Network Working Group
Internet-Draft
Expires May 2002

HT Kung
Scott Bradner
Harvard University
November 2001

A Framework for an Anonymizing Packet Forwarder

<draft-kung-annfwd-framework-00.txt>

Status of This Memo

Abstract

There are a number of situations in the Internet where it would be useful to be able to have an application be able to send traffic to a destination without revealing the IP address of the destination to the source, or the IP address of the source to the destination, or both. One way to do this is to have a network resident set of servers which can forward packets, with encryption and decryption applied to their source and destination addresses when appropriate. We will call this server an anonymizing forwarder. The anonymizing forwarding server intends to contribute to the goal of supporting anonymity at the IP layer [NymIP].

This memo describes several aspects of such an infrastructure based on stateless anonymizing packet forwarders. These include (1) usage examples of the forwarding infrastructure, (2) target servers' registration that receives their encrypted addresses from the

infrastructure and sends them to clients' initialization servers, (3) multi-hop forwarding and the associated registration, and (4) threat analysis for the infrastructure and countermeasure strategies.

The framework discussions here provide background information on the requirements for an anonymizing packet forwarder described in a companion document [RQM-DRAFT].

1. Usage Examples

Notations and Assumptions

C: Client

S: Target Server

S has generated an asymmetric key pair of public and private keys, and holds the private key.

I: Initialization Server

F: Anonymizing Forwarder

F is assumed to be outside of firewalls or NATs of C, S and I, should these firewalls or NATs exist.

For forwarding purposes, F can use either a symmetric key, or an asymmetric key pair of public and private keys. In this document we assume that F uses a symmetric key for all its forwarding operations. F will encrypt an address and later decrypt it, so only F will need to know the symmetric key, and will keep it secret.

To support target servers' registration, F uses an asymmetric key pair of public and private keys, and holds the private key.

The forwarders of the same anycast address ([RFC1546] and [ANYCAST]) all have the same keys.

[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.

If r happens to be the lower case letter of the name of a forwarder or server, then r is 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 or encrypted in A's public key. In the former case A does the signing, whereas in the latter case another entity does the encryption.

A's ctf

A's certificate providing A's public key and vouching for it.

X->Y: [X]{payload}[Y]

X sends packet [X]{payload}[Y] to Y.

X: operation

X performs operation.

Six Forwarding Operations of a Forwarder

Depending on the application, a forwarder may perform one of the following six forwarding operations listed below for a given input packet. 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 target servers' registration (see below).

Baseline Usage Example: Hide [S]

This example illustrates the use of the anonymizing infrastructure to satisfy the following two properties:

(P1) A client C sends request to a target server S without knowing S's address.

(P2) C receives reply from S without knowing S's address.

The client C first interacts with an initialization server I, which may be a local application or a network-based service. In its message to I, C expresses its wish to access a target server S. Then the initialization server I securely sends C a message containing the following three items:

- [F], which is the unicast address of a forwarder F, or the anycast address of a set of forwarders, also denoted by F.

- ([S])f

- S's ctf

When the client wishes to send a request to S, it builds a 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.

- ([S])f

After the packet is sent, C will need to keep ck and S's ctf around for a while, so that they can be used later to verify the reply from S.

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

- (rep, ck)S: reply and cookie encrypted in the private key of S.

- [C]: the source address of the original packet as seen by F.

When F receives the packet, it 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 decides whether or not the received reply is the one corresponding to its original request.

The following summarizes the description above.

Baseline Usage Example: Hide [S]

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 to verify the reply

Usage Example 1.1: Hide [S]

This example is an enhanced version of Baseline Usage Example above. It is designed to defend against a type of replay attacks. Suppose that an adversary repetitively submits C's request:

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

while monitoring packets on links that could be on the path from F to S and vice versa. If the adversary can identify these packets a priori, then he or she will be able to learn [S] by examining destination or source addresses of these packets. It is therefore important to avoid invariant bit strings, such as [C], (req, ck)S and (rep, ck)S in the Baseline Usage Example above, that could be used to identify these packets.

Usage Example 1.1, summarized below, ensures that all these packets will likely have different payloads. Thus it would be difficult to isolate these packets.

Usage Example 1.1: Hide [S]

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

F: DEC-FWD-ENC

F->S: [F]{((req, ck, C's ctf)S, [C])r1, (r1)S, ([C])r2, (r2)f}[S]

S: reply

S->F: [S]{(rep, ck)r3, (r3)C, ([C])r2, (r2)f}[F]

F: DEC-FWD-CLR

F->C: [F]{(rep, ck)r3, (r3)C}[C]

C: decrypt reply and cookie to verify the reply

In the above, r1 and r2 are symmetrical session keys randomly selected by F for an each packet, while r3 is a key selected by S. This means that each resubmitted request

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

will likely result in an entirely different payload in the message from F->S, S->F or F->C.

Usage Example 1.2: Hide [C]

This example illustrates the use of the anonymizing infrastructure to satisfy the following two properties:

(P3) S receives request from C without knowing C's address.

(P4) S sends reply to C without knowing C's address

C will obtain the following two items from the initialization server:

- [F]
- [S]

We summarize Usage Example 1.2 as follows:

Usage Example 1.2: Hide [C]

C->F: [C]{req, C's ctf, [S]}[F]

F: FWD-ENC

F->S: [F]{req, C's ctf, ([C])f}[S]

S: reply

S->F: [S]{rep, C's ctf, ([C])f}[F]

F: DEC-FWD-INC

F->C: [F]{(rep, [S])r, (r)C}[C]

C: decrypt reply and verify the reply

Note that in the packet from F to C, F randomly selects a symmetrical session key r for each packet to defend against replay attacks. In such an attack, an adversary would repetitively resend the same message from S to F, while monitoring links from F to C. If those packets from F to C that are triggered by these repeated resends can be isolated, then their destination addresses will reveal [C]. By dynamically changing their payloads, these packets from F to C can not be isolated easily.

Usage Example 1.3: Hide Both [C] and [S]

This example illustrates the use of the anonymizing infrastructure to satisfy all the following four properties:

(P1) A client C sends request to a target server S without knowing S's address.

(P2) C receives reply from S without knowing S's address.

(P3) S receives request from C without knowing C's address.

(P4) S sends reply to C without knowing C's address.

Thus, this usage intends to hide both addresses of C and S.

The procedure summarized below for Usage Example 1.3 is a combination of procedures for Usage Example 1.1 and 1.2 above. Usage Example 1.3 has the properties of both Usage Example 1.1 and 1.2.

Usage Example 1.3: Hide Both [C] and [S]

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

F: DEC-FWD-ENC

F->S: [F]{((req, ck, C's ctf)S, ([C], C's ctf)r2, (r2)f)r1, (r1)S}[S]

S: reply

S->F: [S]{(rep, ck)r3, (r3)C, ([C], C's ctf)r2, (r2)f}[F]

F: DEC-FWD-CLR

F->C: [F]{((rep, ck)r3, (r3)C)r4, (r4)C}[C]

C: decrypt reply and cookie to verify the reply

2. Target Server's Registration

A target server S interested in hiding its address [S] by using an anonymizing forwarder F may register itself to an initialization server I. The registration will involve S first sending an encryption request to F asking for ([S])f, and then sending the received ([S])f to I via F. After receiving this information from I, C can send its requests to S via F, as described in the Baseline Usage Example earlier. It is instructive to consider a target server's registration as a process for an initialization server to receive an "access ticket", which consists of information such as [F] and [S])f, that allows clients' requests to reach S via F.

Target Server's Registration to I on Use of Forwarder F:

S->F: [S]{S's ctf, ([S])S}[F]

F: decrypt ([S])S to validate [S]'s authenticity, and encrypt [S] in F's symmetric key

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

S: decrypt (([S])f)S using S's private key to recover [F] and ([S])f

optionally, S may send test messages to F using ([S])f to check if F will forward the messages back to S

sign (([S])f, [F]) in S's private key

S->F: [S]{S's ctf, (([S])f, [F])S, [I]}[F]

F: FWD-CLR

F->I: [F]{S's ctf, (([S])f, [F])S}[I]

I: decrypt ([S])f, [F])S to validate the authenticity of [S])f and [F] using S's public key, and then receive [F], ([S])f, and S's ctf

This registration method has three properties worth mentioning. First, it does not require S to know F's symmetric key. Second, S may change to a new forwarder F' it wants to use for a given I by obtaining ([S])f' from F' and sending it to I. Third, S may use different Fs for different Is, for balancing loads or meeting

different security requirements.

Note that F encrypts [S] using that from ([S])S, rather than that extracted from the source address of S's message. This is to defend against possible impersonation of S by an adversary. Suppose that an adversary S1 sends an encryption request [S1](S's ctf, [S1])[F] to F, and S1 receives from F reply [F]{(([S1])f)S}[S1]. If S1 can manage to send the packet [F]{(([S1])f)S}[S] to S, then S would mistakenly use ([S1])f instead of ([S])f in the registration.

3. Multi-hop Registration and Forwarding

In multi-hop forwarding, a sequence of two or more forwarders are used in forwarding a client's packet. These forwarders together are sufficient in decrypting the address of the target server, but any proper subset of them are not.

Multi-hop forwarding can provide additional protection against an adversary who may attempt to compromise a forwarder or monitor its output links. For example, by employing forwarders protected under strong but different security measures, the adversary would need to defeat all these security measures in order to succeed.

Multi-hop Registration

During its registration, a target server S will obtain a sequence of forwarders to use. First, S chooses an F that S trusts, and sends it a request asking for ([S])f. When receiving the request from S, F may find an another forwarder G that F trusts. G may be under a different jurisdiction, so that compromising both F and G would be harder than compromising forwarders in the same jurisdiction. In turn, G may find yet another forwarder H that G trusts, and so on. Finally, the last forwarder will send S the necessary information required by S's registration.

We illustrate below the multi-hop registration, using a three-hop example involving three forwarders F1, F2 and F3 corresponding to F, G and F, respectively. We assume here that the Fs each use a public and private key pair, and hold the private key.

Target Server's Three-hop Registration to I on Use of Forwarders F1, F2 and F3:

S->F1: [S]{S's ctf, ([S])S}[F1]

F1: decrypt ([S])S to validate [S]' authenticity using S's public key, encrypt the [S] in F1's symmetric key, and sign [F1] in F1's private key

F1->F2: [F1]{S's ctf, F1's ctf, ([S])f1, ([F1])F1}[F2]

F2: decrypt ([F1])F1 to validate [F1]'s authenticity using F1's public key, encrypt (([S])f1, [F1]) in F2's symmetric key, and sign [F2] in F2's private key

F2->F3: [F2]{S's ctf, F2's ctf, (([S])f1, [F1])f2, ([F2])F2}[F3]

F3: decrypt ([F2])F2 to validate [F2]'s authenticity using F2's public key, encrypt (([S])f1, [F1])f2, [F2]) in F3's symmetric key, sign [F3] in F3's private key, and encrypt ((([S])f1, [F1])f2, [F2])f3, ([F3])F3 in S's public key

F3->S: [F3]{F3's ctf, (((([S])f1, [F1])f2, [F2])f3, ([F3])F3)S}[S]

S: decrypt ([F3])F3 to validate [F3]'s authenticity using F3's public key, and sign ((([S])f1, [F1])f2, [F2])f3, [F3] in S's private key

S->F3: [S]{S's ctf, (((([S])f1, [F1])f2, [F2])f3, [F3])S, [I]}[F3]

F3: FWD-CLR

F3->I: [F3]{S's ctf, (((([S])f1, [F1])f2, [F2])f3, [F3])S}[I]

I: receive ((([S])f1, [F1])f2, [F2])f3, [F3]

Multi-hop Forwarding

We illustrate below multi-hop forwarding with a simple usage example, that corresponds to the single-hop Baseline Usage Example earlier.

Multi-hop Baseline Usage Example: Hide [S]

C->F3: [C]{(req, ck)S, ((([S])f1, [F1])f2, [F2])f3}[F3]

F3: DEC-FWD-INC

F3->F2: [F3]{(req, ck)S, (([S])f1, [F1])f2, [C]}[F2]

F2: DEC-FWD-INC

F2->F1: [F2]{(req, ck)S, ([S])f1, [C], [F3]}[F1]

F1: DEC-FWD-INC

F1->S: [F1]{(req, ck)S, [C], [F3], [F2]}[S]

S: reply

S->F1: [S]{(rep, ck)S, [C], [F3], [F2]}[F1]

F1: FWD-CLR

F1->F2: [F1]{(rep, ck)S, [C], [F3]}[F2]

F2: FWD-CLR

F2->F3: [F2]{(rep, ck)S, [C]}[F3]

F3: FWD-CLR

F3->C: [F3]{(rep, ck)S}[C]

C: decrypt reply and cookie to verify the reply

The nested encryption is similar to that in onion routing [ONION].

Note that schemes similar to those of Usage Example 1.1 can be applied to prevent replay attacks that exploit the fact that (req, ck)S or (rep, ck)S remains an invariant in all the packets.

4. Threat Analysis and Counter Measure Strategies

Types of Treats

There are various types of threats regarding anonymizing forwarders. We consider the following three types:

Type 1 threat: The forwarding infrastructure leaks address information that it is supposed to hide.

Type 2 treat: The forwarding infrastructure is itself subject to DoS Attacks.

Type 3 threat: The forwarding infrastructure is used as a conduit for DoS attacks.

Countermeasure Strategies

The forwarding infrastructure itself should provide the bulk of the

means of protection against these threats. The methods should not involve clients, initialization servers, and target servers, since they are outside management authorities of the infrastructure. Moreover, the infrastructure should not attempt to protect itself through user authentication, since the infrastructure is supposed to support authentication infrastructure, not vice versa.

Multi-hop forwarding can help defend Type 1 threats, in the sense that it will make an adversary work harder in order to learn the address of a target server. This is especially true if the forwarders in the multi-hop sequence are under different administrative authorities, because in this case the attacker will need to compromise all the authorities in order to succeed.

To defend against Type 2 threats concerning DoS attacks on the forwarding infrastructure, one can have first-hop forwarders provide high-volume, light-weight filtering of requests with spoofed source IP addresses. These servers working at the wire speed could send challenges to the requestors, so that only those with legitimate IP addresses will be able to respond. Forwarders behind the first-hop ones will have their addresses hidden from users. In addition, via registrations, target servers may change the forwarders they use from time to time.

To defend against Type 3 threats concerning the forwarding infrastructure being used as conduit for DoS attacks, the infrastructure could reject requests from spoofed source IPs. In addition, a forwarder could rate limit its output on a per link, per source, or per destination basis.

References

[RQM-DRAFT] Bradner, S., and Kung, H. T., "Requirements for an Anonymizing Packet Forwarder" <draft-bradner-annfwd-req.txt>, Draft, November 2001

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

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

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

[RFC1546] Milliken, W., Partridge C., and Mendez, T., "Host anycasting service," RFC 1546, November 1993.

8. Authors' Addresses

HT Kung
Harvard University
33 Oxford St.
Cambridge MA 02138

Email: kung@harvard.edu
Phone: +1-617-496-6211

Scott Bradner
Harvard University
29 Oxford St.
Cambridge MA 02138

Email: sob@harvard.edu
Phone: +1-617-495-3864