Network Working Group
Internet-Draft
Expires: March 31, 2004

A. Barbir
Nortel Networks

Oct. 2003

OPES entities and end points communication
draft-ietf-opes-end-comm-05

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on March 31, 2004.

Copyright Notice

Abstract

This memo documents tracing and non-blocking (bypass) requirements for Open Pluggable Edge Services (OPES).

Table of Contents

1. Introduction

The Open Pluggable Edge Services (OPES) architecture [8] enables cooperative application services (OPES services) between a data provider, a data consumer, and zero or more OPES processors. The application services under consideration analyze and possibly transform application-level messages exchanged between the data provider and the data consumer.

This work specifies the requirements for providing tracing functionality for the OPES architecture [8]. Tracing functionality enables a data provider or a data consumer application to detect inappropriate actions that are performed by OPES entities. The work also develops requirements that can be used to fulfill IAB Notification and Bypass (Non-Blocking) requirements [2].

The architecture document requires [8] that tracing is supported in-band. This design goal limits the type of application protocols that OPES can support. The details of what a trace record can convey are also dependent on the choice of the application level protocol.

For these reasons, this work documents requirements for application protocols that need to support OPES traces and non-blocking mechanism. The architecture does not prevent implementers of developing out-of-band protocols and techniques to address these limitation. In this document the key words that are used to signify requirements are based on RFC 2119 [3].

2. OPES System

This sections provides a definition of OPES System. This is needed in order to define what is traceable in an OPES Flow.

Definition: An OPES System is a set of all OPES entities [8] authorized by either the data provider or the data consumer application to process a given application message.

The nature of the authorization agreement determines if authority delegation is transitive (meaning an authorized entity is authorized to include other entities).

If specific authority agreements allow for re-delegation, an OPES system can be formed by induction. In this case, an OPES system starts with entities directly authorized by a data provider (or a data consumer) application. The OPES system then includes any OPES entity authorized by an entity that is already in the OPES system. The authority delegation is always viewed in the context of a given application message.

An OPES System is defined on an application message basis. Having an authority to process a message does not imply being involved in message processing. Thus, some OPES system members may not participate in processing of a message. Similarly, some members may process the same message several times.

The above definition implies that there can be no more than two OPES systems [Client-side and server-side OPES systems can process the same message at the same time] processing the same message at a given time. This is based on the assumption that there is a single data provider and a single data consumer as far as a given application message is concerned.

For example, consider a Content Delivery Network (CDN) delivering an image on behalf of a busy web site. OPES processors and services that the CDN uses to adapt and deliver the message comprise an OPES System. In a more complex example, an OPES System would contain CDN entries as well as 3rd-party OPES entities that CDN engages to perform adaptations (e.g., to adjust image quality).

3. Requirements for OPES Tracing

In an OPES System tracing is defined as the inclusion of necessary information within a message in an OPES Flow that identify the collection of transformations or adaptations that have been performed on it before its delivery to an end point (for example, the data consumer application). An OPES trace represents a snap shot of the tracing information that have been added to a given application message. A trace represents the collections of transformations on an application message in the order that were performed. A traceable entity can appear multiple times in a trace (every time it acts on a message).

In an OPES System tracing is performed on per message basis. Trace format is dependent on the application protocol that is being adapted by OPES. A data consumer application can use OPES trace to infer the actions that have been performed by the OPES System. Actions are the set of OPES services that were performed by OPES entities in an OPES System.

In an OPES System, the task of providing tracing information, can depend on many factors. Some considerations are:

- Providers may be hesitant to reveal information about their internal network infrastructure.

- Within a service provider network, OPES processors may be configured to use non-routable, private IP addresses.

- A data consumer applications would prefer to have a single point of contact regarding the trace information.

In an OPES System some OPES services are message-agnostic and operate on message content or parts of a message. Such services do not manipulate message headers. Other services can manipulate message headers. OPES providers require some freedom in the way they deliver tracing information to an end point.

3.1 What is traceable in an OPES Flow?

This section focuses on identifying traceable entities in an OPES Flow.

Tracing information provides a data consumer application (or a data provider application) with information about OPES entities that adapted the data. There are two distinct uses of OPES traces. First, a trace enables an "end (content provider or consumer) to detect the presence of OPES processors within an OPES System. Such "end" should

be able to see a trace entry, but does not need to be able to interpret it beyond identification of the OPES System.

Second, the OPES System administrator is expected to be able to interpret the contents of an OPES trace. The trace can be relayed to the administrator by an end (data consumer or provider) without interpretation, as opaque data (e.g., a TCP packet or an HTTP message snapshot). The administrator can use the trace information to identify the participating OPES entities. The administrator can use the trace to identify the applied adaptation services along with other message-specific information.

Since the administrators of various OPES Systems can have various ways of looking into tracing, they require the choice of freedom in what to put in trace records and how to format them. Trace records should be easy to extend beyond basic OPES requirements. Trace management algorithms should treat trace records as opaque data to the extent possible.

At the implementation level, for a given trace, an OPES entity involved in handling the corresponding application message is traceable or traced if information about it appears in that trace. OPES entities have different levels of traceability requirements. Specifically,

- An OPES processor MUST add its entry to the trace.

- An OPES service MAY add its entry to the trace.

- An OPES entity MAY delegate addition of its trace entry to another OPES entity. For example, an OPES System can have a dedicated OPES processor for adding System entries; an OPES processor can use a callout service to manage all OPES trace manipulations (since such manipulations are OPES adaptations).

In an OPES context, a good tracing approach is similar to a trouble ticket ready for submission to a known address. The address is printed on the ticket. The trace in itself is not necessarily a detailed description of what has happened. It is the responsibility of the operator to resolve the problems.

3.2 Requirements for OPES System

The following requirements apply for information as related to an OPES System:

- An OPES System MUST trace itself.

- An OPES System MUST include information about its privacy policy.

- An OPES System MUST identify the party responsible for setting and enforcing that policy.

- An OPES System MUST include information pointing to a technical contact.

- An OPES System MUST include information that identifies, to the technical contact, the OPES processors involved in processing the message.

- When providing required information, an OPES System MAY use a single URI to identify a resource containing several required items. For example, an OPES System can point to a single web page with a reference to System privacy policy and technical contact information.

This specification does not define the meaning of the terms privacy policy, policy enforcement, or technical contact and contains no requirements regarding encoding, language, format, or any other aspects of that information. Furthermore, an example of System identification would be something like http://www.examplecompany.com/opes/?client=example.com.

3.3 Requirements for OPES processors

Tracing requirements for OPES processors are:

- OPES processor MUST add its unique identification to the trace. Here, uniqueness scope is the OPES System containing the processor.

- OPES processor MUST be able to trace it's own invocation and service(s) execution.

3.4 Requirements for callout servers

In an OPES system, it is the task of an OPES processor to add trace records to application messages. However, in some cases, callout servers may add trace information to application messages. This should be done under the control of the OPES System provider.

4. Requirements for OPES System Bypass (Non-blocking feature)

IAB recommendation (3.3) [2] requires that the OPES architecture does not prevent a data consumer application from retrieving non-OPES version of content from a data provider application, provided that the non-OPES content exists. IAB recommendation (3.3) suggests that the Non-blocking feature (Bypass) be used to bypass faulty OPES intermediaries (once they have been identified, by some method).

In addressing IAB consideration (3.3), one need to specify what constitute non-OPES content. In this work the definition of "non-OPES" content is provider dependent. However, in some cases, the availability of "non-OPES" content can also be a function of the internal policy of a given organization that has contracted the services of an OPES provider. For example, Company A has as contract with an OPES provider to perform virus checking on all e-mail attachments. An employee X of Company A can issue a non-blocking request for the virus scanning service. However, the request could be ignored by the OPES provider since it contradicts its agreement with Company A.

The above examples illustrates that the availability of non-OPES content can be a function of content providers (or consumers or both) policy and deployment scenarios [1]. For this reason, this work does not attempt to define what is an OPES content as opposed to non-OPES content. The meaning of OPES versus non-OPES content is assumed to be determined through various agreements between the OPES provider, data provider and data consumer application. The agreement will also determine what OPES services can be bypassed and in what order (if applicable).

In an OPES System a bypass request is defined as the act of avoiding the invocation of a service(s) that is identified by a URI within a message in an OPES Flow before its delivery to an end point (for example, the data consumer application). The availability of non-OPES content is a precondition to whether a bypass instruction is honored or not in an OPES System.

4.1 What can be bypassed in an OPES Flow?

In this work, the focus is on developing a bypass feature that allow a user to instruct the OPES System to bypass some or all of its services. The collection of OPES services that can be bypassed is a function of the agreement of the OPES provider with either (or both) the content provider or the content consumer applications. In the general case, a bypass request is viewed as a bypass instruction that contains a URI that identifies an OPES entity or a group of OPES entities that perform a service (or services) to be bypassed. An

instruction may contain more than one such URI. A special wildcard identifier can be used to represent all possible URIs (i.e., all possible OPES services that are relevant to an OPES processor).

In an OPES Flow, a bypass request is processed in a local manner by each involved OPES processor. This means that an OPES processor examines the bypass instruction and if non-OPES content is available, the processor then bypasses services that are local to itself. This may include the act of bypassing itself completely depending on what is specified in the URI. The request is then forwarded to the next OPES processor in the OPES Flow. The next OPES processor would then honor all bypass requests that are relevant to it provided that the non-OPES content is available. The processing chain continues throughout the whole processors that are involved in the OPES Flow. If an OPES processor does not know how to honor a bypass request it forwards the message to the next processor in the OPES Flow. At the OPES system level, a bypass instruction is honored when at least one OPES processor bypasses the services that are specified by the URI that is specified in the instruction (provided that the non-OPES content is available).

4.2 Bypass requirements for OPES System

In an OPES System bypass requests are generally client centric and go in the opposite direction of tracing requests. Bypass can be performed out of band or in-band. This work requires that the Bypass feature be performed in-band as an extension to an application specific protocol. Non-OPES entities should be able to safely ignore these extensions. The work does not prevent OPES Systems from developing their own out of band protocols.

The following requirements apply for Bypass feature as related to an OPES System (the availability of a non-OPES content is a precondition) :

- An OPES system MUST support a Bypass feature. This means that the OPES System bypasses an entity whose URI is identified by an OPES end (usually data consumer application).

In order to facilitate the debugging (or data consumer user experience) of the bypass feature in an OPES System, it would be beneficial if non-bypassed entities include information related to why they ignored the bypass instruction. It is important to note that in some cases the tracing facility itself may be broken and the whole OPES System (or part) may need to be bypassed through the issue of a bypass instruction.

4.3 Bypass requirements for OPES processors

For a given application protocol, in an OPES System there can be services that operate on application message headers and those that just operate on content. This mix of service requires that an OPES processor that is calling the service(s) to handle the bypass request. In some cases, the first OPES processor that will get the bypass request may not be the first OPES processor that will know whether a non-OPES version of the content is available or not.

Bypass requirements for OPES processors are (the availability of a non-OPES content is a precondition):

- o OPES processor SHOULD be able to interpret and process a bypass instruction. This requirement applies to all bypass instructions, including those that identify known-to-recipient services.

- o OPES processors that do not know how to process a bypass request MUST forward the request to the next application hop provided that the next hop speaks application protocol with OPES bypass support.

- o OPES processor SHOULD be able to bypass it's service(s) execution.

Provided that non-OPES content is available, those OPES processors that know how to process and interpret a bypass instruction have the following requirements:

- o The recipient of a bypass instruction with a URI that does not identify any known-to-recipient OPES entity MUST treat that URI as a wildcard identifier (meaning bypass all applicable local services).

4.4 Bypass requirements for callout servers

In an OPES system, it is the task of an OPES processor to process bypass requests. However, in some cases, callout servers may be involved in processing Bypass requests. This should be done under the control of the OPES System provider.

5. Protocol Binding

The task of encoding tracing and bypass features is application protocol specific. Separate documents will address HTTP and other protocols.

6. Compliance Considerations

This work specifies high level requirements for tracing and bypass. Protocol binding specifications MUST consider and follow all MUSTs in this draft. Protocol binding specifications MUST be compliant to this draft. As such any implementation compliant to the binding specification is also compliant to this draft.

7. IANA considerations

This specification contains no IANA considerations. Application bindings MAY contain application-specific IANA considerations.

8. Security Considerations

The security considerations for OPES are documented in [7]. This document is a requirement document for tracing and Bypass feature. The requirements that are stated in this document can be used to extend an application level protocol to support these features. As such, the work has security precautions.

8.1 Tracing security considerations

The tracing facility for OPES architecture is implemented as a protocol extension. Inadequate implementations of the tracing facility may defeat safeguards built into the OPES architecture. The tracing facility by itself can become a target of malicious attacks or used to lunch attacks on an OPES System.

Threats caused by or against the tracing facility can be viewed as threats at the application level in an OPES Flow. In this case, the threats can affect the data consumer and the data provider application.

Since tracing information is a protocol extension, these traces can be injected in the data flow by non-OPES entities. In this case, there are risks that non-OPES entities can be compromised in a fashion that threat the overall integrity and effectiveness of an OPES System. For example, a non-OPES proxy can add fake tracing information into a trace. This can be done in the form of wrong, or unwanted, or non existent services. A non-OPES entity can inject large size traces that may cause buffer overflow in a data consumer application. The same threats can arise from compromised OPES entities. An attacker can control an OPES entity and inject wrong, or very large trace information that can overwhelm an end or the next OPES entity in an OPES flow. Similar threats can result from bad implementations of the tracing facility in trusted OPES entities.

Compromised tracing information can be used to launch attacks on an OPES System that give the impression that unwanted content transformation was performed on the data. This can be achieved by inserting wrong entity (such OPES processor) identifiers. A compromised trace can affect the overall message integrity structure. This can affect entities that use message header information to perform services such as accounting, load balancing, or reference-based services.

Compromised trace information can be used to launch DoS attacks that can overwhelm a data consumer application or an OPES entity in an OPES Flow. Inserting wrong tracing information can complicates the debugging tasks performed by system administrator during trouble

shooting of OPES System behavior.

As a precaution, OPES entities ought to be capable of verifying that the inserted traces are performed by legal OPES entities. This can be done as part of the authorization and authentication face. Policy can be used to indicate what trace information can be expected from a peer entity. Other application level related security concerns can be found in [7].

8.2 Bypass security considerations

The bypass facility for OPES architecture is implemented as a protocol extension. Inadequate implementations of the bypass facility may defeat safeguards built into the OPES architecture. The bypass facility by itself can become a target of malicious attacks or used to lunch attacks on an OPES System.

Threats caused by or against the bypass facility can be viewed as threats at the application level in an OPES Flow. In this case, the threats can affect the data consumer and the data provider application.

There are risks for the OPES System by non-OPES entities, whereby, these entities can insert bypass instructions into the OPES Flow. The threat can come from compromised non-OPES entities. The threat might affect the overall integrity and effectiveness of an OPES System. For example, a non-OPES proxy can add bypass instruction to bypass legitimate OPES entities. The attack might result in overwhelming the original content provider servers, since the attack essentially bypass any load balancing techniques. In addition, such an attack is also equivalent to a DoS attack, whereby, a legitimate data consumer application may not be able to access some content from a content provider or its OPES version.

Since an OPES Flow may include non-OPES entities, it is susceptible to man-in-the-middle attacks, whereby an intruder may inject bypass instructions into the data path. These attacks may affect content availability or disturb load balancing techniques in the network.

The above threats can also arise by compromised OPES entities. An intruder can compromise an OPES entities and then use man-in-the-middle techniques to disturb content availability to a data consumer application or overload a content provider server (essentially, some form of a DoS attack).

Attackers can use the bypass instruction to affect the overall integrity of the OPES System. The ability to introduce bypass instructions into a data flow may effect the accounting of the OPES

System. It may also affect the quality of content that is delivered to the data consumer applications. Similar threats can arise from bad implementations of the bypass facility.

Inconsistent or selective bypass is also a threat. Here, one end can try to bypass a subset of OPES entities so that the resulting content is malformed and crashes or compromises entities that process that content (and expect that content to be complete and valid). Such exceptions are often not tested because implementers do not expect a vital service to disappear from the processing loop.

Other threats can arise from configuring access control policies for OPES entities. It is possible that systems implementing access controls via OPES entities may be incorrectly configured to honor bypass and, hence, give unauthorized access to intruders.

Tap bypass can also be a threat. This is because systems implementing wiretaps via OPES entities may be incorrectly configured to honor bypass and, hence, ignore (leave undetected) traffic with bypass instructions that should have been tapped or logged. It is also possible for one end to bypass services such as virus scanning at the receiving end. This threat can be used by hackers to inject viruses throughout the network.

Other application level related security concerns can be found in [7].

Normative References

[1] A. Barbir et al., "OPES Use Cases and Deployment Scenarios", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-scenarios-01.txt, August 2002.

Informative References

[2] Floyd, S. and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services", RFC 3238, January 2002.

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

[4] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[5] A. Barbir et al., "Policy, Authorization and Enforcement Requirements of OPES", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-authorization-02.txt, February 2003.

[6] Rousskov, A. et al, "HTTP adaptation with OPES", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-http-00.txt, July 2003.

[7] A. Barbir et al., "Security Threats and Risks for Open Pluggable Edge Services", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-threats-02.txt, February 2003.

[8] A. Barbir et al., "An Architecture for Open Pluggable Edge Services (OPES)", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-architecture-04, December 2002.

[9] A. Barbir et al., "OPES Treatment of IAB Considerations", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-iab-02.txt, September 2003.

Author's Address

Abbie Barbir
Nortel Networks
3500 Carling Avenue
Nepean, Ontario K2H 8E9
Canada

Phone: +1 613 763 5229
EMail: abbieb@nortelnetworks.com

Appendix A. Acknowledgements

Several people has contributed to this work. Many thanks to: Alex Rousskov, Hilarie Orman, Oscar Batuner, Markus Huffman, Martin Stecher, Marshall Rose and Reinaldo Penno.

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.