Blog article

# Windows Authentication in Java

22.02.2018 11:46 by Stefan Kowski

If you want to restrict the users of an application, the first solution is to make them authenticate themselves. The user enters his or her user name and password and after successful verification of the values the program can be used.

Since storing login credentials is a problem, I present a simple solution in this article: delegating the authentication function to a Windows domain controller. The user logs on with his or her Windows user name and password, which is checked by the Windows server.

## Which protocol do I use for authentication?

*„When in Rome, do as the Romans do."*

I use the protocol that Windows itself uses for its domain logon: Kerberos. Kerberos is a challenge-response method designed for use in insecure networks. No passwords are transmitted in the network, and the retrieval of login packages for replay attacks is also ineffective (more about Kerberos here).

I use JAAS (Java Authentication and Authorization Service) as the Java API for my implementation. My implementation is configuration-free, except for the name of the Windows domain, no other data is required. Therefore, there is no JAAS configuration file that needs to be customized or deployed.

You could also perform the logon using an LDAP bind. This is not recommended, however, as the LDAP protocol likes to transfer passwords in clear text and an SSL connection (ldaps:) is therefore mandatory. In addition, there can be considerable configuration effort if the LDAP standard ports are not used or SSL certificates are needed for LDAP client authentication (depending on what local network administration requires).

## How do I find the Windows server to authenticate against?

The available Windows servers that can perform logon functions are registered in the DNS. We determine the servers via a DNS query.

```
/**
  * Get Active Directory domain contro
  *
  * Shell example: nslookup -type=SRV
  *
  * @param domain
  *           Domain name (e.g. "mydo
  * @return Domain controllers (list m
  * @throws NamingException
  */
private static Collection<InetSocketA

    final String typeSRV = "SRV";
    final String[] types = new String

    DirContext ctx = new InitialDirCo

    Attributes attributes = ctx.getAt
    if (attributes.get(typeSRV) == nu
        return Collections.emptyList(
    }

    NamingEnumeration<?> e = attribut
    TreeMap<Integer, InetSocketAddres

    while (e.hasMoreElements()) {

        String line = (String) e.next

        // The line is: priority weig
        String[] parts = line.split("

        int prio = Integer.parseInt(p
        int port = Integer.parseInt(p
        String host = parts[3];

        result.put(prio, new InetSock
    }

    return result.values();
}
```

In large domains, the function may return 20 or more servers. Since a DNS load balancing usually takes place anyway and our function is only rarely called, I do not evaluate the server priorities in the further program and simply use the first delivered server.

## Configure JAAS

The JAAS API requires a configuration. We do not, so we simply implement an empty configuration class.

```
/**
  * JAAS configuration.
  */
 public static class StaticConfigurati

     final AppConfigurationEntry stati

     public StaticConfiguration(String

         Map<String, ?> options = new
         staticConfigEntry = new AppCo
                 AppConfigurationEntry
     }

     @Override
     public AppConfigurationEntry[] ge

         return new AppConfigurationEn
     }
 }
```

The JAAS configuration requires a handler that is usually used to interactively query logon data from the user. In our case, however, the data comes via API, so that we implement a handler that transfers these values to the JAAS.

```
/**
  * JAAS callback handler.
  */
 public static class StaticCallbackHan

```

```
    /**
     * Constructor.
     *
     * @param username
     *            Windows user name
     * @param password
     *            Windows password
     */
    public StaticCallbackHandler(Stri

        this.username = username;
        this.password = password;
    }

    @Override
    public void handle(Callback[] cal

        for (int i = 0; i < callbacks

            if (callbacks[i] instance

                // unused

```

```
            } else if (callbacks[i] i

                PasswordCallback pc =
                pc.setPassword(passwo

            } else {

                throw new Unsupported
            }
        }
    }

    /** User name. */
    private String username;
```

```

        /** Password. */
        private String password;
    }
```

## Perform the Windows logon

The following code example contains the actual logon function. First a suitable Windows server is searched for, then the JAAS subsystem is configured for Kerberos. The login context object is then used to perform the actual logon.

```
/**
  * Constructor.
  *
  * @param domainName
  *            domain name (e.g. "mydo
  */
 public ActiveDirectoryAuthentication(

     this.domainName = domainName;
 }

 /**
  * Authenticate user.
  *
  * @param username
  *            Windows user name
  * @param password
  *            Windows password
  * @throws ValidationException
  */
 public void authenticate(String usern

     LoginContext lc;
     try {

         // get domain controller for
         Collection<InetSocketAddress>
```

```
            if (result.isEmpty()) {
                throw new ValidationExcep
            }
            String loginServer = result.i
            System.setProperty("java.secu
            System.setProperty("java.secu

            // perform login
            lc = new LoginContext("", nul
                    new StaticConfigurati
            lc.login();

            // logout (we want to check t
            lc.logout();

        } catch (LoginException le) {

            // error
            throw new ValidationException

        } catch (SecurityException se) {

            // error
            throw new ValidationException

        } catch (NamingException ne) {

            // error
            throw new ValidationException
        }
    }

    /** Windows domain name. */
    private String domainName;
}
```

## Conclusion

The example shows that Windows authentication is easy to implement in Java. Since there are no complex configurations and dependencies, the code example can be used in very different environments.

Source code and sample application (4.8 KiB)

Go back

## Address & Contact

Parks Informatik GmbH
Girardetstr. 6
45131 Essen
Germany

**Tel.:** +49 (0) 201 54528-0
**Fax:** +49 (0) 201 54528-28
**Email:** parks@parks-informatik.de

N: 51º 25' 52" E: 07º 0' 16"
View on Google Maps

Home Solutions Blog About us Contact

### Important links

- PAM demo manual
- Blog

### Parks Informatik GmbH

Since 1999 we have been developing software, especially in the areas of Java EE and C++. In addition, we support you in various areas of IT security, e.g. the creation of authorization concepts, request and approval workflows, as-is analyses of user permissions and security in business processes.

### From our Blog

| | |
|---|---|
| 18. MAR | Show group membership in Active Directory |
| 22. FEB | Windows Authentication in Java |
| 12. MAY | Building a group structure - the "AGDLP" principle |
| 09. NOV | Show effective permission groups of a user |

Search Sitemap Legal