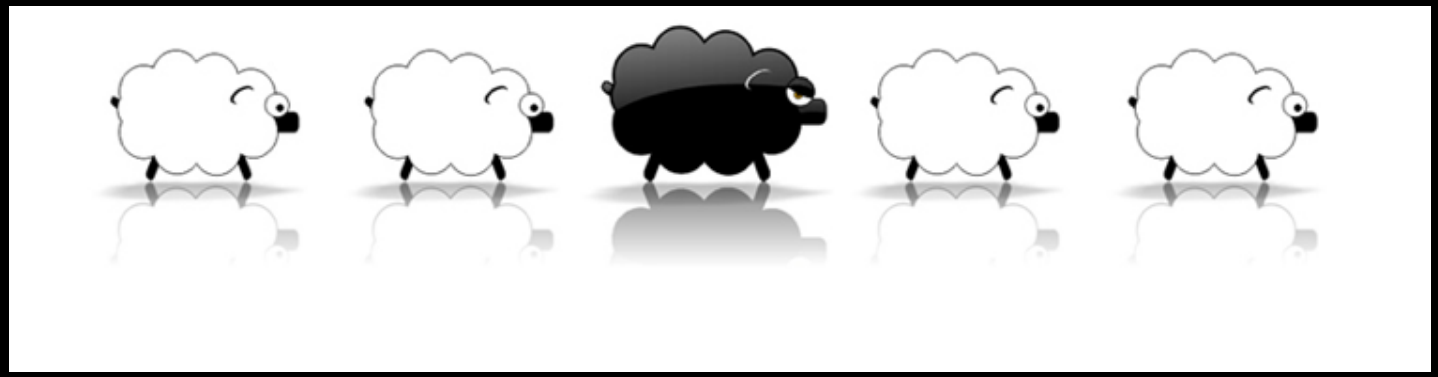
March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

BEYOND SCANNING:

AUTOMATED WEB APPLICATION SECURITY TESTING

Stephen de Vries

- Automated Software Testing
  - Unit Testing
  - Integration Testing
  - Security Testing (Security Assessment)



Tightly integrated:

- Unit and Integration tests
- Developed alongside the code
- The tests are the specification
- Test coverage is measurable
- Automated...
- ...fast, reliable regression tests
- ...integrated with continuous builds

Loosely integrated:

- Security Testing (and QA)
- Independent of the specification
- Undocumented tests
- Partially automated...
- ...not repeatable
- ...no reliable regression tests

How?

Behaviour Driven Development

Page Objects

Burp as Scanner

Example Integration Test

```
public void login(String username,String password) {
        driver.get("http://www.ispatula.com:8081/ispatula/shop/signonForm.do");
        driver.findElement(By.name("username")).sendKeys(username);
        driver.findElement(By.name("password")).sendKeys(password);
        driver.findElement(By.name("update")).click();
}

@Test
public void testLoginWorks() {
        login("bob","password");
        assertTrue(driver.getPageSource().contains("Welcome"));
}

@Test
public void testLoginFailsForWrongPassword() {
        login("bob","balls");
        assertFalse(driver.getPageSource().contains("Welcome"));
}
```

Example Integration Test

```
@Test
public void testLoginBypassWithSQLinjection() {
    login("bob';--",””);
    assertFalse(driver.getPageSource().contains("Welcome"));
}
```

Example Integration Tests

- `testThatLoginWorks()`
- `testThatLoginWithWrongPasswordFails()`
- `testThatLogoutWorks()`
- `testAddItemToShoppingCart()`
- `testCheckoutWithEmptyCart()`
- `testCheckoutWithFullCart()`

Tests = Functional Specification

Bevahiour Driven Development is a communication tool

Behaviour Driven Development

- Let business stakeholders define the specifications
- Write specifications (stories) in English
- Let the developer wire the spec to code

BDD with JBehave

```
@Test
public void testLoginBypassWithSQLinjection() {
    login("bob';--",””);
    assertTrue(driver.getPageSource().contains(”Invalid"));
}
```

```
Scenario: Test login can’t be bypassed using SQL injection

Given the login page
When the user logs in with username: bob’;-- and password: blah
Then the word: Invalid should be on the page
```

BDD with JBehave

```
@Given("the login page")
public void gotoLoginPage() {
    driver.get("http://www.ispatula.com:8081/ispatula/shop/signonForm.do");
}

@When("the user logs in with username: $username and password: $password")
public void login(String username,String password) {
    driver.findElement(By.name("username")).sendKeys(username);
    driver.findElement(By.name("password")).sendKeys(password);
    driver.findElement(By.name("update")).click();
}

@Then("the word: $theWord should be on the page")
public void findWordInPage(String theWord) {
    assertThat(driver.getPageSource(),containsString(theWord));
}
```

BDD with JBehave

```
Scenario: Test the login works with valid credentials

Given the login page
When the user logs in with username: bob and password: password
Then the word: Welcome should be on the page


Scenario: Test login fails with wrong password

Given the login page
When the user logs in with username: bob and password: blah
Then the word: Invalid should be on the page


Scenario: Test login can't be bypassed using SQL injection

Given the login page
When the user logs in with username: bob';-- and password: blah
Then the word: Invalid should be on the page
```

BDD with JBehave

```
Scenario: Test login with valid and invalid data

Given the login page
When the user logs in with <username> and <password>
Then the <matchWord> should be on the page

Examples:
|username   |password    |matchWord  |
|bob        |password    |Welcome    |
|bob        |blah        |Invalid    |
|bob';--    |blah        |Invalid    |
```

Demo: JBehave

BDD with JBehave

```
@When("the user logs in with username: $username and password: $password")
public void login(String username,String password) {
        driver.findElement(By.name("username")).sendKeys(username);
        driver.findElement(By.name("password")).sendKeys(password);
        driver.findElement(By.name("update")).click();
}
```

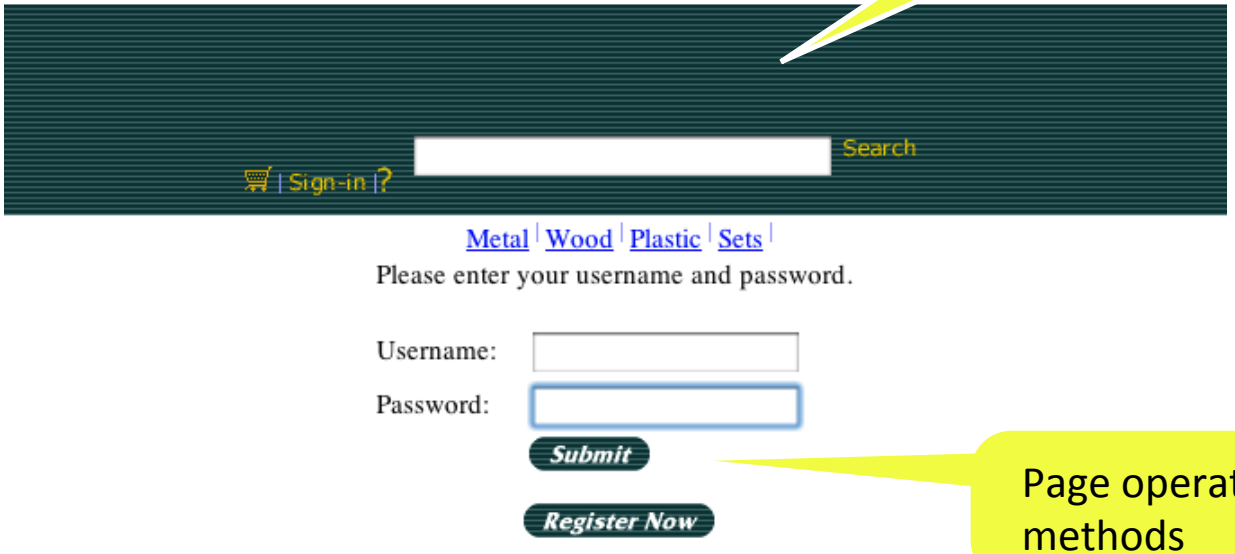
Implementation details leak into the tests

- Tests are brittle
- Difficult to maintain

Page Object Pattern

Page Object Pattern

– **Object Oriented Design for web pages**



Page itself is object

Page operations are methods

Page Object Pattern

```
@When("the user logs in with username: $username and password: $password")
public void login(String username,String password) {

                loginPage.login(username,password);
}
```

Page Object Pattern

```
public class LoginPage extends Page {
        String url=“http://www.ispatula.com/ispatula/shop/signonForm.do";

        WebElement username;
        WebElement password;
        WebElement update;

        public LoginPage(WebDriver driver) {
                super(url,driver);
        }

        public void login(String usernameParam,String passwordParam) {
                username.sendKeys(usernameParam);
                password.sendKeys(passwordParam);
                update.click();
        }
}
```

BDD = Convenient way to write automated security specifications

Page Objects = Modular way to navigate a web application

Introducing BDD-Security

blackhat EUROPE
March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

BDD-Security

- Security Specification Templates for Web Applications
- Specifications are understandable
- No need to modify default specifications between applications
- Framework to write custom specifications
- Integrates with build tools (Maven)

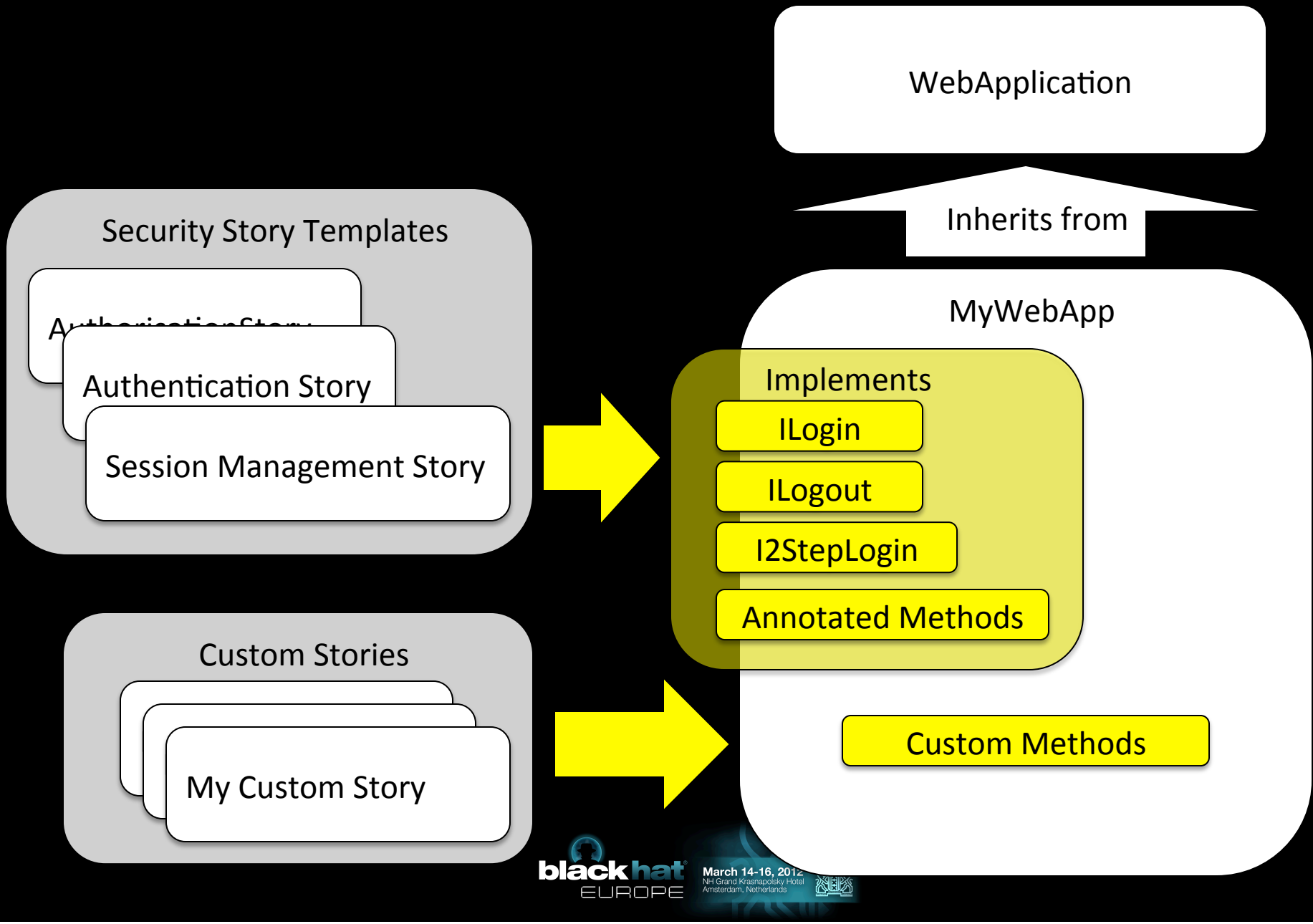
BDD-Security: Authentication Story

- Passwords should be case sensitive
- The user account should be locked out after 4 incorrect authentication attempts
- Login should be secure against SQL injection bypass attacks in the password field
- Login should be secure against SQL injection bypass attacks in the username field
- The login form itself should be served over SSL to reduce the risk of Phishing attacks
- The authentication credentials should be sent over SSL
- When authentication credentials are sent to the server, it should respond with a 3xx status code

BDD-Security: Session Management Story

- When the user logs out then the session should no longer be valid
- Sessions should timeout after a period of inactivity
- The session ID should be changed after authentication
- The session cookie should have the secure flag set
- The session cookie should have the httpOnly flag set

BDD-Security: Overview



Getting started with BDD-Security

```
public class IspatulaApplication extends WebApplication implements ILogin, ILogout {

        public LoginPage openLoginPage() {
            return new LoginPage(driver).open();
        }

        public Page login(Credentials credentials) {
            return openLoginPage().login(credentials);
        }

        public Page logout() {
            return new LogoutPage(driver).open().logout();
        }


        public boolean isLoggedIn(String role) {
            try {
                new AccountInfoPage(driver).open();
                return true;
            } catch (Exception e) {
                return false;
            }
        }

}
```

Getting started with BDD-Security

```
<web-app>
        <defaultDriver>HtmlUnit</defaultDriver>
        <baseUrl>http://www.ispatula.com:8081/ispatula/</baseUrl>
        <class>IspatulaApplication</class>

        <storyUrl>src/main/stories/</storyUrl>
        <reportsDir>reports</reportsDir>
        <latestReportsDir>target/jbehave/</latestReportsDir>

        <sessionIds>
                <name>JSESSIONID</name>
        </sessionIds>

        <users>
                <user username="bob" password="password"/>
                <user username="alice" password="password"/>
                <user username="admin" password="password"/>
        </users>

</web-app>
```

Demo: BDD-Security

BDD-Security: Authorisation

```
public class IspatulaApplication extends WebApplication implements ILogin,ILogout
{

    ...

    @Roles({"admin"})
    public ListOrdersPage listAllOrders() {

         return new ListOrdersPage(driver).open();

    }

}
```

BDD-Security: Authorisation

```
<web-app>
          ...
                    <user username="bob" password="password">
                         <role> user </role>
                    </user>

                    <user username="alice" password="password">
                         <role> user </role>
                    </user>

                    <user username="admin" password="password">
                         <role> user </role>
                         <role> admin </role>
                    </user>
          …
</web-app>
```

BDD-Security: Authorisation

```
Scenario: Verify that only authorised users can view
restricted resources

Given the login page
And the username <username>
And the password <password>
When the user logs in
Then they should not be able to access the restricted
resource <method>

Examples:
unauthorised.resources.table
```

What about scanning ?

Resty-Burp

- REST/JSON interface to Burp
- Released with Java client
- GNU Affero license
- Currently supports:
  - Scan $target
  - getPercentComplete
  - getIssues
  - getProxyHistory
  - get/set/update Config
  - reset

Demo: Resty-Burp

BDD-Security

Automated Scanning Story

- Passively scan for vulnerabilities while the application is navigated
- Scan for SQL injection vulnerabilities
- Scan for Cross Site Scripting vulnerabilities
- Scan for Command Injection vulnerabilities
- Scan for LDAP Injection vulnerabilities
- Scan for XML-SOAP Injection vulnerabilities
- Scan for Miscellaneous header and server vulnerabilities

BDD-Security

```
public class IspatulaApplication extends WebApplication implements IScanWorkflow,
ILogin, ILogout {

   ...

    public void navigateAll() {
        search("hello");
        openLoginPage().login(Config.instance().
                getUsers().getDefaultCredentials("user"));
        new AccountInfoPage(driver).open();
    }

   ...
}
```

Demo: The Whole Shebang

Summary

- BDD for Security Testing:
  - Understandable Security Specifications
  - Automated Regression testing
  - Templated security tests for common web app features
- Page Object Pattern:
  - Re-usable navigation objects (Integration, performance testing)
  - Tests are separated from page navigation = Maintainability
- Resty-Burp
  - HTTP/JSON interface to Burp Suite
  - Automated security scanning integrated with functional testing

Questions?

www.continuumsecurity.net

stephen@continuumsecurity.net

@stephendv

Please complete the speaker feedback surveys