Internet Engineering Task Force	P. Hallam-Baker
Internet-Draft	Comodo Group Inc.
Intended status: Standards Track	July 05, 2013
Expires: January 06, 2014	

OmniBroker Protocol

draft-hallambaker-omnibroker-05

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

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. Use

[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].]

[Alice establishes her connection binding to an omnibroker service provider]

3.1. 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.

3.1.1. 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.

3.1.1.1. Service Connection Broker Example

Alice uses her Web browser to access the URL http://www.example.com/. The Web browser sends a QueryConnectRequest request to obtain the best connection parameters for the http protocol at www.example.com:

```
POST /.well-known/OmniQuery/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=uoqaIp8BUTBrq-VZXo6PSwrcRExH767XH1QYvNKv6fY;
  Id=pDtAQnl5tjPzS5r0JiuOL0pC9zlhJwdyKQCyfZPtIfnWFM8Ok5Tv074Kt8bn
  q7DWBJVGnyqHtvJHBKIhT1t1hbTlN1So9w8Q3IEkLGEqWtU
Host: localhost
Content-Length: 123
Expect: 100-continue

{
  "QueryConnectRequest": {
    "Identifier": {
      "Name": "Example.com",
      "Service": "_http",
      "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 OK
Content-Length: 335
Date: Fri, 05 Jul 2013 04:58:00 GMT
Server: Microsoft-HTTPAPI/2.0

{
  "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"}]}}
```

3.1.2. 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.

3.1.2.1. Service Connection Broker Example

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 /.well-known/OmniQuery/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=KucYLcAtB_BTKSFkc4BmUTuMJeqcR3SqaOGO6_m9PnU;
  Id=pDtAQnl5tjPzS5r0JiuOL0pC9zlhJwdyKQCyfZPtIfnWFM8Ok5Tv074Kt8bn
  q7DWBJVGnyqHtvJHBKIhT1t1hbTlN1So9w8Q3IEkLGEqWtU
Host: localhost
Content-Length: 1110
Expect: 100-continue

{
  "ValidateRequest": {
    "Service": {
      "Identifier": [{
          "Name": "example.com"}]},
    "Credential": [{
        "Data":
        "MIIC0DCCAbigAwIBAgIQQut6m1F0PodIjIzop_d1uDANBgkqhkiG9w0B
        AQUFADARMQ8wDQYDVQQDEwZWb29kb28wHhcNMTMwNjI2MTczOTQyWhcNM
        TQwNjI2MDAwMDAwWjARMQ8wDQYDVQQDEwZWb29kb28wggEiMA0GCSqGSI
        b3DQEBAQUAA4IBDwAwggEKAoIBAQCdc7Qgx71o6Tq5dFUUhcCn8Nt-2Y9
        SGhm3WvsMYIqOIcHq3gjIKN9FWvXzpBbTjz4lCwx-CJT82RBLNDFtsysf
        c0G7K_RsNKosYaM-L-DshO6R_314tptn9gnT9tjTPXuiiICQlAP83BuTI
        148iEJWL36vbmv5AG6vrtk3T6ah5r2hBXQjt46sLQYweiM-peYIhPTIy9
        OYugogfqdzPvaJpDfAukqJBXqMxfscagKPYAGPaICKhobKr11aPam1Tch
        k2cBbtuYgSDz6ZGttsKE2omDbcmhbF7gBpRug-E2OH79Q4EVlSSoO9gZ6
        AF4Km1A9uK9W_Pg8EPugY3Mgns6lAgMBAAGjJDAiMAsGA1UdDwQEAwIEM
        DATBgNVHSUEDDAKBggrBgEFBQcDATANBgkqhkiG9w0BAQUFAAOCAQEACK
        9LQNkewOOugaYhs4LfE3xdrRzrcaR0w5cf3wVcgR0ZZo98rDOtu3FAexp
        dh6vNaIdU4zAzNJPKKSso3XF2LpQZovKIpUuN9pkZqslqZ0TLXqlyXMbh
        eShcqIP1-m6qjZOp95N7jwgxBlEmi_ne-rg1DicXFtAu90LpAZludaQGA
        yrj-LC37gzeMo2AG7BAuyFURXJFfxjpGmnueuYfzZIMIQY-lNl6qm_vSM
        Iz4uUKqq4lWndahnkJAwI2p5zUM0z3O6OMr_zr8eyrdAL__H4NnG3gVyB
        bNoSbvbkxUt_C3oBwFFTupzRMQqJVjzbApyw5H0OzJPJKKkxxhmIYTg"}
        ]}}
```

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

```
HTTP/1.1 OK
Content-Length: 45
Date: Fri, 05 Jul 2013 04:58:00 GMT
Server: Microsoft-HTTPAPI/2.0

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

3.1.3. Credential Validation

The credential validation query provides certificate path validation and status checking.

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.

3.2. 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.

[Examples TBS]

4. OBPQuery

4.1. Message: Payload

4.2. Message: Message

4.3. Message: Request

Every query request contains the following common elements:

Index : Integer [0..1]

Index used to request a specific response when multiple responses are available.

4.4. Message: Response

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.

4.5. Structure: Identifier

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.

4.6. Structure: Connection

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.

4.7. Structure: Advice

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.

4.8. Structure: Service

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.

4.9. QueryConnect

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

4.10. Message: QueryConnectRequest

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.

4.11. Message: QueryConnectResponse

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.

4.12. Advertise

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

4.13. Message: AdvertiseRequest

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.

4.14. Message: AdvertiseResponse

Specifies the connection(s)

Service : Service [0..Many]

Describes a connection that was established.

4.15. 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.

4.16. 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.

4.17. Message: ValidateResponse

Reports the status of the credential presented.

Policy : Label [0..Many]

Policy under which the credentials have been verified.

4.18. QueryCredentialPassword

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

4.19. Message: CredentialPasswordRequest

Requests a password for the specified account.

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

4.20. Message: CredentialPasswordResponse

Returns a password for the specified account.

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

5. 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.

5.1. 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.

5.2. 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.

5.2.1. 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)>

5.2.2. Example

[To be generated from spec]

5.3. 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.

5.3.1. 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.

5.3.2. 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)>

5.3.3. Example

[To be generated from spec]

5.4. 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.

5.4.1. 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]

5.4.2. 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?]

5.4.3. Example

[To be generated from spec]

6. Acknowledgements

[List of contributors]

7. Security Considerations

7.1. Denial of Service

7.2. Breach of Trust

7.3. Coercion

8. 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

9. IANA Considerations

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

10. References

10.1. Normative References

[I-D.hallambaker-httpsession] Hallam-Baker, P., "HTTP Session Management", Internet-Draft draft-hallambaker-httpsession-01, May 2013.

[I-D.hallambaker-jsonbcd] Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", Internet-Draft draft-hallambaker-jsonbcd-00, June 2013.

[I-D.hallambaker-wsconnect] Hallam-Baker, P., "JSON Service Connect (JCX) Protocol", Internet-Draft draft-hallambaker-wsconnect-02, 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. 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.

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

[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.

10.2. 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