Internet Engineering Task Force	P. Hallam-Baker
Internet-Draft	Comodo Group Inc.
Intended status: Standards Track	May 17, 2013
Expires: November 18, 2013	

TOC

OmniBroker Protocol
draft-hallambaker-omnibroker-03

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

Copyright Notice

Table of Contents

1. Definitions
    1.1. Requirements Language
2. Purpose
    2.1. A Curated Service
    2.2. Connection Broker
        2.2.1. Service Connection Broker
        2.2.2. Peer Connection Broker
        2.2.3. Connection Broker API
    2.3. Service Advertisement
        2.3.1. Connection Advertisement API

2.4. Credential Validation Broker
2.5. Authentication Gateway
2.6. Credential Announcement
3. Omnibroker Query Service
3.1. Illustrative example
3.2. OBPQuery
3.2.1. Message: Payload
3.2.2. Message: Message
3.2.3. Message: Request
3.2.4. Message: Response
3.2.5. Structure: Identifier
3.2.6. Structure: Connection
3.2.7. Structure: Advice
3.2.8. Structure: Service
3.2.9. QueryConnect
3.2.10. Message: QueryConnectRequest
3.2.11. Message: QueryConnectResponse
3.2.12. Advertise
3.2.13. Message: AdvertiseRequest
3.2.14. Message: AdvertiseResponse
3.2.15. Validate
3.2.16. Message: ValidateRequest
3.2.17. Message: ValidateResponse
3.2.18. QueryCredentialPassword
3.2.19. Message: CredentialPasswordRequest
3.2.20. Message: CredentialPasswordResponse
4. Transport Bindings
4.1. HTTP over TLS
4.1.1. Message Encapsulation
4.1.2. Example
4.2. DNS Tunnel
4.2.1. Request
4.2.2. Response
4.2.3. Example
4.3. UDP
4.3.1. Request
4.3.2. Response
4.3.3. Example
5. Acknowledgements
6. Security Considerations
6.1. Denial of Service
6.2. Breach of Trust
6.3. Coercion
7. To do
8. IANA Considerations
9. Normative References
Appendix A. Example Data.
A.1. Ticket A
A.2. Ticket B
§ Author's Address

TOC

1. Definitions

TOC

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].

TOC

TOC

2. Purpose

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 and providing autheticated, curated results to the endpoint client.

Unlike the traditional trusted information services that are expected to deliver the same response to a query regardless of who asks the question, OBP permits an Omnibroker to return a response that is tailored to the specific party asking the question. This permits the use of an authentication approach that has negligible impact on performance and permits an Omnibroker to answer questions that a traditional public Internet information service could not. In particular, an Omnibroker can broker peer to peer connections

---

TOC

2.1. A Curated Service

In the traditional configuration, an Internet host accepts DNS service from the IP address specified by the local network provider, frequently the DNS server advertised by the local DHCP service. This approach creates an obvious security risk as DNS is clearly a trusted service and a random DNS service advertised by the local DNS is clearly not trustworthy.

A policy of only using a chosen DNS service provides a reduction in risk but only a modest one since the standard DNS service does not provide authenticated results. Many local area networks intercept all DNS traffic and process it through the local DNS server regardless of the intended destination IP address. This practice is highly desirable if it would prevent a client from accessing an untrustworthy DNS service in place of a trustworthy local DNS service and extreemly undesirable in the contrary case.

In addition to ensuring the authenticity of DNS resolution responses, such services frequently provide filtering of Internet addresses the network provider considers undesirable. Many workplaces block access to Web sites that are considered detrimental to productivity. Many parent subscribe to services that filter content they consider undesirable. While the value of such services is debatable they are services that those customers have chosen to deploy on their networks to meet their perceived security requirements. New security proposals that do not support such capabilities or seek to actually circumvent them will not be acceptable to those constituencies.

While DNS filtering is a form of censorship, not all forms of DNS filering are intrinsically undesirable censorship. Spam filtering is also a form of censorship albeit one that is not normally regarded as such because it most Internet users now consider it to be an essential security control. Anti-Virus tools are also a form of censorship. DNS filtering tools that block access to sites that distribute malware are also a form of censorship and are rapidly gaining popularity for the same reason.

While all forms of censorship raise civil liberties concerns, censorship should not automatically raise civil liberties objections. It is not the fact that filtering is taking place but the party that is in control of the filtering that should raise civil liberties concerns. In an Internet of 2 billion users, all users are obliged to perform some filtering. OBP is designed to make the party that is in control of the filtering process apparent and provide the end user with the ability to select the Omnibroker of their choice.

DNSSEC provides a means of determining that a DNS record is the authentic record published by the source but this capability alone does not meet the security requirements for DNS resolution services as they have come to be understood since the protocol was first proposed.

Internet users want to be safe from all forms of attack, not just the DNS resolver mani-in-the-middle attack that 'end-to-end' deployment of DNSSEC is designed to address. An Internet user is far more likely to be directed to a site with a DNS name controlled by a criminal gang than be subject to a man-in-the-middle attack.

Most users would prefer to have an Internet connection that is 'curated' to remove at least some of the locations they consider to be undesirable. The fact that an organised criminal gang has put a host on the Internet does not mean that any other Internet user should be obliged to allow it to connect to any of the machines that they own.

The same argument for curating the raw results applies to other forms of trusted information service. The fact that a Certificate Authority has issued a digital certificate and considers it to be valid should not mean that the end party is automatically considered trustworthy by anyone and for any purpose.

The deployment of security policy capabilities presents another case in which direct reliance on raw data is undesirable. While security policies such as 'host always offers TLS' or 'mail server always signs outgoing mail with DKIM' can provide considerable security advantages, only some of the security policy information that is published is accurate and kept up to date. Curating such data sources typically proves essential if an unacceptable rate of false positives is to be avoided.

Although a client is permitted to curate its own data sources it rarely has a sufficient amount of data to make decisions as accurately as a network service that can draw on a wide variety of additional data including tracking of communication patterns, historical data series and third party reputation services.

Curation in the network offers better asgility than the client approach. Agility is an important consideration when an attacker changes their strategy rapidly and repeatedly to counter new defensive controls.

While curating trusted data sources is an established and proven practice, current practice has been to curate each source individually. This approach avoids the need to write a new protocol but limits the information a curator can leverage to detect potential danger. Leveraging multiple data sources simultaneously allows better accuracy than applying each individually.

TOC

2.2. Connection Broker

The OBP service connection broker answers the query 'what connection parameters should be used to establish the best connection to interract with party X according to protocol Y. Where 'best' is determined by the Omnibroker which MAY take into account parameters specified by the relying party.

TOC

2.2.1. Service Connection Broker

The OBP service connection broker supports and extends the traditional DNS resolution service that resolves a DNS name (e.g. www.example.com) to return an IP address (e.g. 10.1.2.3).

When using an Omnibroker as a service connection broker, a client specifies both the DNS name (e.g. www.example.com) and the Internet protocol to be used (e.g. _http._tcp). The returned connection parameters MAY include:

> The IP protocol version, address and port number to establish a connection to.
>
> If appropriate, a security transport such as TLS or IPSEC.
>
> If appropriate, a description of a service credential such as a TLS certificate or a constraint on the type of certificates that the client should consider acceptable.
>
> If appropriate, application protocol details such as version and protocol options.

If an attempt to connect with the parameters specified fails, a client MAY report the failure and request a new set of parameters.

TOC

2.2.2. Peer Connection Broker

Each OBP request identifies both the account under which the request is made and the device from which it is made. An OBP broker is thus capable of acting as a peer connection broker service or providing a gateway to such a service.

When using Omnibroker as a peer connection broker, a client specifies the account name and DNS name of the party with which a connection is to be established (e.g. alice@example.com) and the connection protocol to be used (e.g. _xmpp-client._tcp)

The returned connection parameters are similar to those returned in response to a service broker query.

TOC

2.2.3. Connection Broker API

In the traditional BSD sockets API a network client performs a series of calls to resolve a network name to a list of IP addresses, selects one and establishes an IP connection to a port specified by the chosen application protocol.

OBP anticipates a higher level abstract API that encapsulates this complexity, hiding it from the application code.

In the case that one (or more) OBP services are configured, the library supporting the SHOULD obtain connection parameters from the OBP service. Otherwise, it SHOULD establish a connection using the traditional calling sequence of resolving a host name to obtain an IP address, etc.

TOC

2.3. Service Advertisement

Service advertisement is the converse of service resolution. An Internet application wishing to accept inbound connections specifies one or more sets of connection parameters for the Omnibroker to register with whatever naming, discovery or other services may be appropriate.

TOC

2.3.1. Connection Advertisement API

OBP anticipates the use of a high level API for connection advertisement that is the converse of the Connection broker API. Instead of establishing a connection, the API is used to advertise that a connection is offered either as a service or a peer.

An application MAY offer multiple types of connection with different connection properties (e.g. IPv4/IPv6, with and without SSL, etc.). This MAY be supported by marking certain properties as being optional and/or by permitting the API to be called multiple times with different properties specified.

TOC

2.4. Credential Validation Broker

A credential validation broker reports on the validity and trustworthiness of credentials presented to a client by Internet services and/or peers.

The service provided by OBP is similar to that provided by OCSP and SCVP. Like SCVP, OBP is an agent selected by the relying party to validate certificates and/or construct trust paths on its behalf.

TOC

2.5. Authentication Gateway

Every OBP request is strongly authenticated by means of a shared secret that is unique to the account and the device. This may be leveraged to permit use as an authentication gateway, providing access to other credentials that the client has established the right to use.

An Authentication Gateway MAY provide access to account names and passwords that the account holder has chosen to store in the cloud for access to sites that do not support a stronger, cryptographically based form of authentication such as OAuth.

2.6. Credential Announcement

TOC

An Authentication Gateway can only provide access to credentials that it has notice of. A client uses the Credential Announcement transaction to advise the service of a new credential.

3. Omnibroker Query Service

TOC

3.1. Illustrative example

TOC

[For illustrative purposes, all the examples in this section are shown using the Web Services Transport binding. Examples of other transport bindgins are shown in section [TBS].]

The Alice of the previous example uses her Web browser to access the URL www.example.com://www.example.com/. Assuming this was done while the prior binding was still active (i.e. before the UnbindRequest message was sent), the Web browser would send a QueryConnectRequest request to obtain the best connection parameters for the http protocol at www.example.com:

```
Post / HTTP/1.1
Host: example.com
Cache-Control: no-store
Content-Type: Application/json;charset=UTF-8
Content-Length: 131
Content-Integrity:
  mac=9ZSkLYKFMYenvqt/MwkAtvetqvM7Nydh6Rc2bvbKTbM=;
  ticket=jpBXvI7/0WTmwI2NN4n7Vvw96nbS9LpSsSNMIkdapiUoLikSkjpgMrtbVK
    z5lHOPloCgAyZXxfZpQRsp4oPY4BcRaMw6F5na62IHnBVDeXg=

{
  "QueryConnectRequest": {
    "Identifier": {
      "Name": "http",
      "Service": "www.example.com",
      "Port": 80}}}
```

The service responds with an ordered list of possible connections. In this case the site is accessible via plain TCP transport or with TLS. Since TLS is the preferred protocol, that connection is listed first.

```
HTTP/1.1 200
Content-Type: application/json;charset=UTF-8
Content-Length: 347
Content-Integrity:
  mac=1oFa8fNsRbKiCCnwd4feSqq+h/by+tCLbw2bzf235TU=;
  ticket=jpBXvI7/0WTmwI2NN4n7Vvw96nbS9LpSsSNMIkdapiUoLikSkjpgMrtbVK
    z5lHOPloCgAyZXxfZpQRsp4oPY4BcRaMw6F5na62IHnBVDeXg=

{
  "QueryConnectResponse": {
    "Status": 200,
```

```
    "Connection": [{
        "IPAddress": "10.3.2.1",
        "IPPort": 443,
        "Transport": "TLS",
        "TransportPolicy": "TLS=Optional",
        "ProtocolPolicy": "Strict"},
      {
        "IPAddress": "10.3.2.1",
        "IPPort": 80,
        "ProtocolPolicy": "Strict"}]}}
```

Although the QueryConnectResponse returned the hash of a PKIX certificate considered valid for that connection, the server returns a different certificate which the client verifies using the ValidateRequest query.

```
Post / HTTP/1.1
Host: example.com
Cache-Control: no-store
Content-Type: Application/json;charset=UTF-8
Content-Length: 124
Content-Integrity:
  mac=S+xlW2U6z1REQmbNHiOnAw4xpUXP8wJXZiCJzMzQelc=;
  ticket=jpBXvI7/0WTmwI2NN4n7Vvw96nbS9LpSsSNMIkdapiUoLikSkjpgMrtbVK
    z5lHOPloCgAyZXxfZpQRsp4oPY4BcRaMw6F5na62IHnBVDeXg=

{
  "ValidateRequest": {
    "Credential": {
      "Type": "application/x-x509-server-cert",
      "Data": "AAECAwQ="}}}
```

The service validates the certificate according to the Omnibroker service policy.

```
HTTP/1.1 200
Content-Type: application/json;charset=UTF-8
Content-Length: 47
Content-Integrity:
  mac=jrsfxojksHBVs1WWxVbX3nn+CaIIix2JrrTTQn0X43k=;
  ticket=jpBXvI7/0WTmwI2NN4n7Vvw96nbS9LpSsSNMIkdapiUoLikSkjpgMrtbVK
    z5lHOPloCgAyZXxfZpQRsp4oPY4BcRaMw6F5na62IHnBVDeXg=

{
  "ValidateResponse": {
    "Status": 200}}
```

3.2. OBPQuery

TOC

3.2.1. Message: Payload

TOC

3.2.2. Message: Message

TOC

3.2.3. Message: Request

TOC

Every query request contains the following common elements:

Index : Integer [0..1]
: Index used to request a specific response when multiple responses are available.

3.2.4. Message: Response

TOC

Every Query Response contains the following common elements:

Status : Integer [1..1]
: Status return code value

Index : Integer [0..1]
: Index of the current response.

Count : Integer [0..1]
: Number of responses available.

3.2.5. Structure: Identifier

TOC

Specifies an Internet service by means of a DNS address and either a DNS service prefix, an IP port number or both. An Internet peer connection MAY be specified by additionally specifying an account.

Name : Name [1..1]
: The DNS name of the service to connect to.
: Internationalized DNS names MUST be encoded in punycode encoding.

Account : Label [0..1]
: Identifies the account to connect to in the case that a peer connection is to be established.

Service : Name [0..1]
: The DNS service prefix defined for use with DNS records that take a service prefix including SRV.

Port : Integer [0..1]
: IP Port number.
: A service identifier MUST specify either a service or a port or both.

3.2.6. Structure: Connection

TOC

IPVersion : Integer [0..1]
: Contains the IP version field. If absent, IPv4 is assumed.

IPProtocol : Integer [0..1]
: Contains the IP protocol field. If absent, TCP is assumed.

IPAddress : Binary [0..1]
: IP address in network byte order. This will normally be an IPv4 (32 bit) or IPv6 (128 bit) address.

IPPort : Integer [0..1]
: IP port. 1-65535

TransportPolicy : String [0..1]
: Transport security policy as specified in [TBS]

ProtocolPolicy : String [0..1]
: Application security policy specification as specified by the application protocol.

Advice : Advice [0..1]
: Additional information that a service MAY return to support a service connection identification.

3.2.7. Structure: Advice

TOC

Additional information that a service MAY return to support a service connection identification. For example, DNSSEC signatures chains, SAML assertions, DANE records, Certificate Transparency proof chains, etc.

Type : Label [0..1]
: The IANA MIME type of the content type

Data : Binary [0..1]
: The advice data.

3.2.8. Structure: Service

TOC

Describes a service connection

Identifier : Identifier [0..Many]
: Internet addresses to which the service is to be bound.

Connection : Connection [0..1]
: Service connection parameters.

3.2.9. QueryConnect

TOC

Requests a connection context to connect to a specified Internet service or peer.

3.2.10. Message: QueryConnectRequest

TOC

Specifies the Internet service or peer that a connection is to be established to and the acceptable security policies.

Identifier : Identifier [0..1]
: Identifies the service or peer to which a connection is requested.

Policy : Label [0..Many]
: Acceptable credential validation policy.

ProveIt : Boolean [0..1]
: If set the broker SHOULD send advice to permit the client to validate the proposed connection context.

3.2.11. Message: QueryConnectResponse

TOC

Returns one or more connection contexts in response to a QueryConnectRequest Message.

Connection : Connection [0..Many]
: An ordered list of connection contexts with the preferred connection context listed first.

Advice : Advice [0..1]
: Proof information to support the proposed connection context.

Policy : Label [0..Many]
: Policy under which the credentials have been verified.

3.2.12. Advertise

TOC

Advises a broker that one or more Internet services are being offered with particular attributes.

3.2.13. Message: AdvertiseRequest

TOC

Specifies the connection(s) to be established.

The attributes required depend on the infrastructure(s) that the broker is capable of registering the service with.

Service : Service [0..Many]
: Describes a connection to be established.

TOC

3.2.14. Message: AdvertiseResponse

Specifies the connection(s)

Service : Service [0..Many]
: Describes a connection that was established.

TOC

3.2.15. Validate

The Validate query requests validation of credentials presented to establish a connection. For example credentials presented by a server in the process of setting up a TLS session.

TOC

3.2.16. Message: ValidateRequest

Specifies the credentials to be validated and the purpose for which they are to be used.

Service : Service [0..1]
: Describes the service for which the credentials are presented for access.

Credential : Credential [0..1]
: List of credentials for which validation is requested.

Policy : Label [0..Many]
: Policy under which the credentials have been verified.

TOC

3.2.17. Message: ValidateResponse

Reports the status of the credential presented.

Policy : Label [0..Many]
: Policy under which the credentials have been verified.

TOC

3.2.18. QueryCredentialPassword

The QueryCredentialPassword query is used to request a password credential that the user has previously chosen to store at the broker.

TOC

3.2.19. Message: CredentialPasswordRequest

Requests a password for the specified account.

Account : String [0..1]
: The account for which a password is requested.

TOC

3.2.20. Message: CredentialPasswordResponse

Returns a password for the specified account.

Password : String [0..1]
The requested password.

TOC

4. 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.

TOC

4.1. 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.

TOC

4.1.1. 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:

Integrity: <base64 (authenticator)> ; ticket=<base64 (ticket)>

TOC

4.1.2. Example

[To be generated from spec]

4.2. DNS Tunnel

TOC

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.

4.2.1. Request

TOC

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.

4.2.2. Response

TOC

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)>

4.2.3. Example

TOC

[To be generated from spec]

4.3. UDP

TOC

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.

4.3.1. Request

TOC

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]

4.3.2. Response

TOC

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?]

4.3.3. Example

TOC

[To be generated from spec]

5. Acknowledgements

TOC

[List of contributors]

6. Security Considerations

TOC

6.1. Denial of Service

TOC

6.2. Breach of Trust

TOC

6.3. Coercion

TOC

7. To do

TOC

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 a tool to transclude the example and other xml data into the document source.

Fully document the API section.

TOC

8. IANA Considerations

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

TOC

9. Normative References

[RFC1035] Mockapetris, P., “**Domain names - implementation and specification**,” STD 13, RFC 1035, November 1987 (**TXT**).

[RFC2119] **Bradner, S.**, “**Key words for use in RFCs to Indicate Requirement Levels**,” BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).

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

[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.

TOC

Appendix A. Example Data.

TOC

A.1. Ticket A

TOC

A.2. Ticket B

TOC

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.
Email: phillip@comodo.com