Defying Logic

Theory, Design, and Implementation of Complex Systems for Testing Application Logic

Rafal Los & Prajakta Jagdale – HP Application Security

Act 1 – The Pledge

FOUNDATIONS

LET'S TALK ABOUT LOGIC …FOLLOW THE RABBIT.

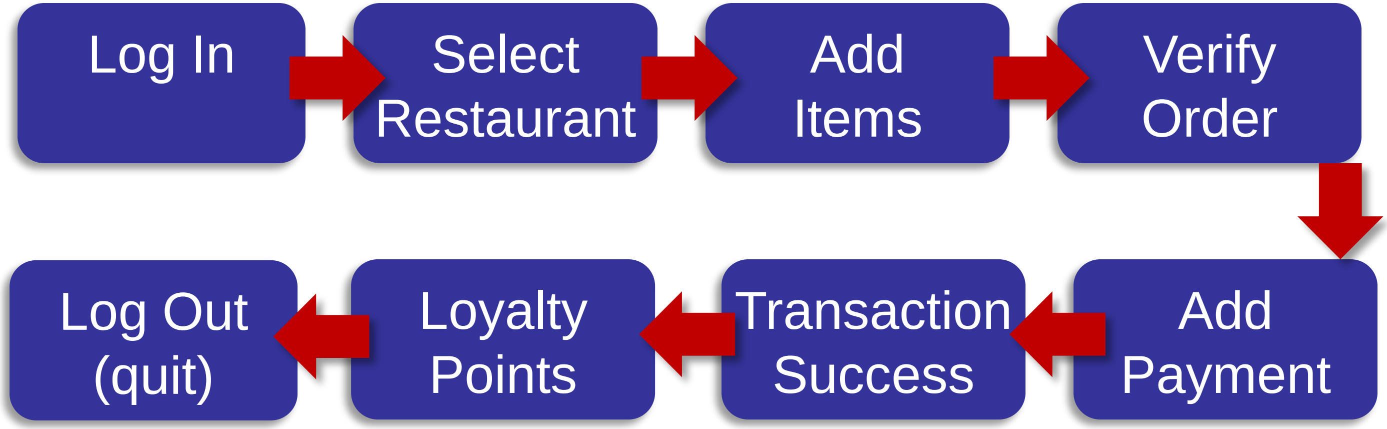
Define: Application Logic

Just what is “application logic”?

- “Design components of a program, including the sequencing of steps prescriptive for how to execute intended business processes in a piece of software”

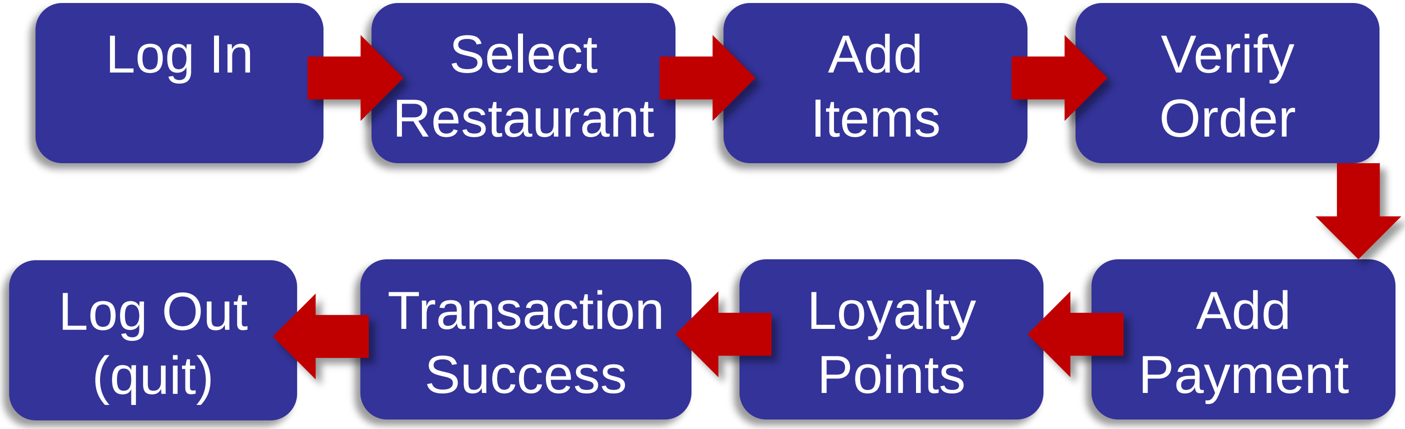
Logic Primer

A business process, ordering



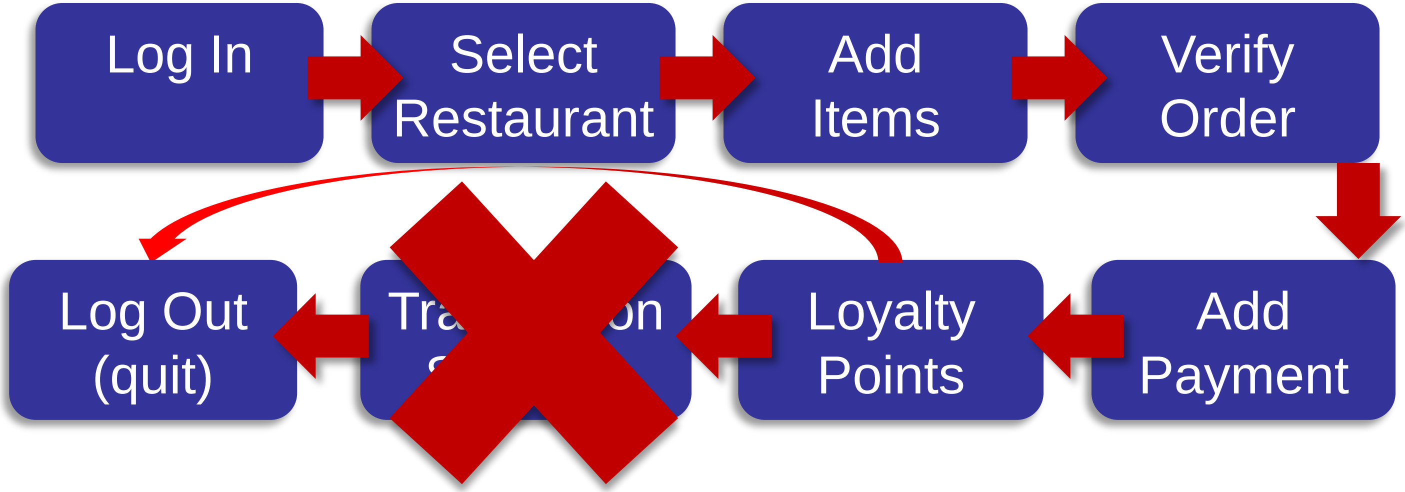
Logic Primer

A broken business process?



Logic Primer

Can we manipulate intended process?



Log In

Select Restaurant

Add Items

Verify Order

Add Payment

Loyalty Points

Tra on S

Log Out (quit)

Logic Primer

In this simple example:

- Manipulate a business process
- Early loyalty points without paying
  - – Points = fraud (free stuff!)
  - – Free meals = financial lo$$ to vendor
- Business process manipulated = theft, fraud, financial loss

Define: Logic Defect

A defect that exposes the component business processes (or execution flows) to manipulation from the attacker perspective to achieve unintended and undesirable consequences from the design perspective; without disrupting the general function or continuity of the application.

New Classifications

How is this different than existing classifications?

- OWASP Top 10
- WASC Threat Classification v2
- MITRE CWE Top 25 ← Best “fit”

A **fundamentally different way** of looking at software vulnerabilities.

Building Onto CWE

– MITRE CWE Top 25

"Classification of business-logic weaknesses is worth deeper investigation by researchers. While business logic rules are domain-specific, there may be some domain-independent patterns to them. We are likely to find new wrinkles to old problems, and maybe some new problems that will be easier to find once we know what to call them." –Steven Christey (MITRE)

“Taxonomy”

2 Types of logic-based defects

- Privilege manipulation
- Transaction control manipulation

Privilege Manipulation

Flaw based on broken or incomplete authentication/authorization mechanism

- Horizontal/vertical privilege escalation
- Access unauthorized content
- Perform privileged system functions

Privilege Manipulation

```
<input type="text" name="username" size="30"
  tabindex="100" value="Wh1t3Rabbit"
  id="username" Role="user">
```

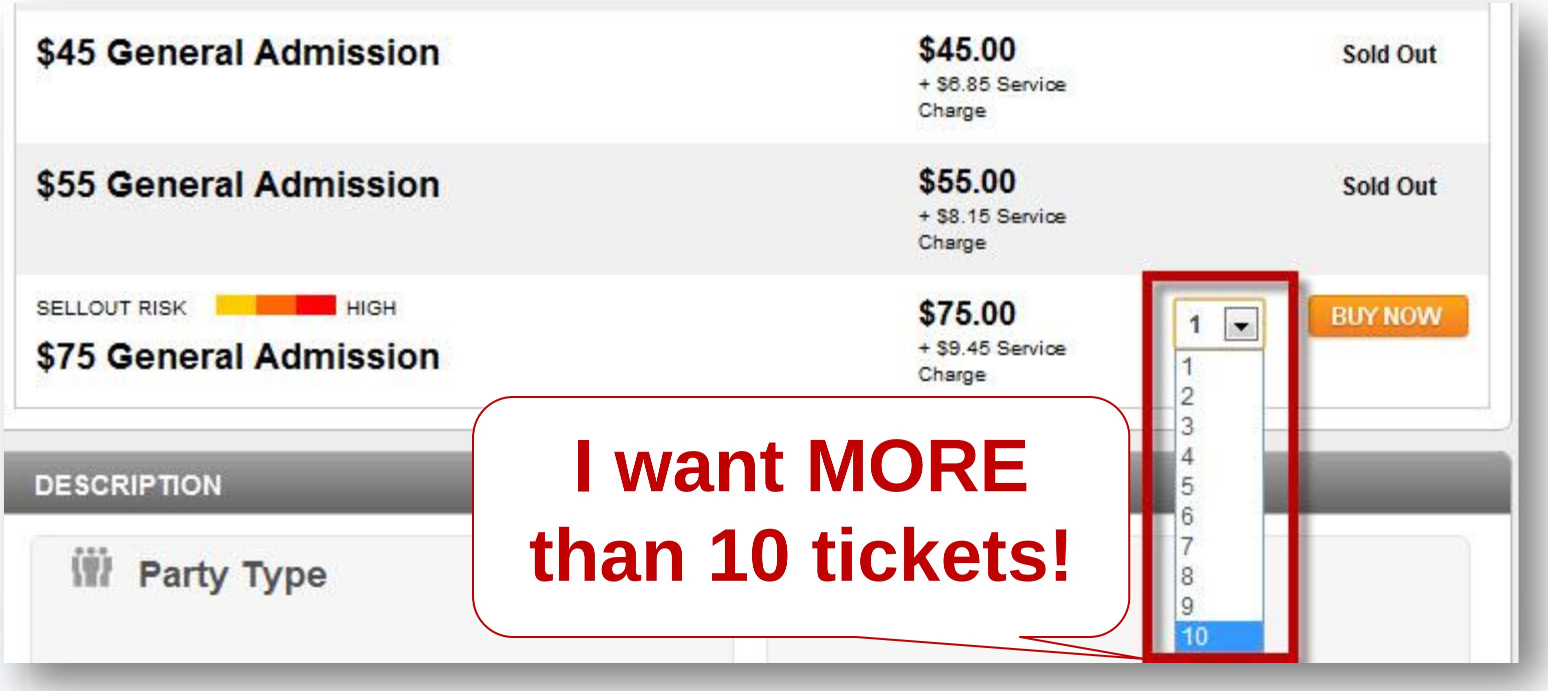
```
<input type="text" name="username" size="30"
  tabindex="100" value="Wh1t3Rabbit"
  id="username" Role="admin">
```

Transaction Control Manipulation

Flaw based on broken business process continuity allowing for manipulation of intended business process/flow

- Circumvent system limitations
- Tamper with application flow
- Manipulate business resources

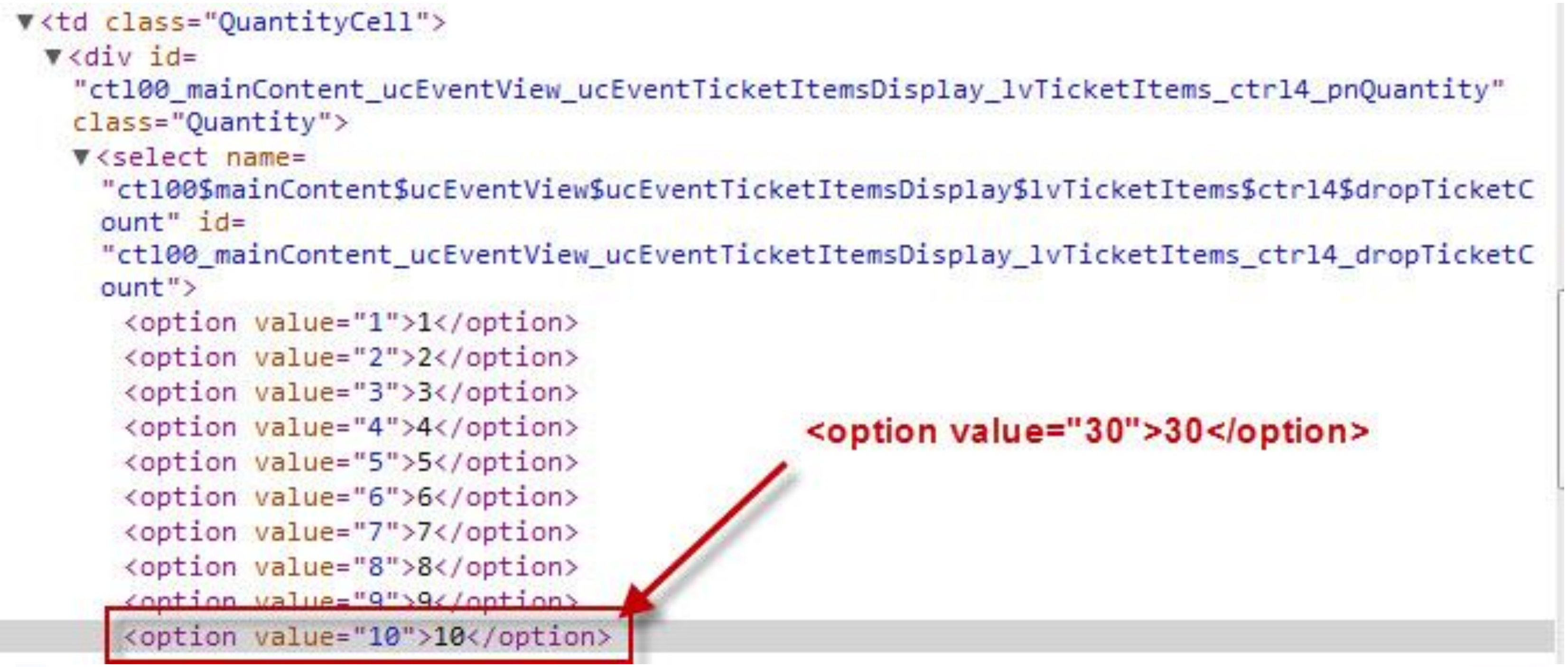
Transaction Control Manipulation



$45 General Admission	$45.00 + $6.85 Service Charge	Sold Out
$55 General Admission	$55.00 + $8.15 Service Charge	Sold Out
SELLOUT RISK HIGH $75 General Admission	$75.00 + $9.45 Service Charge	1 BUY NOW

I want MORE than 10 tickets!

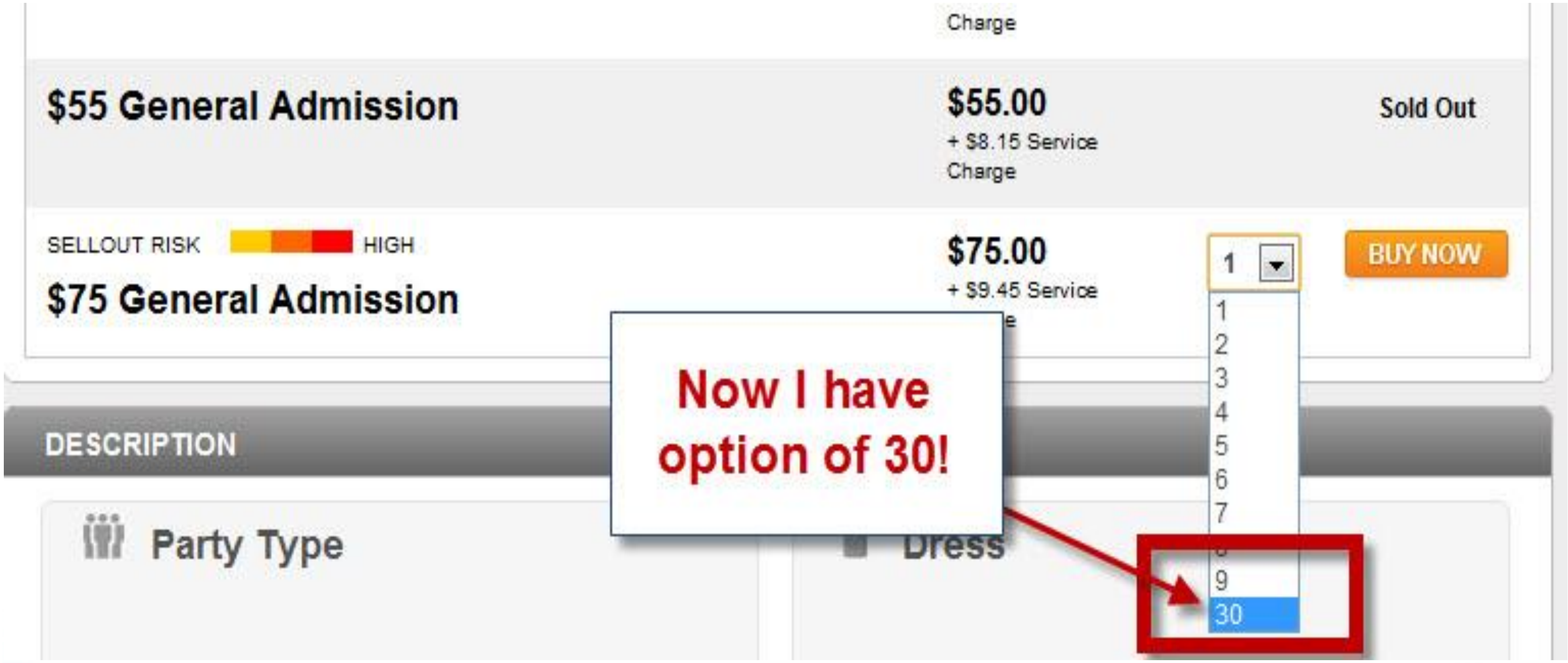
Transaction Control Manipulation



```
▼<td class="QuantityCell">
  ▼<div id=
   "ctl00_mainContent_ucEventView_ucEventTicketItemsDisplay_lvTicketItems_ctrl4_pnQuantity"
   class="Quantity">
    ▼<select name=
     "ctl00$mainContent$ucEventView$ucEventTicketItemsDisplay$lvTicketItems$ctrl4$dropTicketC
     ount" id=
     "ctl00_mainContent_ucEventView_ucEventTicketItemsDisplay_lvTicketItems_ctrl4_dropTicketC
     ount">
       <option value="1">1</option>
       <option value="2">2</option>
       <option value="3">3</option>
       <option value="4">4</option>
       <option value="5">5</option>
       <option value="6">6</option>
       <option value="7">7</option>
       <option value="8">8</option>
       <option value="9">9</option>
       <option value="10">10</option>
```

<option value="30">30</option>

Transaction Control Manipulation



Charge

$55 General Admission

$55.00

+ $8.15 Service Charge

Sold Out

SELLOUT RISK HIGH

$75 General Admission

$75.00

+ $9.45 Service ...e

DESCRIPTION

Party Type

Dress

Now I have option of 30!

Manipulating “Logic”

The server should control the **logic** of the application.

In this case, the server should know 30 is not a valid number …right?!

…………………………..Nope.

To Be Clear

The key point to remember:

This research seeks to better enable the tester through automation.

We **are** addressing ‘designed processes’

We **are not** addressing ‘back-end business process’ *(non-visible)*

Starting a Wave

- **FEW** previously given talks or papers on logic vulnerabilities
  - Ideas → talks → papers → experiements?
  - A more tangible/usable effort is needed
- **NO** existing R&D effort by a vendor in this space as far as we could tell — proprietary **or** open source

Lots of Talking…

“So why aren’t there any readily available, even semi-mature tools or frameworks for testing application logic?”

Act 2 – The Turn

APPLICATION LOGIC VS. AUTOMATION

Logic vs. Automation

Application Logic

- ✓ hard to define!
- ✓ domain-specific
  - ✓ Industry
  - ✓ Business process
- ✓ not pattern-based
  - ✓ not easy to 'RegEx'

Automation

- ○ pattern-dependent
- ○ programmatic
- ○ scale is in repeatability
- ○ no concept of process

Challenges

For humans

- understand the application
  - Business processes
  - ‘Application flow’
- drive automation
  - work with technology
- document & repeat

For technology

- map the application processes
  - simple, flexible maps
- identify control logic
  - critical parameters
- differentiate test success or failure

Application Logic

Application “logic” is tricky…

- found on the server side
- found in the client “cache” (offline use?)
- no consistent patterns to match/test
  - Remember the AT&T/iPad email hack?
- *logic* implies **human thought required**

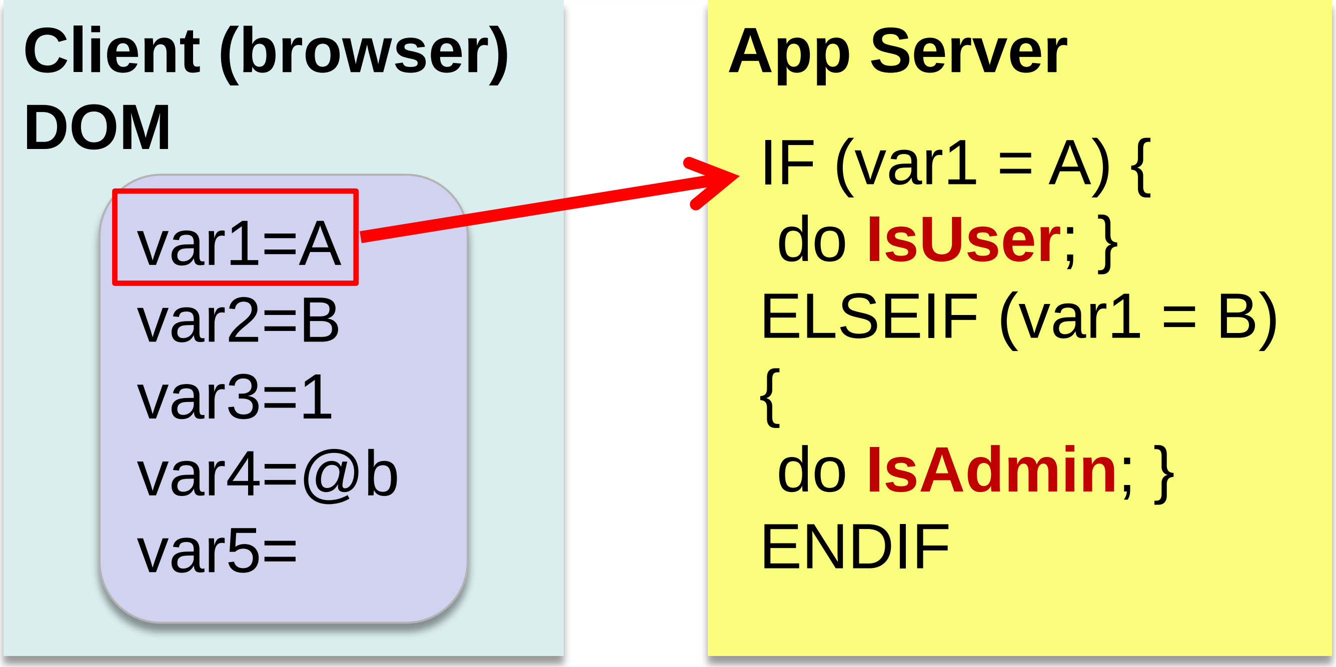
Simple Logic

Simplest logic block example…

```
If <expression> Then
   do something
ElseIf <expression> Then
   do something
Else
   do something else
End If
```

Challenges

Bridging a vast disconnect



Client (browser) DOM

```
var1=A
var2=B
var3=1
var4=@b
var5=
```

App Server

```
IF (var1 = A) {
 do IsUser; }
ELSEIF (var1 = B)
{
 do IsAdmin; }
ENDIF
```

Challenges

How to overcome random “fuzzing”

- identify **var1** as a critical parameter
  - Human or automation-driven
- manipulate **var1**
  - human or technology driven
  - “fuzzing” or data-set driven

Technology enables scalability & repeatability

Future State → Fantasy

- Point ‘n’ shoot application logic testing
  - This was never a good strategy anyway*
- Dynamic testing tool ‘learns’ business processes, logic flow and *thinks*
  - I’ve seen this movie before …”SkyNet?”
- Security continues to operate independent of business

Future State → Reality

- Logic testing enabled through automation
  - Base logic mapping on *QA methodology*
  - Continue to evolve combined functional, exploratory & security testing
- Technology allows repeatable testing throughout application after initial setup

This Has Been Coming

- Initially I talked about mapping application execution flow…
- Dynamic testing technology matured
- Static testing technology matured
- New "**hybrid**" technology gives even greater insight into application function

Act 3 – The Prestige

BUILDING TEST FRAMEWORKS

Overview

Test framework will constitute 3 phases

I. Capturing the assumed application behavior

II. Manipulation of the workflows to evoke (un)expected behavior patterns

III. Analyzing the results of the workflow manipulations to detect non-conformance with defined business processes

3 Step Process

- Model the business process
  - – create valid workflows
- Manipulate application workflow
  - – ‘fuzzing logic’
- Analyze the results
  - – Were we successful in evoking a non-standard response?

Modeling the Business Processes

- Purpose: Discover business processes that define the application workflow
- Challenge: Application workflow specifications are rarely documented and are often unknown to the testers
- Solution: Passively monitor & record normal user behavior through the application

Modeling the Business Processes

Proposed Approach

Devise a mechanism to:

- Capture permissible user actions
- Store the state of the application before and after the invocation of each user event
- Record expected transitions between application states

Manipulating the Application Workflow

- Purpose: Induce deviations in the application behavior that do not conform with the assumed business rules
- Challenge: Meaningfully manipulating application’s behavior with minimal knowledge about its purpose & context
- Solution: Apply pre-defined modifications to the state identifiers and state transactions recorded in phase I

Manipulating the Application Workflow

Proposed Approach

Discover logic defects by:

- Modifying the state of the application before passing it as an input to a transaction
- Fuzz the sequence of user actions captured during the phase I to detect bugs in transaction control

Analyzing the Results

- Purpose: Determine the success of workflow manipulation
- Challenge: Measuring the deviation in application behavior with minimal understanding of application context
- Solution: Apply comparison metrics to infer deviations in the application workflows

Analyzing the Results

Proposed approach

- Measure the impact of workflow manipulations on the application state
  - compare state identifiers (e.g. system variables, page DOM) in the tainted workflow with the original
- Apply pre-defined set of rules to detect violations of the business rules governing the application flow
  - E.g. An acceptable application end state can be reached despite inaccuracies in one or more intermediate states of the workflow

DEMO!

a rough idea of how this could
work one day

Work In Progress

If you are interested in discussing, contributing, or expanding this research in any way – contact us.

Rafal@HP.com
twitter.com/Wh1t3Rabbit