Security Patterns revealed

Mike Wiesner
Senior Consultant
SpringSource
Speaker’s qualifications

- Mike Wiesner
- is a Senior Consultant for SpringSource Germany
- is specializing in Application Security and SSO
- thinks also about good software design (not only secure design)

Overall Presentation Goal

How to use Security Patterns
What's the problem?

- XSS: 18.5%
- SQL-Injection: 13.6%
- Directory Traversal: 4.5%
- Information Leak: 3.1%
- DoS Malformed Input: 2.0%
- Int Overflow: 1.2%
- Permissions: 1.1%
- Webroot: 1.0%
- Authentication: 0.9%

What's the problem?

- Validation: 34.1%
- Directory Traversal: 4.5%
- Information Leak: 3.1%
- Int Overflow: 1.2%
- Permissions: 1.1%
- Webroot: 1.0%
- Authentication: 0.9%
Agenda

- Security Patterns?
- Validation
- Access control
- Data driven security

Security Patterns?

- Possible way to solve a specific problem
- Explain the problems
- Explain the tradeoffs
- Are proven solutions
- Are generic (independent of implementation technology)
Read it and you have it?

- No concrete (proven) implementation
- No use of frameworks and libraries
- No use of modern technology
- There are plenty of them out there ;-)
Agenda

- Security Patterns?
- Validation
- Access control
- Data driven security

Unchecked input

- Cause of many attacks
- Unchecked to the back-end
  - SQL
  - LDAP
  - XPath
- Unchecked to the client
  - Cross-site scripting (XSS)
**SQL Injection**

```sql
select * from users
where user = 'user'
and password = "' or '1' = '1"
```

**JDBC Prepared Statement?**

- Dangerous characters should be encoded there
- But that's not part of the spec!
- And it's most likely a black list
  - Different character sets
  - Different versions
  - Bugs
- Just Outsourcing, no real solution!
Directory Traversal

Solutions

- Blacklist :-(
- Encoding :-|
- Whitelist :-) and encode the rest
- No input ;-) Through preselection
And how do we do that?

**Intercepting Validator**

- Client
  - invokes
  - SecureBaseAction
    - validates
  - Target
    - invokes
    - Validator
      - validates
      - InterceptingValidator
        - creates
Trust Zones

Don't trust

Client

BBI

Trust

Server

JSR-303: Bean Validation

public class Address {

    @NotNull @Length(max=30)
    private String addressline1;

    @Length(max=30)
    private String addressline2;

}
JSR 303: Bean Validation

- Validation information inside of the domain object
- No vendor lock-in
- No SecureBaseAction
  - Use AOP as InterceptingValidator

Agenda

- Security Patterns?
- Validation
- Access control
- Data driven security
Check Point Code

SecurityManager sm = System.getSecurityManager();
if (sm.isUserInRole("ROLE_USER") != true) {
    throw new AccessDeniedException("No Access")
}
// do something

How about this?

@Secured("PERM_DELETE_USER")
public void deleteUser(User user);

@Secured("PERM_SHOW_USER")
public User showUser(long id);
Reference Monitor with AOP

And there’s a lot more

- Role-Based Access Control
- Role Rights Definition
- Controlled Object Monitor
- Multilevel Security
- Security Session
Agenda

- Security Patterns?
- Validation
- Access control
- Data driven security

Data Driven Authorization

Service

1. show(1)
2. show(2)
XACML

- Security Policy Management Standard from OASIS
- Policies in XML
- Extensible
- Flexible
- Independent of programming language
- Java implementations
  - Sun's XACML Toolkit
  - HERAS AF (www.herasaf.org)
Login between 9am and 5pm

<Apply FunctionId="time-greater-than-or-equal">
   <Apply FunctionId="function:time-one-and-only">
      <EnvironmentAttributeSelector AttributeId="current-time"/>
   </Apply>
   <AttributeValue>09:00:00</AttributeValue>
</Apply>

<Apply FunctionId="time-less-than-or-equal">
   <Apply FunctionId="time-one-and-only">
      <EnvironmentAttributeSelector AttributeId="current-time"/>
   </Apply>
   <AttributeValue>17:00:00</AttributeValue>
</Apply>

But that’s not all ...

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
      <Apply FunctionId="function:time-one-and-only">
         <EnvironmentAttributeSelector
            DataType="http://www.w3.org/2001/XMLSchema#time"
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
      </Apply>
      <AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#time">
         09:00:00
      </AttributeValue>
   </Apply>
</Condition>

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
      <EnvironmentAttributeSelector
         DataType="http://www.w3.org/2001/XMLSchema#time"
         AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
   </Apply>
   <AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#time">
      17:00:00
   </AttributeValue>
</Condition>
Pros / Cons

Pros

- Extensible / Flexible (even during Runtime)
- Language agnostic
- Report generation via XSLT

Cons

- Very detailed syntax
- High costs during runtime
- New language to learn

And now the whole XML ...

```xml
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:context">
  <Target />
  <Condition />
  <Rule RuleId="FinalRule" Effect="Deny" />
</Policy>
```
Policies in Java

```java
public class MyRule implements DomainRule<HolidayRequest>{

    public ACCESS decide(Authentication auth, HolidayRequest hr) {
        if (auth.getName().equals(hr.getRequester())) {
            return ACCESS.GRANTED;
        }
        return ACCESS.DENIED;
    }
}
```

Pros/Cons

**Pros**
- Easy to understand
- Same language as the rest of the codebase
- Extensible/Flexible
- Changes during Runtime with Groovy possible

**Cons**
- Language dependent
- Report generation is more difficult
Conclusion

Security Patterns are a good starting point.

But don‘t trust them blindly!

Q&A

Mike Wiesner
SpringSource Germany

mike.wiesner@springsource.com
Skype: mikewiesner

http://www.springsource.com/de
http://www.mwiesner.com
Thanks for your attention!