Securing WLS Resources and Applications

At the end of this module you will be able to:

- ✓ Describe WLS security architecture
- ✓ Configure users, groups, and roles
- ✓ Configure security realms
- ✓ Secure Web Applications with declarative security
- ✓ Configure policies and SSL
- ✓ Create and manage certificates
- ✓ Protect WLS against several types of attacks

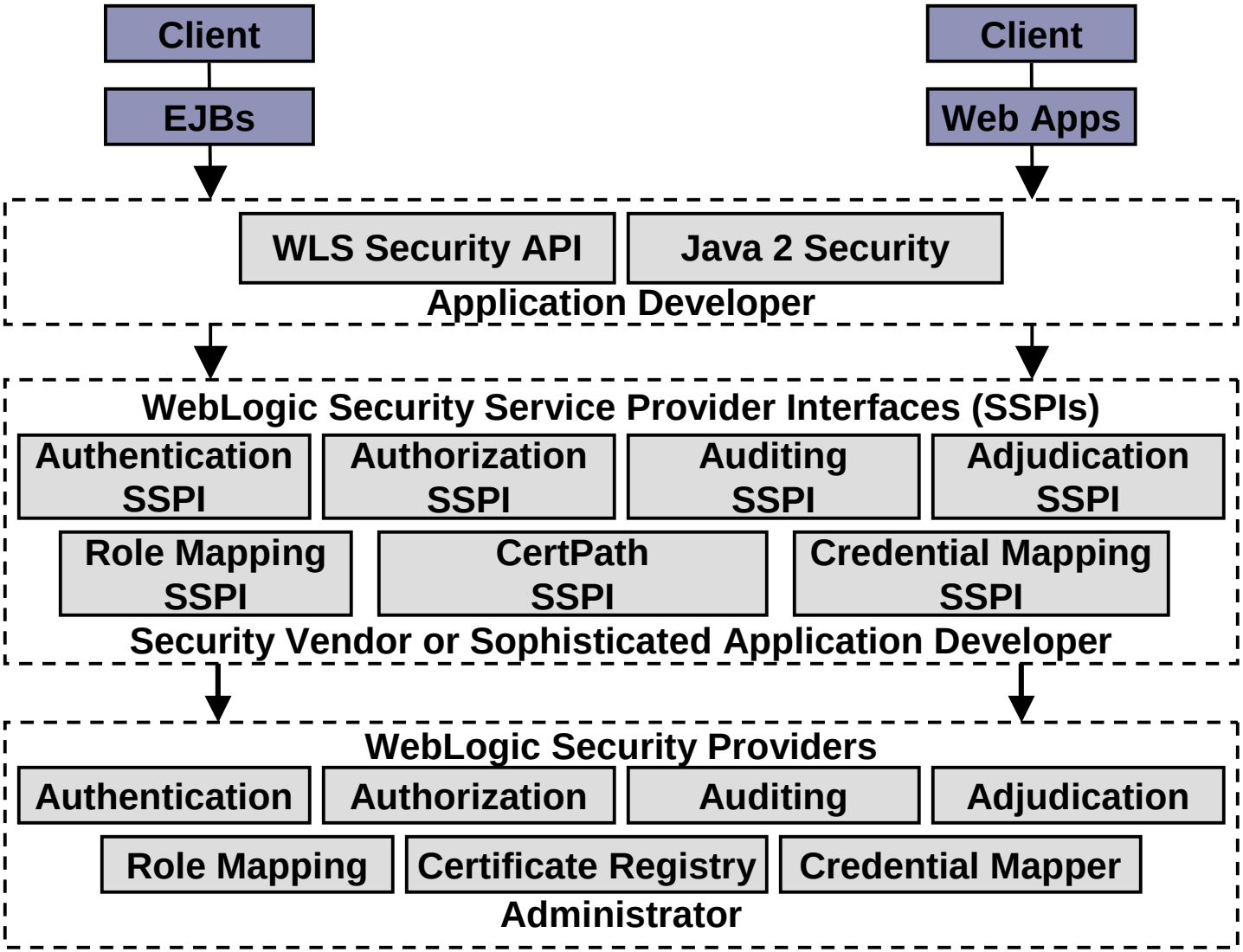
Road Map

1. **WLS Security Architecture Overview**
   - Security Architecture
   - Security Terminology
   - Compatibility with Previous WLS Versions
2. Users and Groups
3. Protecting Application Resources
4. Protecting Communications
5. Protecting Against Attacks

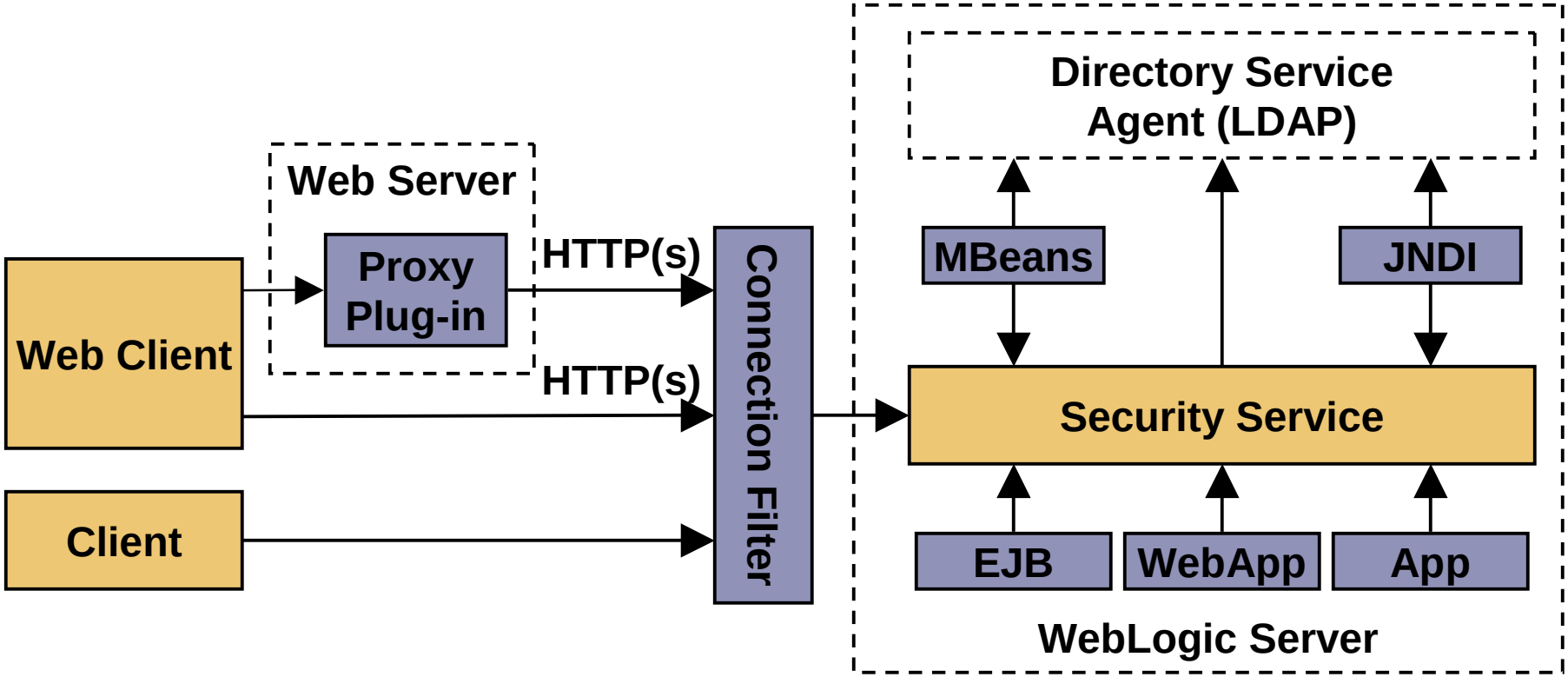
Architecture Goals

- Using Java standards (where applicable) create an architecture that unifies security enforcement and present it as a service to other components.
- Provide a security framework that allows integration of 3rd party security products with minimum restrictions on functionality.
- Provide consistent and unified protection for all resources hosted on WebLogic Server:
  - EJBs
  - Web Applications (Servlets, JSP)
  - Web Services
  - Miscellaneous J2EE Resources
    - RMI objects JDBC, JNDI, MBeans

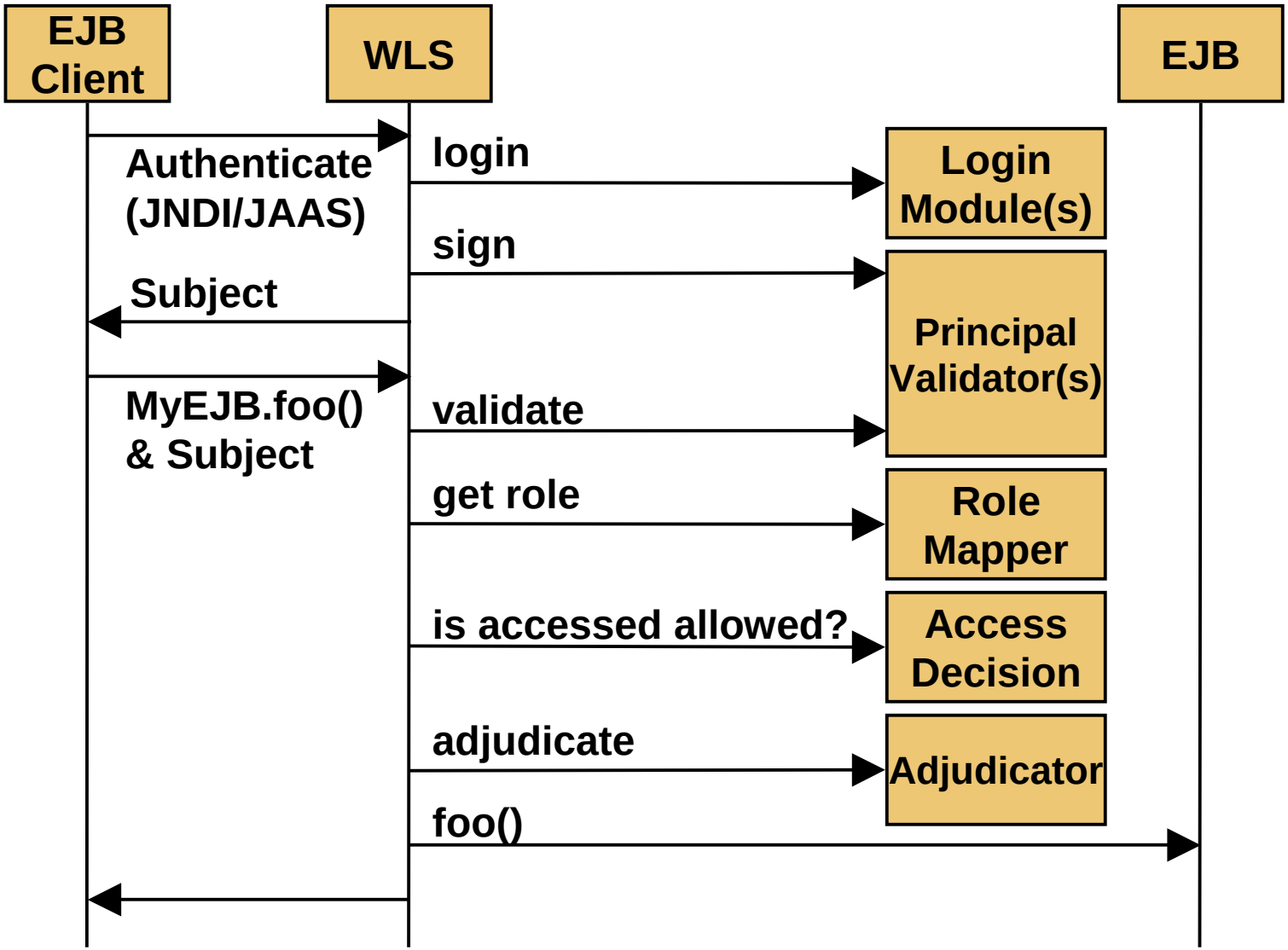
Security Architecture



Process Architecture



Security Services



Authentication Providers

- An *authentication provider* uses `LoginModules` to authenticate users within a security realm.
- An authentication provider transports identity information and makes it available to components with *Subjects*.
- The *Principal Validation provider* provides additional protection by signing and verifying the authenticity of the principals.
- *Identity Assertion providers* as a `LoginModule` to map a valid token to a WebLogic server user.

Authorization Providers

- An *authentication provider* is a process to control the interactions between users and resources based on user identity.
- The *role mapping provider* supplies the security role with information to determine whether access is allowed for role-based resources, such as EJBs.
- The *access decision* component answers the question, is access allowed?
- The access decision returns a result of PERMIT, DENY or ABSTAIN.
- The *adjudication provider* can be used to tally the results that multiple access decisions return to determine the final decision.

Confidentiality

- WebLogic Server supports the *Secure Sockets Layer (SSL)* protocol to secure the communication between clients and server.
- SSL *client authentication* allows a server to confirm a user's identity by checking that a client's certificate and public ID are valid and are issued by a *certificate authority* (CA).
- SSL server authentication allows a user to confirm a server's identity by checking that the server's certificate and public ID are valid and are issued by a CA.

Credential Mapping

- *The credential mapping* process is initiated when application components must access a legacy system authentication mechanism to obtain a set of credentials.
- The requesting application passes the Subject as part of the call and information about the type of credentials required.
- Credentials are returned to the security framework which is then passed to the requesting application component.
- The application component uses the credentials to access the external system.

Auditing

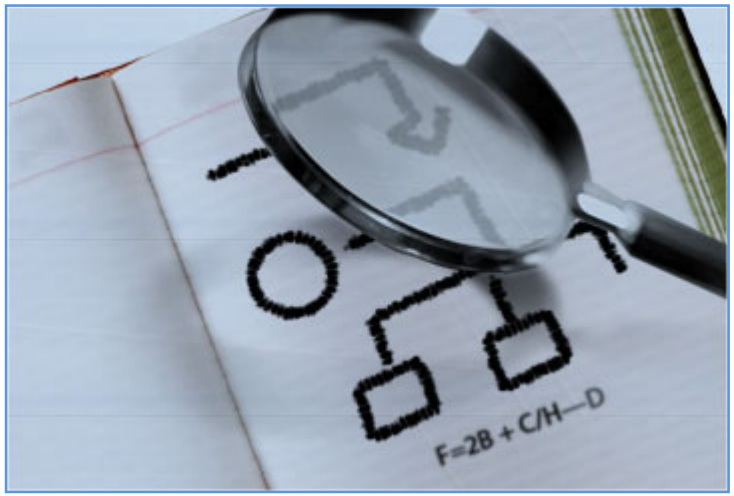
- *Auditing* provides a trail of activity.
- The *Auditing provider* is used to log activity before and after security operations.
- The default auditing provider records the event data associated with the security requests and the outcome of the requests.

```
17 PM>  <Severity =FAILURE>  <<<Event Type = Authentication Audit Event><rbrown>
                                                <AUTHENTICATE>>> Audit Record End ####
#### Audit Record Begin <Feb 7, 2004 8:15:23 PM>  <Severity =FAILURE>  <<<Event Type = Authentication Audit Event>
                                                <paulc><AUTHENTICATE>>> Audit Record End ####
#### Audit Record Begin <Feb 7, 2004 8:15:27 PM>  <Severity =SUCCESS>  <<<Event Type = Authentication Audit Event>
                                                <paulc><AUTHENTICATE>>> Audit Record End ####
```

Section Review

In this section we discussed:

- ✓ Security architecture
- ✓ Security terminology



Road Map

1. WLS Security Architecture Overview
2. **Users and Groups**
   - WLS Embedded LDAP
   - Security Realms
   - Configuring Users, Groups, and Roles
3. Protecting Application Resources
4. Protecting Communications
5. Protecting Against Attacks

Security Realms

- A *Security Realm* is a collection of system resources and security service providers.
- Only one security realm can be active at a given time.
- A single security policy is used in any realm.
- Users must be recognized by an authentication provider of the security realm
- Admin tasks include creating security realms.

What Is LDAP?

- LDAP is:
  - The Lightweight Directory Access Protocol
  - Derived from X.500
  - Provides a hierarchical lookup service
  - Supports sophisticated searching
  - Can be secured via SSL

Embedded LDAP Server

- In WLS users, groups, and authorization information is stored in an embedded LDAP server.
- Several properties can be set to manage the LDAP server, including:
  - Credentials
  - Backup settings
  - Cache settings
  - Replication settings

Configuring Embedded LDAP

Settings for humanresources

Configuration | Monitoring | Control | Security | WebService Security | Notes

General | Filter | Unlock User | **Embedded LDAP** | Roles | Policies

Save

This page allows you to configure the embedded LDAP server for this WebLogic Server domain.

Credential:	••••••••••••••••••	The credential (usually a password) used to connect to
Confirm Credential:	••••••••••••••••••	Enter the credential again. More Info...
Backup Hour:	23	The hour (between 0 and 23) at which the embedded L
Backup Minute:	5	The minute (between 0 and 59) at which the embedde
Backup Copies:	7	The maximum number of backup copies (between 0 an server. More Info...
☑ **Cache Enabled**		Specifies whether a cache is used with the embedded L
Cache Size:	32	The size of the cache (in kilobytes) that is used with th
Cache TTL:	60	The time-to-live of the cache (in seconds) that is used
☐ **Refresh Replica At Startup**		Specifies whether a Managed Server should refresh all

Connecting an external LDAP browser

Change this password to allow an external browser to connect. The password can be different from the admin password.

Users and Groups

- *Users* are entities that use WLS such as:
  - Application end users
  - Client applications
  - Other WebLogic Servers
- *Groups* are:
  - Logical sets of users
  - Are more efficient for managing a large number of users

Configuring New Users

Create a New User

OK | Cancel

User Properties

The following properties will be used to identify your new User.

What would you like to name your new User?

Name:

How would you like to describe the new User?

Description:

Please choose a provider for the user.

Provider: DefaultAuthenticator

The password is associated with the login name for the new User.

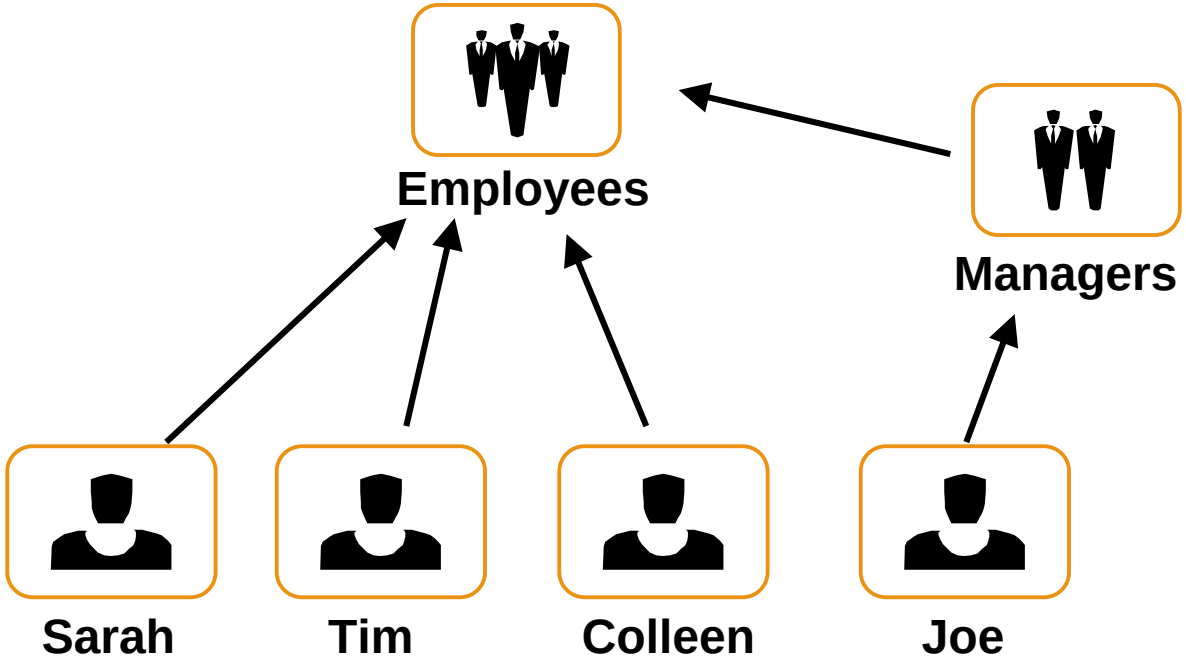
Password:

Confirm Password:

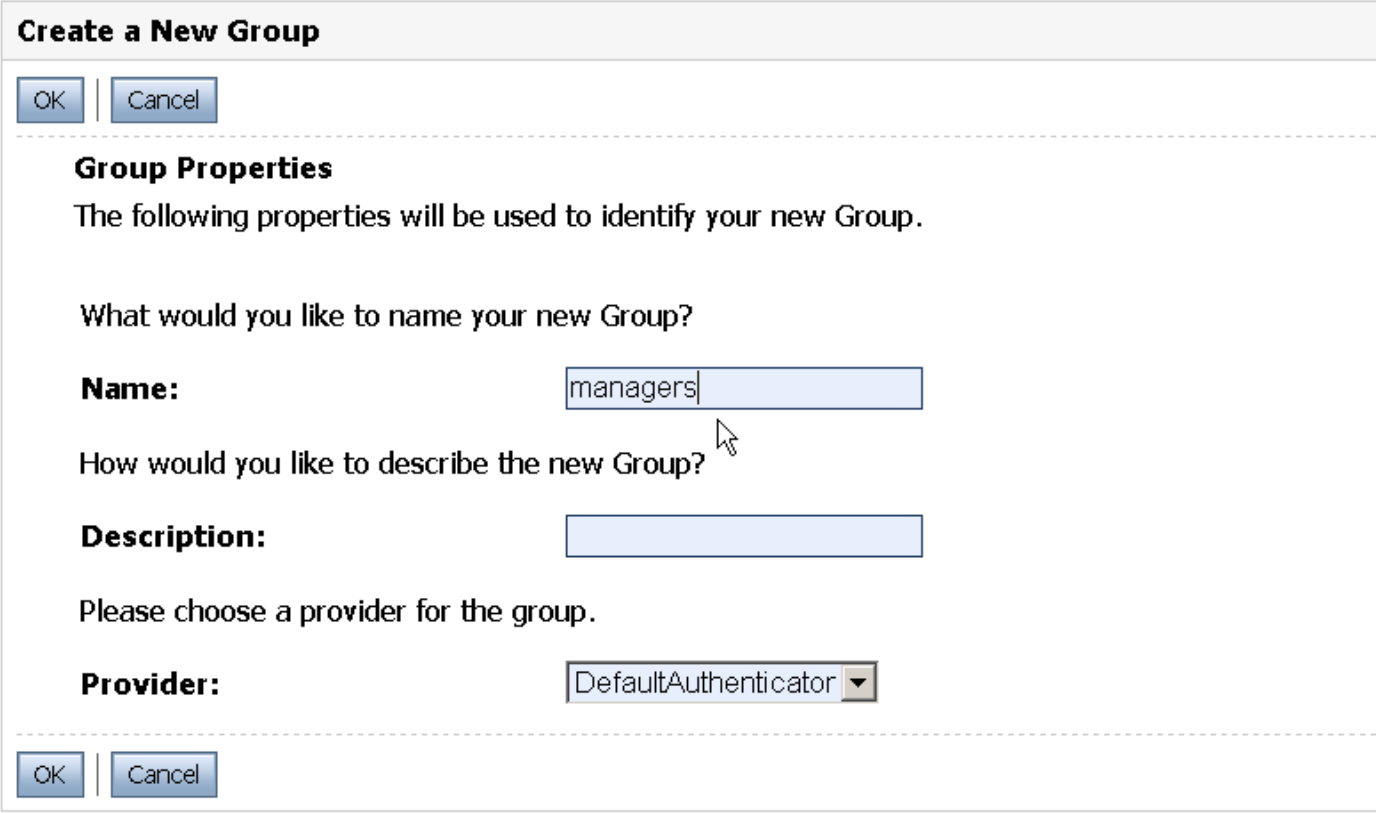
OK | Cancel

Groups

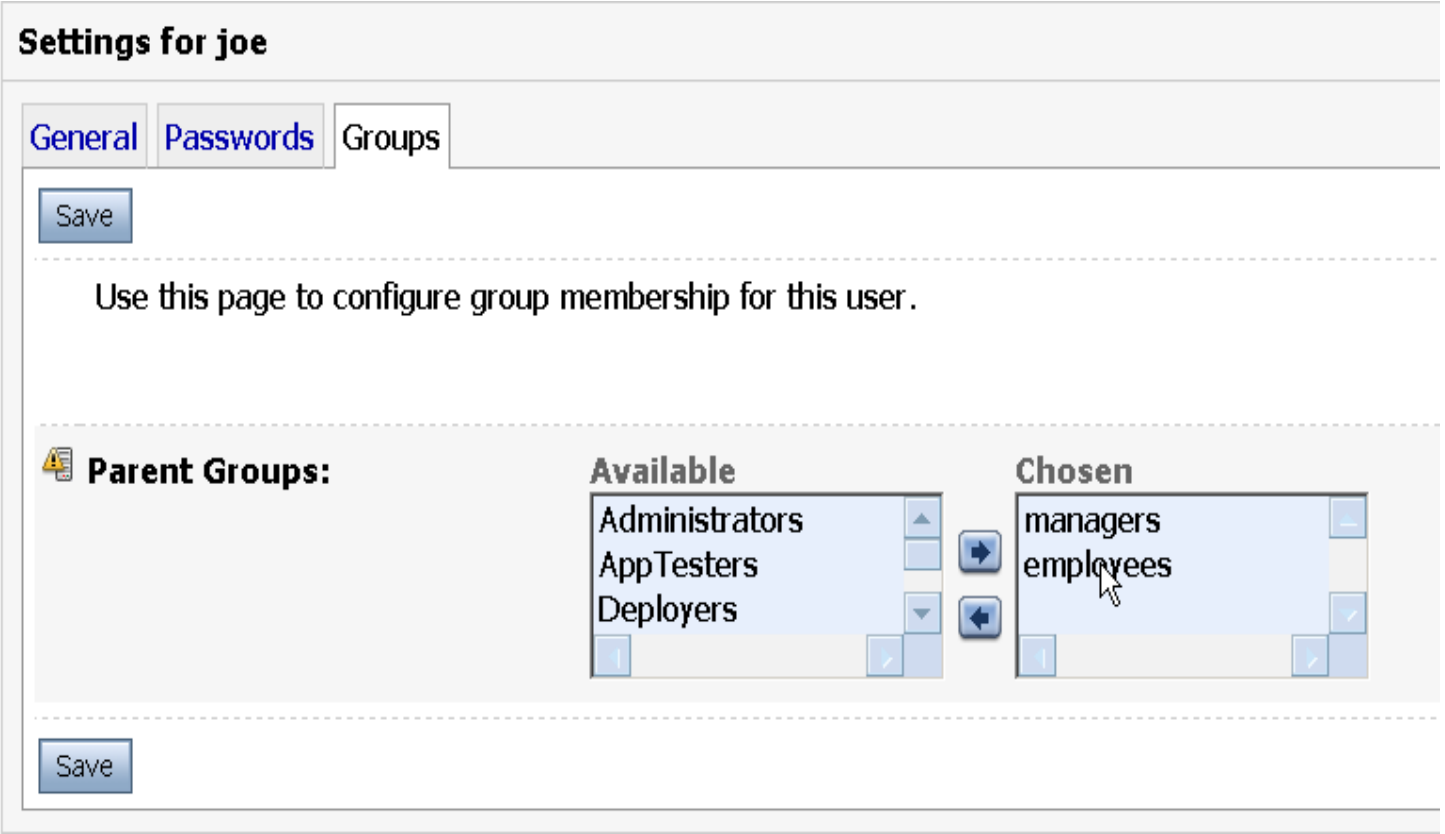
- WLS provides the flexibility to organize groups in various ways:
  - Groups can contain users
  - Groups can contain other groups



Configuring New Groups



Adding Groups to Users



Roles

- A *role* refers to a set of users who have the same permissions.
- A role differs from a group; a group has static membership; a role is conditional.
- A user and group can be granted multiple roles.
- There are two types of roles: global-scoped roles and resource-scoped roles.
- These global roles are available by default: Admin, Operator, Deployer and Monitor.
- Roles defined in deployment descriptors can be *inherited*.
  - Occurs at deployment time
  - Can be disabled
- You can manage role definitions and assignments without editing deployment descriptors or redeploying.

Configuring New Roles

1

Welcome, system — Connected to: humanresources

Home > Summary of Security Realms > Summary of Deployments > timeoff

Settings for timeoff

Overview | Configuration | Security | Targets | Control | Testing | Monitoring | Notes

Application Scope | URL Patterns

Roles | Policies

Use this page to add, edit, or remove application-scoped security roles for this Web Application.

Customize this table

Scoped Roles

New | Delete

	Role Name	Provider Name
	director	DefaultRoleMapper
	Role-0	DefaultRoleMapper
	Role-1	DefaultRoleMapper

New | Delete

2

Welcome, system — Connected

Home > Summary of Security Realms > Summary of Deployments > timeoff > Summary of Deployments > timeoff

Edit URL Pattern Role

Back | Next | Finish | Cancel

Choose a Predicate

Choose the predicate you wish to use as your new condition

The predicate list is a list of available predicates which can be used to make up a security policy c

Predicate List: Access occurs between specified hours

Back | Next | Finish | Cancel

Migrating Security Data

- Can export users/groups, security policies, security roles, or credential maps between security realms or domains.
- Useful, for example, in transitioning from development to QA to production.
- Use migration constraints (key/value pairs) to specify the export/import options.
- Currently only supports migrating security data between WLS security providers.

Exporting WLS Default Authenticator provider

Settings for DefaultAuthenticator

Configuration | Performance | Migration

Import | **Export**

Save

This page allows you to export users and/or groups from the Default Authentication provider's database to a file.

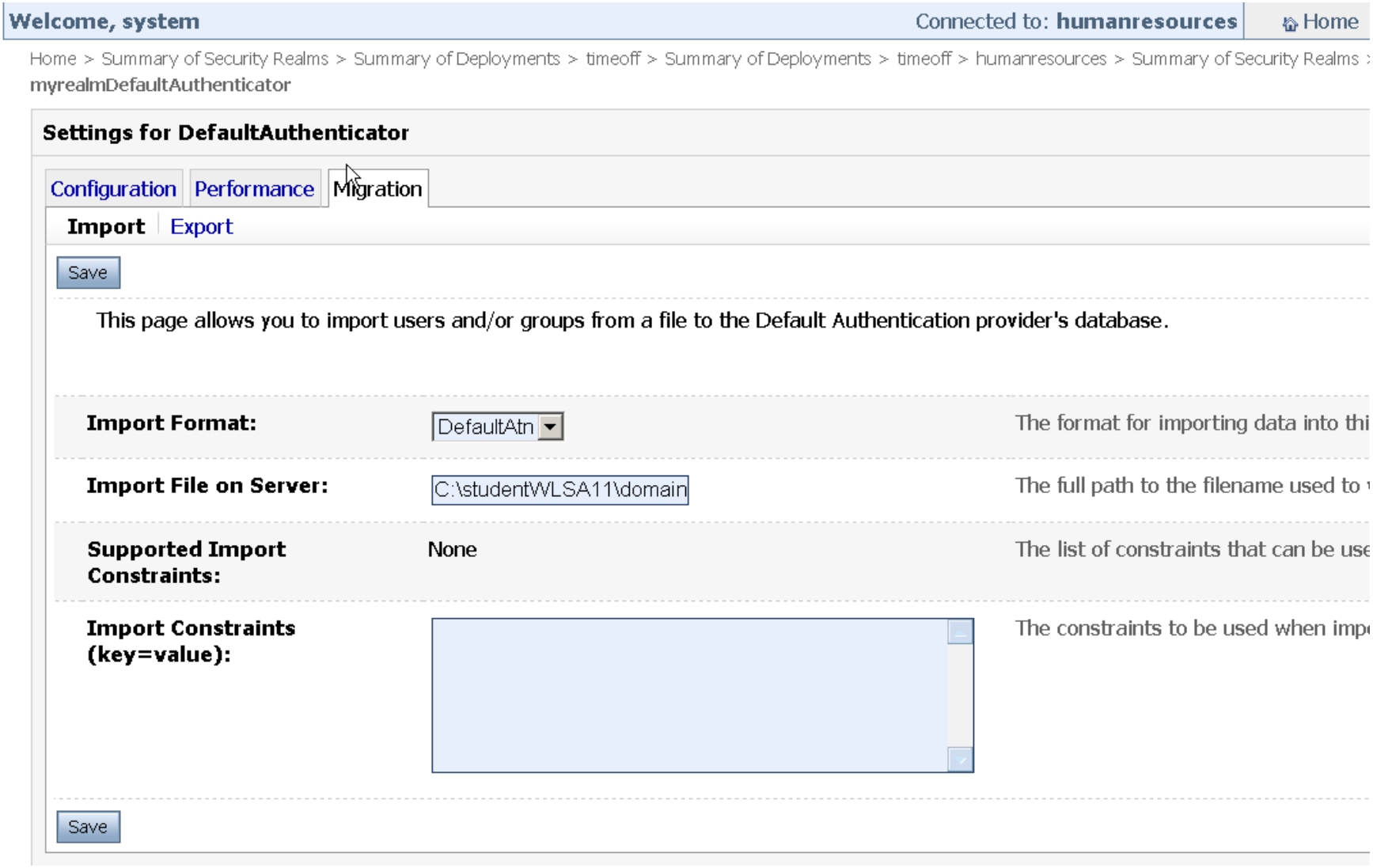
Export Format:	DefaultAtn	The format for exporting
Export File on Server:	C:\studentWLSA11\domain	The full path to the filena
Supported Export Constraints:	users, groups, passwords	The list of constraints th
Export Constraints (key=value):		The constraints to be us

Save

DefaultAuthenticator has had its data exported properly

Continue

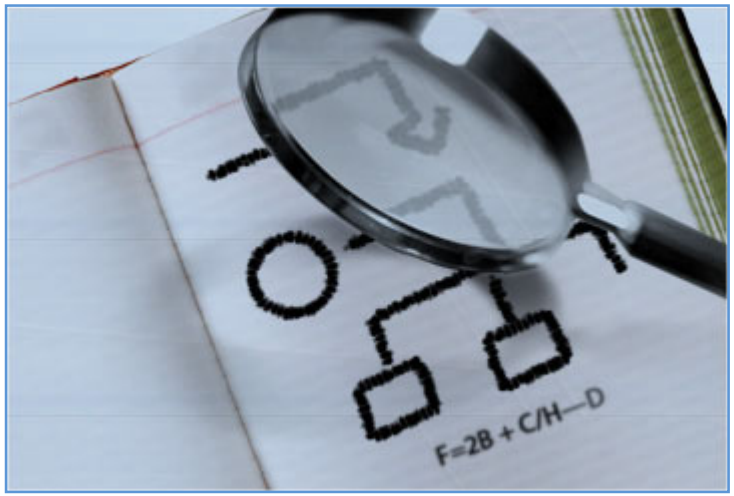
Importing into a different domain



Section Review

In this section we discussed:

- WLS embedded LDAP
- Security realms
- Configuring users, groups, and roles



Exercise

Managing Users and Groups

- In this lab you are going to configure groups and users.
- Ask the instructor for any clarification.
- The instructor will determine the stop time.

Lab Exercise

Road Map

1. WLS Security Architecture Overview
2. Users and Groups
3. **Protecting Application Resources**
   - Declarative Security
   - Protecting Web Applications
   - Defining Policies and Roles
4. Protecting Communications
5. Protecting Against Attacks

J2EE Declarative Security

- *Declarative security*:
  - Is a means to describe an application's access control in a form that is external to the application
  - Involves defining security roles and constraints on web application resources
  - Uses lazy authentication to protect application resources
  - Implemented in XML-based deployment descriptors
  - Applies to all types of application

Using Deployment Descriptors

- The security realm definition determines how the deployment descriptors will be used

Name:	TestRealm	The name of this security realm. More Info...
Check Roles and Policies:	Web applications and EJBs protected in DD	Specifies when this security realm should perform authorization checks on requests for access to Web applications and Enterprise JavaBeans (EJBs). More Info...
On Future Redeploys:	Initialize roles and policies from DD	Used with the Advanced security model. Specifies whether security data is copied from the deployment descriptors into the appropriate security provider databases each time the Web application or EJB is deployed. More Info...
☐ **Ignore Deploy Credential Mapping**		Specifies whether the Credential Mapping providers in this security realm will use only credential maps created using the Administration Console. By default, this box is unchecked, meaning that the Credential Mapping provider will load credential maps specified in a weblogic-ra.xml deployment descriptor. More Info...
Security Model:	DD Only	Specifies the default security deployment model for applications deployed in this security realm. Security models apply to applications containing EJBs or WARs. More Info...
☐ **Combined Role Mapping Enabled**		Specifies whether application role mappings are combined by the J2EE containers. If false the containers need internally defined mappings to use application role mappings. The setting is provided for backward compatibility with version (8.x) of WebLogic Server. For all applications initially deployed in version 9.x, the default value for this setting true (enabled). For all applications previously deployed in version 8.1 and upgraded to version 9.x, the default value is false (disabled). More Info...

Protecting Web Applications

- To protect a Web Application with declarative security:
  1. Define roles that should access the protected resources
  2. Determine Web Application resources that must be protected
  3. Map protected resource to roles that should access them
  4. Map roles to users/groups in the WLS security realm
  5. Set up an authentication mechanism

Define Security Roles

- Define types of users that exist in your Web Application.
- Use the `web.xml` deployment descriptor to define security roles.

Defining Roles:

```
<security-role>
        <role-name>Users</role-name>
</security-role>

<security-role>
        <role-name>Managers</role-name>
</security-role>
```

`<web.xml>`

Determine Protected Resources

- Web resources are defined based on URL patterns.
- URL patterns provide a flexible way to define a single resource or a group of resources.

Example URL Patterns:

URL Pattern	Role Name
`/*`	Some Role Name (i.e. director)
`/*.jsp`	“ “
`/EastCoast/*`	“ “

Map Roles to Resources…

- Apply security constraints to specified resources in your web application.
- Users must be authenticated when accessing resources by these URL patterns.

Configuring Security Constraints:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>East Coast Sales</web-resource-name>
            <url-pattern>/EastCoast/*</url-pattern>
      <http-method>POST</http-method>
      <http-method>GET</http-method>
  </web-resource-collection>
   ...
```

<web.xml>

…Map Roles to Resources

- Define which role(s) may access this collection of resources.

Configuring Security Constraints, Continued

```
<security-constraint>
   ...

   <auth-constraint>
       <role-name>Users</role-name>
       <role-name>Managers</role-name>
   </auth-constraint>
</security-constraint>
```

<web.xml>

Map Roles to Users in Realms

- Use `weblogic.xml` to map your web application roles to entities in the WebLogic security realm.
- Map to Groups or individual Users.

Assigning Roles:

```
<security-role-assignment>
        <role-name>Users</role-name>
        <principal-name>employees</principal-name>
</security-role-assignment>

<security-role-assignment>
        <role-name>Managers</role-name>
        <principal-name>mary</principal-name>
        <principal-name>paul</principal-name>
</security-role-assignment>
```

`<weblogic.xml>`

Setup Authentication

- Configure how a Web application determines users' security credentials:
  - `BASIC` – web browser displays a dialog box
  - `FORM` – use a custom HTML form
  - `CLIENT-CERT` – request a client certificate

Configuring Authentication:

```
<login-config>
   <auth-method>BASIC, FORM, or CLIENT-CERT</auth-method>
<form-login-config>
      <form-login-page>login.jsp</form-login-page>
      <form-error-page>badLogin.jsp</form-error-page>
   </form-login-config>
</login-config>
```

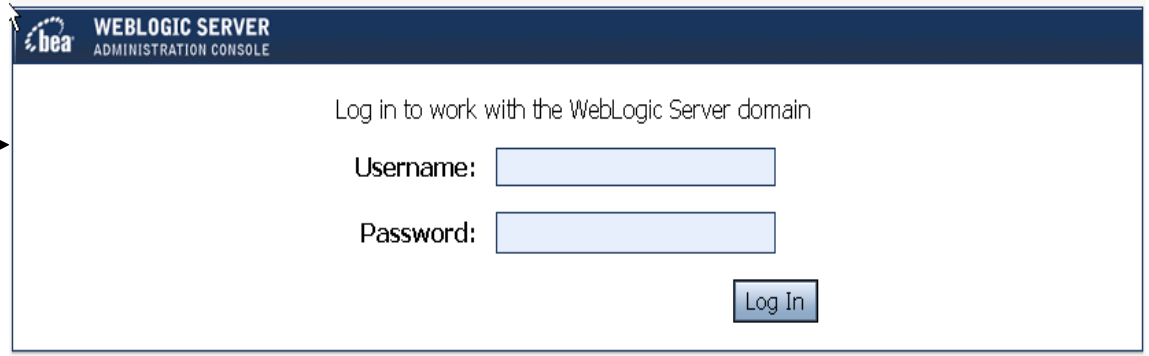
For type FORM only

`<web.xml>`

Authentication Examples

BASIC Authentication

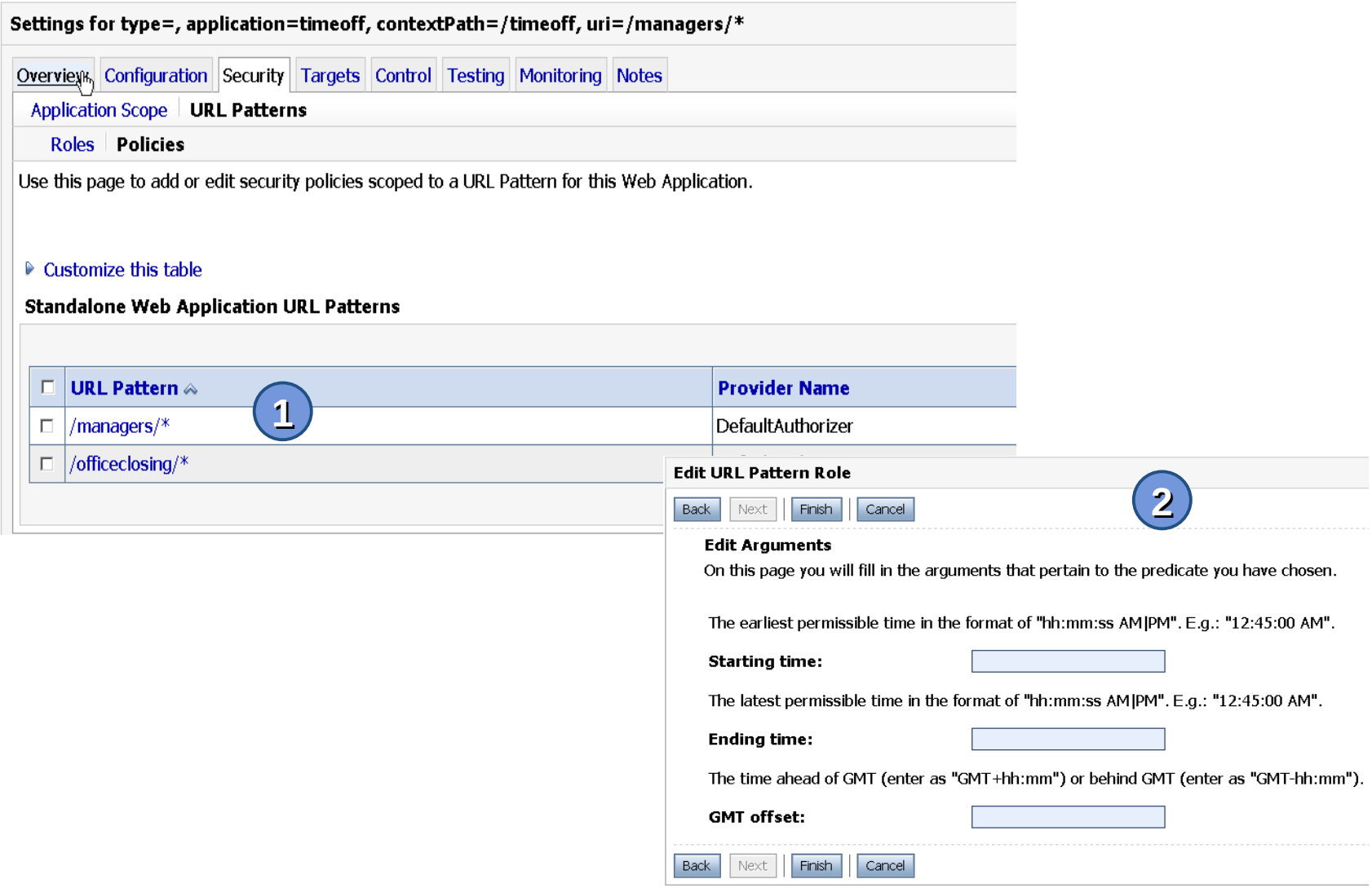
FORM based Authentication



Policies and Roles

- Security roles:
  - Are an abstraction of users and groups
  - Can be determined dynamically for different resources
- Security policies:
  - Are rules/conditions that users/groups must adhere to, to be granted/denied authorization
  - Implement parameterized authorization

Defining Policies and Roles for Web Resources



Defining Policies and Roles for Other Resources…

- You can define roles and policies on other resources, e.g. JDBC, JMS and more….

…Defining Policies and Roles for Other Resources

1

Policy Conditions

These are the conditions that determine the access control to your JDBC data source.

Add Conditions | Combine | Uncombine | Move Up | Move Down | Remove | Negate

No Policy Specified

Add Conditions

Save

2

Settings for examples-demo

Configuration | Targets | Monitoring | Control | Security

Roles | **Policies** | Credential Mappings

Back | Next | Finish | Cancel

Choose a Predicate

Choose the predicate you wish to use as your new

The predicate list is a list of available predicates

Predicate List: Access occur

Back | Next | Finish | Cancel

3

Settings for examples-demo

Configuration | Targets | Monitoring | Control | Security | Notes

Roles | **Policies** | Credential Mappings

Back | Next | Finish | Cancel

Edit Arguments

On this page you will fill in the arguments that pertain to the pre

The earliest permissible time in the format of "hh:mm:ss AM|PM

Starting time:

The latest permissible time in the format of "hh:mm:ss AM|PM"

Ending time:

The time ahead of GMT (enter as "GMT+hh:mm") or behind GM

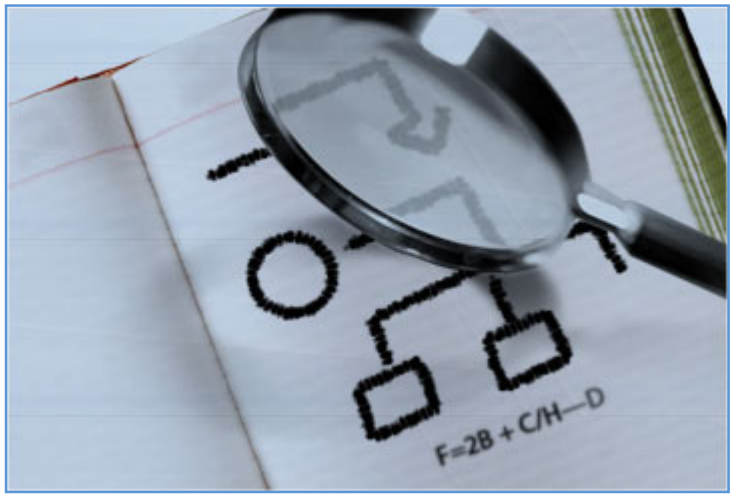
GMT offset:

Back | Next | Finish | Cancel

Section Review

In this section we discussed:

- Protecting Web Applications with declarative security
- Defining policies and roles



Exercise

Securing Web Applications

- In this lab you are going to secure a Web Application.
- Ask the instructor for any clarification.
- The instructor will determine the stop time.

Lab Exercise

Exercise

Configuring Additional Conditions

- In this lab you are going to configure additional conditions for securing web applications.
- Ask the instructor for any clarification.
- The instructor will determine the stop time.

Lab Exercise

Road Map

1. WLS Security Architecture Overview
2. Users and Groups
3. Protecting Application Resources
4. **Protecting Communications**
   - What Is Secure Socket Layer (SSL)
   - Configuring SSL
   - Certificates
   - Keytool
5. Protecting Against Attacks

What Is SSL?

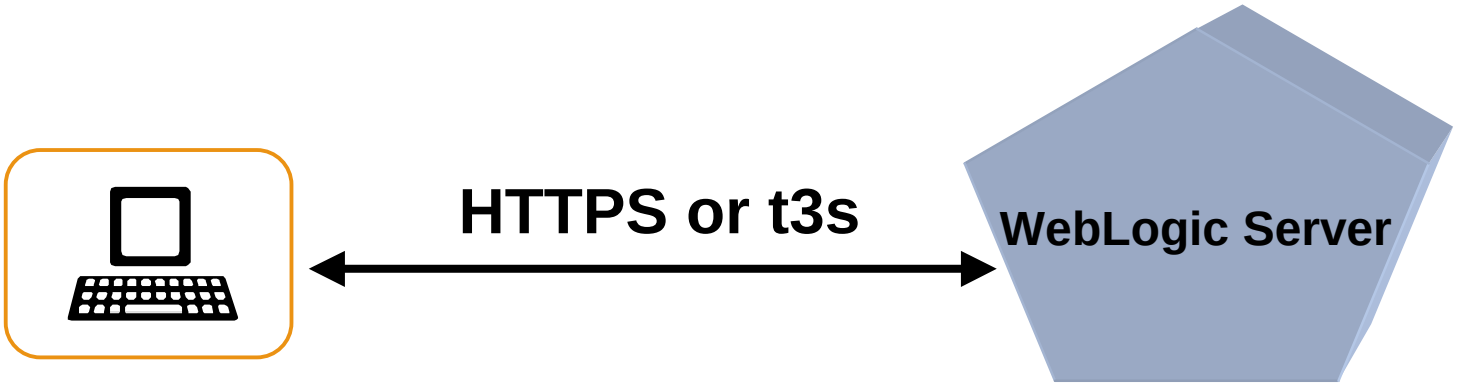
- Secure Socket Layer (SSL) is a protocol that enables:
  - Connection security through encryption
  - A server to authenticate to a client
  - A client to authenticate to a server (optional)
  - Data integrity such that data that flows between a client and server is protected from tampering by a third party

Trust and Identity

- SSL and keystore are configured independently.
- For the purpose of backward compatibility, this release of WebLogic Server supports private keys and trusted CA certificates stored in files, or in the WebLogic Keystore provider
- Identity
  - Private Key and Digital Certificate (can now be looked up directly from the keystore, not necessarily as a standalone file outside the keystore)
- Trust
  - Certificates of Trusted Certificate Authorities

Using a SSL Connection

- SSL is used by WLS to secure HTTP and t3 communication.
- To use SSL, clients access WLS via the https or t3s protocols.
  - `https://localhost:7002/orderStock`
  - `t3s://localhost:7002/useCreditCard`



Enabling Secure Communication

- With SSL data is encrypted using a negotiated symmetric session key.
- A public key algorithm is used to negotiate the symmetric session key.
- In SSL digital certificates are used to provide a trusted public key.

WebLogic Server SSL Requirements

- To enable WebLogic Server SSL you must:
  - Obtain an appropriate digital certificate
  - Install the certificate
  - Configure SSL properties
  - Configure two-way authentication(if desired)
  - SSL impacts performance

The `keytool` Utility

- `keytool` is a standard J2SE SDK utility for managing:
  - Generation of *private keys* and corresponding *digital certificates*
  - *Keystores* (databases) of private keys and associated certificates
- The `keytool` utility can display certificate and keystore contents.
- For documentation, see:

  `http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html`

  `http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html`

Obtaining a Digital Certificate: `keytool` Examples

Generate a new self-signed digital certificate:

```
keytool –genkey –alias dwkey –keyalg RSA –keysize 512
    -keystore dw_identity.jks
```

Generate a CSR:

```
keytool –certreq -v –alias dwkey –file dw_cert_request.pem
    -keypass dwkeypass -keystore dw_identity.jks
    –storepass dwstorepass
```

Import a signed certificate reply from a CA:

```
keytool –import –alias dwkey –file dw_cert_reply.pem
    -keypass dwkeypass -keystore dw_identity.jks
    –storepass dwstorepass
```

Configuring SSL for a WebLogic Server

Summary of Servers

A server is an instance of WebLogic Server that runs in its own Java Virtual Machine (JVM) and has its own configuration.

This page summarizes each server that has been configured in the current WebLogic Server domain.

Customize this table

Servers

New | Clone | Delete

Name
adminserver(admin
mainserver

New | Clone | Delete

1

2

Settings for mainserver

Configuration | Protocols | Logging | Debug | Monitoring | Control | Deployments | Services | Security | Notes

General | Cluster | Services | Keystores | SSL | Deployment | Migration | Tuning | Overload | Health Monitoring | Server Start

Save

This page lets you view and define various Secure Sockets Layer (SSL) settings for this server instance. These settings help you to r transmissions.

Identity and Trust Locations:	Keystores	Indicates where SSL should find the server's identi the server's trust (trusted CAs). More Info...
Identity		
Private Key Location:	from Custom Identity Keystore	The keystore attribute that defines the location of
Private Key Alias:	dwkey	The keystore attribute that defines the string alias private key. More Info...
Private Key Passphrase:	••••••••••••••••••	The keystore attribute that defines the passphrase More Info...
Certificate Location:	from Custom Identity Keystore	The keystore attribute that defines the location of
Trust		
Trusted Certificate Authorities:	from Java Standard Trust Keystore	The keystore attribute that defines the location of

Advanced

Discussions not available on http://localhost:7011/

Configuring Keystores

Configuration | Protocols | Logging | Debug | Monitoring | Control | Deployments | Services | Security | Notes

General | Cluster | Services | **Keystores** | SSL | Deployment | Migration | Tuning | Overload | Health Monitoring | Server Start

Save

Keystores ensure the secure storage and management of private keys and trusted certificate authorities (CAs). This page lets you view configurations. These settings help you to manage the security of message transmissions.

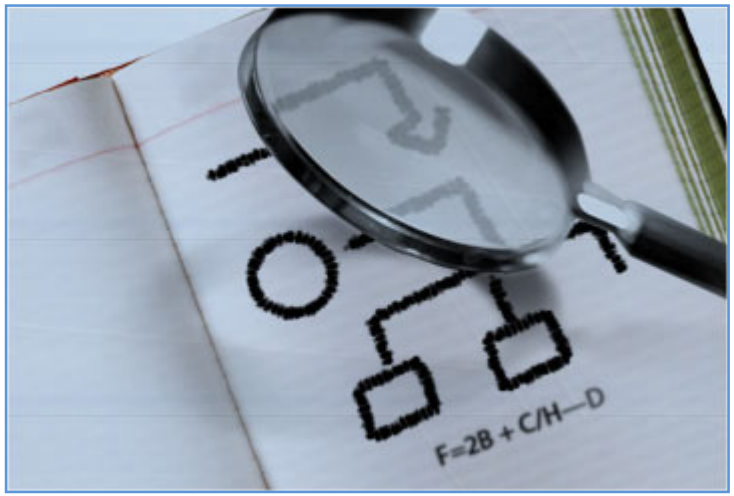
Keystores:	Custom Identity and Java Standard Trust	Which configuration rules should be used for finding th More Info...
Identity		
Custom Identity Keystore:	dw_identity.jks	The path and file name of the identity keystore. More
Custom Identity Keystore Type:	JKS	The type of the keystore. Generally, this is JKS. More
Custom Identity Keystore Passphrase:	••••••••••••••••••	The encrypted custom identity keystore's passphrase. be opened without a passphrase. More Info...
Trust		
Java Standard Trust Keystore:	JAVA_HOME\jre\lib\security\cacerts	The path and file name of the trust keystore. More Inf
Java Standard Trust Keystore Type:	jks	The type of the keystore. Generally, this is JKS. More
Java Standard Trust Keystore Passphrase:	••••••••••••••••••	The password for the Java Standard Trust keystore. Th keystore is created. More Info...

Save

Section Review

In this section we discussed:

- Secure Socket Layer
- Configuring SSL
- Creating certificates
- Managing certificates with `keytool`



Exercise

Configuring SSL

- In this lab you are going to configure Secure Socket Layer.
- Ask the instructor for any clarification.
- The instructor will determine the stop time.

Lab Exercise

Road Map

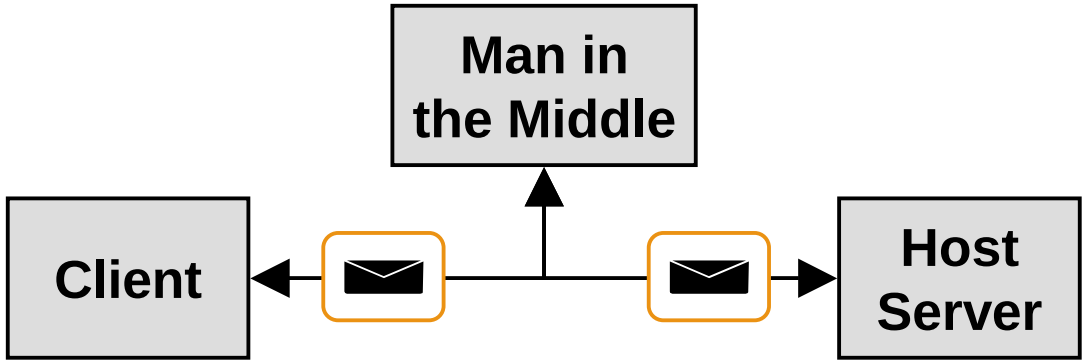
1. WLS Security Architecture Overview
2. Users and Groups
3. Protecting Application Resources
4. Protecting Communications
5. **Protecting Against Attacks**
   - Types of Attacks
   - Protecting Against Man in the Middle Attacks
   - Protecting Against Denial of Service Attacks
   - Protecting Against Large Buffer Attacks
   - Protecting Against Connection Starvation

Protecting Against Attacks

- WLS can help protect applications against several attacks:
  - Man in the middle attacks
  - Denial of service (DoS) attacks
  - Large buffer attacks
  - Connection starvation attacks
- The set of slides that follow detail countermeasures WLS provides to address these attacks.

Man in the Middle Attacks

- In a *man in the middle* attack, a third party poses as a destination host intercepting messages between client and the real host.
- Instead of issuing the real destination host's SSL certificate, the attacker issues his own hoping that client accepts it as being from the real destination host.



Man in the Middle Countermeasures

- Man in the Middle attacks can be resisted by using a host name verifier.
- A *Host Name Verifier* validates that the host to which an SSL connection is made is the intended or authorized party.
- WLS provides a Host Name Verifier by default.
- A custom Host Name Verifier can be created by implementing the interface

  `weblogic.security.SSL.HostnameVerifier`

Denial of Service Attacks

- DoS attacks are attempts by attackers to prevent legitimate users of a service from using that service.
- There are three basic types of attack:
  - Consumption of scarce, limited, or non-renewable resources
  - Destruction or alteration of configuration information
  - Physical destruction or alteration of network components

DoS Countermeasures

- Harden WLS against Denial of Service attacks by:
  - Filtering incoming network connections
  - Configuring consumable WLS resources with appropriate threshold and quotas
  - Limiting access to configuration information and backing up configuration files
  - Preventing unauthorized access by protecting passwords against password guessing attacks

Filtering Network Connections

- WLS can be configured to accept or deny network connections based on the origin of the client.
- This feature can be used to:
  - Restrict the location from which connections to WLS are made
  - Restrict the type of connection made, i.e., only allow SSL connections and reject all others
- To filter network connections, create a class that implements the `ConnectionFilter` interface and install it using the Administration Console.

Connection Filter

Settings for humanresources

Configuration | Monitoring | Control | Security | WebService Security | Notes

General | **Filter** | Unlock User | Embedded LDAP | Roles | Policies

Save

This page allows you to define connection filter settings for this WebLogic Server domain.

☐ **Connection Logger Enabled**

Specifies whether this W
connections. More Info..

Connection Filter:

The name of the Java cla
weblogic.security.net.Co
no connection filter will b

Connection Filter Rules:

```
192.168.0.0/16 127.0.0.1 8002 deny http
```

The rules used by any co
ConnectionFilterRulesList
and when no rules are sp
implementation rules are
protocols. More Info...

Save

Consuming WLS Resources

- Denial of Service can come from consuming server side resources used by Web Applications:
  - Intentionally generating errors that will be logged consuming disk space
  - Sending large messages, many messages, or delaying delivery of messages in an effort to cripple JMS
  - Disrupting network connectivity by "connection starvation"
  - Consuming system memory through "large buffer attacks"
- The effect of these attacks can be reduced by setting appropriate quotas and threshold values.

Large Buffer Attacks…

- Individuals can try and take down a Web site by sending a large buffer of data, which starves the system of memory.
- Administrators can combat this attack by setting a threshold for incoming data.

…Large Buffer Attacks

Settings for mainserver

Configuration | Protocols | Logging | Debug | Monitoring | Control | Deployments | Services | Security | Notes

General | **HTTP** | jCOM | IIOP | Channels

Save

Web-based clients communicate with WebLogic Server using HTTP (HyperText Transfer Protocol).

Use this page to define the HTTP settings for this server.

Default WebApp Context Root:		Returns the original context-root for the default Web applicatic use the context-root attributes in application.xml or weblogic.x context-root for a default Web application is /. If "" (empty str More Info…
Post Timeout:	30	The amount of time this server waits between receiving chunk out. (This is used to prevent denial-of-service attacks that atte More Info…
Max Post Size:	-1	The maximum post size this server allows for reading HTTP PC 0 indicates an unlimited size. More Info…
☑ **Enable Keepalives**		Indicates whether there should be a persistent connection to t of your Web applications.) More Info…
Duration:	30	The amount of time this server waits before closing an inactive
HTTPS Duration:	60	The amount of time this server waits before closing an inactive

Connection Starvation…

- Individuals can try and take down a Web site by sending small, incomplete messages that cause the server to wait.
- Administrators can combat this attack by setting a threshold.
- Connections time out while waiting for the remainder of the data if they have reached the threshold set by the administrator.

…Connection Starvation

Save

Web-based clients communicate with WebLogic Server using HTTP (HyperText Transfer Protocol).

Use this page to define the HTTP settings for this server.

Default WebApp Context Root:		Returns the original context-root for the default Web application for this Web server. Alternatively, you can use the context-root attributes in application.xml or weblogic.xml to set a default Web application. The context-root for a default Web application is /. If "" (empty string) is specified, the Web server defaults to /. More Info...
Post Timeout:	30	The amount of time this server waits between receiving chunks of data in an HTTP POST data before it times out. (This is used to prevent denial-of-service attacks that attempt to overload the server with POST data.) More Info...
Max Post Size:	-1	The maximum post size this server allows for reading HTTP POST data in a servlet request. A value less than 0 indicates an unlimited size. More Info...
☑ **Enable Keepalives**		Indicates whether there should be a persistent connection to this server. (This may improve the performance of your Web applications.) More Info...

User Lockout

- Individuals attempt to hack into a computer using various combinations of usernames and passwords
- Administrators can protect against this security attack by setting the lockout attributes
- Administrator has an option to unlock a locked user through the console

Configuring User Lockout

Settings for myrealm

Configuration | Users and Groups | Roles and Policies | Credential Mappings | Providers | Migration

General | **User Lockout** | Performance

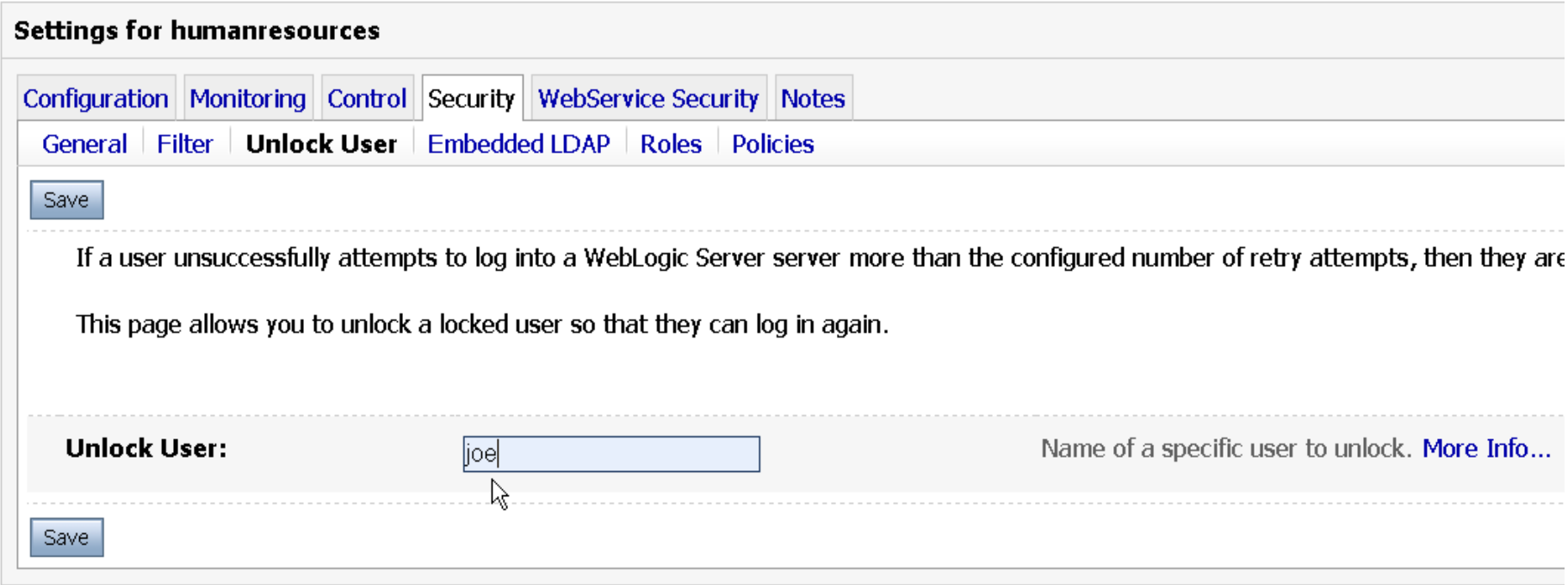
Save

Password guessing is a common type of security attack. In this type of attack, a hacker attempts to log in to a c
passwords. Weblogic Server provides a set of attributes to protect user accounts from intruders. This page allov
security realm.

☑ **Lockout Enabled**		Specifies whether the server locks users out wh Info...
Lockout Threshold:	5	The maximum number of consecutive invalid log out. More Info...
Lockout Duration:	30	The amount of time that a user's account is locl
Lockout Reset Duration:	5	The amount of time within which consecutive in More Info...
Lockout Cache Size:	5	The number of invalid login records (between 0
Lockout GC Threshold:	400	The maximum number of invalid login records th

Save

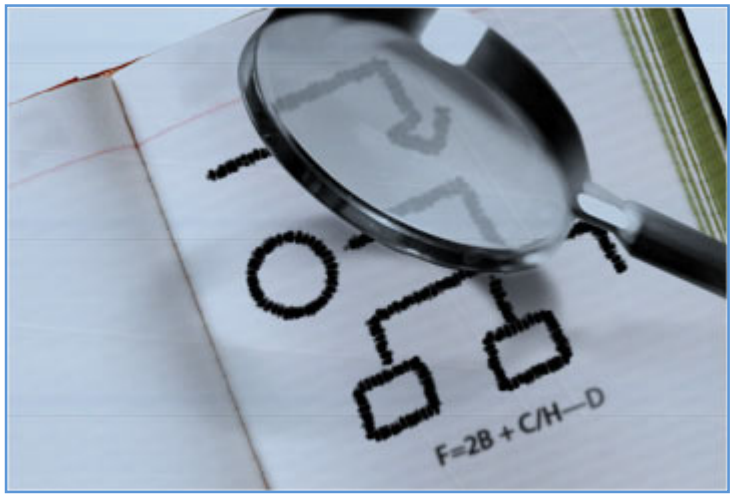
Unlocking Users



Section Review

In this section we discussed:

- Types of attacks
- Protecting WLS against man in the middle attacks
- Protecting WLS against denial of service attacks
- Protecting WLS against large buffer attacks
- Protecting WLS against connection starvation attacks



Protecting Against Attacks

- In this lab you are going to configure WLS features for hardening against password attacks.
- Ask the instructor for any clarification.
- The instructor will determine the stop time.

Lab Exercise

Module Review

- In this module we discussed:
  - WLS security architecture
  - Configuring users, groups, and roles
  - Configuring security realms
  - Securing Web Applications with declarative security
  - Configuring policies and SSL
  - Creating and managing certificates
  - Protecting WLS against several types of attacks