

Benutzerauthentifizierung und Zugriffsschutz mit JAAS

Werner Eberling

werner.eberling@mathema.de

www.mathema.de

- Altbekanntes kurz erwähnt
 - Java Security

- Des Kaisers neue Kleider
 - JAAS

- Zu Ihren Diensten
 - JAAS angewendet

- **Altbekanntes kurz erwähnt**
 - Java Security

- Des Kaisers neue Kleider
 - JAAS

- Zu Ihren Diensten
 - JAAS angewendet

■ Java Security

- build-in (`java.lang.SecurityManager`)
- Permission based
- Abfragen müssen explizit erfolgen (`checkPermission()`)
- SecurityManager muss beim Start der VM aktiviert werden !!!

■ Rechtevergabe erfolgt

- über Positivlisten (Policies)
- auf Grund der Codebase
- oder einer digitalen Signatur

■ Definition von Permissions

```
package de.mathema.campus.jaas;

import java.security.BasicPermission;

public class CampusPermission extends BasicPermission {

    public CampusPermission(String name) {
        super(name);
    }

    public CampusPermission(String name, String actions) {
        super(name, actions);
    }
}
```

■ Rechteabfrage über den SecurityManager

```
...
SecurityManager security = System.getSecurityManager();
if (security != null) {
    security.checkPermission(new CampusPermission("mathema.campus.goodGuy"));
}
...
```

■ java.security

```
...
#
# Class to instantiate as the system Policy. This is the name of the class
# that will be used as the Policy object.
#
policy.provider=sun.security.provider.PolicyFile

# The default is to have a single system-wide policy file,
# and a policy file in the user's home directory.
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
...
```

■ java.policy

```
...
// Standard extensions get all permissions by default

grant codeBase "file:${java.home}/lib/ext/*" {
    permission java.security.AllPermission;
};
...
```

- Bisher: Alles oder Nichts-Prinzip
 - Eine Klasse hat ein bestimmtes Recht oder sie hat es nicht
- Aber: Was ist mit den Rechten der Benutzer?
 - Nicht: Woher kommt der Code?
 - Sondern: Unter wessen Namen läuft der Code?
- J2EE bietet nur eine unbefriedigende Lösung
 - Autorisierung auf User-Ebene durch eigenes Konzept möglich (noch eine Stelle zur Sicherheitskonfiguration)
 - Bis Version 1.3 keine Standardisierung der Authentifizierung (=> Security nur eingeschränkt portabel !!!)

- Altbekanntes kurz erwähnt
 - Java Security
- Des Kaisers neue Kleider
 - JAAS
- Zu Ihren Diensten
 - JAAS angewendet

- Java Authentication and Authorization Service (JAAS)
 - Add-on zu j2sdk-1.3 / Bestandteil von j2sdk-1.4
 - Ermöglicht Autorisierung auf User-Ebene
 - Standardisierter Mechanismus zur Benutzer-Authentifizierung
 - Konsequente Weiterführung des Policy/Permission-Konzeptes

- Wichtigste Bestandteile
 - `javax.security.auth.Subject`
 - `javax.security.auth.login.LoginContext`
 - `javax.security.auth.callback.CallbackHandler`
 - `javax.security.auth.spi.LoginModule`

■ Repräsentation einer Entität mit ihren Sicherheitsattributen

```
package javax.security.auth;

public final class Subject implements java.io.Serializable {
    ...

    public static Object doAsPrivileged(final Subject subject,
        final java.security.PrivilegedAction action,
        final java.security.AccessControlContext acc) {
        ...
    }

    public Set getPrincipals() {
        ...
    }

    ...
}
```

■ Schnittstelle zur client-seitigen Authentifizierung

```
package javax.security.auth.login;

public class LoginContext {

    public LoginContext(String name, CallbackHandler callbackHandler)
        throws LoginException {
        ...
    }

    public void login() throws LoginException {
        ...
    }

    public void logout() throws LoginException {
        ...
    }

    public Subject getSubject() {
        ...
    }
}
```

- Schnittstelle zur Abarbeitung serverseitiger Callbacks während der Authentifizierung (vom Client zu implementieren)

```
package javax.security.auth.callback;  
  
public interface CallbackHandler {  
  
    void handle(Callback[] callbacks)  
        throws java.io.IOException, UnsupportedCallbackException;  
}
```

- Default-Implementierungen
 - com.sun.security.auth.callback.DialogCallbackHandler
 - com.sun.security.auth.callback.TextCallbackHandler
- Wichtige Callbacks
 - NameCallback / PasswordCallback
 - TextInputCallback / TextOutputCallback

- Schnittstelle zur Implementierung eigener Authentifizierungsmechanismen

```
package javax.security.auth.spi;

public interface LoginModule {

    void initialize(Subject subject, CallbackHandler callbackHandler,
                   Map sharedState, Map options);

    boolean login() throws LoginException;

    boolean commit() throws LoginException;

    boolean abort() throws LoginException;

    boolean logout() throws LoginException;
}
```

- JAAS delegiert die Authentifizierung des Benutzers an die dafür konfigurierten LoginModule

■ java.security

```
...  
#  
# Class to instantiate as the javax.security.auth.login.Configuration  
# provider.  
#  
login.configuration.provider=com.sun.security.auth.login.ConfigFile  
  
#  
# Default login configuration file  
#  
login.config.url.1=file:${user.home}/.java.login.config  
...
```

■ login.config

```
...  
  campus {  
    de.mathema.campus.jaas.server.CampusLoginModule required  
  };  
...
```

■ java.policy

```
...  
grant Principal de.mathema.campus.jaas.server.CampusPrincipal  
  "Neo" {  
    permission java.security.AllPermission;  
  };  
...
```

- J2EE-1.4 compliant Server müssen JAAS zur Authentifizierung anbieten
 - Standardisierte client-seitige Authentifizierung über CallbackHandler
 - Anbindung eigener Realms über die Implementierung serverunabhängiger LoginModule
- Aber: J2EE Autorisierung bleibt „eigenes Süppchen“
 - Konfiguration weiter über DD anstatt über Policies

- Altbekanntes kurz erwähnt
 - Java Security

- Des Kaisers neue Kleider
 - JAAS

- Zu Ihren Diensten
 - JAAS angewendet

■ Implementierung der login()-Methode aus dem LoginModule-Interface

```
package de.mathema.campus.jaas.server;

public class CampusLoginModule implements LoginModule {

    ...

    public boolean login() throws LoginException {
        String[] userNameAndPass = getUserUserNameAndPass();
        if (isPassValid(userNameAndPass[0], userNameAndPass[1])) {
            identity = new CampusPrincipal(userNameAndPass[0]);
            isLoggedIn = true;
        } else {
            identity = null;
            isLoggedIn = false;
            throw new FailedLoginException("Wrong username/Password");
        }
        return isLoggedIn;
    }

    ...
}
```

■ Noch mehr Methoden aus dem Interface

```
...  
    public boolean commit() throws LoginException {  
        if (isLoggedIn) {  
            subject.getPrincipals().add(identity);  
        }  
        return isLoggedIn;  
    }  
  
    public boolean abort() throws LoginException {  
        subject.getPrincipals().remove(identity);  
        identity = null;  
        if (isLoggedIn) {  
            isLoggedIn = false;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public boolean logout() throws LoginException {  
        subject.getPrincipals().remove(identity);  
        identity = null;  
        isLoggedIn = false;  
        return true;  
    }  
...  
}
```

■ Username und Passwort holen wir über einen Callback

```
...
private String[] getUserAndPass() throws LoginException {
    Callback[] callbacks = new Callback[3];
    callbacks[0] =
        new TextOutputCallback(
            TextOutputCallback.INFORMATION,
            "Authentication required. Please enter username
                                                and password.");
    callbacks[1] = new NameCallback("username>>");
    callbacks[2] = new PasswordCallback("password>>", true);

    try {
        callback.handle(callbacks);
    } catch (IOException e) {
        throw new LoginException("Callbackerror");
    } catch (UnsupportedCallbackException e) {
        throw new LoginException("Callbackerror");
    }

    return new String[] {
        ((NameCallback) callbacks[1]).getName(),
        String.valueOf(((PasswordCallback)
            callbacks[2]).getPassword())};
}
}
```

■ Implementierung eines (sehr) einfachen CallbackHandlers

```
package de.mathema.campus.jaas.client;

public class CampusCallbackHandler implements CallbackHandler {

    private String password;
    private String name;

    public CampusCallbackHandler(String name, String password){
        this.name=name;
        this.password = password;
    }

    public void handle(Callback[] arg0)
        throws IOException, UnsupportedCallbackException {
        for(int i=0;i<arg0.length;i++){
            handle(arg0[i]);
        }
    }

    ...
}
```

■ Verarbeitung der einzelnen Callbacks

```
...  
  
private void handle(Callback callback)  
    throws UnsupportedOperationException {  
    if(callback instanceof TextOutputCallback){  
        String msg = ((TextOutputCallback)callback).getMessage();  
        if(((TextOutputCallback)callback).getMessageType() ==  
            TextOutputCallback.ERROR){  
            System.err.println(msg);  
        }else{  
            System.out.println(msg);  
        }  
        return;  
    }  
    if(callback instanceof NameCallback){  
        ((NameCallback)callback).setName(name);  
        return;  
    }  
    if(callback instanceof PasswordCallback){  
        ((PasswordCallback)callback).setPassword(  
            password.toCharArray());  
        return;  
    }  
    throw new UnsupportedOperationException(callback);  
}  
...  
}
```

■ Einer muss ja die Arbeit machen...

```
package de.mathema.campus.jaas.client;

public class CampusClient {
    public CampusClient() {
        try {
            LoginContext lc = new LoginContext(
                "campus",
                new CampusCallbackHandler("Werner", "Campus"));
        }
        for (int i = 3; i != 0; i--) {
            try {
                lc.login();
                Subject.doAsPrivileged(lc.getSubject(), new CampusAction(), null);
                lc.logout();
                return;
            } catch (LoginException le) {
                continue;
            } catch (SecurityException se) {
                lc.logout();
                continue;
            }
        }
    } catch (LoginException e) {
        e.printStackTrace();
    }
}
...
```

■ java.security

```
...
#
# Class to instantiate as the javax.security.auth.login.Configuration
# provider.
#
login.configuration.provider=com.sun.security.auth.login.ConfigFile

#
# Default login configuration file
#
login.config.url.1=file:${campus.home}/2003/JAAS/config/login.config
...
```

■ login.config

```
...
campus {
    de.mathema.campus.jaas.server.CampusLoginModule required
    users.properties="${campus.home}/config/users.properties";
};
...
```

■ java.policy

```
...
grant Principal de.mathema.campus.jaas.server.CampusPrincipal
    "Werner" {
    permission java.security.AllPermission;
};
...
```

```
package de.mathema.campus.jaas.client;

import javax.security.auth.Subject;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import de.mathema.campus.jaas.CampusAction;

public class CampusClient {

    public CampusClient() {
        try {
            LoginContext lc;
            if (System.getProperty("useGUI") != null) {
                lc = new LoginContext("campus", new CampusGUICallbackHandler());
            } else {
                lc =
                    new LoginContext(
                        "campus",
                        new CampusCallbackHandler("Werner", "Campus"));
            }

            for (int i = 3; i != 0; i--) {
                try {
                    lc.login();
                    Subject.doAsPrivileged(lc.getSubject(), new CampusAction(), null);
                    lc.logout();
                    return;
                } catch (LoginException le) {
                    System.err.println("le:" + le.getMessage());
                    continue;
                } catch (SecurityException se) {
                    System.err.println("se:" + se.getMessage());
                    lc.logout();
                    continue;
                }
            }
        }
    }
}
```

loading...
Initialized with [Werner=Campus, Nicole=Beilngries]
logging in...
Authentication required. Please enter username and password.
username>>Werner
password>>*****
committing...
Hey, you're the chosen one !
logging out...

■ JAAS bietet

- Einfaches Framework zur standardisierten Authentifizierung (j2sdk und j2sdkee)
- Konsistente Erweiterung des JavaSecurity-Konzepts um User-basierte Autorisierung (j2sdk)

■ Schade eigentlich

- j2sdkee bleibt bei der Autorisierung leider noch immer aussen vor

■ Ausblick

- JAAS ermöglicht den Aufbau eines leichtgewichtigen Resource Access Decision Dienstes allein auf Basis des j2sdk
=> Mehr dazu am Samstag

Vielen Dank!

werner.eberling@mathema.de

www.mathema.de

meet the
experts
of enterprise infrastructure