Kerberos V5

Technical Description
ASN.1 Data Representation Language

Basic Encoding Rules ( BER ) allow:
- optional fields of data structures
- variable-length data structures
- typed data structures

Motivation:
- independence of hardware data structure encodings
e.g., big- or little-endian byte ordering
- standard definition

Example: Specification of an IP address

```
HostAddress ::= SEQUENCE {
  addr-type [0] INTEGER
  address [1] OCTET STRING
}
```

Does redundancy help?

```
tag len    tag len    tag len    val
 8        8        8        ≥ 8
```

```
tag len  tag len  val
 8        8        ≥ 8
```
Delegation of Rights

• Forwarding of TGTs
  – *forwardable* TGT => it can be exchanged for a TGT with one or more different network addresses (i.e., *forwarded* TGT)
  – limited time
  – option for transitive forwarding (adequate control?)

• Proxying of tickets
  – *proxyable* TGT => it can be used to request tickets with one or more different network addresses (i.e., *proxy* tickets)
  – limited time
  – no transitive proxying

• Forwarded and Proxy flags set in all derived TGT / tickets
  – application awareness of delegation
Forwarding Protocol

• **AS_REQ** includes:
  – “forwardable” option

• **AS_REP** includes:
  – “forwardable” TGT

• **TGS_REQ** includes:
  – “forwardable” TGT
  – “forwardable” option
  – “caddr list”

• **TGS_REP** includes:
  – “forwarded” TGT and “forwardable” flag (if requested)
  – set of “caddr”

• Forwarder constructs a KRB_CRED message to pass the forwarded ticket and ticket’s session key to recipient
Proxying Protocol

• **AS_REQ** includes:
  – “proxyable” option

• **AS_REP** includes:
  – “proxyable” flag in TGT

• **TGS_REQ** includes:
  – “proxy” ticket for a *specific* application service
  – “caddr list”
  – additional access restrictions in “authorization_data”

• **TGS_REP** includes:
  – “proxy” ticket and set “authorization_data” (if requested)
  – set of “caddr”

• Proxy-er constructs a KRB_CRED message to pass the proxy ticket and ticket’s session key to recipient
Flag Checking in AS_REP and TGS_REP

- What if the “forwardable” requested option is not checked against the “forwardable” flag?
  - non-forwardable TGTs may become forwardable
  - forwarded - only TGTs may become forwardable

- Note: the use of the “forwardable” feature may be dangerous as it may cause unrestricted propagation of a party’s identity and permissions

- Other requested options vs. flag checks are necessary
  - non-proxyable tickets may become proxyable
  - non-renewable tickets may become renewable
Ticket Lifetimes

- starttime = time the ticket becomes valid
- endtime = time the ticket expires
- authtime = KDC (AS) time when TGT of AS_REP is created

starttime > authtime => postdated tickets

- postdated tickets are useful for batch / absentee computations
- long-lived tickets are necessary

Problem

- long-lived tickets make revocation impossible
- postdated tickets must allow revocation before first use

Solution

- introduce renewable tickets and “renew_til” limit
  - renewal => endtime = min { max_renewable_life, renew_until }
- introduce “invalid” ticket status, postdate limit, and ticket validation
- tickets are renewed and/or validated unless
  - they are placed on ticket revocation list already
- finite ticket holding time on revocation list
TGT Lifetime Determination at AS

Starttime

- req.from ≤ authtime
- req.from ≤ tkt.starttime
- invalid

Endtime

- req.till ≤ postdate limit
- option = Postdated (and Postdate allowed by Realm policy)
- req.from - authtime ≤ postdate limit
- tkt.starttime ≤ tkt.endtime ≤ tkt.renew_till
- client_max_life
- server_max_life
- realm_max_life
- min. tkt. life
- rtime
- req.rtime

Renew_till

- option = Renewable
- rtime = req.till
- rtime = req.rtime
Ticket Lifetime Determination at TGS

Starttime
- req.from
- tkt.starttime
- TGS_time
- postdate limit
- time @ TGS

Endtime
- req.till
- ticket.endtime
- realm_max_life
- req.rtime

Renew_till
- ticket.flag = Renewable
- option = Renewable
- (option = Renewable_OK
  => rtime = min (req.till, ticket_renew_till))

- tkt.starttime
- tkt.endtime
- tkt.renew_till

Flag: may_postdate
Option: Postdated (and Postdate allowed by Realm policy)
req.from - authtime ≤ postdate limit

Validity:
- tkt.starttime ≤ req.from ≤ tkt.endtime
- tkt.starttime ≤ ticket.endtime ≤ ticket.renew_till
- ticket.endtime ≤ server_max_life
- ticket.renew_till ≤ realm_max_life

Renewal:
- rtime = min ( req.till, ticket_renew_till )
- ticket_renew_till ≤ server_max_rlife
<table>
<thead>
<tr>
<th>Message Options and Ticket Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ap_options</strong></td>
</tr>
<tr>
<td><strong>USE_SESSION_KEY</strong></td>
</tr>
<tr>
<td><strong>MUTUAL_REQUIRED</strong></td>
</tr>
<tr>
<td><strong>PROXIEABLE</strong></td>
</tr>
<tr>
<td><strong>PROXY</strong></td>
</tr>
<tr>
<td><strong>ALLOW-POSTDATE</strong></td>
</tr>
<tr>
<td><strong>POSTDATED</strong></td>
</tr>
<tr>
<td><strong>RENEWABLE</strong></td>
</tr>
<tr>
<td><strong>RENEWABLE-OK</strong></td>
</tr>
<tr>
<td><strong>ENC-TKT-IN-SKEY</strong></td>
</tr>
<tr>
<td><strong>RENEW</strong></td>
</tr>
<tr>
<td><strong>VALIDATE</strong></td>
</tr>
<tr>
<td><strong>INITIAL</strong></td>
</tr>
<tr>
<td><strong>PRE-AUTHENT</strong></td>
</tr>
<tr>
<td><strong>HW-AUTHENT</strong></td>
</tr>
</tbody>
</table>
Key Versions in Kerberos V5

Per Principal (p) set of triples \(< \{ p\text{-key} \}_{K_{\text{KDC}}}, p_{\text{kvno}}, k_{\text{kvno}} >\>
TGS_REQ returns tickets encrypted in key with \textit{highest} p_{\text{kvno}}
Encoding in the KDC database: separate principal entry per key

Motivation for KDC support of multiple p_{kvno}

Ticket renewal by KDC
• Scenario:
  1. Server changes its key
  2. Client, which has a renewable ticket encrypted in old server key, requests renewal
  3. KDC needs to remember old server key to decrypt ticket and renew it.
  4. KDC verifies ticket renewability, renews ticket, and re-encrypts it in key with highest p_{kvno}

Ticket postdating (similar scenario)
Realm Hierarchy

Naming Path to Target Realm:
eng.umd.edu - umd.edu -edu - berkeley.edu - eecs.berkeley.edu

1. cache = empty
2. client traverses trust path and obtains TGTs (e.g., TGT to target realms eecs.berkeley.edu)
Inter-realm Authentication Algorithm - An Example

Naming Path to Target:
eng.umd.edu - umd.edu - edu - ro - edu.ro - upb.edu.ro - cs.upb.edu.ro

1. cache = non-empty
2. client gets TGT to closest realm on path to cs.upb.edu.ro
3. TGT to umd.edu => no path to target
Naming Path to Target:

eecs.berkeley.edu - berkeley.edu - edu.ro - edu.ro - upb.edu.ro - cs.upb.edu.ro
Pre-authentication
(and other password-discovery countermeasures)

Motivation

- AS_REQ/AS_REP generate any number of known plaintext - ciphertext pairs
- Off-line password guessing attacks

Solution

- PADATA = \{ ctime \} \textsuperscript{K_{client}} required in AS_REQ
- AS_REP sent only if plaintext ctime of AS_REQ = decrypted PADATA

Separation of Human vs. Server Principals

Motivation

- TGS_REQ specifies a human principal instead of a server principal
- Effect of pre-authentication is circumvented

Solution

- no_ap_tkt flag set for human principals
Pre-authentication etc. (continued)

Separation of principal keys per realm

Motivation

- Principals registered in multiple realms may use the same key
- Theft of key in one realm => compromised keys in all realms

Solution

- $k = \{ \text{OWF (p\textunderscore name, p\textunderscore realm, passwd)} \}^{K_{KDC}}$ is stored in KDC database
- default “salt”: < p\textunderscore name, p\textunderscore realm>; new realm name => obsolete “salt”
  => wrong “pdata” in AS\_REQ => user cannot login
- obsolete “salt” => second chance login => KDC includes new “salt” in error message
- “pdata” of AS\_REP contains new “salt” if any
Double TGT Authentication - Motivation

Kerberos V4: User-to-Host Authentication

- User inputs decryption key (i.e., password); Server gets its key from srvtab

PROBLEM:

- User-to-User Authentication
  - Workstations cannot offer authenticated services; srvtab cannot be protected
  - Idle public workstations cannot be authenticated

- Scaling Constraints
  - Neither added state nor added load to Kerberos
  - No added frequently changing fields to KDC database
  - One transaction per connection

SOLUTION:

- Client initiates protocol with Kerberos
- Use Double TGT Authentication (aka. ENC-TKT-IN-SKEY)
Double TGT Authentication - Message Flows

1. req

2. rep

3. B, TGS

4. A, B

5. ... ap_option = use_session_key

TGS

A

B

B, TGS

A, TGS

TGS

B, TGS

B, TGS

A, B

B, TGS

B, TGS
KDC Database

Realm Policies
- max_life
- max_rlife
- postdate
- preauth

Principals’ Dir.
- name = X.R
- \{ key0 = f( passwd0 ) \} \textit{k_keyx}
- p_kvno = 0
- p_max_life
- p_max_renewable_life
- k_kvno (x)
- entry_expiration
- mod_date
- mod_name
- policy flags
- passwd_expiration
- last_pwd_change
- frequently changing fields

Site Directory
- Site M

Site Policies
- rtill - starttime
- endtime - starttime
- zero_addr_allowed

Options allowed
- last_success
- latest_tkt_exp/key
Kerberos V 5

Message Formats and Protocol Flows
AS_REQ / AS_REP

msg_type (11)
pvno (5)
padata
crealm
cname (A)
ticket
client, server (K_a,tgs)
last_req
nonce
p_key_expiration
flags
authtime
starttime
endtime
renew_till
srealm
name (TGS)
caddr (A addr)

TGS

msg_type (1)
tkt_vno (5)
srealm
sname (krbtgt)
flags
client, server (K_a,tgs)
crealm
cname (A)
transited
authtime
starttime
endtime
renew_till
sname (TGS)
caddr (A addr)
(authorization_data)
**TGS_REQ**

msg_type (12)  
ptno (5)  
padata

msg_type (1)  
tkt_vno (5)  
srealm  
sname ( krbtgt )

flags  
client, server ( K_a,tgs )  
crealm  
cname (A)  
transited  
authetime  
starttime  
endtime  
renew_till  
sname (TGS)  
caddr (A addr)  
()

TGS

client, server ( K_a,tgs )

authenticator_vno ( 5 )  
cname ( A )  
crealm  
csum  
ctime  
cusec  
subkey ( sub c,s )  
seq_number  
()

Authorization_data

kdc_opt, cname, sname, realm, from, till, rtime, nonce, etypes, addresses, adtl. tkts

A → KDC
TGS_REP

msg_type (13)
pvno (5)
crealm
cname (A)
ticket

client, server (K_{a,b})
last_req
nonce
flags
starttime
endtime
renew_till
srealm
sname (TGS)
caddr (A addr)

msg_type (1)
tkt_vno (5)
srealm
sname (B)
flags
client, server (K_{a,tgs})
crealm
cname (A)
transited
authtime
starttime
endtime
renew_till
sname (B)
caddr (A addr)
authorization_data
**AP_REQ / AP_REP**

- `msg_type (14)`
- `pvno (5)`
- `ap_options (use_session_key, mutual_required)`

**Flags**
- client, server (K_{a,b})
- crealm
- cname (A)
- transited
- starttime
- endtime
- renew_till
- sname (B)
- caddr (A addr)
- authorization_data

**Client, server (K_{a,b})**

- `client, server (K_{a,b})`
- `crealm`
- `cname (A)`
- `transited`
- `starttime`
- `endtime`
- `renew_till`
- `sname (B)`
- `caddr (A addr)`
- `authorization_data`

**Authenticator_vno**
- `authenticator_vno (5)`
- `cname (A)`
- `crealm`
- `cksum`
- `ctime`
- `cusec`
- `subkey (sub_{A,b})`
- `seq_number`

**Client, server (K_{a,b})**

- `msg_type (15)`
- `pvno (5)`
- `ctime`
- `cusec`
- `subkey (sub_{A,b})`
- `seq_number`
Data Encryption (for Confidentiality)

krb_priv Messages

key
IV = 0

key
IV = 0
**Data Integrity - kerb_safe Messages**

**rsa_md5_des**

```
<table>
<thead>
<tr>
<th>mes. type (20)</th>
<th>pvno (5)</th>
<th>etype</th>
<th>kvno</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>confounder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>user-data</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t-stamp</td>
<td>5 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>seq. no.</td>
<td>sender IP-addr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>receiv. IP-addr</td>
<td>pad</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Encryption: CBC ENC**

```

- IV = 0
- key ⊕ F0F0F0F0F0F0F0F0

**Confounder**

**rsa_md5**

```

```
Data Integrity - kerb_safe Messages (ctnd.)

```
msg_type (20) | pvno (5) | etype | kvno
-------------|---------|-------|-----
```

```
tag | len | user-data | t-stamp | 5 ms | t-stamp | seq. no. | sender | receiv. | IP-addr | IP-addr | pad | confounder | des_mac
-----|-----|-----------|---------|------|---------|----------|--------|---------|---------|---------|-----|-------------|---------

key $\oplus$ F0F0F0F0F0F0F0F0
IV =0

CBC ENC

C

P

Forwarding

AS

AS_REQ
F-able TGT

A

TGS_REQ
tkt for B

A_addr | F-able | A | TGS

TGS

TGS_REP

B

A, B

Cred. Cache

A

A_addr | F-able | A | TGS

TGS

TGS_REQ
addr_list, unF-able

A_addr | F-ed | A | TGS

B

A, TGS

TGS_REP

A, TGS

TGSTGSA_addr | F-able

A, B

TGSTGSA_addr

A, TGS

TGSTGSA_addr | F-ed

A, B

TGSTGSA_addr

A, TGS

TGSTGSA_addr

A, B
nonce
B

nonce
A, B

msg_type (22)
pvno (5)
ticket (s)

A, TGS
prealm
pname (A)
flags
authtime
starttime
endtime
renew_till
srealm
sname (TGS)
caddr (B_addr)

nonce
timestamp
usec
saddr (A_addr)
raddr (B_addr)

addr_list
F-ed
A
TGS

B

A, TGS

A

krb_cred
krb_error

msg_type (30)
pvno (5)
ctime
cusec
stime
susec
error-code
cname (A)
crealm
realm
sname (B)
e-text
e-data