Java Security Myths

Berlin Expert Days, 03.04.2014

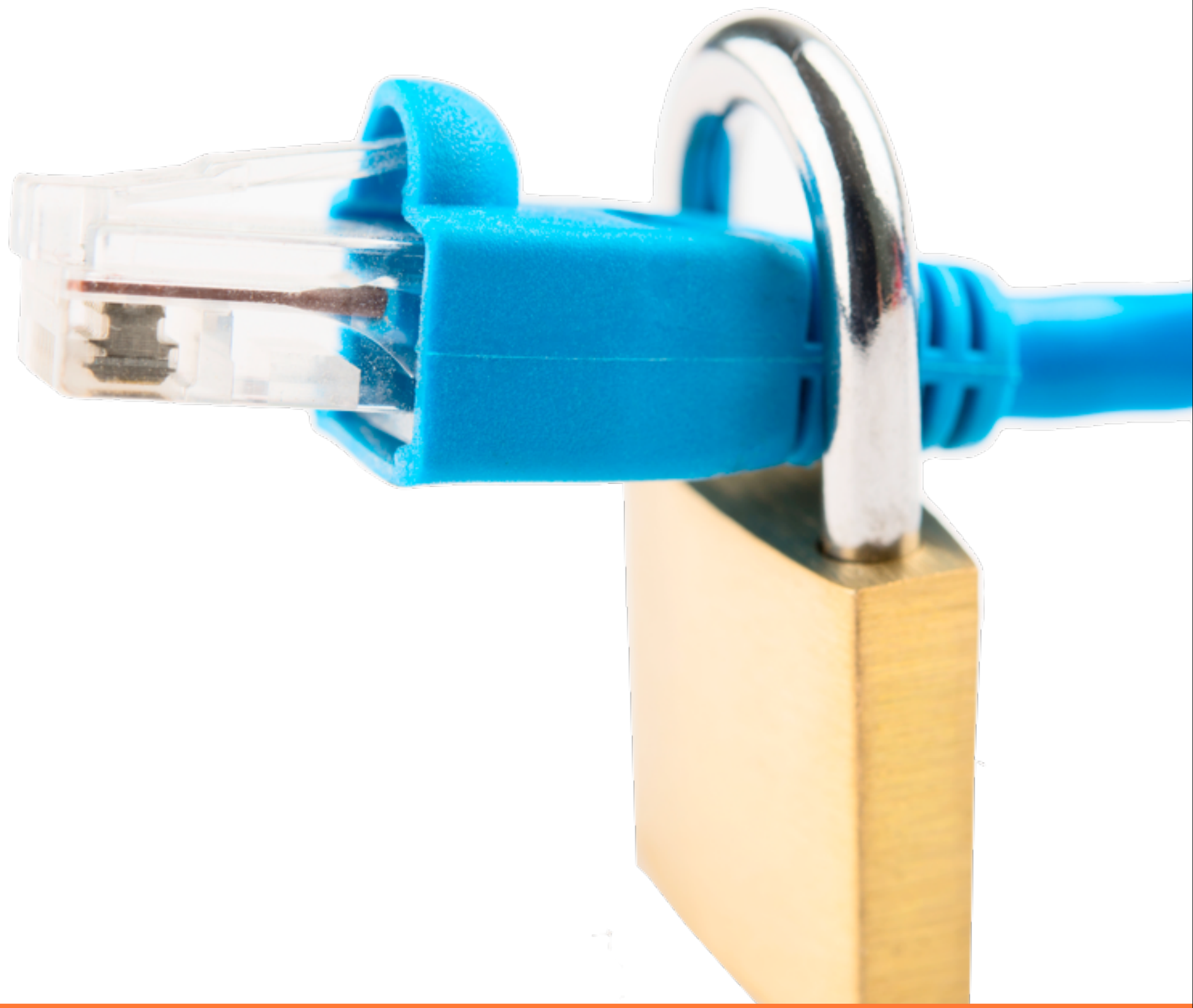
Dominik Schadow
bridgingIT

Secure Communication

Cross-Site Scripting

Cross-Site Request Forgery

Secure Communication



HTTPS after log-in form is enough

Log-in form manipulation and interception

HTTP Strict Transport Security Header (HSTS)

```
protected void doPost(HttpServletRequest request,
  HttpServletResponse response) {
  response.setHeader("Strict-Transport-Security",
    "max-age=2592000; includeSubDomains");
```

HSTS policy requires browser support

Cross-Site Scripting (XSS)

XSS - what?

Browser executes
code from attacker

DOM Based XSS

Reflected XSS

Stored XSS

Modern frameworks automatically provide protection

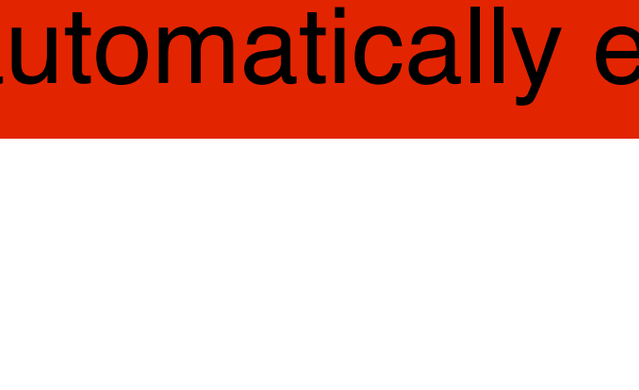
JavaServer Faces automatically escape all output

<script>alert('XSS with JSF')</script>

Submit

<h:outputText/>

<script>alert('XSS with



<h:selectOneMenu/>

<script>alert('XSS with JSF')</script>

XSS in action

Don‘t take framework security for granted

Don‘t regard your code and libraries as separate units

Always update third party libraries

Verify your dependencies are up-to-date

```
Ch06_SQLInjection — bash — 70×5
Last login: Sat Mar 29 12:02:14 on ttys004
marvin:Ch06_SQLInjection dos$ mvn dependency:copy-dependencies
```

Maven

```
Ch06_SQLInjection — bash — 60×5
marvin:Ch06_SQLInjection dos$ dependency-check.sh --app XSS
-out . --scan target/dependency/
```

Dependency-Check Report

Project: XSS

Scan Information (show all):

- *dependency-check version*: 1.1.3
- *Report Generated On*: 29.03.2014 12:08:29
- *Dependencies Scanned*: 37
- *Vulnerable Dependencies*: 1
- ...

Dependency Display: show all

- commons-fileupload-1.2.jar

OWASP
Dependency Check

Input **validate**, output **escape**

Use escaping libraries with security focus

OWASP Java Encoder * Coverity Security Library

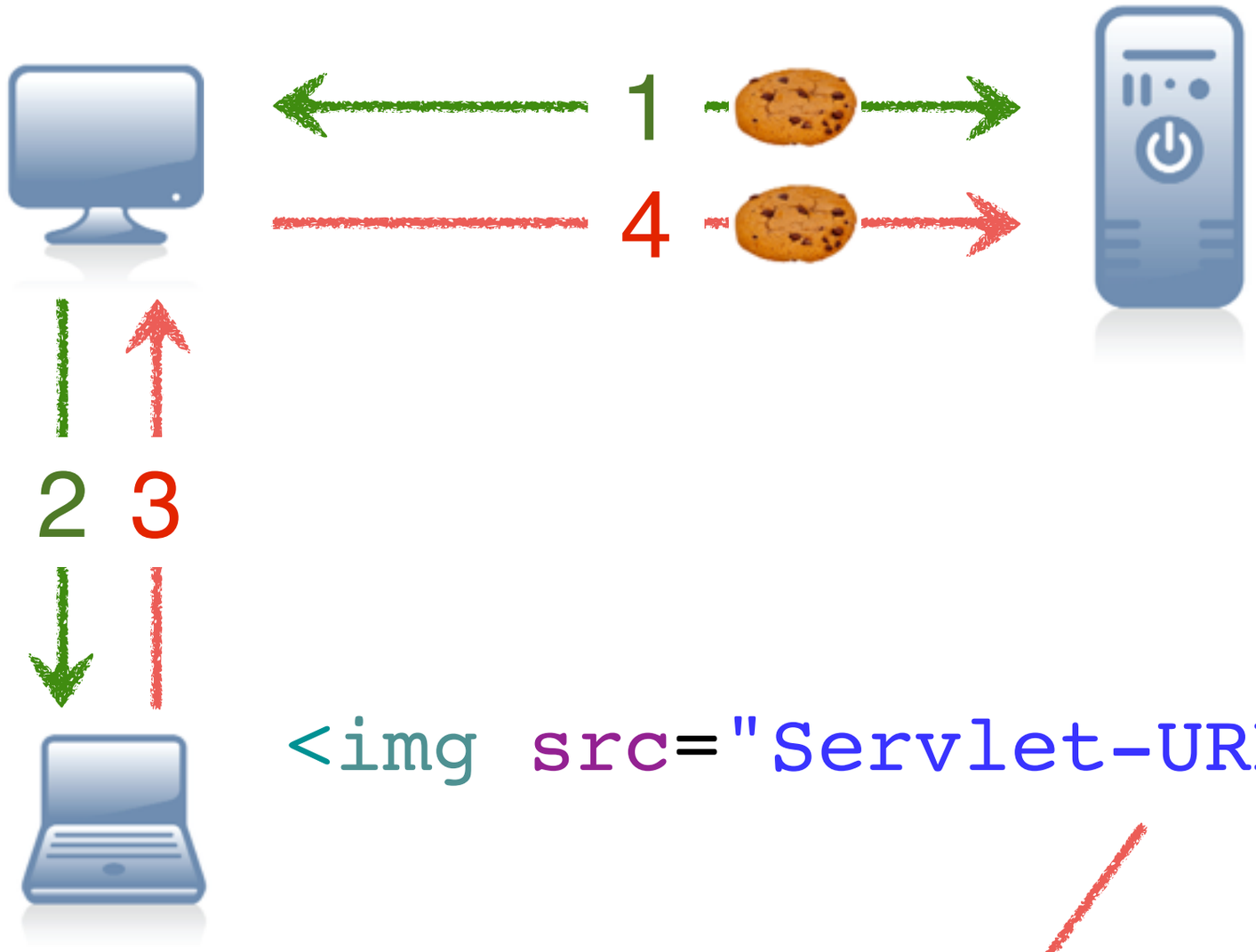
OWASP HTML Sanitizer

Intercepting proxy to avoid validation when testing

Cross-Site Request Forgery (CSRF)

CSRF - what?

POST forms protect from **CSRF**



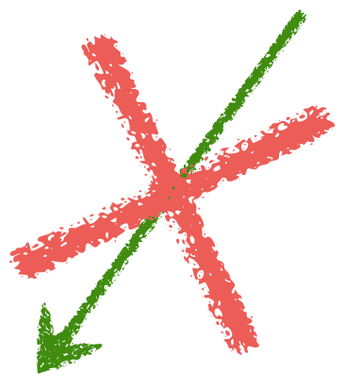

```
<img src="Servlet-URL?name=ATTACK">
```

```
protected void doGet(request, response)
```

Replace GET with **POST requests**

```
<form method="post" action="MyServlet">

    <img src="Servlet-URL?name=ATTACK">
```



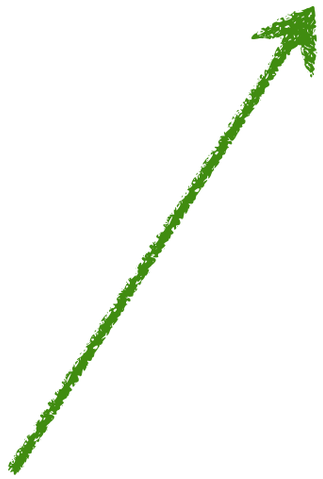
```
protected void doPost(request, response)
```

Add more dynamic with **XMLHttpRequest**

JavaScript may submit POST forms

```
<form method="post" action="MyServlet">
```

```
  protected void doPost(request, response)
```



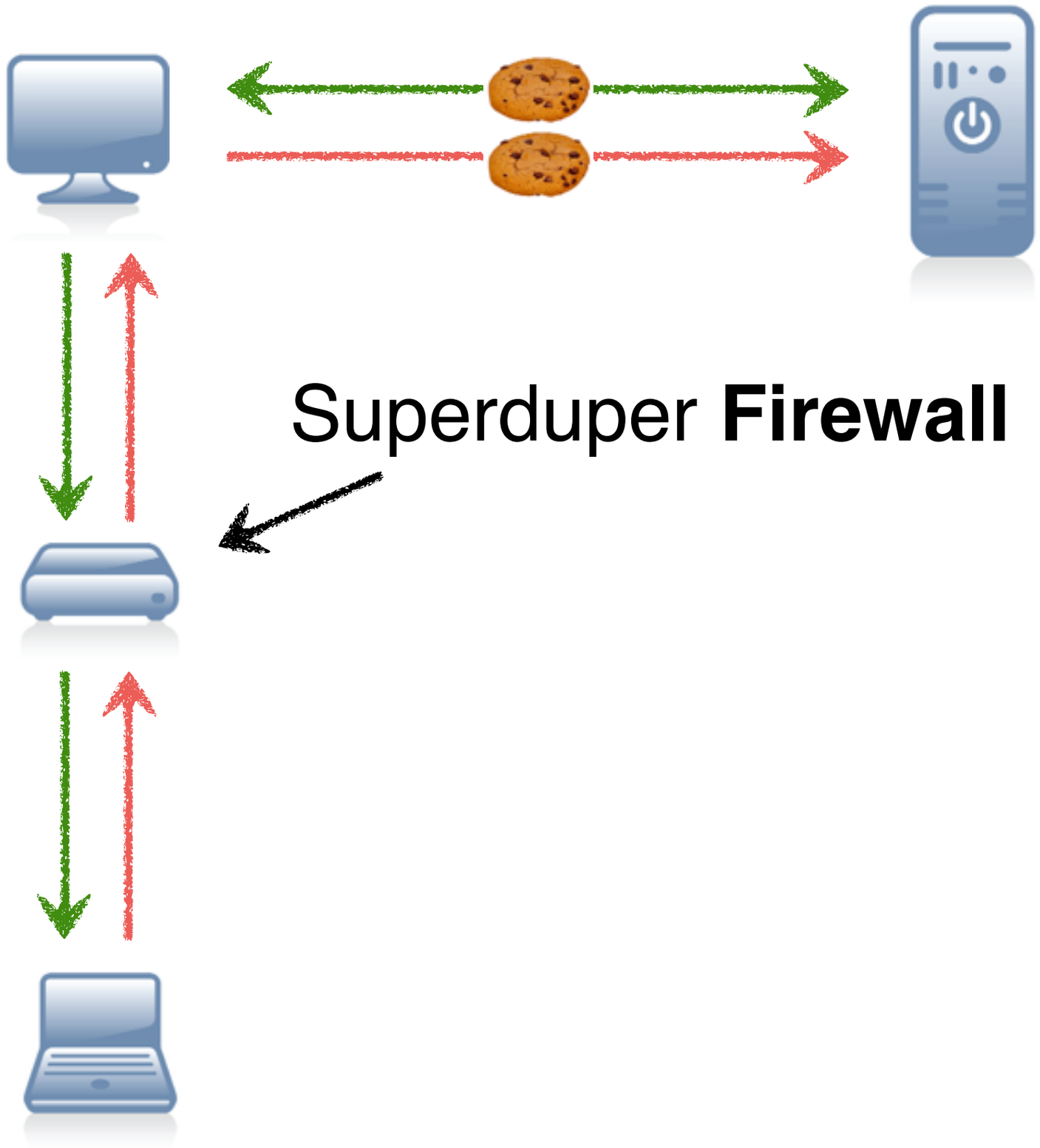
```
var request = new XMLHttpRequest();
request.open("POST", "Servlet-URL");
request.send("name=ATTACK");
```

Add hidden anti CSRF forgery token to each form

CSRF in action

Intranet applications are safe

CSRF attacks don‘t stop at firewalls



Developers make
the difference

Dominik Schadow

dominik.schadow@bridging-it.de
www.bridging-it.de

BridgingIT GmbH
Königstraße 42
70173 Stuttgart

Blog blog.dominikschadow.de
Twitter/ADN @dschadow

Demo Projects
github.com/dschadow/JavaSecurityMyths

Coverity Security Library
github.com/coverity/coverity-security-library

JSF Escaping Bug
java.net/jira/browse/JAVASERVERFACES-2747

OWASP HTML Sanitizer
www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

OWASP Java Encoder
www.owasp.org/index.php/OWASP_Java_Encoder_Project

OWASP Dependency Check
www.owasp.org/index.php/OWASP_Dependency_Check

OWASP Zed Attack Proxy
www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Pictures
www.dreamstime.com www.istockphoto.com