCES

[1] Boucher, P., A. Shostack, I. Goldberg, *Freedom system 2.0 architecture*, Dec. 2000, http://www.freedom.net/products/whitepaperFreedom_System_2_Architecture.pdf

[2] Chaum, D., *Untraceable electronic mail, return addresses, and digital pseudonyms*, *CACM*, vol. 24, pp. 84 – 88, Feb. 1981.

[3] Cottrell, L., *Frequently Asked Questions about Mixmaster Remailer*, http://www.obscura.com/~loki/remailer/mixmaster-faq.html, 1996.

[4] Freedman, M. J., E. Sit, J. Cates, R. Morris, *Introducing Tarzan, A Peer-to-Peer Anonymizing Network Layer*, in Proceedings of 1st Intl. Workshop on Peer-to-Peer Systems, Cambridge, MA. March 2002.

[5] Ioannidis, J. and S. M. Bellovin, *Pushback: Router-Based Defense against DDoS Attacks*, NDSS, February 2002.

[6] Katabi, D., and J. Wroclawski, *A Framework for Global IP-Anycast (GIA)*, in Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, 2000.

[7] Kent, S. and R. Atkinson, *IP Encapsulating Security Payload (ESP)*, RFC 2406, November 1998.

[8] Milliken, W., C. Partridge, T. Mendez, *Host anycasting service*, RFC 1546, November 1993.

[9] The NymIP Effort, http://nymip.velvet.com

[10] Reed, M., P. Syverson, D. Goldschlag, *Anonymous Connections and Onion Routing*, IEEE Journal on Selected Areas in Communications, vol. 16 no. 4, May 1998, pp. 482-494.