Into the Rabbithole—
Evolved Web Application Security Testing

Rafal M. Los
Security Evangelist
HP Application Security Center

Rafal@hp.com - @Wh1t3Rabbit
+1 (404) 606-6056
Let’s descend down the rabbit-hole

OR

Better testing through evolution
Automation: Love & Hate

Web App Sec has a LOVE | HATE relationship with automation

**LOVE**

- Automation speeds defect identification
- Scanning is fast, quickly producing results

**HATE**

- Attack surface coverage unclear*
- Confuse automation’s purpose

*More on the coverage problem shortly...
Understanding Automation

Battle lines *(the classic arguments)*
- Humans offer intelligence
- Automation offers limited scope

Benefits of automation
- Scalability: Analysis speed, coverage, processing
- Complexity: Applications are increasingly process-driven
So What?

We’ve reached a tipping point
Why Did My Scanner Miss X?

Two _real_ reasons

- X required a specific sequence, or **FLOW**
- X required **DATA** to get there

Data + Flow → no excuses

- IF tools have **data** + **logic**… the result is “smarter” automation
- No more “crawl n’ hope”
“Radical” Testing Methodology

ENLIGHTENED METHODOLOGY

• Application functional mapping w/data
• Layered automation-infused testing
• Concrete metrics & KPIs

point n’ scan web application security testing
Do what you do...

only smarter
Application Functional Mapping with Data
Defect vs. Vulnerability

How many of you have ever performed functional testing?
## Functional vs. Security Testing

<table>
<thead>
<tr>
<th>QA TEAM</th>
<th>INFOSECURITY TEAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions known</td>
<td>Functions unknown</td>
</tr>
<tr>
<td>Application understood</td>
<td>Application unknown</td>
</tr>
<tr>
<td>Rely on functional</td>
<td>Rely on crawlers + experience + luck</td>
</tr>
<tr>
<td>specifications</td>
<td></td>
</tr>
<tr>
<td>Coverage known</td>
<td>Coverage unknown</td>
</tr>
<tr>
<td>Highlight key business</td>
<td>Highlight “found” functionality</td>
</tr>
<tr>
<td>logic</td>
<td></td>
</tr>
</tbody>
</table>
Hard Lessons Learned

Security analysts, tools [today] aren’t equipped to properly test **highly complex** applications...

**MISSING PIECES**
- Understanding of application
- Functional mapping of application
- Application execution flow
- Valid test data
Bridging the Gaps

Is the kitchen-sink attack working?

Hint: It used to…not anymore

YOU ARE HERE

IDEAL

THEY ARE THERE
As All This Is Happening—Technology Drives Forward...
Application State Is Changing

HTTP State
- Session/Cookie State
- Server State

Client State
- JavaScript State
- Silverlight/Flash State

Impossible to decouple HTTP from Client State
You can’t just crawl/guess your way through a modern, complex application
Proposed Approach

Combine **functional** + **security** testing, compensating for technology

- Address **technology** complexities
  - Session states
  - Code-complexity

- Address **functional** complexities
  - Mapping application function as execution flows
  - Mapping data for driving execution flows
Incoming *New Automation Technology!*
Standards & Specifications

**EFD**

*Execution Flow Diagram* – Functional paths through the application logic

**ADM**

*Application Data Mapping* – Mapping data requirements against functional paths
Improving the Testing Process

Functional Specification

Function-based automated testing

Manual result & coverage validation

Application functional mapping [EFD]

Application data mapping [ADM]
Basics of the EFD & ADM
Basic EFD Concepts

Graph(s) of flows through the application
- Nodes represent application states
- Edges represent different actions
- Paths between nodes represent state changes
- A set of paths is a flow
Execution Flow Action Types

What is an action?

- Something that causes a change in state
- A human, server or browser-driven event

Three types of actions

- Direct
- Supplemental
- Indirect
Direct Flow Actions

Actions which change the browser’s document context

- Causes an entirely new browser page

Examples-

- Following hyperlink
- Click login button
Supplemental Flow Actions

Actions that change the state of the current document

- Client-side action, maintaining browser page

Examples:
- JavaScript menu
- Flash client event
Indirect Flow Actions

Actions **automatically triggered** by document context

- Usually for supporting data, modifying document state

Examples:
- Site analytics (js)
- Stock ticker
- XMLHTTPRequest
Basic ADM Concepts

An Application Data Map [ADM] defines flows with the context of data

WHY?

- Flows mean nothing without **DATA**
- Data should be **interchangeable**
  - Monitoring requests make this impossible – no context
- Data can be direct or indirect

*Where not specifically defined within an action (at the edge) the data values are assumed to be arbitrary*
ADM + EFD Visually

Retrieve something from a safe:

1. Map the action
2. Add data (context) necessary to execute
3. Execute action using data

I need something from that safe

ACTION (open safe)

Combination: R23, L12, R31, L9
ADM & EFD

Another example: Web site registration
Putting It All Together (1)

Functional Level

Technical Level
Putting It All Together (2)

**EFD**

<table>
<thead>
<tr>
<th>JS DOM</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>GET /</td>
</tr>
<tr>
<td>b</td>
<td>GET /?Login</td>
</tr>
<tr>
<td>c</td>
<td>GET /?Compose</td>
</tr>
<tr>
<td>d</td>
<td>onKeyPressed</td>
</tr>
<tr>
<td>e</td>
<td>DIV.onMouseOver</td>
</tr>
<tr>
<td>f</td>
<td>LI.onChange</td>
</tr>
<tr>
<td>g</td>
<td>FORM.submit()</td>
</tr>
</tbody>
</table>

GET /?Send
## Putting It All Together (3)

<table>
<thead>
<tr>
<th>JS DOM</th>
<th>HTTP</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>GET /?Login</td>
<td>User, Pass, Captcha</td>
<td></td>
</tr>
<tr>
<td>GET /?Compose</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>onKeyPressed</td>
<td>Email_Text</td>
<td></td>
</tr>
<tr>
<td>(160 times)</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>DIV.onMouseOver</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>LI.onChange</td>
<td>Send_To_Address</td>
<td></td>
</tr>
<tr>
<td>BTN.onClick</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>
Applications of Execution
Flow Diagrams
Flow Based Threat Analysis

- Markup flow with Threat Information
  - Prioritize testing
  - Prioritize verified vulnerabilities
- Detect dangerous information flows

Partners Only

Checkout with Credit Card

Viewing Items
Coverage Analysis

Flows defined by functional specification can be compared to security testing to determine gaps!

Q: “How much of the application was tested?”

A: “The scanner was able to test 8 of the 12 flows, we need to find out why/where it broke down”

→ EFD can be referenced to determine where
→ ADM can be referenced to determine why
Flow-Based Reproduction

Demonstrate exactly how to reproduce a defect...

- Demonstrate where application failed
  - Steps executed
  - Data used
Dysfunctional Use of EFD

Vulnerabilities happen when using the application in an unintended way.

If we know the right logic paths...
Next Generation Automation

Automation of execution flows

• Build maps from user-driven functional scripts

• Recording/Playback
  • Record HTTP requests
  • Record JavaScript events
  • Recording Client UI events

• Attacking
  • [Re]Play Flows
  • Auditing HTTP Parameters and HTML Inputs
Next: Automatic Exploration

- Similar paths can be easily enumerated
- JS Static Analysis to find other entry points to paths
For Next Time…

**Layered automation-infused testing**

Testing must be layered to fully understand the attack surface of the application, including multiple levels of authentication, business logic, data sets.

**Concrete metrics & KPIs**

In order to concretely prove functional coverage, application surface area coverage, defect remediation and ultimately risk reduction business-oriented metrics and KPIs must be gathered.
Get to it.

Insert cheesy cliché here...

...or you could just go do it.

Rafal Los
Email: Rafal@HP.com
Twitter: @Wh1t3Rabbit
Voice: (765) 247-2325
Blog: http://echp.com/go/white-rabbit
Outcomes that matter.