secureendpoints

Kerberos v5 Tutorial

Ken Hornstein

with special guests
Russ Allbery &
Jeffrey Altman

Scope of Tutorial

- Will cover basic concepts of Kerberos v5 authentication.
- Will lean heavily toward open-source Kerberos v5 implementations
- Will cover more advanced topics such as:
  - Kerberos-AFS interactions.
  - Cross-realm authentication.
  - Encryption type negotiation.
  - Security concerns with Kerberos.
  - Introduction to SASL and GSSAPI.
  - Discussion of PAM, Network Identity Manager, OpenSSH and Russ' Kerberos tools.

secureendpoints

Basic Introduction: Kerberos v5 Design Goals

- Mutual authentication of trusted parties in untrusted environments.
- Single Sign-On: authenticate to multiple services after an initial credential acquisition.
- Federated authentication between clients and services managed within independent authentication domains.
- Secure generation and distribution of session keys.

secure**e**ndpoints

Basic Introduction: Kerberos v5 Properties

- Kerberos v5 uses symmetric key encryption.
  - Public key encryption can be used for initial authentication.
- Kerberos v5 is a trusted third-party authentication system.
- Kerberos v5 cross-realm paths permit federation.
- Kerberos v5 provides authentication; not authorization.

secureendpoints

Operating Environments Shipping with Kerberos v5

- Microsoft Windows (2000 and above)
- MacOS X
- AIX
- Java
- DCE

- Linux
- Solaris
- Novell e-Directory
  - (aka Directory Services for Windows)
- others

There is a current lack of mobile device platforms.

secureendpoints

Common Services supporting Kerberos v5 authentication

- Logon Service
  - PAM
  - login
- File Transfer
- Source Repositories
- LDAP
- Printing
- Instant Messaging

- Mail Services
  - IMAP, POP3, SMTP
- File Systems
  - AFS
  - NFSv4
  - CIFS
- Web
  - HTTP, SOAP

secure**e**ndpoints

Entities Involved in Kerberos Authentication Communications

- Client - generally corresponds to a user or long running process:
  - jaltman@SECURE-ENDPOINTS.COM
  - jaltman/cron@SECURE-ENDPOINTS.COM
- Application server - generally corresponds to a service a user wants to access.
  - host/minbar.cs.cmu.edu@CS.CMU.EDU
  - afs/cmf.nrl.navy.mil@CMF.NRL.NAVY.MIL
- Key Distribution Center (KDC)
  - Stores all encryption keys (aka long term secrets).

secureendpoints

Long Term Secrets

- All Kerberos clients and servers are assigned a long term encryption key.
  - Can be either symmetric or asymmetric.
- User keys are generated from a password unless public key initial authentication is used.
- Service keys are usually random and stored on the machine hosting the service.
  - Active Directory is the exception. Generates service keys from stored passwords.

secure**e**ndpoints

Basic Kerberos v5 Message Flow

- Clients send messages to the KDC and obtain "service tickets" that prove their identity to application servers.

- Each service ticket is encrypted with the application server's encryption key (the client doesn't know the application server's key).

- The application server decrypts the ticket and uses the information inside of the ticket to authenticate the client.
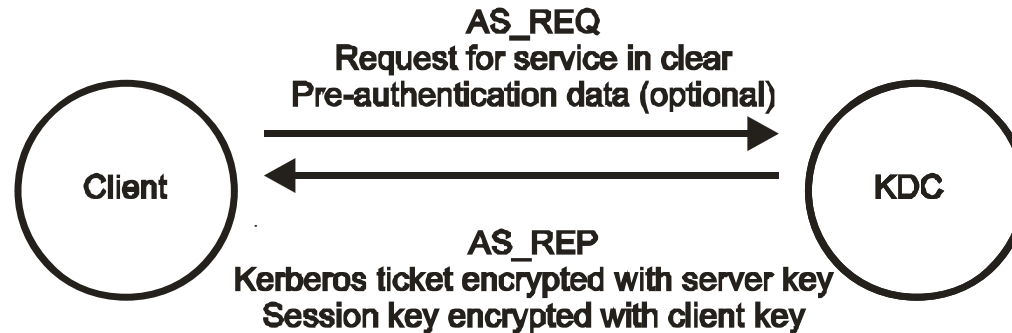
secureendpoints

Kerberos Ticket Contents

- The Kerberos ticket contains:
  - Client identity
    - jaltman@SECURE-ENDPOINTS.COM
  - Application server identity
    - afs/secure-endpoints@SECURE-ENDPOINTS.COM
  - Session encryption key
  - Start time
  - Expiration time
- One Part encrypted with the client key and one part encrypted with the application service key

secure**e**ndpoints
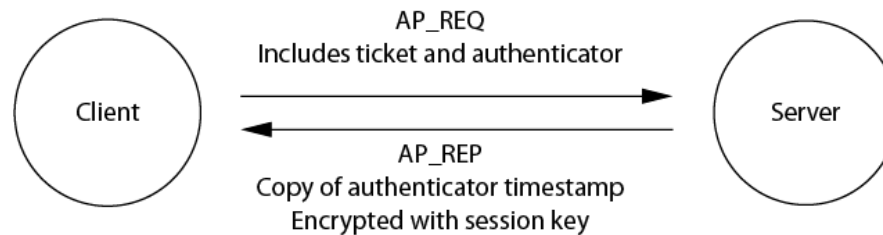
Kerberos Authenticator Contents

- Kerberos Authenticator contains:
  - Client identity
  - Checksum
  - Timestamp
  - Optional sub-session encryption key
  - Optional sequence number
- Kerberos authenticator is created by the client and is encrypted with the session key in the ticket.

secure**e**ndpoints

Kerberos Messages - AS_REQ/AS_REP

AS_REQ
Request for service in clear
Pre-authentication data (optional)

Client → KDC

AS_REP
Kerberos ticket encrypted with server key
Session key encrypted with client key

- The message sent from the client to the KDC is called AS_REQ.
- This is a request for a ticket for the desired service.
- This request is sent in the clear
- This request may include pre-authentication data.
  - For example, a time stamp encrypted with the client's key
- The message sent from the KDC to the client is called AS_REP. This contains the Kerberos ticket and a session key.
- The ticket is encrypted with the application server's key.
- The session key is encrypted with the client's key.
- Note that a copy of the session key is included in the ticket.

secureendpoints

Kerberos Messages - AP_REQ/AP_REP

AP_REQ
Includes ticket and authenticator

Client → Server

AP_REP
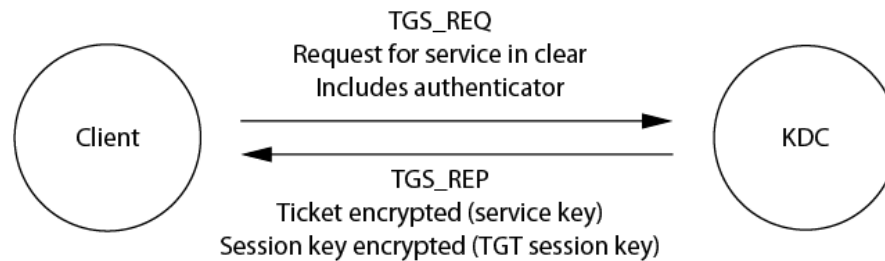Copy of authenticator timestamp
Encrypted with session key

- The message sent from the client to the application server is AP_REQ.
- This includes the ticket (encrypted with server's key), and the authenticator (encrypted with session key).
- A new authenticator is generated by the client for every AP_REQ.
- Application server decrypts ticket, which contains client identity and session key. The session key is then used to decrypt the authenticator. The authenticator timestamp is then verified to be recent (within 5 minutes) in order to reduce the risk of replay attacks. If the authenticator is up-to-date, then the server knows that it is talking to the client, and they both have a session key.

secureendpoints

Ticket Granting Service & Ticket

- The AS_REQ requires that the client knows it's encryption key every time it wants to talk to a new service.

- This isn't convenient for users (they have to keep typing in their password every time they connect to a service).

- In Kerberos there exists a special service called the Ticket Granting Service, who's job it is to issue tickets for other services.
    - This service is located on the KDC.
    - It has a special ticket name: krbtgt/REALM@REALM.

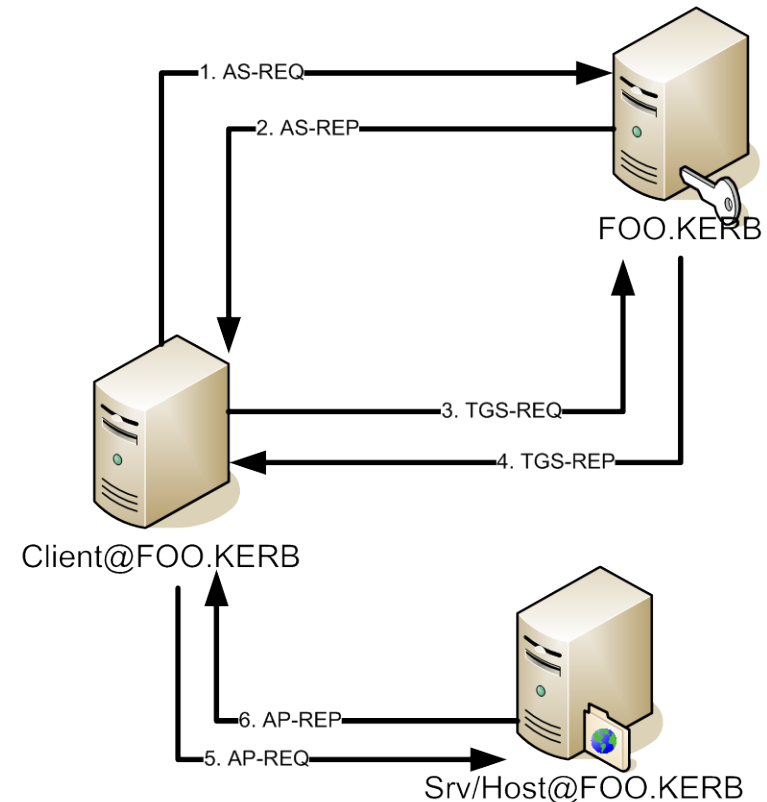- It uses a special messages to talk to the KDC: TGS_REQ and TGS_REP.

secure**e**ndpoints

Kerberos Messages - TGS_REQ/TGS_REP

TGS_REQ
Request for service in clear
Includes authenticator

Client → KDC

TGS_REP
Ticket encrypted (service key)
Session key encrypted (TGT session key)

- A normal AS_REQ and AS_REP exchange takes place to acquire the TGT (krbtgt/REALM@REALM).

secur**e**ndpoints

Main differences between AS_REQ and TGS_REQ

- TGS_REQ includes ticket (for krbtgt service) and authenticator.

- TGS_REP has session key encrypted with TGT session key, NOT the user's key.

secure**e**ndpoints

Kerberos v5:
Summary of Message Exchange

- The Authentication Service (AS) Exchange
  - The client obtains an "initial" ticket from the Kerberos authentication server (AS), typically a Ticket Granting Ticket (TGT).
  - The AS-REQ may optionally contain pre-authentication data to prove the client's identity.
  - The AS-REP, containing an authenticator (aka ticket), is encrypted in the client's long term key.
- The Ticket Granting Service (TGS) Exchange
  - The client subsequently uses the TGT to authenticate and request a service ticket for a particular service, from the Kerberos ticket-granting server (TGS).
- The Client/Server Authentication Protocol (AP) Exchange
  - The client then makes a request with an AP-REQ message, consisting of a service ticket and an authenticator that certifies the client's possession of the ticket session key. The server may optionally reply with an AP-REP message. AP exchanges typically negotiate session specific symmetric keys.

1. AS-REQ

2. AS-REP

FOO.KERB

3. TGS-REQ

4. TGS-REP

Client@FOO.KERB

6. AP-REP

5. AP-REQ

Srv/Host@FOO.KERB

secureendpoints

Kerberos Messages in The Real World

- AS_REQ & AS_REP - Performed by kinit, login.krb5, pam_krb5, etc etc. Sent to UDP/88 and TCP/88 on KDC.

- TGS_REQ &TGS_REP - Performed by Kerberos client application (Kerberized ftp or ssh). Uses TGT acquired during AS_REQ & AS_REP. Sent to UDP/88 and TCP/88 on KDC.

- AP_REQ & AP_REP - Sent between the Kerberos client application and the application server. Generally encapsulated in the application protocol in a protocol-specific manner.

secure**e**ndpoints

Additional Kerberos Messages KRB_SAFE

- Used to integrity-protect (via keyed checksum negotiated by an AP_REQ/AP_REP exchange) protocol data. Additionally can provide replay detection. Designed to be used by clients and application servers.

- Not recommended for use by new application protocols.

secureendpoints

Additional Kerberos Messages KRB_PRIV

- Used to provide data privacy (encryption) using a session key negotiated by an AP_REQ & AP_REP exchange. Also designed to be used by clients and application servers.

- Not recommended for use by new application protocols.

secure**e**ndpoints

Additional Kerberos Messages KRB_CRED

- Holds a ticket (generally a krbtgt ticket) plus the associated session key (session key protected by key negotiated in AP_REQ & AP_REP).

- Can be used to Forward tickets to a remote host as part of an application protocol or insert tickets into the Windows Vista LSA cache using the SubmitTicket LSA operation.

secureendpoints

Kerberos Keys & Version Numbers

- Kerberos supports multiple encryption algorithms (DES, 3DES, RC4, AES-128, AES-256).
- Each principal can have multiple keys of different encryption types.
- In Kerberos messages, the encrypted parts are tagged with the encryption type so Kerberos peers can select the appropriate key.
- Kerberos principals can also have multiple keys of the same encryption type.
- These are distinguished by the key version number (kvno).
- The Kvno is incremented when keys are changed and are used to select the appropriate key for decryption.

secureendpoints

Other Properties of Kerberos Tickets

- Tickets may be flagged with special properties:
  - Forwardable and Forwarded
  - Proxiable and Proxy (useless)
  - May Postdate and Post Dated
  - Invalid
  - Renewable
  - Initial
  - Hardware Authenticated
  - Pre-authenticated
  - Transited Path Checked (cross-realm)

secureendpoints

Authentication versus Authorization

- Kerberos is an authentication service - it answers the question, "Who are you?"
- This is distinct from authorization, which answers the question, "What are you allowed to do?"
- Note that you need both questions answered to properly provide access control!
- Since Kerberos doesn't provide an authorization service, what is really going on?

secureendpoints

Common Authorization using Kerberos v5

- Interactive Kerberos services requesting access to Unix accounts use a simple algorithm to perform authorization checks.
  - If the user matches the Kerberos principal name and they're in the local realm, they're permitted access.
- Most of these interactive services call a function called krb5_kuserok(); this also allows a user to list explicit principals in a .k5login file in their home directory.
- More complex software has an explicit authorization server. In AFS this is provided by the ptserver and the ACLs stored in volumes on the fileserver.
- Windows Kerberos places group membership in the Kerberos ticket, which is used by application services for authorization control.

secureendpoints

Requirements: Kerberos KDCs

- Doesn't have to be a fast machine.
- Past wisdom says the machine should be completely dedicated to KDC function (or at least maintain a consistent security boundary)
  - Microsoft Active Directory and Novell e-Directory are not dedicated
- Should be as secure as you can make it (minimum services).
- For Unix-based KDCs, either one of the two major open-source implementations would be a good choice.
- Might want multiple KDCs for redundancy and reliability.

secure**e**ndpoints

Site Requirements:
Windows Domain Controllers

- Obviously a Windows Domain Controller is not a dedicated machine

- KDC uses a semi-public "directory" for database access

secure**e**ndpoints

Requirements: Kerberos Application Servers

- Generally, replace or install server binaries that support Kerberos authentication.
  - If you're lucky; they already come with your operating system.
  - If you're not, then likely your Kerberos implementation will include a basic set.
- Every application server also needs an encryption key registered with the KDC.
  - The specific details of how this works depends on your \
  - Kerberos implementation.
  - Key is usually stored in a file on disk (keytab).
- Generally, you need a configuration file describing basic information about your Kerberos realm.

secureendpoints

Requirements: Kerberos Clients

- The client (be it a person or a program) needs to have an encryption key assigned.

  - If it's a person, then the encryption key is their password.

- The client needs access to Kerberos client software (kinit and whatever Kerberos client programs you want to give them).

- If you want to use Kerberos for login authentication, you either need an appropriate PAM module or an appropriately modified login program.

- The client also needs a Kerberos configuration file describing the realm information (KDC locations).

secure**e**ndpoints

Basic Kerberos Administration

- The basic program to administer open-source KDCs is kadmin.

- kadmin is specific to a Kerberos implementation (can't use Heimdal kadmin with MIT and vice versa).

- Can generally perform all of the operations an adminstrator needs to perform via kadmin.

- Client talks to special server (kadmind) running on KDC.

secureendpoints

Kerberos Administrative Commands

- Creating users:
  - MIT: addprinc new_user
  - Heimdal: add new_user
- Creating service keys:
  - MIT: addprinc service, ktadd service
  - Heimdal: add -r service, ext_keytab service
- Adjusting password policies (expiration time, history)
  - MIT: add_policy, modify_policy, modprinc -policy policyname
  - Heimdal: No direct equivalant, but allows a specified shared library to act as password quality checker. See documentation for more details:

secureendpoints

Cross-Realm Authentication

- Kerberos allows a user in one realm to authenticate to services in another realm.
- Doing this requires the cooperating realm adminstrators create cross-realm principals in their realms and assign them the same encryption key.
- When a client wants to access a service in another realm, it asks it's KDC for a special cross-realm TGT.
- It then uses that cross-realm TGT to asks for services in the foreign realm.

secure**e**ndpoints

Cross-Realm Authentication Flow

- #1 - Client talks to local KDC, requests cross-realm ticket (krbtgt/REALM-B@REALM-A) using normal TGS_REQ.
- #2 - Client sends TGS_REQ to Realm B KDC, but presents cross-realm ticket. KDC sends ticket back for service, but sets client name in ticket to client@REALM-A.
- #3 - Client sends standard AP_REQ to server.

KDC-A

Client

KDC-B

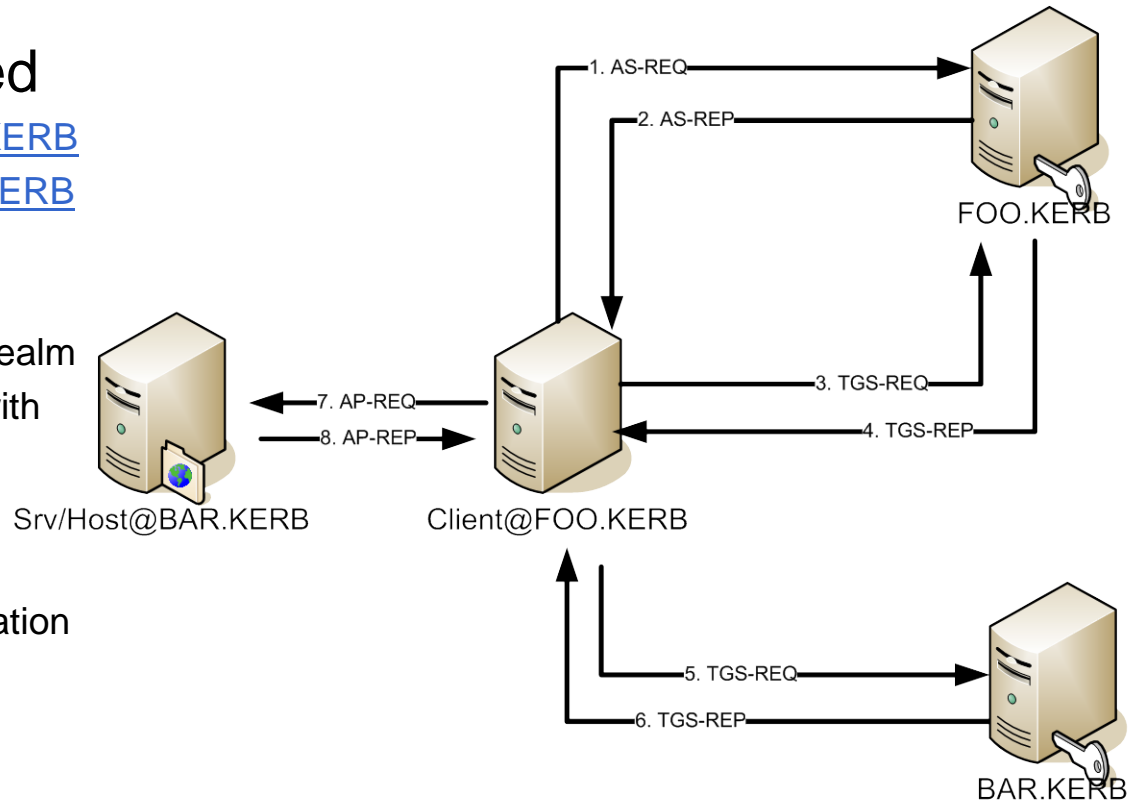Service

secureendpoints
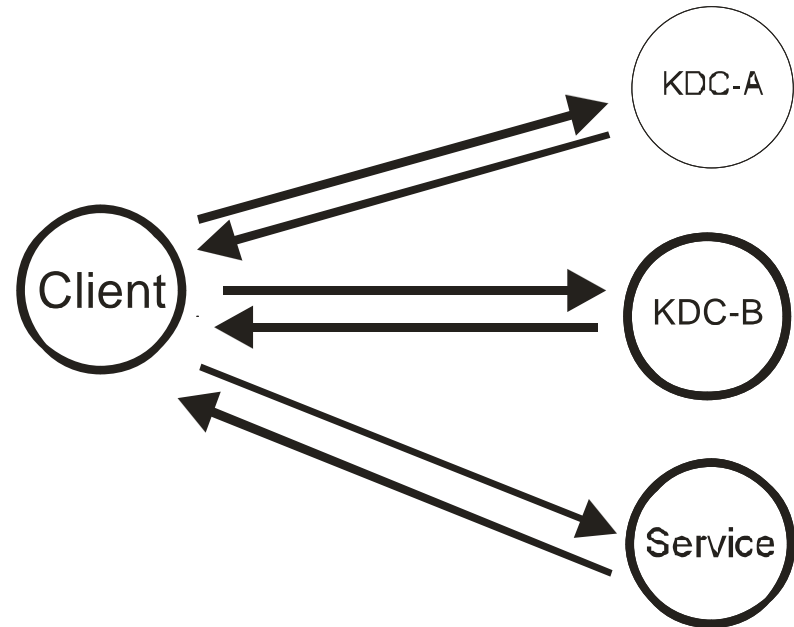
Kerberos v5 Cross Realm

Tickets Obtained

krbtgt/FOO.KERB@FOO.KERB
krbtgt/BAR.KERB@FOO.KERB
Srv/Host@BAR.KERB

Cross Realm works when realm
FOO.KERB shares a key with
realm BAR.KERB.

In all cases, the KDC must
share a key with the application
Service.

1. AS-REQ
2. AS-REP
FOO.KERB

3. TGS-REQ
4. TGS-REP

7. AP-REQ
8. AP-REP
Srv/Host@BAR.KERB   Client@FOO.KERB

5. TGS-REQ
6. TGS-REP
BAR.KERB

secure**e**ndpoints

Client Initiated Cross-Realm: Host Name to Realm Mapping

- The client automatically attempts cross-realm authentication when it detects that the requested server is in another realm.
- The client determines that a server is in another realm by looking at the DNS domain name of the server, and attempting a DNS name to Kerberos realm mapping on it.
  - This mapping can be configured either in the Kerberos configuration file, or via DNS.
- The client then asks the local realm for a cross-realm TGT for the foreign realm.

secure**e**ndpoints

Configuring & Testing Cross-Realm

- Since it requires the coordination of both admins, generally do it while on the phone together or both in front of laptops.
- Create krbtgt/REALM-A@REALM-B and krbtgt/REALM-B@REALM-A.
  - I have a program which creates random passwords for these principals.
- Can test it with "kvno" (MIT only) or use a Kerberos utility to try to connect to a server in the foreign realm.
- If you get a service ticket for the foreign realm, then it works!

secureendpoints

Common Mistakes

- Cross-realm principal kvnos don't match.
- Incompatible list of supported encryption types.
- Some implementations require exact same list of encryption for cross-realm principals in the same order.
- Cross-realm keys don't match.
- In most cases, you need to look at the KDC log files to determine the exact problem (error feedback to the user is not good).

secureendpoints

Using Kerberos Cross-Realm With AFS

- In addition to configuring cross-realm Kerberos, a few extra steps are necessary with AFS.
  - Create a cross-realm PTS group (system:authuser@foreign.realm).
    - Note: the owner of this group must be system:administrators.
  - Give it a high group quota
    - as many people as you think you'll have coming in from that realm
    - Or pre-create the users from the foreign realm that should be given access.
  - Modern version of aklog will automatically create the cross-realm PTS user (user@foreign.realm) the first time they aklog to the foreign realm.

secureendpoints

# Side Effects of Cross-Realm and AFS

- Users end up with a semi-random PTS id (not changeable), and so do files that they create.
- This can cause software that interprets file ownership directly instead of using access() to misbehave
- Cross-realm users cannot appear in the Bos UserList.
- Cross-realm users are not members of system:authuser.
  - They are members of the foreign-realm authuser group.

secureendpoints

Kerberos v5 support for Single Sign-On

- Ticket Granting Ticket permits access to multiple network services from a single machine without re-entering the initial credentials.

- Ticket Forwarding permits tickets to be migrated to additional machines to extend the single sign-on experience.

secure**e**ndpoints

Kerberos v5 Delegation aka Ticket Forwarding

- Delegation utilizes the ability to FORWARD tickets from a client machine to a service.

- The service can then assume the identity of the client in order to authenticate to a subsequent service.

- Constraints can be applied to the forwarded tickets using authorization data.
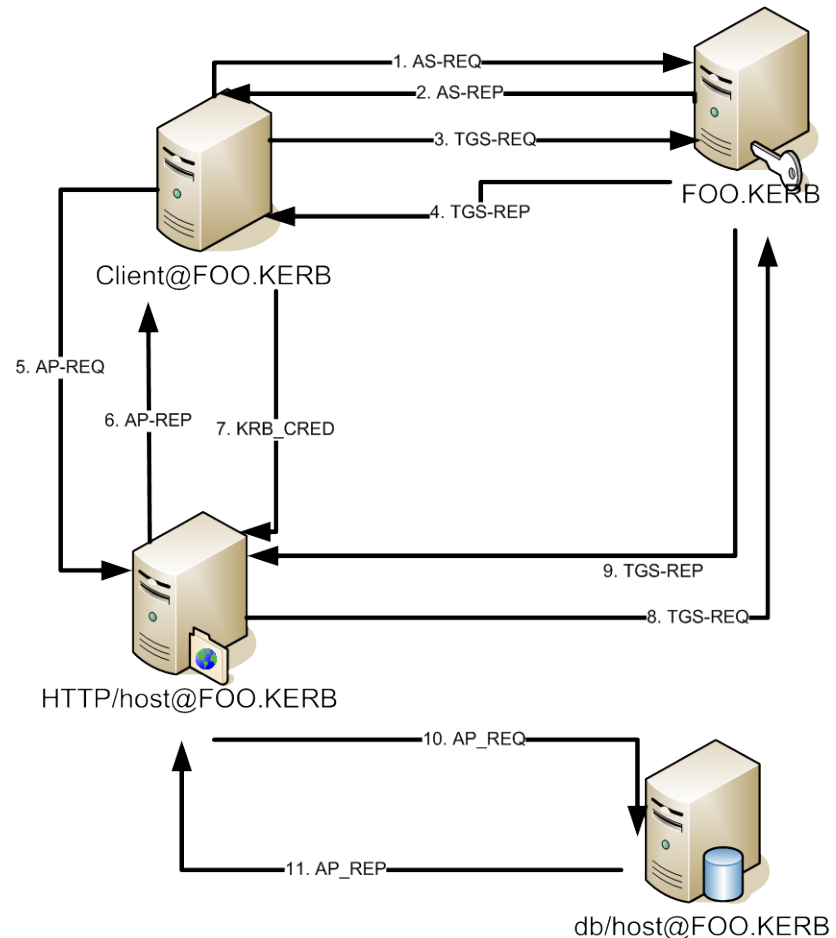
Weaknesses of the Single Sign-on Model

- Support for constraining delegation to particular services and uses is limited.

- Ticket forwarding occurs within application protocols. Few protocols provide any method for refreshing forwarded tickets when they expire.

secure**e**ndpoints

Encryption Type Negotiation

- Kerberos supports multiple encryption types. This is both a good and a bad thing.

- Good: You can easily change/upgrade encryption types for clients and services.

- Bad: It's easy to screw things up.

- Kerberos gives you good tools to handle multiple encryption types, but you need to understand the limitations.

secureendpoints

Basic Encryption Negotiation

- When client sends AS_REQ or TGS_REQ to the KDC, they include a list of all encryption types that the client supports.

- The KDC selects the "best" encryption type (the client lists the ones it prefers first) depending on the keys available and the target server but the KDC chooses.

- However, the "best" encryption type depends on which key or data we're talking about, and which server you're talking to!

- Encryption Types include
  - AES-256, AES-128, RC4-HMAC, 3DES,and DES variants

secure**e**ndpoints

Encryption type #1 Response enctype

- The enc-type used to encrypt the session key when the KDC replies to the client.
- The key used is either the client's password (AS_REQ) or the TGS session key (TGS_REQ).
- The possible enc-types are an intersection of the client requested enc-types and the keys registered for that client on the KDC.
- The client is the only one who cares about this enc-type.
- The only way to get a new enc-type for a client's key is to re-key it.
- This is one reason why regular password changes are important!
- You can get in a situation where an AES session key is encrypted with a single-DES key.

secureendpoints

Encryption type #2
Ticket enctype

- The enc-type used to encrypt the service ticket that is (eventually) sent to the application server.
- The key used is select by the KDC from the "best" enc-type (based on ordered preference list in the KDC) from the encryption types registered for the server.
- Normally, the client does not care about this enc-type (since it cannot decrypt the Kerberos ticket).
- HOWEVER ... some Kerberos client implementations will reject a ticket if it contains an enc-type that they are unfamiliar with. (Old versions of MIT Kerberos and some Java Kerberos implementations).
- Locally, NRL created a patch to the KDC that will only issue tickets for enc-types that were indicated in the client request (only recently turned that off).

secureendpoints

Encryption type #3
Session key enctype

- The enctype of the key that is sent in the response (encrypted, of course) and the service ticket.

- Used for authenticator validation, session encryption, and other application-specific uses.

- Has to be understood by both the client and application server.

- This enc-type is chosen by selecting the "best" enc-type that both the client indicates it supports in the request and the enc-types registered on the server.

secureendpoints

Log File Examples

- MIT:
  - TGS_REQ (5 etypes {23 18 16 3 1}) 1.2.3.4(88): ISSUE: authtime 987654, etypes {rep=23 tkt=16 ses=16}, kenh@REALM for host/elvis.realm@REALM

- Heimdal:
  - AS-REQ foo@MEATBALL.SE from IPv4:1.2.3.4 for krbtgt/MEATBALL.SE@MEATBALL.SE
  - Using des-cbc-md5/des-cbc-md5
  - Requested flags: renewable_ok, renewable, forwardable
  - sending 505 bytes to IPv4:1.2.3.4

secureendpoints

What Enc-types Should I Support?

- Depends on site requirements, policies, etc etc, but general guidelines:
- You should support as many enc-types as possible.
  - Gives you the most flexibility in case security problems are discovered with algorithms.
  - Impossible to add encryption types to clients without password change.
  - Since the client tells the KDC what enc-type it supports, it's relatively safe to add new enc-types to client principals.
- You need to limit enc-types for application servers to what the Kerberos implementation on your servers support.
- You might have other weird reasons to limit enc-types (you like to randomly rename users). You know who you are.

secure**e**ndpoints

Handling Enctype Migration - General Rules

- Make sure your KDC supports all enctypes you would ever want to use.
  - In other words, upgrade it first.
- Have regular password expiration to insure users get latest encryption types.
  - Having your AES key protected by a much weaker enctype is dumb, but it happens.
- Only place keys on application server machines that are supported by that version of Kerberos.
  - You can restrict the enc-types by the -e switch to ktadd (MIT) and using del_enctype (Heimdal).
- If you support MIT, you could write log analysis scripts to determine which clients support which enc-types.

secureendpoints

GSSAPI and Kerberos

- A generic API designed to support different security systems.

- The most common security system supported by GSS today is Kerberos 5.

- The Kerberos 5 mechanism for GSSAPI defines special network messages similar to, but NOT THE SAME AS Kerberos 5 messages.

- Thus, a protocol which speaks GSSAPI cannot receive raw Kerberos messages, and vice versa (the AP_REQ/AP_REP is encapsulated in a GSSAPI token).

- A new AFS Rx security class is being implemented via GSSAPI (rxgk).

secure**e**ndpoints

SASL and Kerberos

- SASL is a generic protocol framework for negotiating different   authentication mechanisms in a protocol.
- This is used by such protocols as IMAP, SMTP, and XMPP.
- One of the supported mechanisms in SASL is GSSAPI.
- Thus, if a protocol uses SASL for authentication, then it supports GSSAPI, which means the protocol has a defined way of supporting Kerberos v5.
- This doesn't mean, unfortunately, that a particular client or server will support GSSAPI/Kerberos v5, unfortunately.

secure**e**ndpoints

Security Considerations

secure**e**ndpoints

What About Kerberos v4?

- MIT announced v4 use as deprecated more than three years ago
- MIT and its OEMs are phasing out support
  - Apple no v4 support in MacOS X 10.6 (Snow Leopard)
  - MIT Kerberos 1.7 and later have no v4 support
- Sun Solaris and Microsoft already have no Kerberos v4 support
- OpenAFS has announced kaserver use as deprecated

secureendpoints

Why is Kerberos v4 bad?

- Built upon 56-bit DES with no ability to migrate to stronger data protection methods

- Serious design flaws in cross-realm protocol leave systems open to attack if cross-realm support is enabled

- All Kerberos v4 tickets include one IPv4 address

  - There are serious problems with NATs and multi-homed clients

secure**e**ndpoints

Still using Kerberos v4 or AFS kaserver?

- Migrate to Kerberos v5 ASAP
  - Ken Hornstein's Kerberos v5 migration kit
    - aklog and asetkey are part of OpenAFS
    - kaforwarder is part of MIT Kerberos
    - afs2k5db remains to be built in-house
  - Implement a new Kerberos v5 realm and begin to migrate users and services
- Time is running out!

secureendpoints

Security Considerations:
Off-line AS_REP decryption

- The AS_REP is encrypted with the user's password.
- Since the AS_REQ is unauthenticated, anyone can ask for a ticket for any user.
- Once you receive an AS_REP for a user, you can perform trial password decryptions for essentially forever.
- Mitigation Strategies
  - Require pre-authentication. Only partial fix (prevents anyone from requesting a ticket, but AS_REPs can still be sniffed).
  - Use hardware pre-auth (combines user's password with token output).
  - Implement password quality checking and regular expiration.

secureendpoints

Security Considerations: KDC Spoofing (the Zanarotti attack)

- Some applications want to just verify the Kerberos password was correct for access control (e.g., screensavers, crappy web server software).

- However, doing an AS_REQ/AS_REP exchange is NOT sufficient!

- An attacker could pretend to be a KDC and send back a AS_REP encrypted with a key known to the attacker.

- Exploits common misunderstanding about Kerberos - authentication is only valid after AP_REQ/AP_REP exchange.

- Solution:
  - Insure that received TGT is used to generate an AP_REQ that is verified against locally-stored service key.

secureendpoints

Security Considerations Client-side Credential Theft

- On most Unix systems, Kerberos tickets are stored in a file in /tmp.

- If the user's account is broken into or root is compromised, the credentials can be stolen by an attacker.

- Have seen this happen (relatively unsophisticated attacker).

- Mitigation Strategies
  - Try to insure as much protection on client systems (difficult)
  - Implement a better credential cache type.
  - No good solution to the untrusted host problem..

secure**e**ndpoints

Security Considerations Authenticator Replay

- The timestamp in the authenticator in the AP_REQ only has to be valid within a 5 minute window.

- An attacker can replay the ticket and authenticator within the 5 minute window and convince the application server that the ticket is still valid.

- MIT implements a replay cache; remembers old authenticators and checks new ones against the list. Heimdal implements a replay cache, but isn't turned on by default.

- Depending on the specific protocol and the use of subkeys, it may not be an issue.

secure**e**ndpoints

Security Considerations: Cross-realm

- If the foreign KDC is compromised or has a malicious admin, they can impersonate anyone in that realm (not other realms).

    - In other words, you trust the administrators of a foreign realm to secure their KDCs. If they do not, your systems are at risk of impersonation.

- The default authorization checks do NOT authorize cross-realm users to log into local accounts.

    - Most software gets this right but not all do.

secureendpoints

Common Deployment Issues

secureendpoints

Common Deployment Issues: Clock Skew

- Your client's clock has to agree with your application server's clock within 5 minutes.
- If it's not, you'll get the error "Clock skew too great".
- Generally resetting the client's clock will solve the problem.
- The latest credential caches store an approximate time offset to minimize the problem:
  - Offset = (Local clock - KDC clock + ½ round trip time)
- Note that if your application server or KDC clock drifts too far off, no one will be able to authenticate!
  - **Use Network Time Protocol!!!**

secure**e**ndpoints

Common Deployment Issues: Wrong Kvno

- If the key version number of the stored key doesn't match the version number in the ticket, you will get "Key Table Entry Not Found" (better error from Heimdal).

- Commonly seen when rekeying hosts and configuring cross-realm.

- Use ktutil and kadmin to check key version numbers on KDC keys to make sure they match.

secureendpoints

Common Deployment Issues: Wrong Key

- If the keys don't match, the error message returned is "Decrypt integrity check failed".

- Most common case is a reinstall of KDC from scratch, but the old keys in the KDC host keytabs were not completely removed and regenerated.

secure**e**ndpoints

Common Deployment Issues: Firewall / Network Address Translation

- Kerberos presents a few "interesting" challenges to firewall admins.

- Need to open UDP/88 and TCP/88 to Kerberos KDC and port(s) used by application protocol.

- Some firewalls that do telnet/ftp protocol spoofing lose hard.

- Most NAT lossage can be fixed by using addressless tickets (password changing is still broken with MIT from behind a NAT).

secureendpoints

Common Deployment Issues: Multi-homing

- Mostly a problem for application servers, depending on your DNS configuration.
- If you give each interface a different name in DNS, the client may get the wrong DNS name to construct the Kerberos server principal name.
- Possible solutions:
  - Construct DNS entries so all DNS entries map back to the canonical hostname.
  - Put server names corresponding to all interfaces into the keytab. This can break some software (like MIT ftpd).

secure**e**ndpoints

Common Deployment Issues: Long Running Jobs

- There are jobs that need to run with Kerberos authentication for long periods of time or on a periodic basis
  - Service daemons
  - End user initiated background jobs
  - Cron jobs

- kstart to the rescue

- Consider providing alternate principals

secure**e**ndpoints

AFS Interactions with Kerberos

- As everyone knows, AFS uses Kerberos for authentication.
- But AFS isn't exactly a normal application service.
- Examples:
  - Vanilla AFS ships with a Kerberos v4 KDC, yet many people use it with Kerberos v5
  - You use a special program "tokens" to view your AFS tickets.
  - You use a special program to get service tickets for AFS.

secure**e**ndpoints

How come AFS is so different than other application services?

- AFS Cache Manager is loaded in the operating system kernel
  - No access to the user's credential cache
- The existing AFS RX Security Class (rxkad) was designed when sharing keys among servers was thought to be ok.
- AFS implements its own distributed authorization service (ptserver)
  - Name format is 'user' or 'user@remote.cell' and is case insensitive.
  - Kerberos (v4 and v5) client principal names must be translated

secureendpoints

AFS Service Principals and existing RX security classes (rxkad)

- Unlike other Kerberized services, there is one AFS service principal for all AFS services within an AFS cell.
- This principal name is either
    - "afs@REALM.NAME" (deprecated)
    - "afs/cell.name@REALM.NAME" (preferred)
- The key for this principal is stored on all machines which host AFS services
    - database and fileservers
- This means if one AFS server is broken into, you need to re-key the entire cell!
    - This is a design weakness of rxkad
    - Per-service keying is one of the design goals for the rxgk security class.

secureendpoints

Kerberos Usage Within the AFS Protocol

- When performing authentication with native AFS tools, the following steps take place:
  - AFS utility (klog or AFS-aware login program) does equivalant of a AS_REQ/AS_REP exchange ... except that there are two important differences:
    - The basic packet format is Kerberos v4.
    - The protocol doesn't use the Kerberos wire protocol, but instead talks to the kaserver using Rx.
  - A TGS_REQ/TGS_REP equivalent exchange then takes place, but again you communicate with the kaserver via Rx.
  - The service ticket from the TGS_REP (for the afs service) and the associated session key is then placed into the kernel so the cache manager can use it.

secureendpoints

Cache Manager Interaction with Kerberos

- When the cache manager wishes to perform a file operation on behalf of a user, it does the following things:
  - Makes an Rx connection to the fileserver.
  - The fileserver sends an "rxkad challenge" packet with a nonce.
  - The client sends the token plus an encrypted version of the nonce in an "rxkad response" packet.
  - If the client requested encryption, it is activated
    - rxkad uses fcrypt, a sibling of DES that uses the same key format as a Kerberos DES session key but requires less computing power.

secure**e**ndpoints

AFS "rxkad" tokens were Kerberos v4 Tickets

- All AFS services can understand old-style Kerberos v4 tickets.
- There are two ways to acquire such tickets: natively, and via a translator.
- Natively, you talk to a kaserver or a v4 KDC. The AFS client program places a normal v4 ticket & session key into the cache manager, and everything proceeds normally.
- With the translator, you run a special service called 524 (MIT has a separate daemon, Heimdal includes it into the KDC).
- This process will take a v5 service ticket, decrypt it, and convert it to a Kerberos v4 ticket.
  - Some sites perform name translation (yuck!!!)
- Note that since it uses UDP/4444, you may have trouble getting it through firewalls. (It can be run on any port, including 80, and can be advertised via DNS SRV records)

secureendpoints

AFS "rxkad" tokens can now use Kerberos v5 Tickets

- Newer AFS services can handle a Kerberos v5 service ticket presented to it by the client.
  - 1.2.10 for DES-CBC-CRC
  - 1.2.13 for DES-CBC-MD5 and DES-CBC-MD4 (used by Active Directory)
- There are two important limitations:
  - The ticket (and session key) have to be single-DES.
  - This requires NO changes to the client (the client just treats the Kerberos ticket as a binary blob).
  - In clients older than 1.4.0, the ticket can't be larger than 344 bytes.
  - When the AFS token contains just the encrypted portion of the Kerberos v5 afs service ticket, it is called a 'rxkad2b' token

secureendpoints

Differences Between "klog" and "aklog"

- "klog"
  - Uses Rx protocol to communicate with kaserver.
  - Takes a Kerberos password, does not keep around Kerberos TGT.
- "aklog"
  - Communicates with Kerberos KDC using standard Kerberos protocol.
  - Uses an existing Kerberos credential cache and TGT.
  - Stores AFS ticket in credential cache before placing into kernel.
  - Can perform cross-realm authentication and pts registration.

secureendpoints

To 524, or not to 524?

- aklog used to talk to the 524 service to convert the ticket from a v5 ticket to a v4 ticket.
  - This is no longer the default.
  - Only use the 524 translator if the AFS cell does not support Kerberos v5 OR if it is dependent upon the use of hideous hacks that perform name translation
- Due to a Windows virus that utilized port TCP/4444, many sites and some stupid ISPs have fire walled UDP/4444, which blocks access to the 524 translation service.
- Remember, Kerberos v4 is dead. Don't use 524 unless you have to!!!!

secureendpoints

AFS Usage of Kerberos Principal Names

- With a Kerberos v4 ticket, the client name is used as-is (except that if the principal is in the local realm, the realm is stripped off, otherwise it is lowercased).

- When a Kerberos v5 ticket is received, the same things happen as they do with v4, except a LIMITED amount of v5 to v4 principal name mapping takes place
  - "host" becomes "rcmd"
  - trailing hostname components
  - Multiple component client names are represented in dotted form.
  - Client principals with a dot in the first component are not permitted.

- In either case, once a client name has been produced, it is looked up in the ptserver and converted to a vice ID. It is the ID that is included in access control and group lists.

secureendpoints

Security Consideration: AFS vs Kerberos Naming

- Kerberos principal names are case-sensitive (unless you are using Windows Active Directory)
- AFS protection service names are case-insensitive
- user@REALM, User@REALM, and USER@REALM are all the same to AFS
- Even worse, user@REALM and user@realm are the same
- When establishing cross-realm relationships, do not permit two realms whose names only differ by case

secureendpoints

Using Multiple AFS Cells with one Kerberos Realm

- One of the strengths of AFS is that the administrators of the AFS cell do not need to be the same as the administrators of the Kerberos realm used for authentication.
- In fact, the AFS cell name does not have to have any relationship to the Kerberos realm name.
- It is therefore possible to use a centralized Kerberos realm to authenticate multiple departmental AFS cells
- To do this
  - place the name of the Kerberos realm you want to treat as the "local" realm on the first line of /usr/afs/etc/krb.conf
  - Then create an afs/cell.name@REALM service principal within the Kerberos database for each AFS cell
- **DO NOT create a single afs@REALM service principal and share the key among multiple AFS cells.**

secure**e**ndpoints

Using Multiple Kerberos Realms with a AFS Cell

- Many organizations manage multiple Kerberos realms and synchronize the account allocation
- These organizations wish to permit users to access their data in AFS with any of their identities without requiring the use of separate vice IDs for each identity and the associated management of groups and access control lists.
- In OpenAFS 1.5 and 1.4.5, multiple Kerberos realms can be listed on the first line of the /usr/afs/etc/krb.conf file. Each listed realm will be treated as a local realm.
  - user@MIT.REALM and user@WIN.DOMAIN are both mapped to "user" in the PTS database.
- DO NOT USE if different entities control name allocation in each realm.

secureendpoints

Differences between KDC implementations

secureendpoints

The AFS Authentication Service (kaserver)

- Can only do v4 – Do not use

- Listens on UDP port 7004 for Rx connections for the AFS authentication service.

- Also listens on UDP ports 88 and 750 for Kerberos v4 requests.

- Database distributed redundantly via Ubik protocol.

secureendpoints

KDC Differences: MIT

- Implements V5 Kerberos protocols.
  - PK-INIT support in 1.7.x
- Ships with kaserver emulator software (fakeka)
- Database is copied over to backup servers in bulk.
  - 1.7 adds incremental backups based on Solaris protocol.
- Supports all enctypes
- Used as the basis for Kerberos distributions in
  - MacOS X, Solaris, AIX, HP-UX, OpenVMS, Red Hat
  - Novell e-Directory

secureendpoints

KDC Differences: Heimdal

- Implements V5 Kerberos protocols.
- Provides kaserver emulator integrated into KDC.
- Can do incremental replication to backup servers.
- Supports all newer enc-types
- Supports PK-INIT
  - Compatible with both the IETF standard and Active Directory
- Kerberized Certificate Authority
- Non-Kerberos authentication methods.
- Better tools for importing database from AFS kaserver format.

secureendpoints

KDC Differences: Active Directory

- Implements v5 only
  - does NOT support AFS-salted keys.
- Has multi-master server replication.
- Administrated via LDAP.
- Stores group membership in Kerberos ticket authorization field.
- Supports RC4 and DES, but not 3DES.
  - AES-256 in Vista/Win7 and 2008 Server
- Supports PK-INIT
  - Pre-standard in 2000 and 2003
  - Standard in 2008
- Case insensitive principal names
- Client and Server Referrals
- Constrained Delegation
- Tools for managing keys for non-Windows services are poor.

secureendpoints

How to Decide Which to Use?

- If you are primarily a Microsoft shop and can restrict your use to the protocols and extensions that Microsoft supports, then Active Directory is a good choice
- If you need to support AES today or need to extend Kerberos to support OTPs, alternate pre-auth mechs, or other site local behaviors, then either Heimdal or MIT
  - MIT has traditionally been focused on OEM requirements. This has produced a focused Kerberos product that is slow to adopt custom functionality
  - Heimdal is more willing to include non-Kerberos functionality and accept patches. Heimdal is easier to use but MIT is more likely to be the version shipped in your OS.

secure**e**ndpoints

General Migration Info (from kaserver)

- You can have multiple AFS keys in the AFS server KeyFile; this allows you to test out different KDCs on the same AFS server without impacting anything.

- Important: make sure that the different AFS keys have different kvnos!

- Once you've verified that it works (test with aklog), you can slowly transition users over, or switch everything at once and provide kaserver compatibility on the KDC.

- One very common problem: make sure that AFS service key is single-DES!

- If you want to support legacy AFS authentication later, enable AFS-salted keys.

secureendpoints

Migrating To MIT Kerberos

- Build Ken Hornstein's afsk5db converter to convert the database to V5 format.
  - Not part of OpenAFS because it may require access to private functions of MIT Kerberos.
- Use asetkey to store the AFS service key into a KeyFile.
- Run fakeka to support kaserver services when you switch (may need to run ka-forwarder if you have kaservers not on same machine as KDC).
- MIT has some bugs related to AFS salted keys. Some principals may require password changes after the migration

secureendpoints

Migrating to Heimdal

- Use hprop to convert AFS database to V5 format.

- Use ktutil to write AFS KeyFile.

- Turn on flag to support kaserver on KDC during the transition.

secureendpoints

Strategies for Dealing with Microsoft Active Directory for AFS

- You can do cross-realm from a Windows domain to a Unix-based domain (use Microsoft Cross-Realm Wizard).
- You can store the AFS service key in a keytab
  - You can use the raw krb5 tickets as tokens, or
  - You can run a krb524d on a Unix machine
- When using raw krb5 tickets, the Kerberos ticket size exceeds the limit in the cache manager, due to the group membership information.
  - Microsoft has a patch to 2003 Server that permits disabling the PAC for specific accounts
  - Doug Engert at Argonne developed a patch to krb524d to strip out the group information from the ticket.

secureendpoints

Kerberos Integration: General Guidelines for Kerberos Domination

- Why is this important? Because like AFS, the more you use it, the more useful it will become.

- Authentication is (at most sites) a political minefield. It's impossible to give guidance to installataions to get them to adopt Kerberos more widely, because each site is different. Here are some ideas that have helped other sites in the past.

secureendpoints

Kerberos Domination Guidance

...

- Try to use Kerberos as single password storage system.
  - Yes, even if it means typing Kerberos passwords into web forms.
  - Single Password first, then Single Sign-On
- Enable as many services as possible with Kerberos authentication.
  - You don't have to require only Kerberos authentication, but if you give users the option, they may want to switch for convenience sake (password expiration can help you here).
  - You can also use this to tout the advantages of single sign-on.
- Use translation services when possible.
  - NRL developed application proxies for POP - they speak regular POP out one side, and GSSAPI-authenticated POP out the other. This same technique could be used for IMAP, SMTP, and other protocols.

secureendpoints

Last Resort Guidance

- "Don't you want to be cool?"

- Supported by diverse vendors as Sun, Apple, Microsoft, Red Hat, Novell, ...

- "No one ever got fired for buying Microsoft"

- Central component of Windows Active Directory.
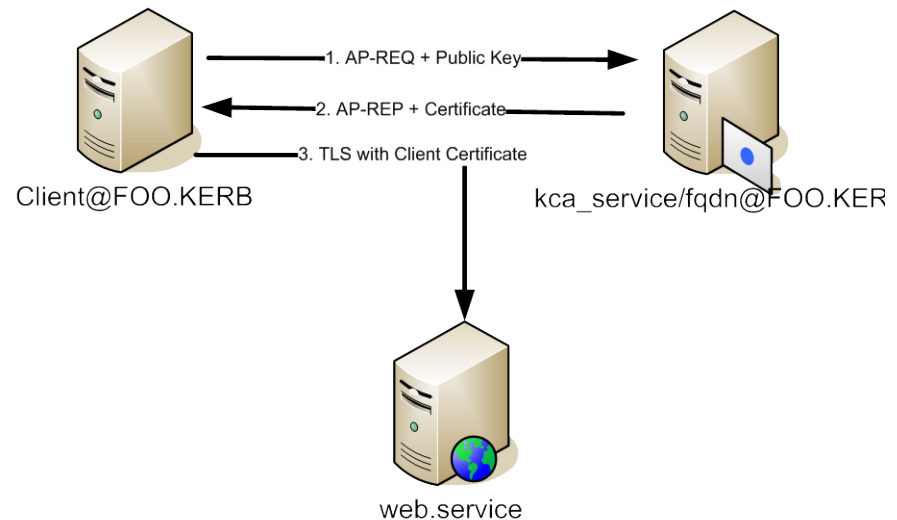
- Note that which one you pick depends on your site!

secureendpoints

Additional Topics

secureendpoints

Kerberos and the Web

- HTTP Negotiate
- Kerberized Certificate Authorities
- Web Sign-On Systems
  - CoSign
  - WebAuth
  - …

secureendpoints

KX.509 / KCA (or How to authenticate using a Kerberos identity to a PKI service)

- KX509/KCA utilizes a Kerberos Application Service authentication to communicate with a special certificate service that issues client certificates with the same identity and valid lifetime as the Kerberos Service ticket.

- The resulting certificate is placed in the certificate store for use by applications such as web browsers.

Client@FOO.KERB

1. AP-REQ + Public Key

2. AP-REP + Certificate

3. TLS with Client Certificate

kca_service/fqdn@FOO.KER

web.service

secure**e**ndpoints

Web authentication

- Started out with POSTing passwords
- Instead of doing this for every page, cookies.
- But doing this sitewide was poor (passwords) or insecure (cookies)

secure**e**ndpoints

Web Single Sign-On

- If you already have Kerberos, leverage it.
  - Sidecar
  - Webauth v1
  - Minotaur

secureendpoints

Try, try again

- Solutions involving browser plugins have portability issues
- Solutions involving add-ons fell to NAT issues, security issues authenticating the local browser connection, or cases where software can't be installed

Web Double Sign-On

- If you can't use Kerberos directly, leverage its model

- Accept passwords in the server and pass back an identifier

- Possibly store acquired Kerberos credentials for later use

- Of course you can't use the credentials you already have

secure**e**ndpoints

Many ways to get there

- Washington pubcookie
- Stanford webauth (v3)
- Yale CAS
- Michigan CoSign

secureendpoints

Really all the same

- Everyone's WebSSO uses the Kerberos metaphors of a KDC granting a granting token, and honoring it when providing authentication tokens for services

- But everyone's WebSSO does it a little differently

secure**e**ndpoints

Forward to the past

- GSSAPI SPNEGO in the browser (Safari, Firefox, IE)
- Leverage the tickets you have
- But this is the least widely deployed solution

secure**e**ndpoints

To the logical extreme

- If you're going to mimic Kerberos, might as well do it fully

- Cross realm Kerberos' equivalent: WebSSO federation

secure**e**ndpoints

Differing goals

- Shibboleth is a common enterprise solution

- OpenID is in some ways an upstart

- Not really the same target audience

secure**e**ndpoints

Shibboleth

- All about control
  - What are you releasing
  - Who are you releasing it to

secureendpoints

OpenID

- Simplicity is key
  - The only question that matters is "does this person control this URL"
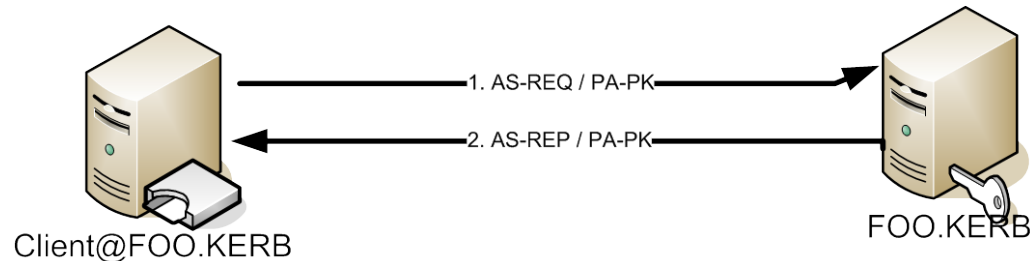 - Good as long as you trust browser redirects

secure**e**ndpoints

The ultimate answer

- None yet

- But then Kerberos hasn't proven itself the only answer in its field yet either, so why would its web-centric twin?

secureendpoints

Pre-authentication Methods

- PK-INIT
- CryptoCard
- SecureID

secureendpoints

PK-INIT: How does it work?

- PK-INIT is implemented as a Kerberos Pre-authentication mechanism
- If the client's request adheres to KDC policy and can be validated by its trusted CAs, then the reply is encrypted either with
  - A key generated by a DH key exchange and signed using the KDC's signature key, or
  - A symmetric encryption key, signed using the KDC's signature key, and then encrypted with the client's public key.
- Any required keying material is returned to the client as part of the AS-REP's PA-PK data.
- If the client can validate the KDC's signature, obtain the encryption key, and decrypt the reply, then it has successfully obtained an Initial Ticket Granting Ticket.

secure**e**ndpoints

PK-INIT: Not Vaporware

- Draft -9 deployed by Microsoft in Windows 2000 and above
- The Proposed Standard (RFC 4556) is being deployed today:
  - Microsoft Vista
  - Heimdal Kerberos
- Future deployments:
  - MIT Kerberos 1.7 and the operating systems that distribute it

secure**e**ndpoints

PK-INIT: Opening the doors to alternative enrollment models

- Trusted CA issued certificate can be enrolled with multiple realms

- Raw public key pairs can be used instead of certs allowing SSH style enrollments

- A single smart card can be enrolled with multiple realms allowing the acquisition of TGTs for multiple service providers

secureendpoints

Single Sign-On General Issues

- How are credentials obtained?
- Where are they placed?
- Can the applications access them?
- What encryption types are used?
- Multiple Kerberos implementations on the same system
  - Operating System Vendor's
  - Third party
    - Heimdal, MIT, Quest, CyberSafe, …
  - Java
    - Sun, IBM, …

secureendpoints

Single Sign-On Credential Cache Types

- FILE
  - Various file formats have been implemented over the years. Not all Kerberos products support all types
- API - MIT/UMich Credential Cache API
  - Per session or Per machine service
  - Implemented on MacOS X and Windows
- MEMORY – per process
- MSLSA – Microsoft Windows session cache
- KEYRING – Linux keyring
- PIPE – Per session cache only accessible to inherited processes

secureendpoints

Single Sign-On Encryption Types

- Many implementations of Kerberos and applications have a very common bug:
  - As part of logging/debugging, the enc-type is used as an index to a table of enc-type names. If the enc-type value is not in the table or is larger than the table size, the application dumps core or aborts the transaction.
  - This even happens when the unknown enc-type is for the key used to encrypt the service portion of the ticket.

secure**e**ndpoints

Single Sign-On and UNIX: PAM

- What is PAM?
- The PAM Groups
- PAM for Login
- PAM for Screen Savers
- Kerberos PAM Modules
- Linux PAM Examples
- Solaris PAM Example
- Special Configurations

secureendpoints

PAM:
What is PAM?

- Pluggable Authentication Modules
- Abstracts the user authentication and session setup process
- Only does authentication and simple authorization
- Developed originally on Solaris
- Enhanced but mostly compatible version on Linux
- Now used by many UNIXes, but implementation varies

secureendpoints

PAM:
The PAM Groups

- PAM divides the login process into groups
  - auth: Prompts for and verifies password
  - account: Simple authorization decisions (only for login)
  - session: Prepares for an interactive session
  - password: Handles authentication token changes
- setcred, the odd step-child
- setcred vs. open session: who knows? who cares?

secure**e**ndpoints

PAM:
PAM for Login

- auth group prompts for password, does basic authentication
  - Store the credentials in a separate temporary cache
  - Don't chown credential cache until setcred
- account group does basic authorization
- setcred stores credentials and adds supplemental groups
- session group creates a login session
- When the user logs out, session group closes the login session

secureendpoints

PAM:
PAM for Screen Savers

- auth group prompts for password, does basic authentication

- account group could do authorization, but frequently ignored

- setcred to refresh credentials (REINITIALIZE/REFRESH)

- session group not called

- Bad screen savers don't call setcred and thereby lose

secureendpoints

PAM:
Kerberos PAM Modules

- Sourceforge pam krb5
- Red Hat pam krb5
- My pam-krb5, based on Frank Cusack's module
- Solaris native pam krb5

secure**e**ndpoints

PAM: Configuration

- Debian: /etc/pam.d/common-*
- Red Hat: /etc/pam.d/system-auth
- Solaris: /etc/pam.conf
- Whether to use a Kerberos PAM module for password changes

secure**e**ndpoints

PAM: Linux PAM Example

```
auth       sufficient pam_krb5.so
auth       required   pam_unix.so try_first_pass
account    required   pam_krb5.so
account    required   pam_unix.so
session    optional   pam_krb5.so
session    required   pam_unix.so
password   sufficient pam_krb5.so minimum_uid=1000
password   required   pam_unix.so obscure min=6 md5
```

secureendpoints

PAM:
Solaris PAM Example

login auth sufficient /usr/local/lib/security/pam_krb5.so
        minimum_uid=100
login auth required /usr/lib/security/pam_unix_auth.so.1
        use_first_pass
login account required /usr/local/lib/security/pam_krb5.so
        minimum_uid=100
login account required /usr/lib/security/pam_unix_account.so.1
login session required /usr/local/lib/security/pam_krb5.so
        retain_after_close minimum_uid=100
login session required /usr/lib/security/pam_unix_session.so.1

(no wrapping)

secure**e**ndpoints

PAM:
Special Configurations

- minimum uid or ignore root
- MIT Kerberos needs master kdc setting for password expiry
- SSH and ticket cache initialization
- SSH and ChallengeResponseAuthentication
- search k5login and shared role accounts
- PKINIT
- AFS — see talk on Friday

secureendpoints

Single Sign-On and Microsoft Windows

- Goals:
  - User logs into Windows and enters password once
  - All applications can use the same TGT
  - Credentials are automatically renewed

secure**e**ndpoints

Single Sign-On
KFW Integrated Logon

- MIT KFW installs a Network Provider Credential Manager
  - Obtains a TGT using the username, password, default realm (krb5.ini)
  - Pushes TGT into the logon session ccache API:<user>@<REALM>
- Only works with interactive logons
  - If logon scripts are not executed, the ccache will not be created
- Can be used for non-domain accounts or domain accounts with passwords that are synchronized with accounts in alternate realms

secureendpoints

Network Identity Manager
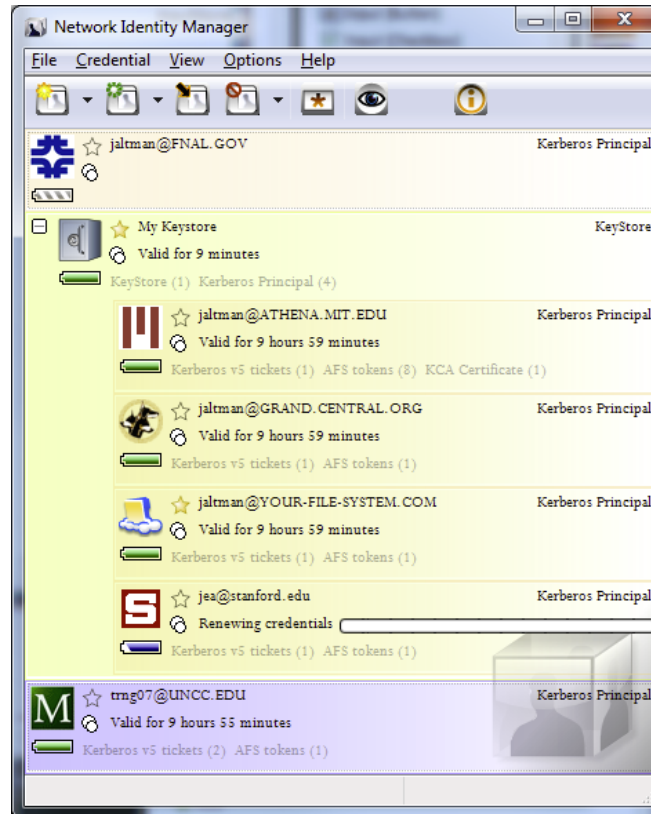
Network Identity Manager

- Multiple identity credentials manager
  - One identity at a time can be "default"
  - Applications that are identity aware can access non-default identities
  - Manages API, MSLSA and FILE ccaches
- Credentials of one type can be used to obtain credentials of another type
  - Kerberos v5 -> Kerberos v4
  - Kerberos v5 -> AFS
  - Kerberos v5 -> 524 -> AFS
  - Kerberos v5 -> Kerberos v4 -> AFS
  - Kerberos v5 -> X.509 certificate (KCA/kx509)
- Additional credential types will be supported in the future
  - X.509 -> Kerberos v5 (PKINIT)
- Automated credential renewal
- For detailed documentation see http://www.secure-endpoints.com/#Network%20Identity%20Manager

secureendpoints

Network Identity Manager Credential Acquisition
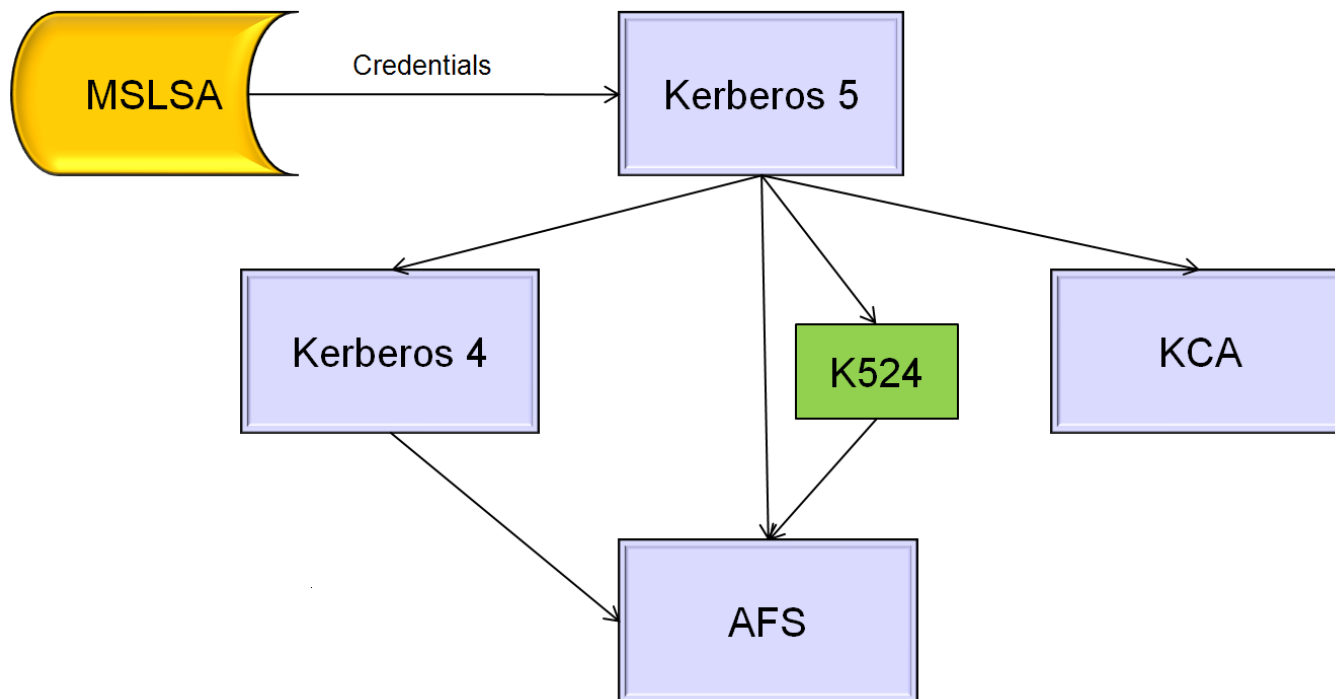
Network Identity Manager and Microsoft Vista

- Microsoft Vista provides the necessary functionality for NIM to push identities into the MSLSA ccache

- This will permit NIM to be used to synchronize the user selected default identity for both MIT API applications and SSPI applications

secure**e**ndpoints

Single Sign-On Challenges

- Many applications come with their own Kerberos implementations that do not integrate with the MSLSA or KFW API ccaches
- Examples include:
  - Cygwin applications
    - MIT Kerberos v5 built without support for Winsock, MSLSA and API ccache, and registry configuration
  - Attachmate
    - Can import configuration information from KFW or Windows domain logon. Does not re-use existing TGT.
  - Hilgraeve
    - Ships with a custom build of KFW called "Connectivity Kerberos" use a real MIT KFW release instead

secure**e**ndpoints

Single Sign-On and MacOS X

- MacOS X has the potential to provide the best single sign-on experience
- Java, Kerberos, and applications are all provided by Apple
- No multiple Kerberos stacks to deal with
- Unfortunately, its not quite there yet.
  - See Henry Hotz' talk from BPW 2007 http://www.pmw.org/afsbpw06/talks/hotz.html

secure**e**ndpoints

Single Sign-On and MacOS X Kerberos.app
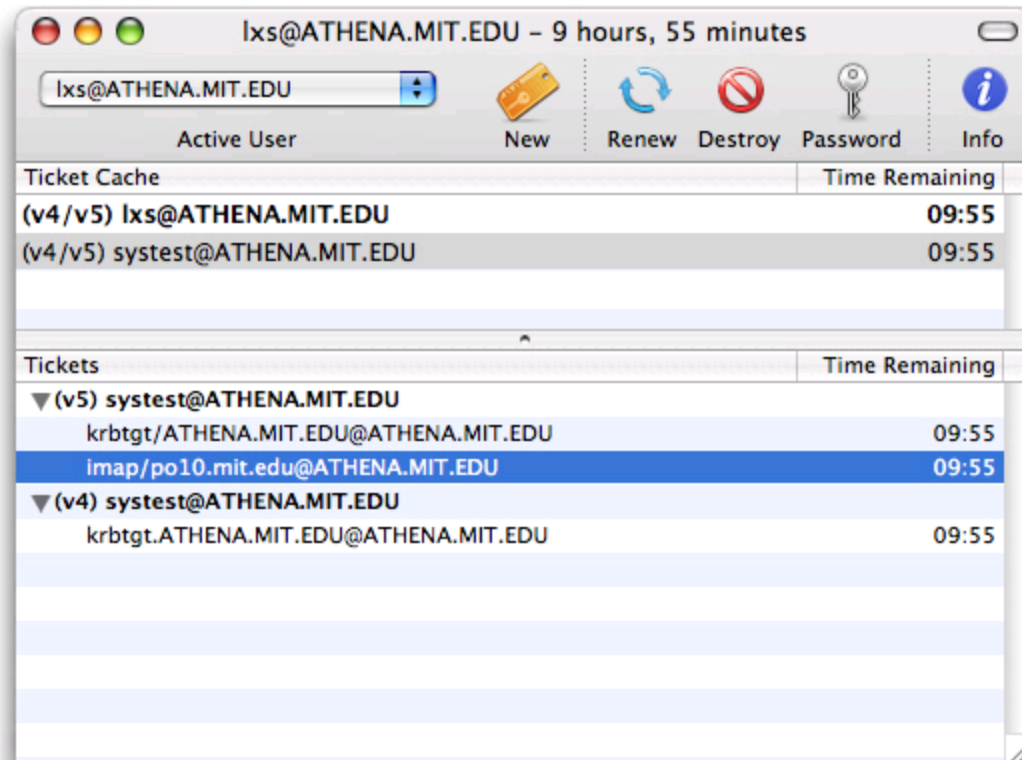
Single Sign-On and MacOS X Kerberos.app

- Multiple identity credentials manager
  - One identity at a time can be "default" or "active"
  - Applications that are identity aware can access non-default identities
  - Manages AP ccaches
- No support for non-Kerberos credential types
  - 10.4 supports v5 and v4.
  - 10.5 only supports v5
- Excellent overview at http://web.mit.edu/macdev/KfM/KerberosClients/KerberosApp/Documentation/using-osx.html

secureendpoints

Kerberos and Sun Java GSS

- Java 6.0 provides the most functional implementation of Kerberos/GSS
- Reads Kerberos profile (krb5.conf)
- Enc-types:
  - AES-128 (AES-256 with JCE Crypto Policy)
  - RC4-HMAC
  - 3DES-CBC-SHA1
  - DES-CBC-CRC, DES-CBC-MD5

- GSS SPNEGO
- Pre-authentication support for alternate salts, enc-types, …
- Native GSS-API library on Solaris and Linux. (KFW and SSPI support on Windows soon to be announced.)
- IPv6 support (5.0)
- TCP support (4.2)
- TGT renewals (5.0)

secure**e**ndpoints

OpenSSH:

- OpenSSH has some Kerberos support, using the GSSAPI, in its core distribution
- Support for GSSAPI Key Exchange, and for cascading credentials, is available in patches from
  - http://www.sxw.org.uk/computing/patches/openssh.html

secure**e**ndpoints

OpenSSH: Why Key Exchange?

- The bundled support authenticates the user to the server

- Server to user authentication is done using the normal SSH public key

- The public key is verified by checking the client's known_hosts file

- Maintaining known_hosts files is a key management headache for large sites

secure**e**ndpoints

OpenSSH: GSSAPI Key Exchange

- GSSAPI key exchange uses Kerberos for both server to user, and user to server authentication
- SSH servers no longer require keys
- All key management is performed by Kerberos
- Significantly reduces the key management overhead
- And allows other things to happen, too ...

secure**e**ndpoints

OpenSSH: Cascading Credentials

- Power users are typically logged into, and through, many machines in a single session

- Delegation lets them propagate out tickets when they first log in

- But what happens when they all expire?

secure**e**ndpoints

OpenSSH: Cascading Credentials II

- SSH allows key exchange to be performed multiple times in the life of a connection

- GSSAPI key exchange can delegate tickets as a side-effect of key exchange

- If the client rekeys the connection every time its Kerberos credentials are updated ...

- ... every host gets the new credentials

secure**e**ndpoints

OpenSSH:
When things go wrong

- Virtually all error output gets supressed, unless you're in debug mode

- Start client with -v -v -v, and server with -d -d -d

- Watch out for GSSAPI errors, bearing in mind that they can span multiple lines

secureendpoints

OpenSSH: DNS round robin hosts

- DNS round robin pointing ssh at
  - ssh-A
  - ssh-B
  - ssh-C
- Kerberos canonicalises name, and gets *ssh-A*
- OpenSSH then looks up name, and gets *ssh-B*
- Using *ssh-A*'s key to talk to *ssh-B* doesn't work
- Solution: Set the GSSAPITrustDNS option to on.

secureendpoints

OpenSSH:
Multi-homed hosts

- On a host which has multiple different canonical names, only the server principal containing the machines hostname is accepted.

- Solution: Place keys for host principals for all of the machines canonical names in the keytab, and set GSSAPIStrictAcceptorCheck to off

secureendpoints

Russ' Very Useful Tools

secureendpoints

Authenticated Daemons

- Need to obtain Kerberos credentials automatically
- Need to refresh credentials automatically
- Need to refresh tokens automatically
- Must not interfere with each other
- Must not be interfered with by users (including root)
- Application can't do any of the work

secure**e**ndpoints

Why kstart?

- K4 *kinit* completely deficient
- K5 *kinit* much better, but *kstart* had evolved
- Runs as daemon to maintain credentials
- Forks token-getting program as needed
- Checks ticket expiration (-H)
- Can run command with credentials, PAG
- *k4start* and *k5start*
- Similarities to Heimdal *kcm*

secureendpoints

remctl

- GSS-API-based RPC layer (command and response)
- Simplest thing that could possibly work
- Check an ACL and run a command, nothing more
- Simpler than web services, no HTTP required, Kerberos auth
- Documented protocol
- No output size limits, streaming output
- Client library with simple API
- Bindings for Perl, PHP, and Python
- Independent Java implementation

- Extensive test suite

secure**e**ndpoints

Wallet

- Manages any type of secure data
  - Keytabs and simple files implemented
  - WebAuth keyrings and X.509 CA planned
  - Extensible system for adding new types
- Rich ACL mechanism
  - Fine-grained access control for operations
  - Simple identity and NetDB implemented
  - Extensible system for adding more verifiers
  - PTS, LDAP, and nested groups planned
- Built on remctl, but server and client can be replaced with any other authenticated RPC layer (SOAP, REST, etc.) without major server changes

secureendpoints

MIT Kerberos v5 API Overview

MIT API:
Client Side

- krb5_init_context()
  - Initialize Kerberos library
- krb5_sname_to_principal()
  - Create server principal name
- krb5_cc_default()
  - Access Kerberos credential cache
- krb5_cc_get_principal()
  - Get client principal name
- krb5_get_credentials()
  - Get service ticket and session key (perform TGS_REQ if necessary).
- krb5_mk_req_extended()
  - Generate AP_REQ.
- krb5_rd_rep()
  - Process AP_REP.
- krb5_sendauth()
  - Can be used instead of krb5_get_credentials(), krb5_mk_req_ext(), and krb5_rd_rep().

secureendpoints

MIT API:
Server Side

- krb5_init_context()
- krb5_rd_req()
- Process AP_REQ.
- krb5_mk_rep()
- Generate AP_REP.
- Client principal name ends up in ticket->enc_part2->client

secure**e**ndpoints

MIT API: Encryption

- krb5_auth_con_getlocalsubkey()
- krb5_auth_con_getlocalsubkey()
  - Extract subkey to use for encryption.
- krb5_c_encrypt()
- krb5_c_decrypt()
  - Perform encryption/decryption.
- Also need to set up initial vectors and/or key usage numbers. More fun: encrypted length is longer than plaintext length.

secureendpoints

MIT API: Kinit/Password Verification

- krb5_init_context()
- krb5_cc_default()
- krb5_parse_name()
  - Form client principal name.
- krb5_get_init_creds_password()
  - AS_REQ/AS_REP exchange.
- krb5_cc_initialize()
- krb5_cc_store_cred()
  - Store new credentials.
- krb5_verify_init_creds()

secureendpoints

Q&A

secureendpoints