ATOCA
Internet-Draft
Intended status: Informational
Expires: May 3, 2012

R. Barnes
BBN Technologies
October 31, 2011

Lightweight Emergency Alerting Protocol (LEAP)
draft-barnes-atoca-delivery-01.txt

Abstract

Emergency alerts need to be delivered reliably from one source to many recipients at once. TCP is unsuitable for this style of delivery, because the large number of acknowledgements would likely cause network congestion. This document defines a UDP-based protocol for delivering alerts that supports fragmentation and retransmission for reliability, and allows the sender of a datagram to control whether acknowledgements are sent.

Please send feedback to the atoca@ietf.org mailing list.

Table of Contents

1. Introduction

Servers that provide emergency alerts to end hosts have two conflicting requirements. They need to deliver alerts reliably to a large number of hosts, but in a scalable fashion that does not cause undue network congestion. In particular, TCP is unsuitable for delivering alerts because of the overhead imposed by connection establishment and acknowledgement messages [RFC0793]. Sending alerts directly in a UDP datagram is not appropriate either, because of the size limits imposed by link maximum transmission units (MTUs) [RFC0768].

This document defines the Light-weight Emergency Alerting Protocol (LEAP) as a simple, UDP-based way to deliver emergency alerts. This protocol defines a simple fragmentation layer over UDP, and retransmission and reassembly algorithms that allow for reliable transmission of alerts without a need for acknowledgements. We also define a URI format for specifying alert sources, so that alert servers can inform alert recipients about what sorts of alerts they should accept over this protocol.

1.1. Open Questions

Should we randomize the order in which fragments are transmitted in order to deal with correlated loss?

How should we manage UDP ports? Require that destination==source? Require that destination==default? If there is any flexibility in port selection, should the URI format allow these to be indicated?

2. Definitions

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 RFC 2119 [RFC2119].

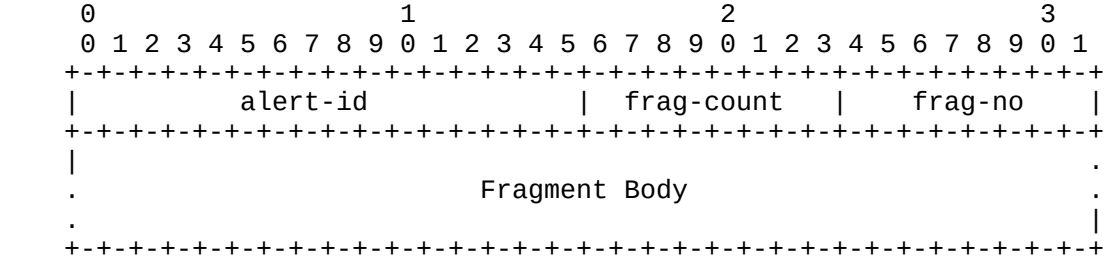
3. Packet Format

LEAP transmits ESCAPE-encoded CAP alerts as a collection of fragments [I-D.barnes-atoca-escape]. Alert servers divide alerts into fragments that are small enough to fit into an MTU, and clients reassmeble these fragments to obtain the complete alert. (See Section 5 and Section 6 for details on the fragmentation and reassembly processes.

LEAP payloads are encapsulated in UDP datagrams with source and

destination ports equal to XXX. Each datagram comprises a 4-octet LEAP header, followed by alert data:

[[Note to RFC Editor: Please replace the XXX above with the port number assigned by IANA]]



The LEAP header has the following fields:

- alert-id: A 16-bit unsigned integer uniquely identifying this alert among alerts sent from the server IP address and port for this packet

- frag-count: An 8-bit unsigned integer describing the total number of fragments in an alert

- frag-no: An 8-bit unsigned integer describing the position of this payload in the sequence of alert fragments

The remainder of the UDP payload contains the body of the alert fragment itself. The reassembled fragments of a LEAP-transmitted alert MUST comprise a valid ESCAPE-formatted alert. Note that because each alert can be split into at most 256 fragments, the total size of the alert is still limited to a multiple of the MTU. If the available payload size after IP, UDP, and LEAP headers is 1KB, then the maximum alert that can be transmitted is 256KB.

4. URI Format

A LEAP URI describes an alert server that will transmit alerts using LEAP. Clients can use these URIs to determine which LEAP messages they should accept based on a list of authorized LEAP URIs.

[[TODO: ABNF for URI format leap:[host/IP]]]

5. Server Processing

An alert server transmits an ESCAPE-encoded alert according to the following steps:

1. Choose a 16-bit pseudo-random alert ID.

2. Divide the alert into fragments that are sufficiently small that they are likely to be less than the MTU on all links between the server and end clients. A 512-octet maximum fragment size is RECOMMENDED.

3. Attach to each fragment a LEAP header with the following values:

    * alert-id: The 16-bit value chosen in step 1

    * frag-count: The number of fragments generated in step 2

    * frag-no: The index of this fragment in the sequence of fragments, starting at zero

4. Transmit each fragment (with its header) in a UDP datagram to the client(s)

5. Re-transmit the fragment sequence as necessary to achieve the desired level of reliability

Servers increase the reliability of alert delivery by retransmitting the sequence of alert fragments. Servers SHOULD compute the number of retransmissions R based on three factors:

- p: The estimated probability of a packet successfully reaching the client from the server (one minus the loss rate)

- q: The probability that a client receives all fragments successfully

- F: The number of fragments in the alert

When clients apply the reassembly algorithm described below, the probability of receiving an entire alert after R retransmissions is given by the following formula:

$$q = (1 - (1-p)^R)^F$$

Solving this equation for R, the number of retransmissions required to achieve a resiliency q is as follows:

R = log(1-q^(1/F)) / log(1-p)

For example, if the server estimates that there is a 10% loss rate to clients (p=.9) and wishes to transmit a 10-fragment alert (F=10) with 99% reliability (q=.99), then it should transmit the entire sequence of alert fragments at least 3 times (R=2.998).

6. Client Processing

LEAP clients reassemble alert fragments from alert servers in order to obtain a complete alert. A LEAP client maintains a set of alert buffers (possibly empty) to hold fragments of incomplete alerts. Each buffer is identified by the IP address of the alert server and the 16-bit alert ID of the alert being reassmbled. Each alert buffer contains the following data elements:

- IP address of the alert server
- Alert ID for this alert
- Number of fragments in this alert
- List of fragment numbers that have been received
- List of fragment bodies that have been received

A LEAP client processes an incoming LEAP datagram according to the following steps:

1. Search for an existing alert buffer that matches this datagram's IP address and alert ID
2. If there is no current alert buffer, initialize one with the following values:
    * IP address: The source IP address of the incoming datagram
    * Alert ID: The alert ID from the LEAP header in the incoming datagram
    * Number of fragments: The fragment count from the LEAP header in the incoming datagram
    * Received fragment number list: A one-element list containing the fragment number from the LEAP header in the incoming datagram

* Received fragment body list: A one-element list containing the fragment body in the incoming datagram

3. If there is a current alert buffer, add this datagram to the buffer:

   A. If the fragment count field in the datagram differs from the fragment count field in the buffer, discard the datagram

   B. Add the fragment number from the incoming datagram to the list of fragment numbers

   C. Add the fragment body from the incoming datagram to the list of fragment bodies

   D. If all fragments have been received, re-assemble the fragment bodies in order by fragment number and return the reassembled alert

In order to limit the amount of state that needs to be stored, clients SHOULD apply access controls before accepting incoming datagrams and limit the time that an individual buffer is stored. When a client has been configured with local alert servers (e.g., using the Alert Metadata Protocol [I-D.barnes-atoca-meta]), then it SHOULD only accept LEAP datagrams from configured servers.

Clients MUST apply a buffer timeout T1 to incoming alerts. If all fragments for a buffer do not arrive within T1 milliseconds, then the buffer is discarded. The RECOMMENDED default value for T1 is 5000 milliseconds.

Clients MAY also impose an absolute limit on the number of buffers they will store at one time, although this may cause them to miss a legitimate alert if an attacker sends many false alerts. If a client wishes to limit the number of buffers stored, it SHOULD place limits on a per-IP-address basis, rather than on a global basis. This will prevent attackers from creating many buffers, but still allow a legitimate alert server to transmit the few alerts that it needs to get through.

7. IANA Considerations

[TODO: Request a default port number]

[TODO: Register URI scheme]

8. Security Considerations

The primary risk for alerting systems is the introduction of false alert information, either by injecting false alerts or by modifying valid alerts. This protocol addresses these risks by using the authentication and integrity features of the ESCAPE alert format [I-D.barnes-atoca-escape].

The main security concern for this protocol is denial of service on the client, both in the sense of resource exhaustion and in the sense of preventing legitimate alerts from arriving. Clients are required to maintain state, so there is a risk that this state will be exhausted. Rejecting LEAP datagrams based on resource limits, however, can lead to legitimate alert datagrams being dropped.

Several DOS mitigations are described in Section 6 above. The LEAP protocol itself also imposes an absolute upper bound on the amount of data stored per source IP address. Due to the limited set of alert IDs and fragment numbers available, the worst-case amount of buffer is 2^24 times the link MTU, for example 4GB for a 1KB MTU. An attacker can only force a client to accept more data than this by spoofing IP addresses or sending alerts from multiple hosts.

As discussed above, clients SHOULD apply resource constraints to limit the amount of state that an attacker can require a client to store. These resource contraints must be constructed so that legitimate alerts are still likely to get through. Since there is no authentication in LEAP, it is not possible to apply access controls based on cryptographic credentials. But if alert server IP addresses can be pre-provisioned, then the client can choose to accept datagrams only from those IP addresses. Limiting resources on a per-IP-address basis also increases the likelihood that legitimate alerts will be received. While attackers may try to send many alerts simultaneously in order to exhaust resources, real alert servers are much more likely to only send a few alerts at any given time.

9. Acknowledgements

Thanks to Martin Thomson, Brian Rosen, Hannes Tshofenig for help in developing and refining the ideas in this document.

10. References

10.1. Normative References

[I-D.barnes-atoca-escape]
Barnes, R., "Encoding of Secure Common Alert Protocol Entities (ESCAPE)", draft-barnes-atoca-escape-00 (work in progress), October 2011.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[I-D.barnes-atoca-meta]
Barnes, R., "Alert Metadata Protocol (AMP)", draft-barnes-atoca-meta-00 (work in progress), October 2011.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

Author’s Address

Richard Barnes
BBN Technologies
9861 Broken Land Parkway
Columbia, MD 21046
US

Phone: +1 410 290 6169
Email: rbarnes@bbn.com