Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 25, 2014

P. Hallam-Baker
Comodo Group Inc.

January 21, 2014

# OmniBroker Protocol
# draft-hallambaker-omnibroker-07

Abstract

An Omnibroker is an agent chosen and trusted by an Internet user to provide information such as name and certificate status information that are in general trusted even if they are not trustworthy. Rather than acting as a mere conduit for information provided by existing services, an Omnibroker is responsible for curating those sources to protect the user.

The Omnibroker Protocol (OBP) provides an aggregated interface to trusted Internet services including DNS, OCSP and various forms of authentication service. Multiple transport bindings are supported to permit efficient access in virtually every common deployment scenario and ensure access in any deployment scenario in which access is not being purposely denied.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on July 25, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

## 1. Definitions

### 1.1. Requirements Language

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].

## 2. Purpose

Today, a network client is required to make queries against multiple information sources to establish a secure connection to a network resource. A DNS query is required to translate network names to Internet addresses. If TLS transport is used, an OSCP query may be required to validate the server certificate. Support for client authentication may require interaction with another service.

Servers require similar support when accepting Internet connections. Even though most networking infrastructure supports some form of network administration, it is left to the network administrator to fill in the gap between server applications and network infrastructure. Making use of such facilities is rarely cost effective except at the very largest installations.

An Omnibroker is a trusted agent that acts as a single point of service for client queries requesting a connection to a named network resource and server advertisements accepting connections to a named network resource.

### 2.1. Omnibroker Implementation

Omnibroker is implemented as a JSON/REST Web service using Jason Connect (JCX) to establish and manage the long term trust relationship with the Omnibroker provider.

#### 2.1.1. Establishing service

In normal use, an omnibroker client receives service from a single Omnibroker service provider. For performance and reliability reasons, an Omnibroker service provider is expected to provide multiple Omnibroker service instances.

An Omnibroker client acquires the network address information and credentials necessary to access an omnibroker service using the JCX

Web Service to establish a connection binding. To ensure reliabilty and the ability to access the service in all circumstances, an Omnibroker connection binding SHOULD specify multiple service instances.

### 2.1.2. Protocol Bindings

Due to the need for low latency and the need to function in a compromised network environment, three protocol bindings are defined:

A HTTP binding using HTTP [RFC2616] for session layer framing and HTTP Session Continuation [I-D.hallambaker-httpsession] for message authentication and JSON encoding [RFC4627] of protocol messages.

A UDP Binding using JSON-B [I-D.hallambaker-jsonbcd]for framing and encoding of protocol messages.

A DNS Binding using DNS [RFC4627] TXT record queries with Base32 and Base64 encoding for transport and framing and JSON-B for encoding of protocol messages.

The implementation overhead of support for three different protocol bindings is reduced by the choice of a binary encoding for JSON (JSON-B) that is very close in structure to JSON encoding allowing encoders and decoders to support both encodings with minimal additional code.

Regardless of the protocol binding used, all Omnibroker messages are authenticated with protection against replay attack under the cryptographic credentials established in the connection binding service instance.

## 2.2. Omnibroker Query Service

Directing queries through a single point of contact has performance, relability and security advantages. Directing queries to multiple network information sources degrades performance and may cause a connection request to fail if an information resource is not available. This has led many application providers to limit the information sources they consult. Directing queries through an Omnibroker allows as many information sources to be brought to bear as the broker has local cached data for without loss of performance or reliability.

Making use of additional data sources allows the broker to 'curate' the response. If the broker knows that a Web site always returns a redirect to a TLS secured version of the same site, it can tell a Web

Browser to go straight to the secure version. If a Web Server is hosted on a known botnet, the Omnibroker can tell the client that it really does not want to visit that location.

Unlike the traditional DNS configuration, an Ombinbroker client decides which source(s) of trusted information to use rather than relying on whatever happens to be the nearest source to hand.

The traditional DNS approach creates an obvious security risk as DNS is a trusted service and deciding to choose a random DNS service advertised by the local DHCP service is clearly a poor decision process for a trusted service.

## 2.3. Omnibroker Advertisement Service

Directing service advertisements to a single point of contact has similar benefits. The single point of contact can take responsibility for managing load balancing. Firewall and router configurations can be set automatically. Anti-abuse technologies such as IP address filters and rate limiting devices can be switched in as required.

In simple network configurations such as a residential

## 2.4. Walled Gardens

IETF culture has traditionally resisted attempts to establish partitions within the open Internet with restricted access to network resources or compromised security. Such 'Walled Gardens' models typically exist for the benefits of those who own the walls rather than those forced to live inside them.

While virtually all residential Internet users reject such controls, most find them acceptable, if not desirable in workplaces and schools.

Omnibroker simplifies the process of establishing such a walled garden but does not make the walls any easier to defend.

### 2.4.1. Censorship

From a censorship point of view, the censorship concerns of running an Omnibroker are essentially the same as those of running a DNS service. The party who decides which discovery service to use can determine which content is visible to the users.

### 2.4.2. Trust Substitution

Like SCVP [RFC5055] and XKMS [TBS] , Omnibroker permits an Internet client to delegate some or all aspects of PKIX [RFC5280] certificate path chain discovery and validation.

In the normal mode of operation, the Omnibroker service performs only path chain discovery, leaving the client to re-check the PKIX certificate path before relying on it. This gives the Omnibroker the power to veto a client connection to a server that it considers to be unsafe but not the power to tell the client to trust a site of its own choosing.

This ability to veto but not assert trust is appropriate and sufficient for the vast majority of network applications. It allows the broker to make use of additional path validation checks that are not supported in the client such as DANE [RFC6698] or Certificate Transparency [RFC6962].

There are however some workplace environments where the ability to access external network resources with strong encryption is not permissible by enterprise policy or in some cases by law. An intelligence analyst working at the NSA may have a need to access external Web sites that contain important information but must on no account have access to a covert channel that could be used to exfiltrate information. Certain Financial institutions with access to valuable commercial information are required to monitor and record all communications into and out of the company to deter insider trading.

The traditional response to such needs has been to tell the parties affected to look elsewhere for support. As a consequence the techniques used to satisfy such requirements are generally unfriendly to network applications in general and have in some cases put the public Web PKI trust infratructure at risk.

There is an argument to be made that rather than attempting to prohibit such activities entirely, it would be better to provide a principled method of achieving those ends and for mainstream software providers to support it in such a fashion that ensures that network applications configured for that mode of use can be readilly identified as such by end users.

### 2.4.3. Censorship Bypass

As the preceeding examples demonstrate, a party with control over the Omnibroker service chosen by a user has full control over the network activities of that user. An important corrolary of this fact is that

all a user need do to achieve full control over their network activities is to run their own Omnibroker service and connect to that.

For example such an Omnibroker service might be configured to return connection data for permitted domestic Web sites as normal but direct attempts to connect to forbidden foreign news or social media through a privacy network such as TOR.

## 3. Transport Bindings

To achieve an optimal balance of efficiency and availability, three transport bindings are defined:

Supports all forms of OBP transaction in all network environments.

Provides efficient support for a subset of OBP query transactions that is accessible in most network environments.

Provides efficient support for all OBP query transactions and is accessible in most network environments.

Support for the HTTP over TLS binding is REQUIRED.

An OBP message consists of three parts:

Ticket [As necessary] If specified, identifies the cryptographic key and algorithm parameters to be used to secure the message payload.

Payload [Required] If the ticket context does not specify use of an encryption algorithm, contains the message data. Otherwise contains the message data encrypted under the encryption algorithm and key specified in the ticket context.

Authenticator [Optional] If the ticket context specifies use of a Message Authentication Code (MAC), contains the MAC value calculated over the payload data using the authentication key bound to the ticket.

Note that although each of the transport bindings defined in this specification entail the use of a JSON encoding for the message data, this is not a necessary requirement for a transport binding.

### 3.1. JSON Payload Binding

Protocol schema types are mapped to JSON encoding as follows:

Integer Data of type Integer is encoded using the JSON number encoding.

Name Data of type Name is encoded using the JSON string encoding.

String Data of type String is encoded using the JSON string encoding.

Binary Data of type Binary is converted to strings using the Base64url encoding specified in [RFC4648] and encoded using the JSON string type.

DateTime Data of type DateTime is converted to string using the UTC time conversion specified in [RFC3339] with a UTC offset of 00:00.

### 3.2. HTTP over TLS

OBP requests and responses are mapped to HTTP POST requests and responses respectively. Java Script Object Notation (JSON) encoding is used to encode requests and responses.

#### 3.2.1. Message Encapsulation

Requests and responses are mapped to HTTP POST transactions. The content of the HTTP message is the message payload. The Content-Type MUST be specified as application/json. The Character set MUST be specified as UTF-8.

The Ticket and Authenticator are specified using the Integrity header as follows:

Session: <base64url (authenticator)> ; ticket=<base64url (ticket)>

#### 3.2.2. Example

[To be generated from spec]

### 3.3. DNS Tunnel

The DNS Tunnel mode of operation makes use of DNS TXT resource record requests and responses to tunnel OBP Query requests. Due to the constraints of this particular mode of operation, use of this transport is in practice limited to supporting transactions that can be expressed within 500 bytes. These include the QueryConnect and ValidateRequest interactions.

### 3.3.1. Request

Requests are mapped to DNS TXT queries. The request is mapped onto the DNS name portion of the query by encoding the Ticket, Authenticator and JSON encoded Payload using Base32 encoding and appending the result to the service prefix to create a DNS name as follows:

<base32(payload)>.<base32(authenticator)>.<base32(ticket)>.Suffix

The payload MAY be split across multiple DNS labels at any point.

### 3.3.2. Response

Responses are mapped to DNS TXT records by encoding the Authenticator and JSON encoded Payload using Base64 encoding and cocatenating the result with a periods as a separator as follows:

<base32(payload)>.<base32(authenticator)>

### 3.3.3. Example

[To be generated from spec]

## 3.4. UDP

The UDP transport MAY be used for transactions where the request fits into a single UDP packet and the response can be accomadated in 16 UDP packets. As with the Web Service Binding, Java Script Object Notation (JSON) encoding is used to encode requests and responses.

### 3.4.1. Request

The request consists of four message segments containing a Header, Ticket, Payload and Authenticator. Each message segment begins with a two byte field that specified the length of the following data segment in network byte order. The Payload is encoded in JSON encoding and the remaining fields as binary data without additional encoding.

The header field for this version of the protocol (1.0) contains two bytes that specify the Major and Minor version number of the transport protocol being 1 and 0 respectively. Future versions of the transport protocol MAY specify additional data fields.

[TBS diagram]

### 3.4.2. Response

The response consists of a sequence of packets. Each packet consists of a header section and a data section.

The header section consists of a two byte length field followed by two bytes that speciofy the Major and Minor version number of the transport protocol (1 and 0), two bytes that specify the frame number and the total number of frames and two bytes that specify the message identifier.

[TBS diagram]

[Question, should the authenticator be over the whole message or should each packet have its own authenticator?]

### 3.4.3. Example

[To be generated from spec]

## 4. Acknowledgements

[List of contributors]

## 5. Security Considerations

### 5.1. Denial of Service

### 5.2. Breach of Trust

### 5.3. Coercion

## 6. To do

The specification should define and use a JSON security object.

Formatting of the abstract data items needs to be improved

Need to specify the UDP transport binding

Should specify how each data item is represented in JSON format somewhere. This is obvious for some of the data types but needs to be fully specified for things like DateTime.

Run the code to produce proper examples.

Write an API document

## 7. IANA Considerations

[TBS list out all the code points that require an IANA registration]

## 8. References

### 8.1. Normative References

[I-D.hallambaker-httpsession]
Hallam-Baker, P., "HTTP Session Management", draft-hallambaker-httpsession-01 (work in progress), May 2013.

[I-D.hallambaker-jsonbcd]
Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", draft-hallambaker-jsonbcd-00 (work in progress), June 2013.

[I-D.hallambaker-wsconnect]
Hallam-Baker, P., "JSON Service Connect (JCX) Protocol", draft-hallambaker-wsconnect-02 (work in progress), June 2013.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5055] Freeman, T., Housley, R., Malpani, A., Cooper, D., and W. Polk, "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, December 2007.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[X.509] International Telecommunication Union, "ITU-T Recommendation X.509 (11/2008): Information technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, November 2008.

[X.680] International Telecommunication Union, "ITU-T Recommendation X.680 (11/2008): Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, November 2008.

### 8.2. Non Normative References

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, June 2013.

## Appendix A. Example Data.

### A.1. Ticket A

### A.2. Ticket B

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com