

Attacking Java Clients

Stephen de Vries



Introduction

Why are Java clients interesting from a security perspective?

- Old technology is trusted technology
- Security controls often built into the client
- Critical business logic often built into the client
- Modern IDE blur the lines between client and server

Introduction

Placing implicit trust in the client is an all too easy mistake to make:

...

```
if (loggedIn) {  
    int userId = userService.getLoggedInUserId();  
    orderService.placeOrder(userId,theOrder);  
    ....  
}
```

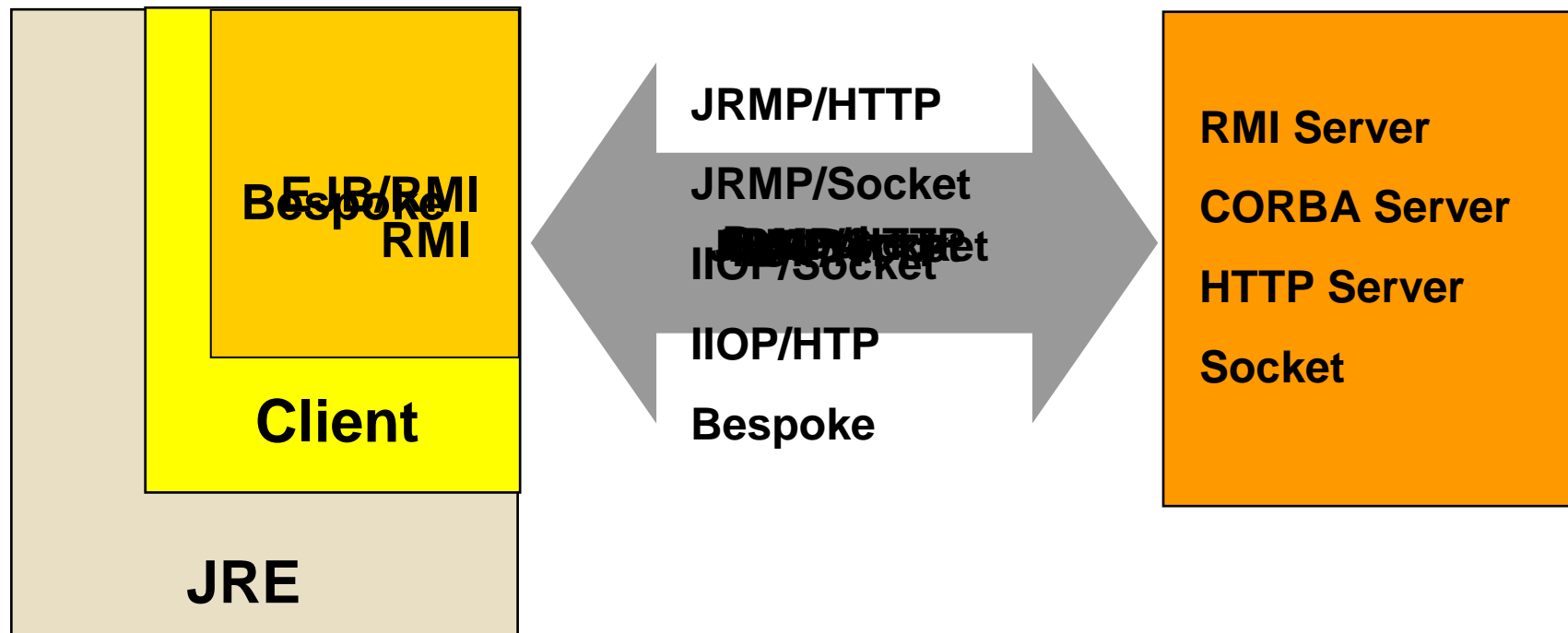


Overview of Java client technology

- Java Applets – In browser, restricted permissions by default
- Java Applications – Unrestricted by default
- JNLP – Application loaded over the network, semi-restricted by default
- JavaFX – Runs on desktop JRE or on specific runtime on mobile devices

....in any case, it's all Java Bytecode running on a JRE

Common remoting technologies



Attack the client, not the transport



Introduction

Problems when security testing Java clients:

- Input validation in the client can prevent injection attacks
- GUI makes automated attacks difficult (e.g. brute force)
- Remoting transport is difficult to intercept
 - Burp plugin by Manish Saindane for Java RMI (BlackHat Europe 2010)
- Decompilers don't work 100%
- Decompilers don't allow you to manipulate the client

Objectives

1. Understand the client logic
2. Manipulate fields and methods in order to subvert the security

Attack Approach

1. Information gathering
2. Probing & Analysis
3. Exploit

Information gathering

- What are the interesting classes?
- What are the server side methods?
- Where is the comms layer?

Probing & Analysis

- What does the execution flow look like?
- Where is the security logic?
- Where is key business logic?
- Which classes are the most convenient to inject a shell?

Exploit

- Inject shell or Static patching
- Bypass client side controls
- Attack server side
 - Injection attacks, e.g. SQL injection
 - Brute force/dictionary attacks
 - Bypass access control

Tools

- Eclipse Test and Performance Tools Platform (TPTP)
- Eclipse plugins
 - JD Decompiler plugin
 - AspectJ Development Tools plugin
- BeanShell
- Java Object Inspector
- AspectJ

Demo Application

CORSAIRE

Some Potential Attacks

- Subvert access control to view other users' orders
- SQL injection attacks against server side
- Brute force attack of login credentials

Step 1: Information gathering

- What are the interesting classes?
- Where are the interesting methods?
- Which remoting technology is in use?

- JAR file inspection
- Class file inspection in IDE
- Class file inspection with javap
- Decompile classes

Step 1: Information gathering

Demo



Step 2: Probing & Analysis

- What does the execution flow look like?
- Where is the security logic?
- Where is key business logic?
- Which classes are the most convenient to inject a shell?

- Profiling with Eclipse TPTP
- Tracing with Eclipse TPTP
- Tracing with AspectJ

Step 2: Probing & Analysis

Demo Profiling



Step 2: Probing & Analysis

Tracing with Eclipse TPTP

- Insert print/log statements
- TPTP supports instrumenting bytecode directly using probes
 - Callsite probe: inserted into calling code
 - Method probe: inserted into called code

Step 2: Probing & Analysis

Demo Tracing with Eclipse



Step 2: Probing & Analysis

Tracing with AspectJ

...before we begin...

...before we begin: AOP and AspectJ

- Programming paradigm to isolate cross-cutting functionality from main business logic, e.g.:
 - Logging
 - Access control
- AspectJ started with source weaving, now does bytecode weaving
- Terminology:
 - **Advice** : New code to insert into the application
 - **Pointcut** : Defines when “advice” should be executed
 - **Aspect** : Advice + pointcut

...before we begin: AOP and AspectJ

```
public aspect ShowSets {  
    pointcut sets() : call( void set* (..) );  
  
    before() : sets() {  
        System.out.println("About to set something");  
    }  
  
    after() : sets() {  
        System.out.println("Completed setting something");  
    }  
}
```

Step 2: Probing & Analysis

Tracing with AspectJ

- Insert print/log statements
- Can log field assignment!
- Define pointcuts and advice

Step 2: Probing & Analysis

Demo Tracing with AspectJ

Step 3: Exploit

- Subvert access control to view other users' orders
- SQL injection attacks against server side
- Brute force attack of login credentials

- Inject shell or Static patching
 - Bypass client side controls
 - Perform attacks

...before we begin

Quick introduction to:

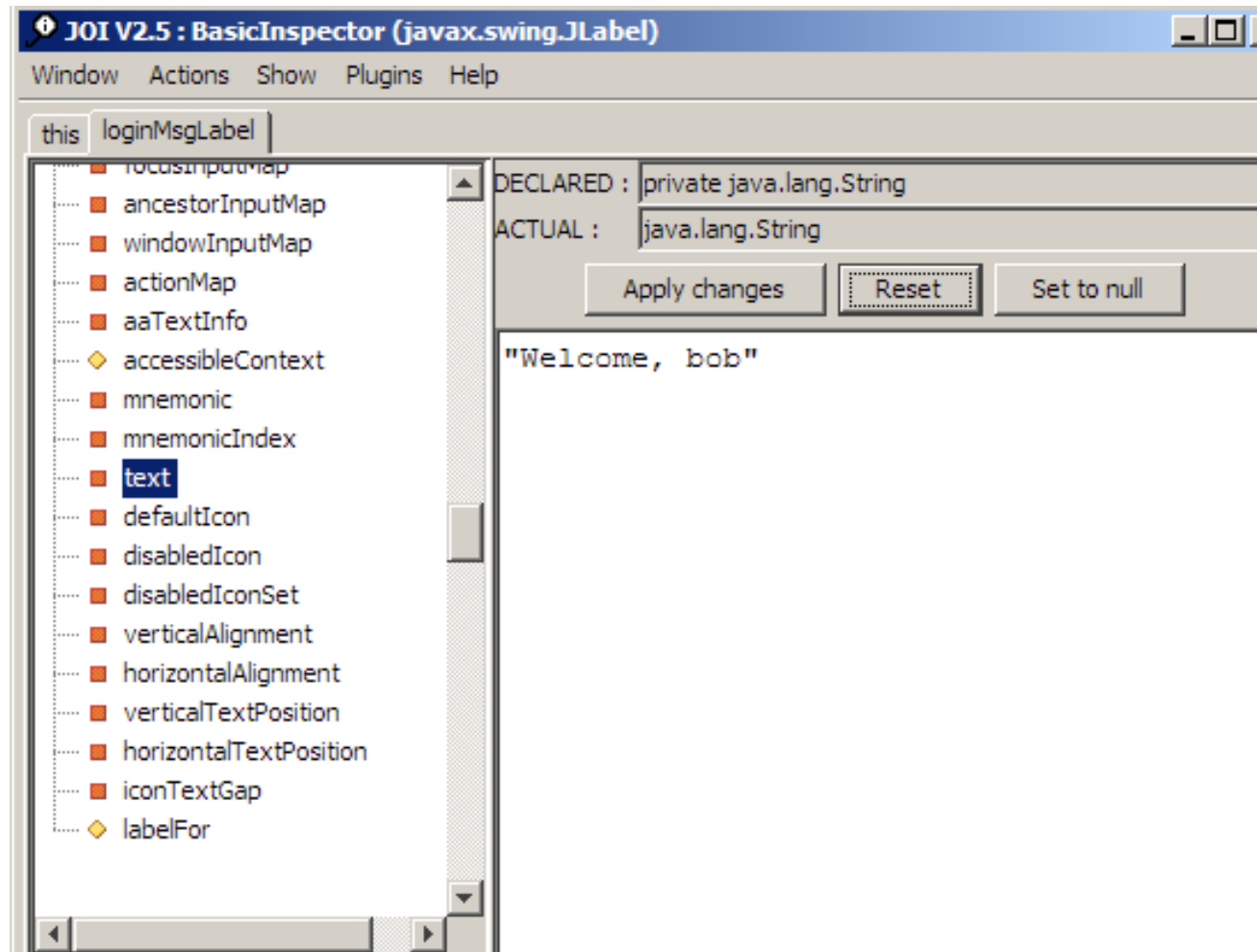
- Java Object Inspector
- BeanShell

...before we begin: Java Object Inspector

- Inserted into application
- View and edit objects

```
import org.pf.joi.Inspector;  
...  
Inspector.inspect(myObject);
```

...before we begin: Java Object Inspector



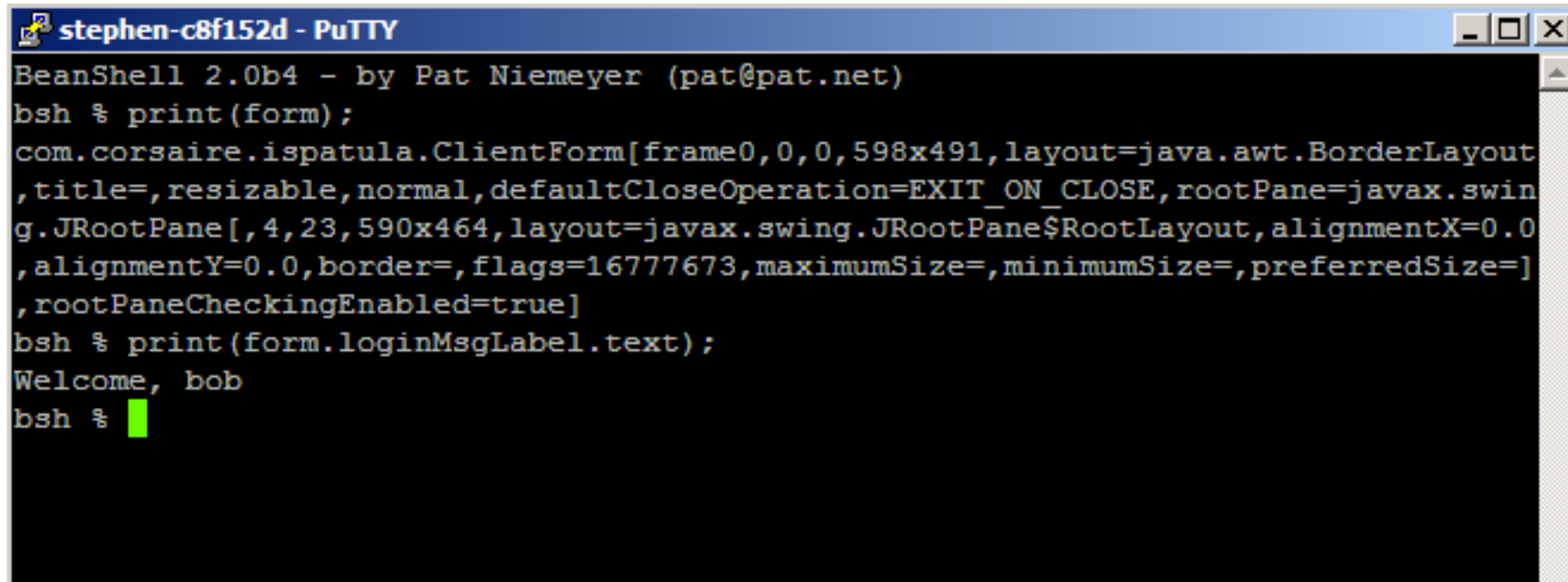
...before we begin: BeanShell

- BeanShell is an embeddable Java source interpreter
 - Provides Java like scripting language
 - For debugging: provides a shell inside the running Java program

```
Interpreter i = new Interpreter();  
try {  
    i.set("myObject", myObject);  
    i.eval("server(7777)");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

...before we begin: BeanShell

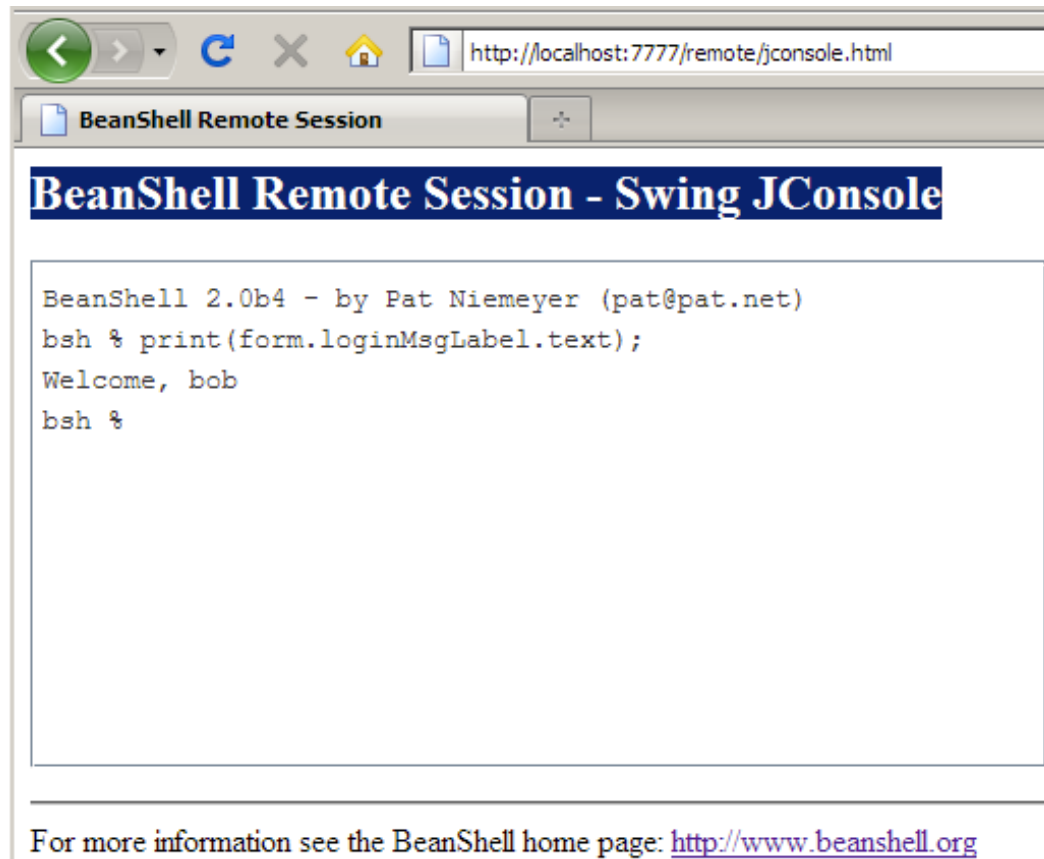
- Telnet server on port 7778

A screenshot of a PuTTY terminal window titled "stephen-c8f152d - PuTTY". The terminal displays the output of a BeanShell session. The first command is "bsh % print(form);", which outputs a detailed Java Swing component description for "com.corsaire.ispatula.ClientForm". The second command is "bsh % print(form.loginMsgLabel.text);", which outputs "Welcome, bob". The prompt "bsh %" is followed by a green cursor.

```
stephen-c8f152d - PuTTY
BeanShell 2.0b4 - by Pat Niemeyer (pat@pat.net)
bsh % print(form);
com.corsaire.ispatula.ClientForm[frame0,0,0,598x491,layout=java.awt.BorderLayout
,title=,resizable,normal,defaultCloseOperation=EXIT_ON_CLOSE,rootPane=javaw.swing.JRootPane[,4,23,590x464,layout=javaw.swing.JRootPane$RootLayout,alignmentX=0.0
,alignmentY=0.0,border=,flags=16777673,maximumSize=,minimumSize=,preferredSize=]
,rootPaneCheckingEnabled=true]
bsh % print(form.loginMsgLabel.text);
Welcome, bob
bsh % █
```

...before we begin: BeanShell

- HTTP server on port 7777



```
BeanShell 2.0b4 - by Pat Niemeyer (pat@pat.net)
bsh % print(form.loginMsgLabel.text);
Welcome, bob
bsh %
```

For more information see the BeanShell home page: <http://www.beanshell.org>

...before we begin: BeanShell

- View the state of objects
- Change values
- Execute methods
- Write scripts to automate tasks

...before we begin: Putting it all together

- BeanShell – Rich shell environment that can be inserted into code
- Java Object Inspector – View and edit fields
- AspectJ – Weave new functionality directly into bytecode

Step 3: Exploit

Demo



Step 3: Exploit

...or statically patch with AspectJ

Step 3: Exploit

- Static patching with AspectJ
 - Redefine methods and return values

Step 2: Exploit

Demo: Static patching with AspectJ

Conclusions

- Developer tools to aide reverse engineering
 - Javap
 - Eclipse TPTP
 - AspectJ
- Trace application flow:
 - AspectJ and Eclipse TPTP
- Manipulate the client:
 - AspectJ
 - BeanShell
 - Java Object Inspector

JavaSnoop

**Don't miss Arshan Dabirsiaghi's
JavaSnoop presentation
15h15
in Neopolitan Room**

- GUI to intercept and modify fields and method calls
- Attaches to running processes: no need to inject anything

Questions ?

stephen@corsaire.com

CORSAIRE