Simple Mass Market 2-Factor Authentication Using Java™ Technologies

Alex Karasulu, Apache Foundation
Michael Yuan, PhD., JBoss Inc
The Safehaus Triplesec project

http://triplesec.safehaus.org/

TS-9862

Keep It Simple; Keep It Safe

Develop secure thin and rich client applications with easy-to-use 2-factor authentication for mass market users

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

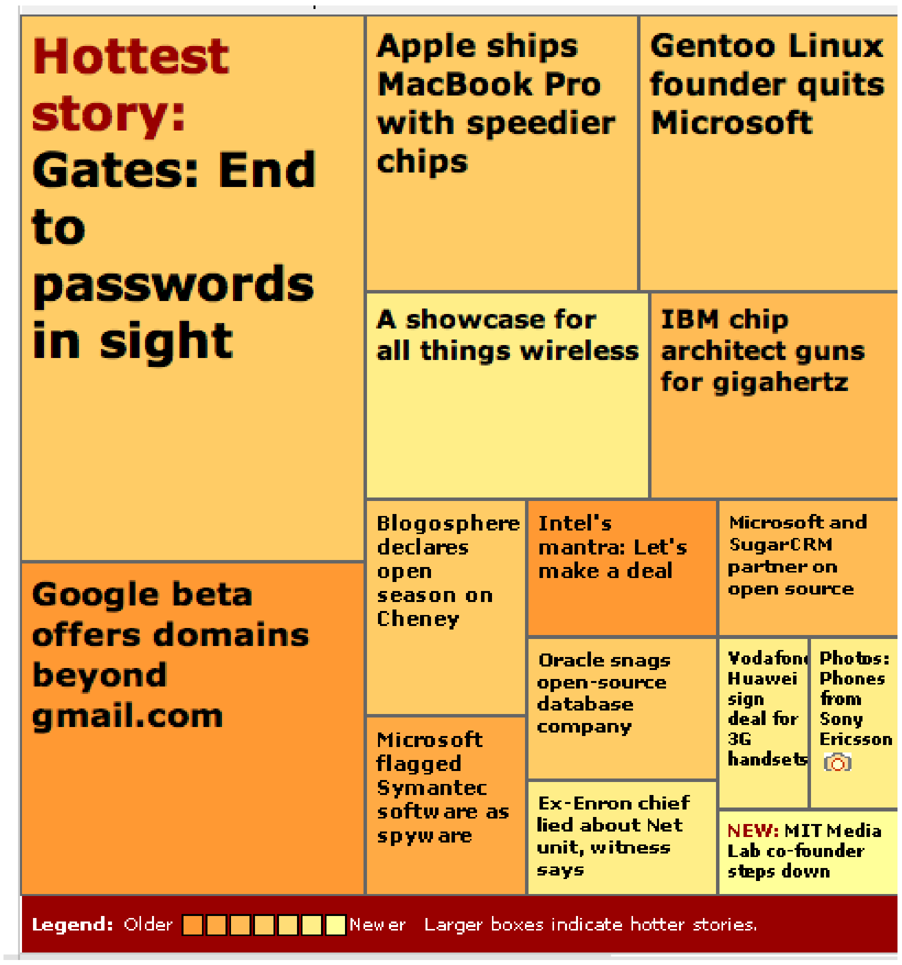
Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

Passwords Are Sooo 20th Century

- Hard to remember
- Easy to guess/steal
- Gets much worse when you have multiple accounts



Hottest story: Gates: End to passwords in sight

Apple ships MacBook Pro with speedier chips

Gentoo Linux founder quits Microsoft

A showcase for all things wireless

IBM chip architect guns for gigahertz

Google beta offers domains beyond gmail.com

Blogosphere declares open season on Cheney

Intel's mantra: Let's make a deal

Microsoft and SugarCRM partner on open source

Oracle snags open-source database company

Vodafone Huawei sign deal for 3G handsets

Photos: Phones from Sony Ericsson

Microsoft flagged Symantec software as spyware

Ex-Enron chief lied about Net unit, witness says

NEW: MIT Media Lab co-founder steps down

Legend: Older ■■■■■■■ Newer Larger boxes indicate hotter stories.

The 3 Factors

- What you know
  - Passwords
- What you have
  - A card, device or token
- Who you are
  - Fingerprint
  - Eye scan? DNA? Implanted RFID?
- Most secure sites uses two factors

In the Real World

- Password + hardware token
  - Corporate VPNs
  - Online brokerage houses
- Bio-metrics
  - The government
  - Data center
- Is online banking next?
  - Government regulation already mandates 2-factor authentication for banks in 2007

One Time Password (OTP)

- Password generated by a hardware token
- Changes with each use
- Algorithms:
  - HMAC (Hashed Message Authentication Code)
  - S/Key (MD4/MD5)
  - Time Based
- HOTP (HMAC-based One-Time Password Algorithm) an IETF standard

HOTP Algorithm (RFC 4226)

- Shared secret
- Counter
- Throttling parameter
- Look-ahead parameter: re-sync
- Bi-directional authentication
- Low resources utilization
- Ideal for self service
- Security considerations

Why Not a Mass Market Solution

- User resistance
  - Extra token device to carry
  - Multiple token devices are inconvenient
  - Lost/stolen devices are hard to replace
- The token device is expensive
- Device provisioning and management is hard
- Requires invasive changes to existing security infrastructure

Propose a Solution

- Use the mobile phone to generate one time passwords
  - Pervasive
  - No new hardware to buy
- Simple provisioning process
  - Safely transmit the mobile client over the internet
- Use standard protocols for authentication
- Provide simple integration with existing Java™ Platform, Enterprise Edition (Java EE) application servers, Java Platform, Standard Edition (Java SE) rich applications, and native operating systems
- Integrated identity management

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

Triplesec “Strong Identity” Server

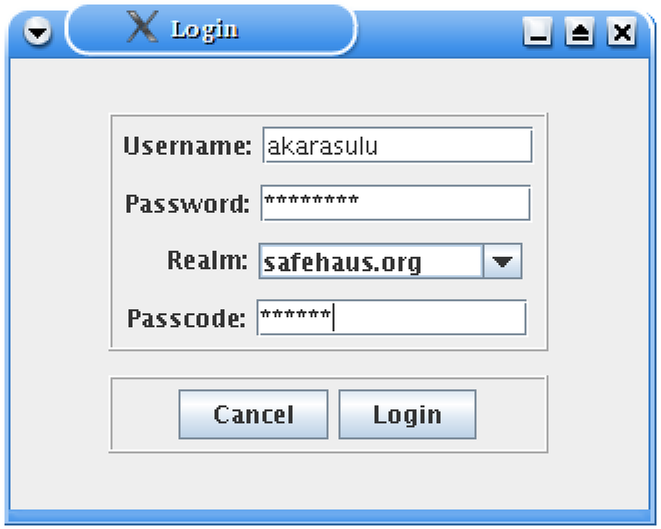
- Completely Free and Open Source from Safehaus
- Fully featured Identity Management
  - 2-Factor authentication
  - Authorization (Role-Based Access Control, RBAC)
  - Auditing
  - SSO (Single-Sign On)
- Java Platform, Micro Edition (Java ME) for mobile token
- Apache Directory Server to support standard protocols:
  - Kerberos 5 and LDAPv3
- Simple integration with Java EE and SE platform-based applications

Hauskeys Mobile Token

- Java ME platform-based application
  - MIDP 1.0
  - Footprint 33k
  - Runs on low end phones like Nokia S40
- Connectionless OTP generation
  - No need to have mobile data service subscription!
  - HMAC-based OTP
  - HOTP RFC 4226
- OATH (Open AuTHentication) member and reference implementation

Authentication

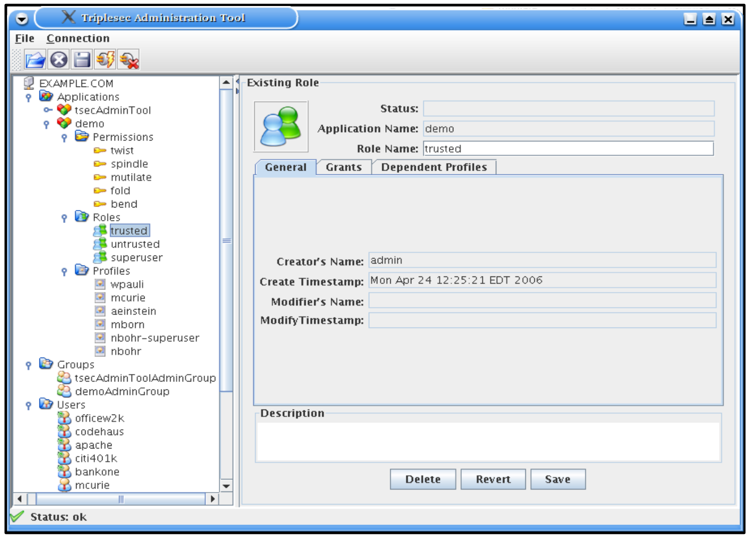
- Additional fields
  - OTP (mandatory)
  - Realm (optional, application dependent)
- Authentication uses
  - LDAP v3
  - Kerberos 5



Username: akarasulu
Password: *********
Realm: safehaus.org
Passcode: *******

Authorization

- Authorization Policy Store
  - Applications
    - Permissions
    - Roles
    - Profiles
  - Users
  - Groups
- Guardian API for Policy Access
  - Only the owning application can access its policy
  - Control application operations with fine granularity



```
EXAMPLE.COM
  Applications
    tsecAdminTool
    demo
      Permissions
        twist
        spindle
        mutilate
        fold
        bend
      Roles
        trusted
        untrusted
        superuser
      Profiles
        wpauli
        mcurie
        aeinstein
        mborn
        nbohr-superuser
        nbohr
  Groups
    tsecAdminToolAdminGroup
    demoAdminGroup
  Users
    officew2k
    codehaus
    apache
    citi401k
    bankone
    mcurie
```

Existing Role

Status:	
Application Name:	demo
Role Name:	trusted

Creator's Name:	admin
Create Timestamp:	Mon Apr 24 12:25:21 EDT 2006
Modifier's Name:	
ModifyTimestamp:	

Description

Status: ok

Guardian API

- Secure, simple read only API
- Uses role-based model (RBAC)
- Default backing store is LDAP
- Applications login and see only their authorization information for regulatory compliance
- Swing based and web administration available

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

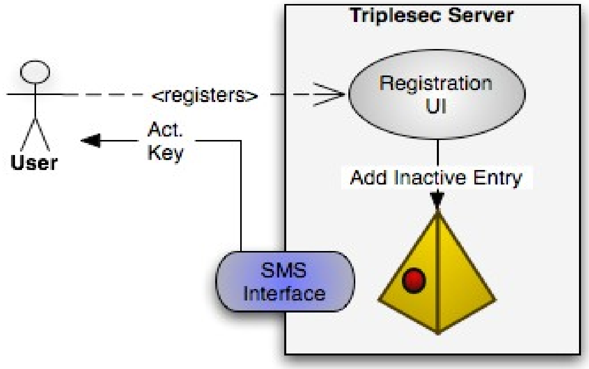
Triplesec Installation and Administration

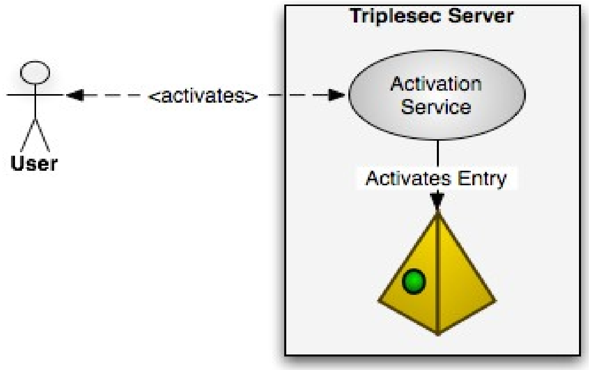
Token Provisioning

- Self-server web based account registration
- Random generation of shared secret
- Java ME application dynamically generated to embed the secret, and user credentials
- Multiple ways to provision the Java ME application to handsets
  - WAP PUSH
  - SMS
  - Email
  - Desktop --> Bluetooth

Provisioning Workflow

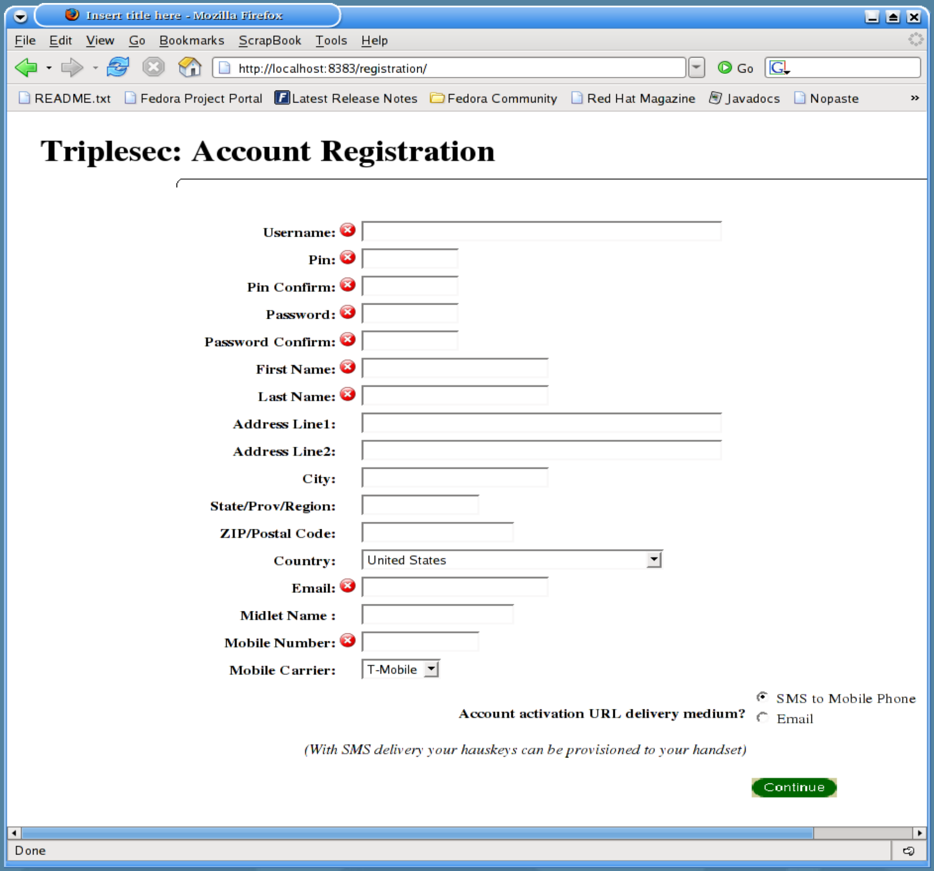
- User registers
- Account is created
- User receives SMS WAP Push of new MIDlet
- User downloads MIDlet
- User receives activation notice via SMS
- User activates account





DEMO

Online Provisioning


Insert title here - Mozilla Firefox
File Edit View Go Bookmarks ScrapBook Tools Help
http://localhost:8383/registration/
Go
README.txt
Fedora Project Portal
Latest Release Notes
Fedora Community
Red Hat Magazine
Javadocs
Nopaste
Continue
Done

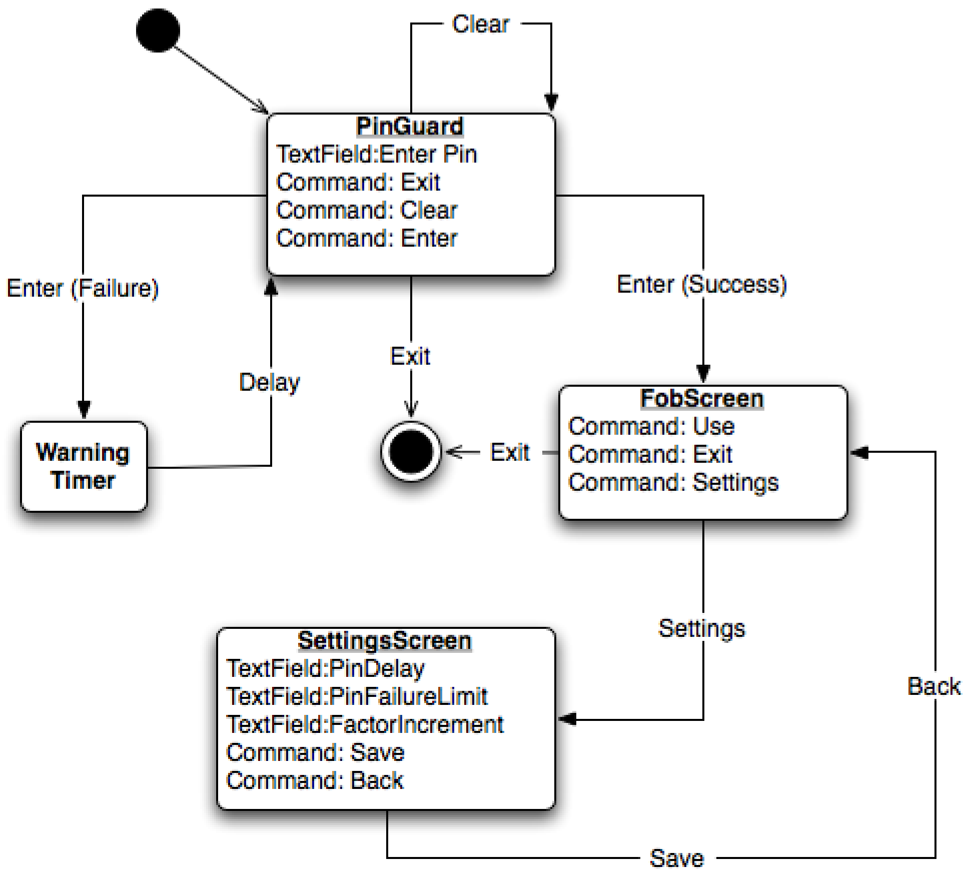
Triplesec: Account Registration

Username:	
Pin:	
Pin Confirm:	
Password:	
Password Confirm:	
First Name:	
Last Name:	
Address Line1:	
Address Line2:	
City:	
State/Prov/Region:	
ZIP/Postal Code:	
Country:	United States
Email:	
Midlet Name :	
Mobile Number:	
Mobile Carrier:	T-Mobile

Account activation URL delivery medium? [✓ SMS to Mobile Phone] Email

(With SMS delivery your hauskeys can be provisioned to your handset)

How Hauskeys Works



Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

Kerberos Configuration

- SafehausLoginModule extends JDK™ software Krb5LoginModule
- Additional OS Kerberos configuration required
- /etc/krb5.conf for UNIX
- c:/WINNT/krb5.ini for Windows
- Both ini and conf files have same syntax

Example krb5 Configuration

```
[libdefaults]
 default_realm = SAFEHAUS.ORG
 ticket_lifetime = 24h
 forwardable = yes
[realms]
  SAFEHAUS.ORG = {
  kdc = localhost:88
  admin_server = localhost:749
  default_domain = karasulu.homeip.net
 }
[domain_realm]
 .karasulu.homeip.net = SAFEHAUS.ORG
 karasulu.homeip.net = SAFEHAUS.ORG
```

Using the SafehausLoginModule

```
// Create new instance of the login module and subject
LoginModule module = new SafehausLoginModule();
Subject subject = new Subject();

// Prepare the shared state and options
Map options = new HashMap();
options.put( SafehausLoginModule.ALLOW_ADMIN, "true" );
Map state = new HashMap();

// Initialize, login and commit the login module
module.initialize( subject, new DemoHandler(),
    state, options );
boolean result = module.login();
result &= module.commit();

// Respond to result ...
```

Implementing a CallbackHandler

```
class DemoHandler implements CallbackHandler
{
  public void handle( Callback[] callbacks ) throws ...
  {
    for ( int ii = 0; ii < callbacks.length; ii++ ) {
      if ( callbacks[ii] instanceof NameCallback ) { ... }
      else if ( callbacks[ii] instanceof PasswordCallback ) {...}
      else if ( callbacks[ii] instanceof RealmCallback ) {
        RealmCallback cb = ( RealmCallback ) callbacks[ii];
        cb.setRealm( realm );
      }
      else if ( callbacks[ii] instanceof PolicyCallback ) {
        PolicyCallback cb = ( PolicyCallback ) callbacks[ii];
        cb.setPolicy( policy );
      }
      else if ( callbacks[ii] instanceof PasscodeCallback ) {
        PasscodeCallback cb = ( PasscodeCallback ) callbacks[ii];
        cb.setPasscode( passcode );
      }
}}}
```

Initializing the ApplicationPolicy

```
// Set up connection parameters
ApplicationPolicy policy = null;
Properties props = new Properties();
props.setProperty( "applicationPrincipalDN",
    "appName=demo,ou=Applications,dc=safehaus,dc=org" );
props.setProperty( "applicationCredentials", "secret" );
String conntionUrl = “ldap://localhost:10389/dc=safehaus,dc=org”;

// Load the driver, and connect to get the policy
Class.forName( "org.safehaus.triplesec.guardian.ldap.LdapConnectionDriver" );
policy = ApplicationPolicyFactory.newInstance( connectionUrl, props );

// Access permissions, roles and profiles
Profile p = policy.getProfile( “akarasulu” );
if ( p.isInRole( “trusted” ) ) {
   ...
}
```

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

Simple for Web Developers

- Package your applications in regular WAR files and deploy on any Java EE container
  - Tomcat is used in this example
- Hook the application up with Triplesec via one of the following ways:
  - Configure a Java EE container managed security domain
  - Access via the Java Authentication and Authorization Service (JAAS)

Using the JAAS Module in a Servlet

- Traditionally application servers would be configured to use a login module
- Principals generated during authentication could be accessible via servlet's getUserPrincipal().
- However you can directly use the SafehausLoginModule
  - Access the application policy on Servlet initialization
  - Use LoginModule to authenticate and authorize user
  - Then use the SafehausPrincipal to authorize fine grained actions

Quick Peek at the web.xml

```
<web-app>
  <servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>org.safehaus.triplesec.demo.LoginServlet</servlet-class>
    <init-param>
      <param-name>realm</param-name>
      <param-value>SAFEHAUS.ORG</param-value>
    </init-param>
    <init-param>
      <param-name>secret</param-name>
      <param-value>secret</param-value>
    </init-param>
    <init-param>
      <param-name>connectionUrl</param-name>
      <param-value>ldap://localhost:10389/dc=safehaus,dc=org</param-value>
    </init-param>
    <init-param>
      <param-name>appDn</param-name>
      <param-value>appName=demo,ou=Applications,dc=safehaus,dc=org</param-value>
    </init-param>
    <load-on-startup/>
  </servlet>
...
```

Servlet Using SafehausLoginModule

```
// Initialize the Servlet with the application's policy
ApplicationPolicy policy = null;
...
public void init( ServletConfig config ) {
  // get init params for realm, connectionUrl, appDn, and secret
  ...
  Properties props = new Properties();
  props.setProperty( "applicationPrincipalDN", appDn );
  props.setProperty( "applicationCredentials", secret );
  try
  {
    Class.forName( "org.safehaus.triplesec.guardian.ldap.LdapConnectionDriver" );
    policy = ApplicationPolicyFactory.newInstance( connectionUrl, props );
  } catch ( Exception e ) { ... }
}
```

Authenticating Users Inside doXXX()

```
// Extract request parameters posted via a form
String username = ( String ) request.getParameter( "username" );
String password = ...
String passcode = ...

// Execute login command that wraps the login module
DemoLoginCommand command = new DemoLoginCommand( username, password,
     realm, passcode, policy );
boolean result = false;
try {
     result = command.execute();
}
catch ( LoginException e ) { ... }
...

// Get the profile of authenticated user and print it out
SafehausPrincipal principal = command.getSafehausPrincipal();
Profile profile = principal.getAuthorizationProfile();
PrintWriter out = response.getWriter();
out.println( "<html><body> ...
```

DemoLoginCommand Looks Familiar

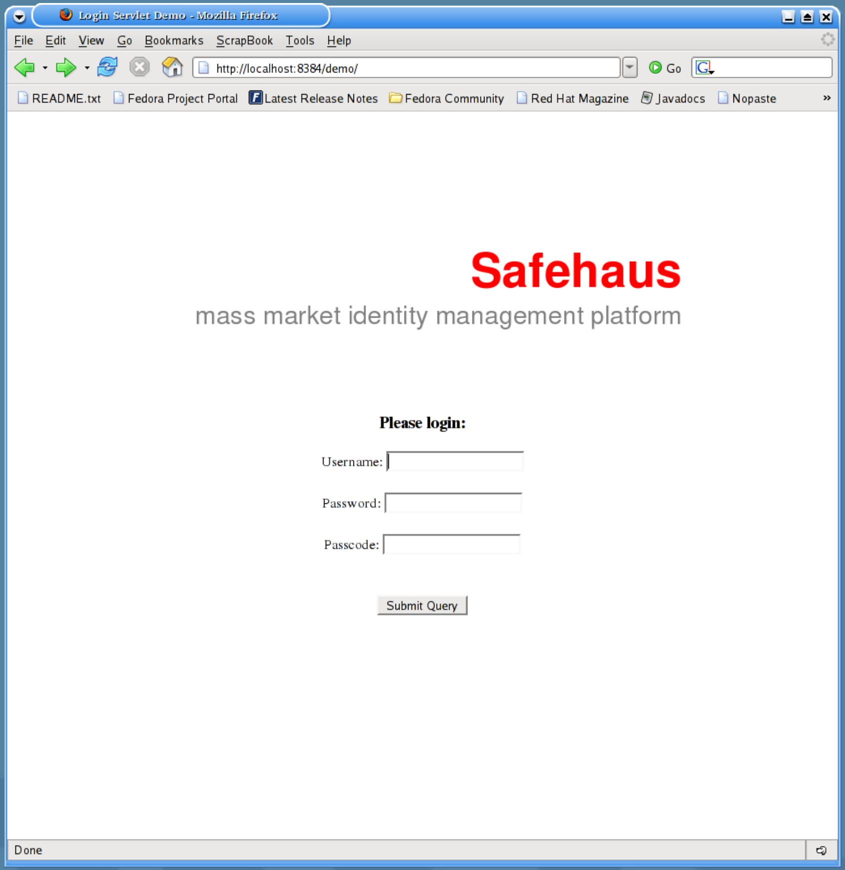
```
...

public boolean execute() throws LoginException
{
  LoginModule module = new SafehausLoginModule();
  Subject subject = new Subject();
  module.initialize( subject, new DemoHandler(), ... );
  boolean result = module.login();
  result &= module.commit();
  ...
  // Extract Principal from the subject if successful
  principal = ( SafehausPrincipal )
    subject.getPrincipals().toArray()[0];
  return result;
}

class DemoHandler implements CallbackHandler {
  ...
```

DEMO

Simple Web Application



Safehaus

mass market identity management platform

Please login:

Username:	
Password:	
Passcode:	

DEMO

Re-sync the Mobile Token

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

Simple for Swing Developers

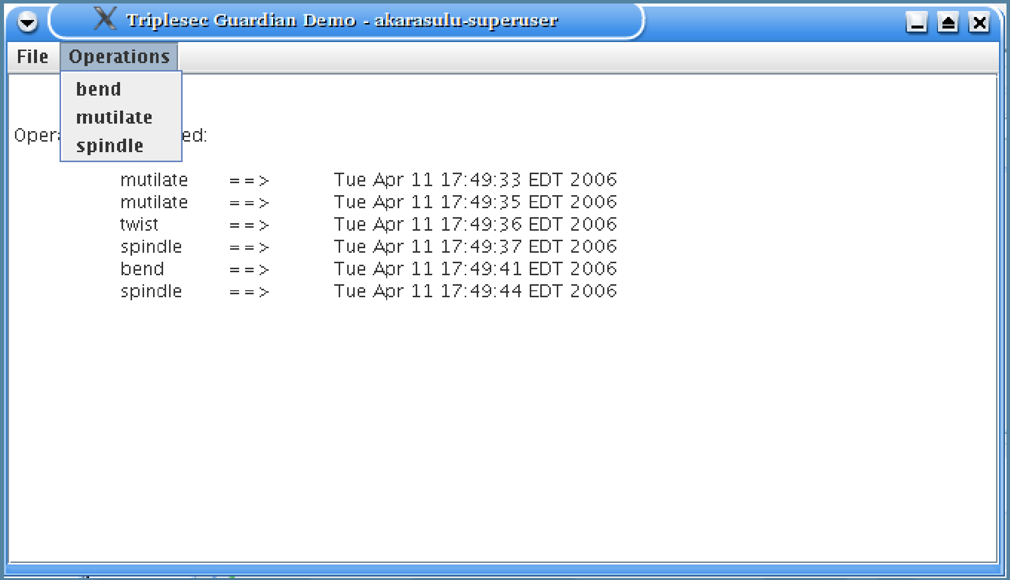
- Access the Triplesec server via the standard JAAS API
- Permission changes on Triplesec server can alter the rich UI client in real time
- The Triplesec administration console itself is a Swing application
- Mechanics are identical to Web example

Most Interesting: Dynamic Updates

```
Class DemoListener implements PolicyChangeListener
{
  public void roleChanged(ApplicationPolicy policy,
    Role role, ChangeType changeType)
  {
    ... if role change effects profile resetMenus( updatedProf );
  }
...
  public void profileChanged(ApplicationPolicy policy,
    Profile profile, ChangeType changeType)
  {
    ... if current profile changed resetMenus( profile );
  }
...
}
```

DEMO

Simple Swing Application with Authorization



```
Oper          ed:

          mutilate      ==>      Tue Apr 11 17:49:33 EDT 2006
          mutilate      ==>      Tue Apr 11 17:49:35 EDT 2006
          twist         ==>      Tue Apr 11 17:49:36 EDT 2006
          spindle       ==>      Tue Apr 11 17:49:37 EDT 2006
          bend          ==>      Tue Apr 11 17:49:41 EDT 2006
          spindle       ==>      Tue Apr 11 17:49:44 EDT 2006
```

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

Seamless OS Integration

- If it can talk LDAP or Kerberos 2-factor integration is seamless
- Authenticating Triplesec users on Linux
- Let’s take a look

DEMO

Authenticate User on a Linux box

Agenda

The Challenges

Triplesec Overview

Provisioning Software Tokens

JAAS LoginModule Use

Writing a Web Application

Writing a Swing Application

Native OS Integration

Triplesec Installation and Administration

DEMO

Install and Start Triplesec

DEMO

The Triplesec Administration Console

Summary

- Triplesec provides a secure and yet easy-to-use 2-factor authentication solution for mass market users
- Triplesec is also an open source and standards-based identity management solution
- For application developers, Triplesec is very easy to work with

What’s Next

- Provisioning and workflow
- User store can be another LDAP server like (AD) or existing RDBMS
- RADIUS and DIAMETER
- Kerberos Trusts between Realms
- SAML and XACML
- Changlog/Auditing interface in the works
- User store and policy store snapshoting and rollback

Special Thanks

- List
- NMSI Messaging: http://nmessaging.com/
- ASF Directory Team: http://directory.apache.org
- Codehaus: http://www.codehaus.org
- Safehaus Triplesec Team: http://triplesec.safehaus.org

Q&A

Triplesec

Simple Mass Market 2-Factor Authentication Using Java™ Technologies

Alex Karasulu, Apache Foundation
Michael Yuan, PhD., JBoss Inc
The Safehaus Triplesec project

http://triplesec.safehaus.org/

TS-9862