Integrated, Kerberized Login on MacOS X

Henry B. Hotz
Jet Propulsion Laboratory

Overview

- Context for this information
- MacOS X login process and available hooks
- Authorization Services configuration
- Authorization Services plug-in’s
- Kerberos plug-in’s
- Other bugs and recommendations

What are We Trying to Do?

- We want to get or refresh our Kerberos tickets transparently whenever we type our password to identify ourself to the machine.

1: Kerberos is authoritative

- All authorization uses Kerberos (if applicable for user)
- Must verify KDC isn't spoofed

2: Kerberos is "extra"

- All machine authorization uses another authority
- Attempt to get tgt when possible for network services

MacOS X Login Process

- Authorization Services
  - Called by loginwindow, screen saver and fast user switching
  - Calls Directory Services
- Login Hook
- Login Items (System Preferences)

Directory Services Hooks

- If Directory Services uses Kerberos to check passwords, we're done, right?
- AuthenticationAuthority attribute is defined for Directory Services
    - `;Kerberosv5;`
- Independently implemented (?) by every plug-in
    - Kerberos only implemented by LDAPv3 plug-in
    - AD plug-in "fakes" it
    - NetInfo (local) plug-in does ***not*** do it

Configuring Authorization Services

- Configuration is in /etc/authorization
  - Editable text file, but format changes with OS version
  - API can be used for changes starting in 10.2
- Consists of a list of "rights" (like `system.login.console`) that are checked by appropriate parts of the system, and "rules" that may be referenced by the rights.
  - Rights or rules can list required mechanisms to execute (a little like pam modules)
    - Mechanisms may be implemented as plug-in's.
    - All mechanisms *must* return success (like pam required).

Authorization Services Key Meanings

- Rights are evaluated according to their class
  - <none> Same as "rule" (but with some restrictions)
  - allow
  - deny
  - user (next slide)
  - rule (slide after next)
  - evaluate mechanisms
    - array of strings of the form `[plugin:]mechanism[,privileged]`
    - If "plugin" is given then the mechanism is in the bundle in `/System/Library/CoreServices/SecurityAgentPlugins`
    - "privileged" makes it uid 0, but doesn't change the security context.
    - Can also have "tries" and "shared" specified (see next slide).

Authorization Services Key Meanings, Continued

- user
  - Can specify the following (defaults in paren's)
    - authenticate-user (true)
    - group (don't care)
    - allow-root (false)
    - session-owner (false)
    - mechanisms (see below)
    - tries (3)
    - shared (false, see TN1277)
    - timeout (infinity)
  - If "mechanisms" is missing then the mechanisms from the "authenticate" rule are used.

Authorization Services Key Meanings, Concluded

- Rules are evaluated recursively.
- Evaluation stops when the result is known
- Specific properties:
  - k-of-n
    - if not present then all listed rules must be satisfied
  - rule
    - the array of strings (or single string) are the names of other rulse that must be satisfied.

Relevant Right Config's

- `system.login.console` (right)

```
<key>system.login.console</key>
<dict>
        <key>class</key>
        <string>evaluate-mechanisms</string>
        <key>mechanisms</key>
        <array>
                <string>builtin:auto-login,privileged</string>
                <string>loginwindow_builtin:login</string>
                <string>builtin:reset-password,privileged</string>
                <string>authinternal</string>
                <string>builtin:getuserinfo,privileged</string>
                <string>builtin:sso,privileged</string>
                <string>HomeDirMechanism:login,privileged</string>
                <string>HomeDirMechanism:status</string>
                <string>MCXMechanism:login</string>
                <string>loginwindow_builtin:success</string>
                <string>loginwindow_builtin:done</string>
        </array>
</dict>
```

Relevant Right Config's, Continued

- system.login.done (right)

```
<key>system.login.done</key>
<dict>
       <key>class</key>
       <string>evaluate-mechanisms</string>
       <key>mechanisms</key>
       <array/>
</dict>
```

- system.login.screensaver (right)

```
<key>system.login.screensaver</key>
<dict>
       <key>class</key>
       <string>rule</string>
       <key>rule</key>
       <string>authenticate-session-owner-or-admin</string>
</dict>
```

Relevant Right Config's, Concluded.

- authenticate-session-owner-or-admin (rule)

```
<key>authenticate-session-owner-or-admin</key>
<dict>
    <key>allow-root</key>
    <false/>
    <key>class</key>
    <string>user</string>
    <key>group</key>
    <string>admin</string>
    <key>session-owner</key>
    <true/>
    <key>shared</key>
    <false/>
</dict>
```

- authenticate (rule)

```
<key>authenticate</key>
<dict>
    <key>class</key>
    <string>evaluate-mechanisms</string>
    <key>mechanisms</key>
    <array>
        <string>builtin:authenticate</string>
        <string>authinternal</string>
    </array>
</dict>
```

Authorization Services Plug-Ins

- `authinternal` is the Authorization Services mechanism that does a Directory Services check password call.
  - Directory Services searches for the user record with the given username.
  - Asks that record's parent node to authenticate it with the given password.

Kerberos A. S. Plug-Ins

builtin: krb5authenticate	kerberos: authenticate	Tries password with Kerberos and verifies against the "`host`" principal in `/etc/krb5.keytab`. If fails, try Directory Services before returning an actual failure.
builtin: krb5authnoverify	kerberos: authenticate-noverify	Same as above, but skip the keytab verification.
builtin:sso (builtin:krb5auth)	<no equiv.>	Same as `login`, but only if the "`kerberos-principal`" context value is set.
builtin: krb5login	kerberos: login	Try Kerberos with password and save `tgt` if acquired. Always return success. (Example needs patch.)
<no equiv.>	kerberos: none	Do nothing. Always return success (for testing).

Fast User Switching

- *Don't do it!*
- I know I don't know what all the bugs are, but. . .
  - Switching to a new user calls AS twice, once in the "from" user context and once in the system context.
    - An existing security context overrides the `seteuid()` back door provided for `KLStoreNewInitialTicketCredentials()`.
  - Switching between users, Kerberos tickets are saved to the "from" user, not the "to" user. (AS only called once.)
    - Bug 4509062 for OSX 10.4, Bug 4395796 for Leopard
  - The `FUSDataKey` authorization hint exists when in the "from" user context (in 10.4.6 at least).

Service Tickets for Ancillary Services (Like AFS)

- Use the loginLogout plug-in interface

```
[libdefaults]
    login_logout_notification = plug-in-name
```

  - Plug-in bundle goes in

    `/Library/Kerberos Plug-Ins/plug-in-name.loginLogout`

  - API documented at

    http://www.opensource.apple.com/darwinsource/10.3/Kerberos-47/KerberosFramework/KerberosLogin/Documentation/LoginLogoutNotification.html

  - Don't call closelog() inside a plug-in.

- Called (twice) every time a tgt is (successfully) acquired, renewed, or destroyed.

  - No need to modify `/etc/authorization`

Recommendations

- In theory it should be possible to do integrated login with MacOS X 10.4. If you want to try. . .
  - In /etc/authorization
    - Add kerberos:login to system.login.console right
    - Add mechanism list to `authenticate-session-owner-or-admin` rule
  - Install Ragnar Sundblad's Kerberos/AFS plug-in
    - See References, last slide
  - Install kerberos:login example plug-in
    - Use patch on next slide
- builtin:krb5login doesn't work for me in 10.4.5

Patch for kerberos Plug-In

```
*** authplugin.c.orig   Sat Mar 25 14:33:02 2006
--- authplugin.c        Sat Mar 25 14:37:08 2006
***************
*** 58,64 ****
        return NULL;
  }

! static bool invoke(MechanismRef *mechanism, int mode)
  {
        bool verifyKDC = (mode == authenticate); // only in this
mode require kdc to be authenticated
        bool successfulAuthentication = false;
--- 58,64 ----
        return NULL;
  }

! static bool invoke(MechanismRef *mechanism, KerberosMode mode)
  {
        bool verifyKDC = (mode == authenticate); // only in this
mode require kdc to be authenticated
        bool successfulAuthentication = false;
***************
*** 181,186 ****
--- 181,190 ----
                case kMechKerberosAuthenticateNoVerify:
                        result = invoke(inMechanism, authnoverify);
                        break;
+               case kMechKerberosLogin:
+                       invoke(inMechanism, login);
+                       result = kAuthorizationResultAllow;
+                       break;
                default:
                        return errAuthorizationInternal;
      }
```

References

- Apple Developer Technical Support
  - Many thanks.
- Documentation

  Authorization Plug-in Reference

  Authorization Services C Reference

  Apple Open Directory (multiple documents)
- Tech Notes and Q&A's

  Security Credentials, QA1277

  Authorization for Everyone, TN2095

  `/etc/authorization` File Format (when issued)

References, Continued. . .

- Example Code

  `CryptNoMore Plugin`

  - How `authinternal` uses Directory Services

  `NullAuthPlugin`

  - Includes list of most authorization hints (except `FUSDataKey`).

  Directory Services LDAPv3 plug-in (real code from Darwin)

  - How Open Directory does Kerberos authentication and uses the `AuthorizationAuthority` attribute.
  - Actual, users' stored `tgt` is acquired by Authorization Services' `builtin:sso` plug-in, not by this one.

References, Concluded.

- Example Code (actually used)

  `afslog.loginLogout`

  - Available from `/afs/nada.kth.se/home/staff/ragge/out/test/`
  - Get's AFS tokens for either Arla or OpenAFS clients whenever Kerberos gets `tgt`'s.

  `kerberosAuthPlugin`

  - Available from Apple
  - Shows most of what the builtin kerberos plug-in's do.
  - README file includes sample code for modifying `/etc/authorization` on 10.2 and up.