Provider-side autoresponder extension (Kaetzchen)

Yawning Angel
Kali Kaneko
David Stainton

Abstract

This interface is meant to provide support for various autoresponder agents "Kaetzchen" that run on Katzenpost provider instances, thus bypassing the need to run a discrete client instance to provide functionality. The use-cases for such agents include, but are not limited to, user identity key lookup, a discard address, and a loop-back responder for the purpose of cover traffic.

Table of Contents

Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

Terminology

The following terms are used in this specification.

SURB	Single use reply block. SURBs are used to achieve recipient anonymity, that is to say, SURBs function as a cryptographic delivery token that you can give to another client entity so that they can send you a message without them knowing your identity or location on the network. See `SPHINXSPEC` and `SPHINX`.
BlockSphinxPlaintext	The payload structure which is encapsulated by the Sphinx body. It is described in KATZMIXE2E, section Client and Provider processing of received packets.

1. Extension Overview

Each Kaetzchen agent will register as a potential recipient on its Provider. When the Provider receives a forward packet destined for a Kaetzchen instance, it will hand off the fully unwrapped packet along

with its corresponding SURB to the agent, which will then act on the packet and optionally reply utilizing the SURB.

1.1 Agent Requirements

- Each agent operation MUST be idempotent.
- Each agent operation request and response MUST fit within one Sphinx packet.
- Each agent SHOULD register a recipient address that is prefixed with (Or another standardized delimiter, agreed to by all participating providers in a given mixnet).
- Each agent SHOULD register a recipient address that consists of an RFC5322 dot-atom value, and MUST register recipient addresses that are at most 64 octets in length.
- The first byte of the agent's response payload MUST be 0x01 to allow clients to easily differentiate between SURB-ACKs and agent responses.

1.2 Mix Message Formats

Messages from clients to Kaetzchen use the following payload format in the forward Sphinx packet:

```
struct {
uint8_t flags;
uint8_t reserved; /* Set to 0x00. */
select (flags) {
case 0:
opaque padding[sizeof(SphinxSURB)];
case 1:
SphinxSURB surb;
}
opaque plaintext[];
} KaetzchenMessage;
```

The plaintext component of a `KaetzchenMessage` MUST be padded by appending “0x00” bytes to make the final total size of a `KaetzchenMessage` equal to that of a `BlockSphinxPlaintext`.

Messages (replies) from the Kaetzchen to client use the following payload format in the SURB generated packet:

```
struct {
opaque plaintext[];
} KaetzchenReply;
```

The plaintext component of a `KaetzchenReply` MUST be padded by appending “0x00” bytes to make the final total size of a `KaetzchenReply` equal to that of a `BlockSphinxPlaintext`

2. PKI Extensions

Each provider SHOULD publish the list of publicly accessible Kaetzchen agent endpoints in its MixDescriptor, along with any other information required to utilize the agent.

Provider should make this information available in the form of a map in which the keys are the label used to identify a given service, and the value is a map with arbitrary keys.

Valid service names refer to the services defined in extensions to this specification. Every service MUST be implemented by one and only one Kaetzchen agent.

For each service, the provider MUST advertise a field for the endpoint at which the Kaetzchen agent can be reached, as a key value pair where the key is `endpoint`, and the value is the provider side endpoint identifier.

```
{ "kaetzchen":
{ "keyserver" : {
"endpoint": "+keyserver",
"version" : 1 } },
{ "discard" : {
"endpoint": "+discard", } },
{ "loop" : {
"endpoint": "+loopback",
"restrictedToUsers": true } },
}
```

3. Anonymity Considerations

In the event that the mix keys for the entire return path are compromised, it is possible for adversaries to unwrap the SURB and determine the final recipient of the reply.

Depending on what sort of operations a given agent implements, there may be additional anonymity impact that requires separate consideration.

Clients MUST NOT have predictable retranmission otherwise this makes active confirmations attacks possible which could be used to discover the ingress Provider of the client.

4. Security Considerations

It is possible to use this mechanism to flood a victim with unwanted traffic by constructing a request with a SURB destined for the target.

Depending on the operations implemented by each agent, the added functionality may end up being a vector for Denial of Service attacks in the form of CPU or network overload.

Unless the agent implements additional encryption, message integrity and privacy is limited to that which is provided by the base Sphinx packet format and parameterization.

Acknowledgments

The inspiration for this extension comes primarily from a design by Vincent Breitmoser.

References

KATZMIXE2E

Angel, Y., Danezis, G., Diaz, C., Piotrowska, A., Stainton, D., “Katzenpost Mix Network End-to-end Protocol Specification”, July 2017, https://github.com/katzenpost/katzenpost/blob/main/docs/specs/old/end_to_end.md

KATZMIXPKI

Angel, Y., Piotrowska, A., Stainton, D., “Katzenpost Mix Network Public Key Infrastructure Specification”, December 2017, ttps://github.com/katzenpost/katzenpost/blob/main/docs/specs/pki.md

RFC2119

Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119

RFC5322

Resnick, P., Ed., “Internet Message Format”, RFC 5322, DOI 10.17487/RFC5322, October 2008, https://www.rfc-editor.org/info/rfc5322

SPHINX09

Danezis, G., Goldberg, I., “Sphinx: A Compact and Provably Secure Mix Format”, DOI 10.1109/SP.2009.15, May 2009, https://cypherpunks.ca/~iang/pubs/Sphinx_Oakland09.pdf

SPHINXSPEC

Angel, Y., Danezis, G., Diaz, C., Piotrowska, A., Stainton, D., “Sphinx Mix Network Cryptographic Packet Format Specification”, July 2017, https://github.com/katzenpost/katzenpost/blob/main/docs/specs/sphinx.md