The Diviner

Digital Clairvoyance Breakthrough
Source Code & Structure Black Box Divination

Shay Chen, @sectooladdict

CTO at Hacktics ASC, Ernst & Young

November 20, 2012

ERNST & YOUNG
Quality In Everything We Do
About Me

► Addictions
About Me

► Security Tools Collector / Addict
About Me

Law of Familiarity
About Hacktics

 ► Hacktics ASC
   ▶ Formerly a boutique company that provided various information security services since 2004.
   ▶ As of 01/01/2011, Ernst & Young acquired Hacktics professional services practice, and the group joined EY as one of the firm’s advanced security centers (ASC).
The Diviner Project

Diviner

- OWASP ZAP extension (v1.4+)
- Requires ZAP to run with Java 1.7+

Development

- 1+ years of development, tons of extra hours by @Secure_ET
- Made possible due to support from the OWASP ZAP project, specifically from Simon Bennetts (@psiinon)
The Problem
The numerous tasks of manual penetration testing
Manual Testing: Attacks & Vulnerabilities

- WASC Threat Classification
  - 34 Attacks
  - 15 Weaknesses
- OWASP Attacks & Vulnerabilities
  - 64 Attacks
  - 165 Vulnerabilities
- CWE, Wiki, OWASP Testing Guide and Additional Lists
<table>
<thead>
<tr>
<th>SQL Injection</th>
<th>NoSQL Injection</th>
<th>SQL Sorting</th>
<th>LDAP Injection</th>
<th>XPath Injection</th>
</tr>
</thead>
<tbody>
<tr>
<td>XQuery injection</td>
<td>XML Injection</td>
<td>HTTP Request Splitting</td>
<td>HTTP Request Smuggling</td>
<td>HTTP Request Header Injection</td>
</tr>
<tr>
<td>HTTP Response Header Injection</td>
<td>Code Injection-General</td>
<td>SSL Injection</td>
<td>Format String Injection</td>
<td>Expression Language Injection</td>
</tr>
<tr>
<td>Remote File Inclusion</td>
<td>Local File Inclusion</td>
<td>Race Conditions</td>
<td>Forceful Browsing</td>
<td></td>
</tr>
<tr>
<td>Integer Overflow</td>
<td>Null-Byte Injection</td>
<td>Temporal Session Race Conditions (TSRC)</td>
<td>Session Hijacking</td>
<td></td>
</tr>
<tr>
<td>Abuse of Functionality</td>
<td>Parameter Tampering</td>
<td>Session Variable Overloading</td>
<td>Insecure Object Mapping</td>
<td></td>
</tr>
<tr>
<td>Session Prediction</td>
<td>Binary Planting</td>
<td>Connection String Parameter Pollution</td>
<td>Open Redirect</td>
<td></td>
</tr>
<tr>
<td>Oracle Padding</td>
<td>Reflected XSS</td>
<td>Persistent XSS</td>
<td>Cross Frame Scripting</td>
<td></td>
</tr>
<tr>
<td>CSRF</td>
<td>Dynamic CSRF</td>
<td>SDRF</td>
<td>Click-Jacking</td>
<td></td>
</tr>
<tr>
<td>Cross Site Tracking</td>
<td>Frame Spoofing</td>
<td>Content Spoofing</td>
<td>CRLF Injection</td>
<td>HTTP Response Splitting</td>
</tr>
<tr>
<td>Policy Abuse</td>
<td>Log Forging</td>
<td>HTTP Verb Tampering</td>
<td>HTTP Methods Abuse</td>
<td>Cross Site History Manipulation</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>Distributed Denial of Service</td>
<td>Numeric Denial of Service</td>
<td>Application Denial of Service</td>
<td>Account Lockout</td>
</tr>
<tr>
<td>Regular Expression Denial of Service</td>
<td>Beast Attack</td>
<td>SSL/TSL Renegotiation Raw</td>
<td>Replay Attack</td>
<td>Man-in-The-Middle</td>
</tr>
<tr>
<td>SQL Flow injection</td>
<td>Information Disclosure</td>
<td>Caching</td>
<td>Auto Complete</td>
<td>Fingerprinting</td>
</tr>
<tr>
<td>Policy Violation</td>
<td>Uncaught Exception</td>
<td>Weak Cryptography</td>
<td>Broken Access Control</td>
<td>Poor Logging Practice</td>
</tr>
<tr>
<td>Source Code Disclosure</td>
<td>Inefficient Logout</td>
<td>Credentials Disclosure</td>
<td>Unrestricted File Upload</td>
<td>Obsolete Files</td>
</tr>
<tr>
<td>Insecure Password Recovery Process</td>
<td>Insecure Transport</td>
<td>Insecure Cookie</td>
<td>Hard-Coded Passwords</td>
<td>HTTP Request Injection</td>
</tr>
<tr>
<td>XXE</td>
<td>Mail Headers Injection</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Limited Time Frame (Cont.)

#tests = \approx 100 \text{ tests per each parameter}

#pages = \text{different web pages in the application}

#params = \text{different parameters in each web page}
The Limited Time Frame (Cont.)

#tests * #pages * #params

= A lot of time (and tests)
The Limited Time Frame (Cont.)

\[ \text{#tests} \times \text{#pages} \times \text{#params} \]

\[
100 \times 20 \times 3 = 6,000 \text{ tests}
\]
The Limited Time Frame (Cont.)

\[
\text{#tests} \times \text{#pages} \times \text{#params} = 100 \times 2 \times 3 = 6,000 \text{ tests}
\]
The Limited Time Frame (Cont.)

\[
\text{#tests} \times \text{#pages} \times \text{#params} \\
100 \times 3 = 6,000 \text{ tests}
\]
The Limited Time Frame (Cont.)

#tests * #pages * #params

100 100 3

= 30,000 tests
The Limited Time Frame (Cont.)

!!!30,000
The Limited Time Frame, Potential Solutions

► **Experience, Intuition and Luck.**

► **Automated Scanners**
  - **Benefit:** Perform multiple tests on a large amount of URLs/Parameters.
  - **Downside:** Can only detect familiar attacks and scenarios, limited accuracy, and potential false positives.

► **Fuzzers**
  - **Benefit:** Collect the responses of numerous payloads from multiple URLs.
  - **Downside:** Presentation method, amount of analysis required.

► **Information Gathering…**
Gazing into the Crystal Ball
The Art of War: Information Gathering
Information gathering processes are used to locate instances of **sensitive information disclosure**, as well as obtaining semi-legitimate information on the application’s structure, underlying infrastructure, and behavior.

“If you know your enemies and know yourself, you will not be imperiled in a hundred battles”

(Sun Tzu, The Art of War, 6th century BC)
Passive Information Gathering

- **Dictionary term:** “accepting or allowing what happens or what others do, without active response or resistance.”

- **Application-level passive analysis** is performed using techniques such as:
  - Google hacking
  - Entry point mapping
  - Content analysis tools:
    - **Watcher**, ZAP, WebFight, Etc.
  - Internet Research
  - Open source code analysis
  - Etc.
Active Information Gathering

► Dictionary Term: “Gathering information that is not available in open sources, sometimes requires criminal activities to obtain.”

► Performed using techniques such as:
  ► Brute-Force Attacks
  ► Resource Enumeration
  ► Intentional Error Generation
  ► Source Code Disclosure Attacks
  ► Etc.

Is it really the limit?
Mr. Big

(?!?)
MrBig

Massive Recursive Behavior Information Gathering

► Application behavior in normal & extreme scenarios
► Indirect cross component effect
► Effect of values in each and every field
► Restrictions
► Behavior analysis

Which can lead to…
The Impact
Black Box
Source Code & Structure
Insight
The Crown Jewel - Source Code Disclosure

- Inherent Security Flaws in the Application Code
- Test a Local Copy of the Application
- Hardcoded Credentials & Encryption Keys
- Disclose the Structure of the Internal Network
- Etc.
Security by Obscurity – Officially Dead?

- Based on Kerckhoffs's principle.
  - "Security by obscurity" makes the product safer and less vulnerable to attack.
  - Written in 1883.
- During the last 130 years, security experts disprove this concept over and over again.
- Diviner puts the last nail in the coffin.
Source Code Divination – Benefits

- The benefits of source code divination are many:
  - Generate a visual representation of the behavior of each page.
  - Generate a pseudo-code representation of language specific source code.
  - Locate and differentiate between direct & indirect effect of input values on entry points.
  - Track the flow of input & output in the application.
  - Track session identifier origin & lifespan.
  - Detection of dormant events, methods, and parameters.
  - Indirect attack vector detection.
Source Code Divination

```
/puzzlemall-private-buypuzzle.jsp

80% String input16 = request.getParameter("id");

80% String input4 = request.getParameter("descr");

70% connection conn = DriverManager.getConnection("connection-string");

70% PreparedStatement Sqlstatement16 = conn.prepareStatement("UPDATE table16 SET target_field16 = ? WHERE {conditions}");

70% PreparedStatement Sqlstatement4 = conn.prepareStatement("UPDATE table4 SET target_field4 = ? WHERE {conditions}");

70% Sqlstatement16.setString(1, input16);

70% Sqlstatement4.setString(1, input4);

70% Sqlstatement16.executeUpdate();

70% Sqlstatement4.executeUpdate();

80% out.println(input4);
```
Direct & Indirect Cross Entry Point Effect
Visual Entry Point Input-Output Correlation

Options
- Results
- Requests

```
/ puzzlemall/private/mainmenu.jsp
Location  Input Parameters
  Session

/ puzzlemall/login.jsp
Location  Input Parameters
  username
  password

/ puzzlemall/private/vieworders.jsp
Location  Input Parameters
  Database

/ puzzlemall/private/buypuzzle.jsp
Location  Input Parameters
  Output
  descr
  id
  purchase

/ puzzlemall/private/viewprofile.jsp
Location  Input Parameters
  Session

/ puzzlemall/register-phase2.jsp
Location  Input Parameters
  Output
  username
  email
  password
  recovery

/ puzzlemall/recovery-phase2.jsp
Location  Input Parameters
  username
```
Divination Attacks
# ZAP’s Request History

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>URL</th>
<th>Status</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/contact.jsp?origin=USA">http://localhost:8080/puzzlemall/contact.jsp?origin=USA</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>4</td>
<td>POST</td>
<td><a href="http://localhost:8080/puzzlemall/login.jsp">http://localhost:8080/puzzlemall/login.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>5</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/viewprofile.jsp">http://localhost:8080/puzzlemall/private/viewprofile.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>7</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/viewpuzzles.jsp">http://localhost:8080/puzzlemall/private/viewpuzzles.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>8</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=2&amp;descr=transaction">http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=2&amp;descr=transaction</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>9</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=2&amp;purchase=true&amp;descr=transaction">http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=2&amp;purchase=true&amp;descr=transaction</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>10</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/vieworders.jsp">http://localhost:8080/puzzlemall/private/vieworders.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>11</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/mainmenu.jsp">http://localhost:8080/puzzlemall/private/mainmenu.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>12</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/sitemap.jsp">http://localhost:8080/puzzlemall/sitemap.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>13</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/recovery-phase1.jsp">http://localhost:8080/puzzlemall/recovery-phase1.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>14</td>
<td>POST</td>
<td><a href="http://localhost:8080/puzzlemall/recovery-phase2.jsp">http://localhost:8080/puzzlemall/recovery-phase2.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>16</td>
<td>POST</td>
<td><a href="http://localhost:8080/puzzlemall/recovery-success.jsp">http://localhost:8080/puzzlemall/recovery-success.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>17</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/register-phase1.jsp">http://localhost:8080/puzzlemall/register-phase1.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>18</td>
<td>POST</td>
<td><a href="http://localhost:8080/puzzlemall/register-phase2.jsp">http://localhost:8080/puzzlemall/register-phase2.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>19</td>
<td>GET</td>
<td><a href="http://localhost:8080/puzzlemall/private/mainmenu.jsp">http://localhost:8080/puzzlemall/private/mainmenu.jsp</a></td>
<td>200</td>
<td>OK</td>
</tr>
</tbody>
</table>
Exploring Different Paths of Execution

Behavior in Different Authentication Modes and History Perquisites

Start

- Login Mode
  - No Login
  - Login First
  - Login After Source EP

- History Access
  - No History
  - Partial History
  - Full History

History

- Request#1
- Request#2
- Login-Request
- Request#4
- ...

Source Entry Point

- Optional Login

Result Analysis

- Target History
  - No History
  - Required History

Target Entry Point

Diviner - Clairvoyance in the Digital Frontier
Exploring Different Paths of Execution, Cont.

Behavior With Different Session Cookies, Identifiers and Tokens

New Session Cookie

Use Updated Cookie

Use Original Cookie

New AntiCSRF Token

Use New Token

New AntiCSRF Token

Update Parameter

New Page Specific Parameter

Use New Token

Use New Token

New Page Specific Parameter

Scenario Execution

Scenario Execution
# Source Code Divination Accuracy

<table>
<thead>
<tr>
<th>ID</th>
<th>Behaviour Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input Reflected from Variable</td>
</tr>
<tr>
<td>2</td>
<td>Input Reflected from Session</td>
</tr>
<tr>
<td>3</td>
<td>Input Reflected from Database</td>
</tr>
<tr>
<td>4</td>
<td>Input Stored in Server Variable</td>
</tr>
<tr>
<td>5</td>
<td>Input Stored in Session Variable</td>
</tr>
<tr>
<td>6</td>
<td>Input Stored in Database Table</td>
</tr>
<tr>
<td>7</td>
<td>New Cookie Value</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Source Code Divination Accuracy

<table>
<thead>
<tr>
<th>ID</th>
<th>Code Description</th>
<th>JSP Code</th>
<th>ASP.Net Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Read Input to Variable</td>
<td>String input$$1$$ = request.getParameter(&quot;#1#&quot;);</td>
<td>String input$$1$$ = Request[&quot;#1#&quot;];</td>
</tr>
<tr>
<td>2</td>
<td>Invalidate Session</td>
<td>session.invalidate();</td>
<td>Session.Abandon();</td>
</tr>
<tr>
<td>3</td>
<td>New Session Identifier</td>
<td>request.getSession(true);</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>New Cookie Value</td>
<td>Cookie cookie = new Cookie(&quot;#1#&quot;, val);</td>
<td>Response.Cookies(&quot;#1#&quot;).Value = &quot;val&quot;;</td>
</tr>
<tr>
<td>5</td>
<td>Get Database Connection</td>
<td>Class.forName(DriverClassName);</td>
<td>SqlConnection conn = new SqlConnection(X);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connection conn = DriverManager.getConnection(X);</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Source Code Divination Accuracy

<table>
<thead>
<tr>
<th>Behavior ID</th>
<th>Code ID</th>
<th>Code Type</th>
<th>Rank</th>
<th>Default Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>1010</td>
<td>50%</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1</td>
<td>10040</td>
<td>70%</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>5550</td>
<td>40%</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2010</td>
<td>90%</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>10000</td>
<td>80%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Verification Process and Probability

For each unique entry point / request, the probability for the existence of specific lines of code is adjusted according to the results of various behavior specific confirmation processes.

Previous session redirects to login after set-cookie instruction?

Behaviour7 -> CodeId2 +40%, CodeId3 +20%, CodeId4 -10%

<table>
<thead>
<tr>
<th>Behavior ID</th>
<th>Code ID</th>
<th>Code Type</th>
<th>Rank</th>
<th>Current Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>1010</td>
<td>70%</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1</td>
<td>10040</td>
<td>60%</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>5550</td>
<td>80%</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2010</td>
<td>90%</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>10000</td>
<td>80%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Source/Target Entry Points Code Correlation

```
String input16 = request.getParameter("id");
String input4 = request.getParameter("desc");
connection conn = DriverManager.getConnection("connection-string");
PreparedStatement Sqlstatement16 = conn.prepareStatement("UPDATE table16 SET target_field16 = ? WHERE [conditions]");
PreparedStatement Sqlstatement4 = conn.prepareStatement("UPDATE table4 SET target_field4 = ? WHERE [conditions]");
Sqlstatement16.setString(1, input16);
Sqlstatement4.setString(1, input4);
Sqlstatement16.executeUpdate();
Sqlstatement4.executeUpdate();
out.println(input4);
```
Diviner
A New ZAP Extension
Live Demo!
Divination Wizard – Record Login Scenario

The login page will be used to create logged in sessions in the analyzing process.

- Application Supports UserLock (the login page will not be analyzed)
- http://localhost:8080
- http://localhost:8080/puzzlemall
- http://localhost:8080/puzzlemall/images
- http://localhost:8080/puzzlemall
- http://localhost:8080/puzzlemall/private-viewprofile.jsp
- http://localhost:8080/puzzlemall/recovery-phase1.jsp
- http://localhost:8080/puzzlemall/recovery-phase2.jsp
- http://localhost:8080/puzzlemall/register-phase2.jsp

Set the parameters for the login request:
http://localhost:8080/puzzlemall/login.jsp
name=user1&password=guessme1232

OK   Cancel
**Divination Wizard – Handle CSRF Barriers**

The image shows a window for handling Cross-Site Request Forgery (CSRF) tokens and replayable parameters. The window includes a table with columns for **Token name**, **Per page**, and **Active**. The table lists various tokens such as `VIEWSTATE`, `EVENTTARGET`, `VIEWSTATEENCRYPTED`, `VSTATE`, `PREVIOUSPAGE`, `REQUESTDIGEST`, `anticsrf`, `CSRFToken`, and `RequestVerificationToken`. The **Per page** column has checkmarks for some entries, indicating they are active per page.

- **VIEWSTATE** is active per page.
- **EVENTTARGET** is active per page.
- **EVENTARGUMENT** is active per page.
- **LASTFOCUS** is active per page.
- **VIEWSTATEENCRYPTED** is active per page.
- **LASTFOCUS** is active per page.
- **VSTATE** is active per page.
- **PREVIOUSPAGE** is active per page.
- **REQUESTDIGEST** is active per page.
- **anticsrf** is not active.
- **CSRFToken** is not active.
- **RequestVerificationToken** is not active.

The window also includes a list of options on the left side, such as **Domain**, **Plugins**, **Scenarios**, **URLs**, **Login**, **Public Session**, and **Choose Request**. The top of the window reads, "CSRF Tokens and Replayable Parameters:"

The window also includes an option to "Add Token."
Divination Wizard – Define Analysis Scope

Analyzing Scope:

Only the selected URLs will be analyzed. It is recommended to remove static files from scope, such as media files, to make the analyze process faster.

- http://localhost:8080
- http://localhost:8080/puzzlemall
- http://localhost:8080/puzzlemall-images
- http://localhost:8080/puzzlemall-login.jsp
- http://localhost:8080/puzzlemall-private
- http://localhost:8080/puzzlemall-private/buypuzzle.jsp?id=1&descr=aaaddaaaaaa
- http://localhost:8080/puzzlemall-private/buypuzzle.jsp?id=1&purchase=true&descr=aaaddaaaaaa
- http://localhost:8080/puzzlemall-private/mainmenu.jsp
- http://localhost:8080/puzzlemall-private/vieworders.jsp
- http://localhost:8080/puzzlemall-private/viewprofile.jsp
- http://localhost:8080/puzzlemall/recovery-phase1.jsp
- http://localhost:8080/puzzlemall/recovery-phase2.jsp
- http://localhost:8080/puzzlemall/register-phase2.jsp

Check/Uncheck All  Filter  OK  Back
Visual Penetration Testing & Payload Reuse
Visual Entry Point Input - Output Correlation
Entry Point Structure & Source Visualization

Diviner - Clairvoyance in the Digital Frontier
Source/Target Entry Points Code Correlation

```java
String input16 = request.getParameter("id");
String input4 = request.getParameter("descr");
connection conn = DriverManager.getConnection("[connection-string]);
PreparedStatement Sqlstatement16 = conn.prepareStatement("UPDATE table16 SET target_field16 = @ WHERE [conditions]");
PreparedStatement Sqlstatement4 = conn.prepareStatement("UPDATE table4 SET target_field4 = @ WHERE [conditions]");
Sqlstatement16.setString(1, input16);
Sqlstatement4.setString(1, input9);
Sqlstatement16.executeUpdate();
Sqlstatement4.executeUpdate();
out.println(input5);
```
Detect Indirect Attack Vectors – Source Page

```java
String input0 = request.getParameter("username");
request.getSession().setAttribute("SessionAttribute1", input1);
out.println(input0);
```
Detect Indirect Attack Vectors – Target Page
Support Different Technologies
Support Different Technologies
Reap the Rewards

Detecting Exposures in Divined Pseudo-code
Live Demo!
Reap the Rewards
Detecting Exposures in Divined Structure
Live Demo!
Reap the Rewards
Parameter Specific Manual
Detection Recommendations
Live Demo!
Reap the Rewards
Using the Payload Manager with Diviner Visual Entry Point
Presentation
Live Demo!
Reap the Rewards
Task List Management (Leads) &
Attack Flow Advisor
Live Demo!
Divination Mechanics
When entry point behaviors are interpreted to language-specific pseudo code, one line of code of each “code type” is added (to enable the process to support multiple interpretations for each behavior), for every behavior potential code collection.
Sorting Divined Source Code

The code is initially sorted according to a predefined behavior specific ranking system, but then re-sorted according to the results of designated sort verification processes (delay of service and behavior stack verification).
Source Code Divination – Structure Analysis

Analyzing the application structure, and tracking the flow of input/output will provide various insights:

- Component behaviors in normal vs. extreme scenarios:
  - Reaction to different sets of characters (abnormality/exception)
  - Reaction to missing content
  - Direct & Indirect effect of input on different entry points
  - Indirect and Direct output reflection

- In addition, the locations
  - Input Database storage vs. Session storage
  - Static Variable Storage and Viewstate storage
Hints on the existence of specific code can be obtained from various sources and behaviors:

- Application behaviors, such as:
  - Direct & Indirect reflection of input in the output
  - Exceptions or abnormal behaviors caused due to specific characters
  - Abnormal access sequences
  - Response variation
- Comparing different behaviors
- Identifying value override junctions
Source Code Divination – Code Prediction

Source Code Divination Sources (Cont.):

- Line-targeted Delay Of Service attacks:
  - RegEx DoS
  - Connection Pool Consumption
  - Numeric DoS
  - Magic Hash, Etc
- Behavior fingerprinting, alongside various verifications
Source Code Divination – Sorting Mechanics

- Sorting the source code can be achieved via:
  - Simultaneous activation of line-targeted **Delay of Service** attacks, while:
    - Accessing the entry point with an exception generating character, located during the structure mapping phase.
  - Exception & behavior fingerprinting
  - Sending erroneous exceptions in different parameters (exception & behavior priority)
  - Comparing multiple information sources
  - Assigning default sort value to each potential line of code
Intentional Latency Increment (Sorting Code)

▶ Delay of Service – intentional extension of the productive latency.

▶ If the line is delayed then it also exists, and occurs before, after or between other lines of code.

```java
session.setAttribute(
    SessionConstants.USERNAME_VARIABLE,
    username);
    
    }  // Productive Latency

session.invalidate();  // invalidate session, erase all variables
The ADoS attack must affect the lines of code before, between or after the behavior/exception specific code.

For example, a denial of service attack that targets the web server is inefficient (since all the code is affected) while a denial of service attack that targets the database (and thus, the database access code) might be.
Layer Targeted ADoS
Layer Targeted Denial Of Service

- Different lines of code might access different digital layers, such as:
  - Databases
  - Web Services
  - External Servers
  - File Operations.

- Furthermore, malicious payloads can be used to increase the latency of code sections:
  - Regular Expressions
  - Loops
  - Search Criteria.
Increasing Latency with RegEx DoS

- RegEx Dos Payloads can increase the latency of validation and search mechanisms. For example:
  - **RegEx:** ([a-zA-Z0-9]+)*
  - **Input:** Admin, aaaaaaaaaaaaaaaaaaaaaaaaaaa!

```java
String username = request.getParameter("username");
String password = request.getParameter("password");

session.setAttribute(SessionConstants.USERNAME_VARIABLE, username);

//input validation
if (!username.matches(ValidationConstants.USERNAME_IV_REGEX)) ||
   !password.matches(ValidationConstants.PASSWORD_IV_REGEX)) {
    session.invalidate(); //invalidate session, remove all variables
    ... 
} else {
    ...
}
```
Use an automated script that consistently accesses modules, which use connections from a size-restricted connection pool for querying the database.

- The script must use a number of threads equal or higher to the maximum connections in the pool.
- In order to continue occupying connections, each thread should re-access the module again, immediately after getting a response.
- The script should use less threads than the amount supported by the server.
- The script should not affect the availability of the server, or any other layer (but the target layer).
Occupying Connections to Increase Latency

- Occupying connections will guarantee that code, which requires a database connection, will experience some latency.

```java
String username =
    request.getParameter("username");
session.setAttribute(
    SessionConstants.USERNAME_VARIABLE,
    username);

Connection conn = ConnectionPoolManager.getConnection();

// Delayed until a connection is released
session.invalidate();
```
And Finally...
Additional Resources

► Diviner Homepage (ZAP 1.4+ Extension)
  ► http://code.google.com/p/diviner/
  ► Structure and input/output flow visualization
  ► Source code & memory structure divination
  ► Advisor and task list manager
  ► Payload manager integrated with ZAP repeater

► Payload Manager .Net
  ► External editor for Diviner’s payload manager database

► OWASP ZAP Proxy:
  ► http://code.google.com/p/zapproxy/
Acknowledgments

The following individuals and groups helped transform Diviner from an idea to reality:

- **Eran Tamari** – The lead developer and a firm believer.
- The OWASP ZAP Project, **Simon Bennetts** and **Axel Neumann** - for the amazing support and for enabling ZAP extensions.
- **Zafrir Grosman** – Material design.
- **Hacktics Employees** - for assisting in the various development phases of the payload manager extension.
- **Ernst & Young**, for investing the resources necessary to publish the research.
Ernst & Young Advanced Security Centers

► Americas
  ► Hacktics Israel
  ► Houston
  ► New York
  ► Buenos Aires

► EMEIA
  ► Dublin
  ► Barcelona

► Asia Pacific
  ► Singapore
  ► Melbourne
About Ernst & Young
Ernst & Young is a global leader in assurance, tax, transaction and advisory services. Worldwide, our 130,000 people are united by our shared values and an unwavering commitment to quality. We make a difference by helping our people, our clients and our wider communities achieve potential.

About Ernst & Young’s Technology Risk and Security Services
Information technology is one of the key enablers for modern organizations to compete. It gives the opportunity to get closer, more focused and faster in responding to customers, and can redefine both the effectiveness and efficiency of operations. But as opportunity grows, so does risk. Effective information technology risk management helps you to improve the competitive advantage of your information technology operations, to make these operations more cost efficient and to manage down the risks related to running your systems. Our 6,000 information technology risk professionals draw on extensive personal experience to give you fresh perspectives and open, objective advice – wherever you are in the world. We work with you to develop an integrated, holistic approach to your information technology risk or to deal with a specific risk and security issue. And because we understand that, to achieve your potential, you need a tailored service as much as consistent methodologies, we work to give you the benefit of our broad sector experience, our deep subject matter knowledge and the latest insights from our work worldwide. It’s how Ernst & Young makes a difference.

For more information, please visit www.ey.com.

© 2012 EYGM Limited. All Rights Reserved.
Proprietary and confidential. Do not distribute without written permission.

Ernst & Young refers to the global organization of member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients.
Questions

shay.chen@il.ey.com (https://twitter.com/#!/sectooladdict)
&
eran.tamari@il.ey.com (https://twitter.com/#!/Secure_ET)