Kitten Working Group
Internet Draft
Intended status: Standards Track
Expires: February 1, 2015

K. Zheng
W. Jiang
Intel Corporation
T. Hardjono
T. Yu
MIT Kerberos Consortium

October 25, 2014

Token Pre-Authentication for Kerberos
draft-ietf-kitten-kerb-token-preauth-01

Abstract

Kerberos provides a pre-authentication framework authenticating client using other authentication mechanisms and credentials instead of password. This document proposes a new pre-authentication mechanism, i.e token-preauth, to allow client to authenticate to KDC using a standard JWT token.

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), 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 February 1, 2015.

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. Introduction..2
2. Conventions Used in This Document.............................3
3. Scope...3
4. Usage Overview..4
   4.1. JWT Token and Token Provider.............................4
   4.2. Bearer Token and Holder-of-Key Token Types...............4
   4.3. Identity Token and Access Token Schemes..................4
   4.4. Identity Account Synchronization.........................5
   4.5. Required Token Attributes................................6
5. Pre-Authentication..6
   5.1. Initial Client Request...................................7
   5.2. KDC Challenge..8
   5.3. Client Response..8
   5.4. Verifying the Pre-Authentication Data....................8
   5.5. Issuing Ticket...9
6. New Types..10
   6.1. PA-TOKEN-CHALLENGE......................................10
   6.2. PA-TOKEN-REQUEST..12
   6.3. AD-TOKEN..12
7. Security Considerations......................................12
8. IANA Considerations..13
9. References...13
   9.1. Normative References....................................13
   9.2. Informative References..................................15
10. Acknowledgments...15

1. Introduction

This document proposes to add a new pre-authentication mechanism similar to OTP and PKINIT for Kerberos, based on the Kerberos pre-authentication framework [RFC6113], allowing user to use JWT token as credential instead of password to authenticate to KDC. Specific token attribute value is required to specify the target Kerberos client principal for issuing ticket. PKI is used to establish the trust relationship between token issuer and KDC. According to the trust setup, KDC validates token and determines to issue ticket or not. A derivation of original token will be wrapped into the issued ticket as new authorization data and carried on to application server side for further checking and authorization usage.

2. Conventions Used in This Document

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].

This document assumes familiarity with the Kerberos pre-authentication framework [RFC6113] and so freely uses terminology and notation from that document.

The word padata is used as shorthand for pre-authentication data.

The word tgt is used as shorthand for Ticket Granting Ticket.

Also note, as this mechanism shares common properties with OTP FAST factor [RFC6560], this document borrows the discussion layout and procedures from it for convenience.

3. Scope

This document describes a FAST [RFC6113] mechanism that allows JWT token to be used in Kerberos pre-authentication in a manner that does not require use of the user's Kerberos password. It discusses two kinds of token, Bearer Token and Holder-of-Key Token; and for both it allows two token schemes, Identity Token and Access Token. This document focuses on defining the token pre-authentication mechanism and the framework to support token for Kerberos, only covering the simple case, i.e. Identity Token scheme in Bearer Token type. It uses Bearer Token and doesn’t provide Replacing-Reply-Key and Strengthening-Reply-Key facilities; Based on the framework future effort COULD be made to extend this document to support Holder-of-Key tokens, and provide such facilities by employing the token security properties. It uses Identity Token to authenticate and request a tgt, and doesn’t support Access Token; Based on the framework future effort COULD be made to support Access Token to be used to request service ticket directly.

4. Usage Overview

4.1. JWT Token and Token Provider

As described above, this document describes a generic Kerberos pre-authentication mechanism supporting JWT token [JWT]. Token used in this mechanism should conform to JWT standards family including [JWT], [JWA], [JWE], [JWS] and [JWK]. To facilitate token validation and processing, the tokens should also contain specific required token attributes. Token providers should issue such tokens to be supported by this mechanism. The required token attributes will be defined later.

4.2. Bearer Token and Holder-of-Key Token Types

The mechanism supports Bearer Token in its first version, allows supporting Holder-of-Key Token in future.

As defined in [RFC6750], any party in possession of a Bearer Token can use it to access service or resources without demonstrating possession of a cryptographic key, and to prevent misuse Bearer Token needs to be protected from disclosure in storage and transport. When using Bearer Token, other mechanism(s) that provides the necessary facilities like KDC-authentication and Replace Reply Key should be deployed together and employed, to protect token from being stolen. At the time of this writing, we recommend PKINIT [PKINIT] and PKINIT Anonymous [PKINIT-ANONYMOUS] in situation that requires fair balance between security and usability.

As defined in [HOK-TOKEN], Holder-of-Key Token associates a cryptographic key, and requires token client when presenting token to other party also demonstrate knowledge of keying material that is bound to the token. The key can be symmetric or asymmetric, and such cryptographic key can be used to compute the Reply Key to protect the KDC-REQ exchange, therefore Holder-of-Key token provides better security properties and in this case the mechanism can be used separately without relying on other mechanisms.

4.3. Identity Token and Access Token Schemes

The mechanism supports Identity Token and Access Token.

Identity Token is quite often mentioned in security field but never gets defined. Simply Identity Token can be thought of as tgt as an authentication result, and can be persisted in cache or cookie for further usage to request an Access Token when the identity user requests access to specific service or resources. Similarly, Access

Token is rather like service ticket and binds access to a specific service with optional resources. Access Token is derived from Identity Token, with limited target audiences and shorter life cycle valid time.

This mechanism allows Identity Token to be used to request tgt in AS exchange, and then the tgt can be used as usually in Kerberos. It allows Access Token to be used to request service ticket targeting the corresponding service associated with the token. To do so a tgt should be obtained first as Identity Token but the tgt can only be used to request service ticket for the targeted service. In both cases, token subject is mapped into the target Kerberos client principal, and particularly for Access Token, the service as token audience is mapped into the target Kerberos service principal (SPN). The both mappings are required to be supported by different token providers.

Both Identity Token and Access Token MUST be in JWT token format, and can be Bearer Token or Holder-of-Key Token.

For Access Token, in practice one MAY additionally implement to support returning service ticket directly in AS exchange avoiding tgt. This SHOULD be much efficient and useful to support OAuth 2.0 Access Token work flow.

4.4. Identity Account Synchronization

In the pre-authentication framework defined in [RFC6113], client principal is assumed to exist in KDC database before a user can employ a mechanism to authenticate to the KDC. Therefore for some mechanisms, principal account synchronization between identity source and KDC is needed, which is not only problematic but also involves maintain overhead and prevents the mechanisms being more widely deployed in practice. In this mechanism, as client principal name can be synthesized from the required token attribute mapping, it doesn’t require the client principal account MUST exist to pass the authentication request if the KDC policy allows. A KDC can be configured to disable this behavior; in this case the mechanism works exactly as others and requires the client principal MUST exist.

[Discussion] Avoiding identity account synchronization as discussed above MAY be general and also desired for more mechanisms like PKINIT, so it would make sense to update the framework [RFC6113] with this consideration.

4.5. Required Token Attributes

The mechanism requires the following attributes MUST exist in the JWT token. The attributes SHOULD conform to the [JWT] spec, and particularly, they SHOULD appear in both plaintext Header Parameters and encrypted Claims Set.

1. "sub" (Subject) Claim. This claim SHOULD specify the client Kerberos principal name including the realm.

2. "aud" (Audience) Claim. This claim SHOULD specify the token audience appropriately, for Identity Token, the value SHOULD be the principal name of the Ticket Granting Service including the realm; for Access Token the value SHOULD be the principal name of the target service including the realm. The mechanism uses this attribute to determine the input token is an Identity Token or an Access Token.

3. "iss" (Issuer) Claim. This claim SHOULD be able to identify the corresponding token provider configured in the KDC side.

4. "exp" (Expiration Time) Claim.

5. "nbf" (Not Before) Claim.

6. "iat" (Issued At) Claim.

When a token is used to request Kerberos ticket, if additional ticket life time is specified via other means, the time MUST fall in the range defined by exp, nbf and iat values in the token. These values SHOULD also be used to frame the time range used to issue the resultant ticket. If a derived ticket from a token is to be renewed, the renew time SHOULD also be in the time range.

5. Pre-Authentication

The mechanism uses pre-authentication data in AS-REQ, AS-REP, and KRB-ERROR messages, and supports both four-pass and two-pass negotiation variants.

In the four-pass system, the client begins by sending an initial AS-REQ to the KDC. The KDC will then determine, in an implementation dependent fashion, whether token authentication is required or supported and if it is, it will respond with a KRB-ERROR message containing a PA-TOKEN-CHALLENGE in the PA-DATA. PA-TOKEN-CHALLENGE contains OPTIONAL information about token providers supported by the KDC. The client will prompt the user to select a token provider and input a token. As described in Section 5.4.1 of [RFC6113], the FAST system uses an Armor Key to set up an encrypted tunnel for use by FAST factors, and FAST factor or mechanism needs to determine two keys: a Client Key to encrypt the KDC's nonce and a Reply Key used to decrypt the KDC's reply. This mechanism determines the two keys as: 1) In case Bearer Token is used as defined by this document, the mechanism uses the Armor Key as the two keys; 2) Further effort can be made to extend this document to support Holder-of-Key tokens and then more securely, the two keys could be generated from the key bound with the token and the Armor Key. Anyway, the input token and OPTIONAL information about the chosen token provider are wrapped in a PA-TOKEN-REQUEST encrypted within the armored-data of a PA-FX-FAST-REQUEST PA-DATA element, and the padata is sent to the KDC as a second AS-REQ.

In the two-pass system, the client sends the PA-TOKEN-REQUEST in the initial AS-REQ instead of sending it in response to a PA-TOKEN-CHALLENGE returned by the KDC.

In both cases, on receipt of a PA-TOKEN-REQUEST, the KDC determines the Client Key and Reply Key as the client does, and uses the Client Key to verify the pre-authentication. The KDC will then authenticate the token against the corresponding token provider, by decrypting the token, verifying the signature of the token, and validating the token life time, according to [JWT], [JWE] and [JWS]. If everything is fine, the KDC proceeds and determines to issue ticket to the client. Different from other pre-authentication mechanisms, the KDC MAY derive a token by escaping the encryption and signature layers of the original token, then wrap the derivation token as a new authorization data type, AD-TOKEN, and put it into the authorization-data field using the AD-KDC-ISSUED container in the ticket.

5.1. Initial Client Request

In the four-pass mode, the client begins by sending an initial AS-REQ, possibly containing some pre-authentication data. If the KDC determines that the token mechanism is required or supported and the request does not contain a PA-TOKEN-REQUEST, then it will respond returning PA-TOKEN-CHALLENGE with supported token providers.

If the client has all the necessary information, knows the mechanism is supported, and the user specifies a token from a certain token provider, it MAY use the two-pass system by constructing a PA-TOKEN-REQUEST and including it in the initial request directly.

5.2. KDC Challenge

If the user is required to authenticate using token per KDC policy, then the KDC SHALL respond to the initial AS-REQ with a KRB-ERROR (as described in Section 2.2 of [RFC6113]), with a PA-TOKEN-CHALLENGE contained within the enc-fast-rep of the armored-data of a PA-FX-FAST-REPLY encrypted under the current Armor Key as described in [RFC6113].

If the mechanism is to be carried out as an individual mechanism, then the PA-TOKEN-CHALLENGE SHALL be carried within the padata of the KrbFastResponse. Alternatively, if the mechanism is required as part of an authentication set, then the PA-TOKEN-CHALLENGE SHALL be carried within a PA-AUTHENTICATION-SET-ELEM as described in Section 5.3 of [RFC6113].

The KDC MAY use the token-vendor field to assist the client in input of token to be used by identifying the purpose of the authentication. For example, the token-vendor field could assist a user in identifying the token provider to be used when a user has multiple token providers that are used for different purposes. The KDC SHALL include a sequence of one or more token-info elements containing information on the desired tokens that the user can use for the authentication. Either one token-info element or multiple elements are included, only one token that matches with one of the token-info is needed and SHOULD be accepted by the KDC. If none token is received or matches with any of the specified token-info, the KDC SHOULD reject the request.

5.3. Client Response

The client response SHALL be sent to the KDC as a PA-TOKEN-REQUEST included within the enc-fast-req of the armored-data within a PA-FX-FAST-REQUEST encrypted under the current Armor Key as described in [RFC6113]. In order to generate its response, the client MUST prompt user to input and provide a token according to the KDC challenge requirement regarding what kinds of token is desired as described in PA-TOKEN-CHALLENGE if any in four-pass system.

5.4. Verifying the Pre-Authentication Data

The KDC validates the pre-authentication data by determining the Client Key and Reply Key in the same way as the client. Then the KDC validates the token according to JWT token specs. There are chances that the validation may fail and then KRB-ERROR SHOULD be returned to client as response. The KRB-ERROR message SHOULD use KDC_ERR_TOKEN_INVALID error code and also contain the concrete error message in the e-text field regarding why the token processing or validation fails. The chances are:

1. Identity token is expected in this mechanism for requesting a tgt, but if instead an Access Token is provided.

2. Similar to 1), Access Token is expected in this extension for requesting service ticket, but if instead an Identity Token is provided.

3. Token SHOULD be correctly decrypted and verified according to [JWE] and [JWS], but if the decryption or verification fails.

4. If the client Kerberos principal can't be determined from the token.

5. Similarly, if service Kerberos principal can't be determined from the Access Token.

6. KDC checks the token’s valid life time in certain clock skew. When such check fails.

7. Particularly, even if token isn't expired, but the left life time isn't enough for the requested tgt or service ticket in desired valid time.

5.5. Issuing Ticket

Once KDC validates the request and determines to issue a ticket to the client, this mechanism SHOULD contribute a new Authorization Data type, AD-TOKEN. AD-TOKEN will be wrapped into AD-KDC-ISSUED, and it contains a derivation of the token with meaningful token attributes for application usage. The token derivation can be resulted from escape of encryption and signature of the original token as defined by the Dt function in the following. The derivation serves two purposes: 1) applications can therefore understand and make use of the token attributes without need to deploy relevant security keys; 2) such derivation can be safely passed to applications without worrying about leakage or restoring of original token since original token SHOULD be carefully protected. Even if the derivation token is presented to KDC by any party, KDC SHOULD fail to validate it as defined by the validation according to [JWE] and [JWS].

```
   De(token) = Escape or remove encryption layer from the JWT token

   Ds(token) = Escape or remove signature layer from the JWT token
```

```
      Dt(token) = Ds(De(token)) or De(Ds(token))
```

6. New Types

6.1. Kerberos Token

KRB-TOKEN is defined for the token type as follows.

```
   TOKEN-FORMAT-JWT 1

   KRB-TOKEN ::= SEQUENCE {

      token-format [0] INTEGER,

      token-value  [1]  OCTET STRING,

   }
```

6.2. PA-TOKEN-CHALLENGE

The padata-type PA-TOKEN-CHALLENGE is returned by the KDC to the client in the enc-fast-rep of a PA-FX-FAST-REPLY in the PA-DATA of a KRB-ERROR when the mechanism is required. The corresponding padata-value field contains the Distinguished Encoding Rules (DER) [X.680] and [X.690] encoding of a PA-TOKEN-CHALLENGE containing token information for the client on how to opt to provide a token.

```
      PA type TOKEN-CHALLENGE     TBD

      PA-TOKEN-CHALLENGE ::= SEQUENCE {

         tokenInfos       [0] SEQUENCE (SIZE(1..MAX)) OF TokenInfo,

      }

      TokenInfo ::= SEQUENCE {

           flags            [0] TokenFlags,

           tokenVendor      [1] UTF8String,

      }

      TokenFlags ::= KerberosFlags
```

-- reserved(0),

-- id-token-required(1),

-- ac-token-required(2),

-- bearer-token-required(3),

-- hok-token-required(4)

token vendor

This MAY be used by the KDC to assist the client to choose to input the appropriate token, when the KDC has multiple token providers configured for different purposes or set of users.

token flags

KDC checks token policy and prepare for token flags to require client to provide appropriate token accordingly when the mechanism is required for a principal request. Once token is input and provided the client needs to validate and process the token according to the flags. Client should try the best to avoid obvious issues that the client can find in its side but fails to do so then is found by KDC side.

If id-token-required is set, then client should prompt user to input and provide an Identity Token for requesting tgt. If ac-token-required is set, then client should prompt user to input and provide an Access Token, for requesting service ticket targeting a specific service. If bearer-token-required is set, then client should prompt user to input and provide a Bearer Token; and in such case, if instead a Hold-of-Key Token is provided the request MAY not fail but the possession of the bound key MIGHT not be proved. If hok-token-required is set, then client should prompt user to input and provide a Holder-of-Key Token. A note, as user may not understand all kinds of these token types and schemes, client MAY not force user to. The prompt info to hint user for token SHOULD not be too specific and technical. However it should log the error in detail to help with troubleshooting.

Token info

It contains token flags and token vendor.

6.3. PA-TOKEN-REQUEST

The padata-type PA-TOKEN-REQUEST is sent by the client to the KDC in the KrbFastReq padata of a PA-FX-FAST-REQUEST that is included in the PA-DATA of an AS-REQ. The corresponding padata-value field contains the DER encoding of a PA-TOKEN-REQUEST.

```
PA type TOKEN-REQUEST     TBD

PA-TOKEN-REQUEST ::= SEQUENCE {

  token          [0]  OCTET STRING,

  tokenInfo      [1]  TokenInfo

}
```

6.4. AD-TOKEN

The new Authorization Data Type AD-TOKEN type contains token derivation and is meant to be encapsulated into AD-KDC-ISSUED type and to be put into tgt or service tickets. Application can safely ignore it if the application doesn't understand it. The token field SHOULD be ASN.1 encoded of the binary representation of the serialization result of the derivation token according to [JWT].

```
AD-TOKEN     TBD

AD-TOKEN ::= SEQUENCE {

   token     [0]  OCTET STRING,

}
```

7. Security Considerations

This document discusses generic token types (Bearer Token and Holder-of-Key Token) and schemes (Identity Token and Access Token), and defines common token support framework for Kerberos. For simplicity it defines the token pre-authentication mechanism using the Bearer Token scheme, which doesn’t provide generating and strengthening Client Key and Reply Key facilities. Therefore it’s not RECOMMENDED that this mechanism be deployed independently. To protect token from leakage, we RECOMMEND it SHOULD be deployed together with other pre-authentication mechanism like PKINIT defined in [PKINIT] and OTP defined in [RFC6560], or whatever means that can provide the good enough Armor Key. Sure also, transport layer security like TLS/SSL COULD also be employed to protect token if it’s available between client and KDC.

Additionally, the following aspects are worth to be noted:

Contrast to Bearer Token, Holder-of-Key Token binds a cryptographic key which can be employed to generate the Client Key and Reply Key both in client and KDC sides, which makes the mechanism with the token type be more secure and separately deployable. Future effort COULD be made to extend this document to support so.

An Access Token is not equivalent to the user's long term password. Therefore Access Token SHOULD not be used to request tgt and this mechanism prevents it from being used for the usage to exchange tgt. Instead, an Access tokens is equivalent to a service ticket and targets for a specific service identified as token audience. Therefore it’s natural to extend this effort further to support Access Token to be used to request service ticket directly.

8. IANA Considerations

The PA-TOKEN-CHALLENGE and PA-TOKEN-REQUEST should be registered in the "Algorithm URI Registry and Related PSKC Profiles" registry [RFC6030].

The following pre-authentication types are defined in this document:

PA-TOKEN-CHALLENGE	TBD
PA-TOKEN-REQUEST	TBD

These values should be registered in a registry created by [RFC6113], but the entries should be updated to refer to this document.

The following error code is defined in this document:

KDC_ERR_TOKEN_INVALID TBD

The meaning is, invalid token credential is provided.

9. References

9.1. Normative References

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

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

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

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

[RFC6112] Zhu, L., Leach, P., and S. Hartman, "Anonymity Support for Kerberos", RFC 6112, April 2011.

[RFC6113] Hartman, S. and L. Zhu, "A Generalized Framework for Kerberos Pre-Authentication", RFC 6113, April 2011.

[RFC6560] G. Richards, "One-Time Password (OTP) Pre-Authentication", RFC 6560, April 2012.

[RFC6749] D. Hardt, Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC6750] D. Hardt, Ed., "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

[JWT] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token-22 (work in progress), June 20, 2014.

[JWE] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption (work in progress), June 2014.

[JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key (work in progress), June 2014.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature (work in progress), June 2014.

[JWA] Jones, M., "JSON Web Algorithms (JWA)", draft-ietf-jose-json-web-algorithms (work in progress), June 2014.

[HOK] J. Bradley, P. Hunt, T. Nadalin, and H. Tschofenig, "The OAuth 2.0 Authorization Framework: Holder-of-the-Key Token Usage", draft-tschofenig-oauth-hotk-03.txt (work in progress), January 14, 2014.

[POPOK] M. Jones, J. Bradley, H. Tschofenig, " Proof-Of-Possession Semantics for JSON Web Tokens (JWTs)", draft-jones-oauth-proof-of-possession-02 (work in progress), July 4, 2014.

[X.680] ITU-T, "Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.", July 2002.

[X.690] ITU-T, "Recommendation X.690 (2008) | ISO/IEC 8825-1:2008, X.690: Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", December 2008.

9.2. Informative References

[ASN] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", Cryptology ePrint Archive Report 2002/163, November 2002.

[IAN] Hornquist Astrand, L., "Kerberos number registry to IANA", Work in Progress, March 2010.

10. Acknowledgments

This initial draft incorporated many important feedbacks from both Greg and Ben Kaduk (MIT) in the early discussion about this mechanism.

Thanks Dey, Avik and Andrew Purtell for the early interesting and confirmation for the prototype of the idea in this direction. And thanks to Xiang Zhong for his initial review and feedback.

Authors' Addresses

Kai Zheng

Intel Corporation

Email: kai.zheng@intel.com

Weihua Jiang

Intel Corporation

Email: weihua.jiang@intel.com

Thomas Hardjono

MIT Kerberos Consortium

Email: hardjono@mit.edu

Tom Yu

MIT Kerberos Consortium

Email: tlyu@mit.edu