Cross-Realm Kerberos Authentication
A study into implementations and compatibility

Esan Wit         Mick Pouw
Supervisor: Michiel Leenaars

A System and Network Engineering RP
University of Amsterdam

July 3, 2014
Introduction

- Around since ancient times ('80s)
  - Version 5 from 1993, revised in 2005
- Offers authentication in networks between clients and services
- Single Sign On
  - “Yesteryear’s OAuth”
- Many implementations exist
  - Active Directory
  - Heimdal
  - MIT Kerberos
  - Shishi
Previous research

- Implementation of cross-realm referral handling in MIT Kerberos client
- Research by Cervesato et al. illustrated the possibility to impersonate users by rogue KDCs
- Much debate about cross-realm options
  - But very little in the way of implementations
- Specifying Kerberos 5 cross-realm authentication
The goal is to check the current status of Kerberos implementations and identifying possibilities for dynamic configuration to enable cross-realm authentication. E.g. using an @OS3.NL account to authenticate a user for their Facebook profile.

- Analyse the interoperability between implementations
- Research default behaviour for edge cases
- Research options for Cross Realm trust configurations
- Analyse cryptographic behaviour
Kerberos recap

- Authentication provider relying on trusted third party
- Based on shared secrets
- Tickets are encrypted so only the intended recipient can decrypt it
- Designed to provide authentication on untrusted networks
- Password is not send over the network
- Supports public key cryptography
Kerberos Illustrated

Client@MIT.OS3.NL

KDC MIT.OS3.NL

1: AS_REQ
2: AS_REP
3: TGS_REQ
4: TGS_REP

5: Ticket authenticated
6: Time authenticated

Service
MIT.OS3.NL
Testing basic functionality

- Testing combinations of all implementations, focused on receiving a valid ticket
- Clients authenticated using password
- Services using keytab via GSS-API

Requirements
- Machines taking role of either client, service, or KDC.
- Configured DNS
- Patience
Testing basic functionality

- Testing combinations of all implementations, focused on receiving a valid ticket
- Clients authenticated using password
- Services using keytab via GSS-API

Requirements

- Machines taking role of either client, service, or KDC.
- Configured DNS
- Patience
  - A lot of it
# Testing basic functionality

<table>
<thead>
<tr>
<th>Client</th>
<th>Active Directory</th>
<th>KDC</th>
<th>MIT</th>
<th>Shishi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Directory</td>
<td>✓</td>
<td>✗¹</td>
<td>✗¹</td>
<td>✗¹</td>
</tr>
<tr>
<td>Heimdal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MIT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shishi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service</th>
<th>Active Directory</th>
<th>KDC</th>
<th>MIT</th>
<th>Shishi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Directory</td>
<td>✓</td>
<td>✗¹</td>
<td>✗¹</td>
<td>✗¹</td>
</tr>
<tr>
<td>Heimdal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MIT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shishi</td>
<td>✗²</td>
<td>✗²</td>
<td>✗²</td>
<td>✗²</td>
</tr>
</tbody>
</table>

**Table:** Compatibility between implementations

¹ No service available for testing  
² Shishi GSSAPI not implemented yet, but service ticket can be requested
## Crypto compatibility

<table>
<thead>
<tr>
<th>Ciphers</th>
<th>Active Directory</th>
<th>Heimdal</th>
<th>MIT</th>
<th>Shishi</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES128/256-SHA1</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>CAMELLIA128/256-CTS-CMAC</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>DES3-CBC-SHA1</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DES-CBC-CRC(^3)</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DES-CBC-MD5(^3)</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DES-CBC-MD4(^3)</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>RC4-HMAC-EXP(^3)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>RC4-HMAC</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table: Ciphers implemented

\(^3\) Considered weak[2]
Testing PKINIT compliance

- Use of public key cryptography for authentication and encryption
- Chain of trust maintained as standard X.509 certificates
- Any certificate authority
- Extended Key Usage (EКU)
  - X.509 Subject Alternative Name (SAN) extension
- Or if you’re Microsoft:
  - dNSName containing a SAN of the hostname of the KDC
PKINIT Results

• Shishi no support.
• Windows has its own format
• MIT EKU tested/confirmed
• Heimdal support for both formats, EKU tested/confirmed
  • Connecting to MIT KDC weak encryption, DH parameters
Kerberos uses DNS to find the KDC servers of a realm. This is accomplished by using SRV records and will make the realm configuration in the configuration

- `_kerberos._tcp.ad.os3.nl. IN SRV 01 00 88 ad.os3.nl.`
- `_kerberos._udp.ad.os3.nl. IN SRV 01 00 88 ad.os3.nl.`

- Behaviour was analysed under several configurations
- MIT Kerberos 5, Heimdal and Shishi clients all use DNS if realm is unknown\(^4\)

\(^4\)provided a user specifies a realm when attempting to perform initial authentication
Cross-Realm setup

- All manually configured, no automatic options available
- Requires shared secret between KDCs
- All cross-realm trusts are one-way
  - Add a principal in the right direction
- Two-way trust is possible
  - Add principals for both directions
## Cross-Realm requirements

<table>
<thead>
<tr>
<th></th>
<th>Active Directory</th>
<th>Heimdal</th>
<th>MIT</th>
<th>Shishi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Directory</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Heimdal</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>MIT</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Shishi</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Table:** Cross compatibility

5 Shishi does not support cross realm configuration
Conclusion

- The implementations adhere to the protocol
  - Most conflicts occur from other variables
- Much remains to be done to enable auto-configuration
  - Public key cryptography for communication between KDCs
- Heimdal and MIT Kerberos 5 are most compatible

Note:
Many documents are outdated when it concerns Kerberos
Future Work

- Finish Shishi
- Better debugging options in the implementations
- Improve interoperability between implementations
- Dynamic configurable trust
- Foreign trust policies
- Asynchronous Cryptography for Cross-Realm trust
  - PKCROSS started as draft but remains unfinished
    - As of this week some activity again on the mailing list
Questions?

Takeaways in Kerberos

- Check your time
- KERBEROS LOVES CAPS (and so do config files)
- When in doubt, DNS!

Special thanks to Michiel Leenaars and Rick van Rein for their input and feedback during this project.