



The Diviner

Digital Clairvoyance Breakthrough
Source Code & Structure Black Box Divination

Shay Chen
Senior Manager, Hacktics CTO
Hacktics ASC, Ernst & Young

October 21st, 2012



A Picture is Worth a Thousand Words...

Clairvoyance – source code divination

JSP ASP ASP.NET

/puzzlemall/private/buypuzzle.jsp

80%	String input101 = request.getParameter("descr");
70%	connection conn = DriverManager.getConnection("[connection-string]");
70%	PreparedStatement Sqlstatement18 = conn.prepareStatement("UPDATE tabel18 SET target_field18 = ? WHERE [conditions]");
70%	Sqlstatement18.setString(1, input18);
70%	Sqlstatement18.executeUpdate();
90%	out.println(input101);
90%	String output51 = request.getSession().getAttribute(SessionAttribute51);
90%	String output24 = request.getSession().getAttribute(SessionAttribute24);
90%	String output0 = request.getSession().getAttribute(SessionAttribute0);
90%	out.println(output51);
90%	out.println(output24);
90%	out.println(output0);

Complete Analysis

Show Decision Path



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).



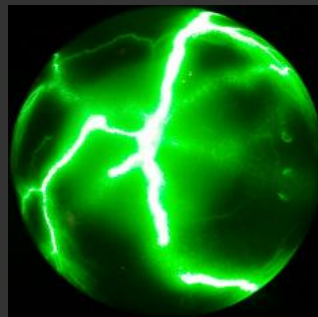
The Diviner Project

▶ Diviner

- ▶ OWASP ZAP extension (v1.4+)
- ▶ Requires ZAP to run with Java 1.7+
- ▶ Homepage: <http://code.google.com/p/diviner/>

▶ 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)



Agenda

- ▶ **The Problem – The Challenges of Manual PT**
- ▶ **The Art of War – Information Gathering**
- ▶ **The Impact – BB Source Code & Structure Insight**
- ▶ **Divination Attacks – Predicting the Server Structure**
 - ▶ Memory Structure Divination
 - ▶ Source Code Divination
 - ▶ Server Side Process Visualization
- ▶ **Divination Mechanics**
- ▶ **The Diviner Project**
- ▶ **Q & A**



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**



SQL Injection	NoSQL Injection	SQL Sorting	LDAP Injection	XPath Injection
XQuery Injection	XML Injection	HTTP Request Splitting	HTTP Request Smuggling	HTTP Request Header Injection
HTTP Response Header Injection	SMTP Injection	Code Injection-General	Code Injection-ASP	Code Injection-PHP
Code Injection-JSP	OS Command Injection	SSI Injection	Format String Injection	Expression Language Injection
Remote File Inclusion	Local File Inclusion	Directory Traversal	PHP File Inclusion	Buffer Overflow
Integer Overflow	Null-Byte Injection	Race Conditions	Temporal Session Race Conditions (TSRC)	Forceful Browsing
Abuse of Functionality	Parameter Tampering	Session Variable Overloading	Session Fixation	Session Hijacking
Session Prediction	Binary Planting	Connection String Parameter Pollution	HTTP Parameter Pollution	Insecure Object Mapping
Oracle Padding	Reflected XSS	Persistent XSS	DOM XSS	Open Redirect
CSRF	Dynamic CSRF	SDRF	Click-Jacking	Cross Frame Scripting
Cross Site Tracing	Frame Spoofing	Content Spoofing	CRLF Injection	HTTP Response Splitting
Policy Abuse	Log Forging	HTTP Verb Tampering	HTTP Methods Abuse	Cross Site History Manipulation
Denial of Service	Distributed Denial of Service	Numeric Denial of Service	Application Denial of Service	Account Lockout
Regular Expression Denial of Service	Beast Attack	SSL/TSL Renegotiation Flaw	Replay Attack	Man-In-The-Middle
SQL Row Injection	Information Disclosure	Caching	Auto Complete	Fingerprinting
Policy Violation	Uncaught Exception	Weak Cryptography	Broken Access Control	Poor Logging Practice
Source Code Disclosure	Inefficient Logout	Credentials Disclosure	Unrestricted File Upload	Obsolete Files
Insecure Password Recovery Process	Insecure Transport	Insecure Cookie	Hard-Coded Passwords	HTTP Request Injection
XXE	Mail Headers Injection			



The Limited Time Frame (Cont.)

#tests = ~100 tests per each parameter

#pages = different web pages in the application

#params = 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.)

#tests * #pages * #params

100

20

3

=

6,000 tests



The Limited Time Frame (Cont.)

#tests * #pages * #params

100

2

3

=

6,000 tests



The Limited Time Frame (Cont.)

$$\begin{array}{ccc} \#tests & * & \#pages * \#params \\ 100 & & 3 \\ & = & \\ & 6,000 \text{ tests} & \end{array}$$



The Limited Time Frame (Cont.)

#tests * #pages * #params

100

100

3

=

30,000 tests



The Limited Time Frame (Cont.)

!!!30,000



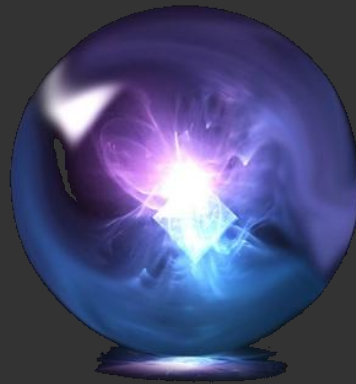
The Limited Time Frame, Potential Solutions

- ▶ **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...**



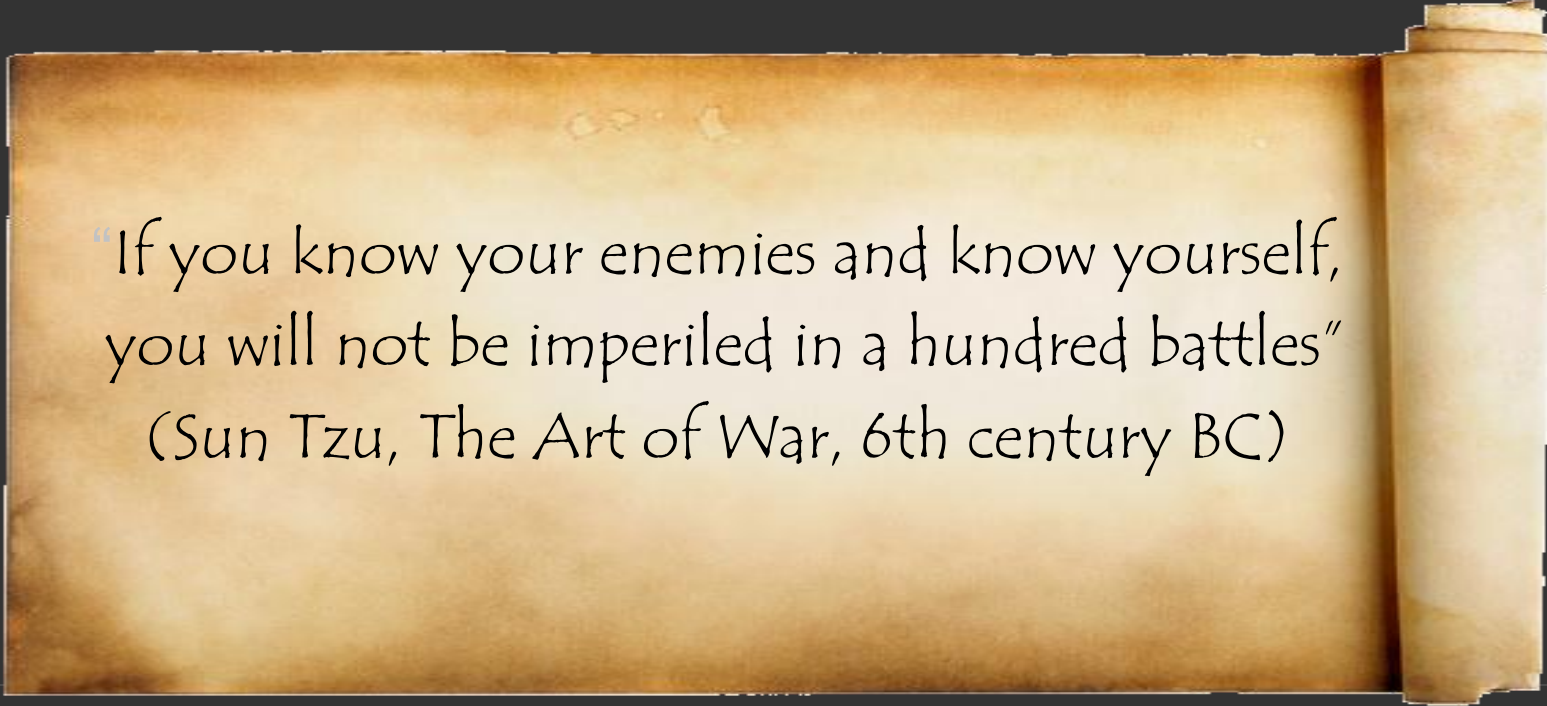
Gazing into the Crystal Ball

The Art of War: Information Gathering



Introduction to Digital 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)



The Risks of Information Disclosure

- ▶ Underlying Infrastructure and Associated Vulnerabilities
- ▶ Entry Point Structure
- ▶ Technology-in-Use
- ▶ Hidden Interfaces
- ▶ Potential Flaws
- ▶ Etc.



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.



“Secure” App. Behavior...

- ▶ Employs, and in some cases relies, on the use of **obscurity** to improve the state of **security**.
- ▶ Does **not** reveal sensitive information on the underlying infrastructure, or the internal structure of the application.
- ▶ Does **not** disclose exceptional erroneous information.
- ▶ Does **not** disclose source code.

Was it all in vain...?



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.

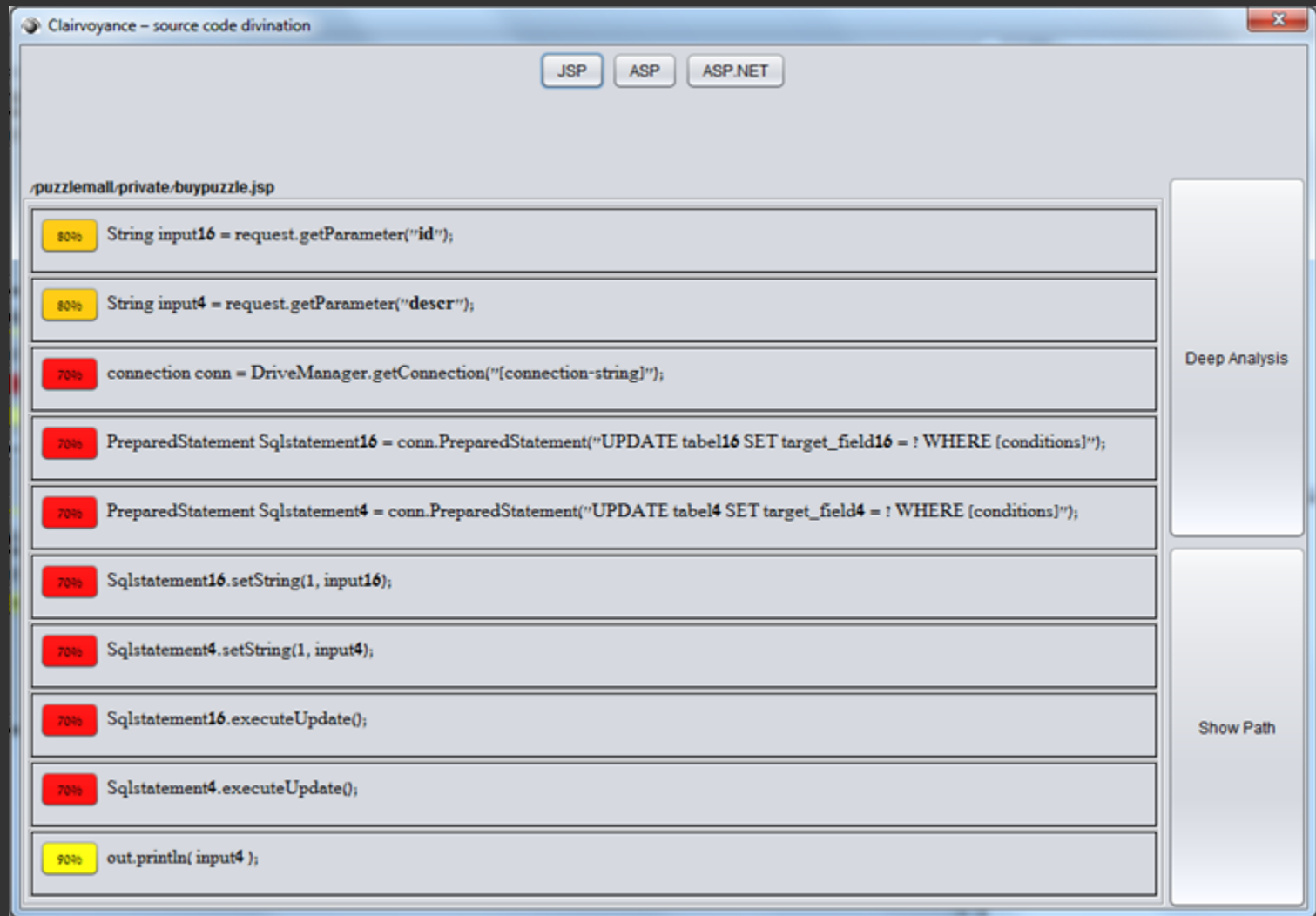


Source Code Divination – Benefits

- ▶ 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



Direct & Indirect Cross Entry Point Effect

Visual Entry Point Input-Output Correlation

Options

Results Requests

Location	Input Parameters
Session	

./puzzlemall/private/mainmenu.jsp

Location	Input Parameters
	username password

./puzzlemall/login.jsp

Location	Input Parameters
Database	

./puzzlemall/private/vieworders.jsp

Location	Input Parameters
Output	descr id purchase

./puzzlemall/private/buypuzzle.jsp

Location	Input Parameters
Session	

./puzzlemall/private/viewprofile.jsp

Location	Input Parameters
Output	username email password recoverya...

./puzzlemall/register-phase2.jsp

Location	Input Parameters
	username

./puzzlemall/recovery-phase2.jsp

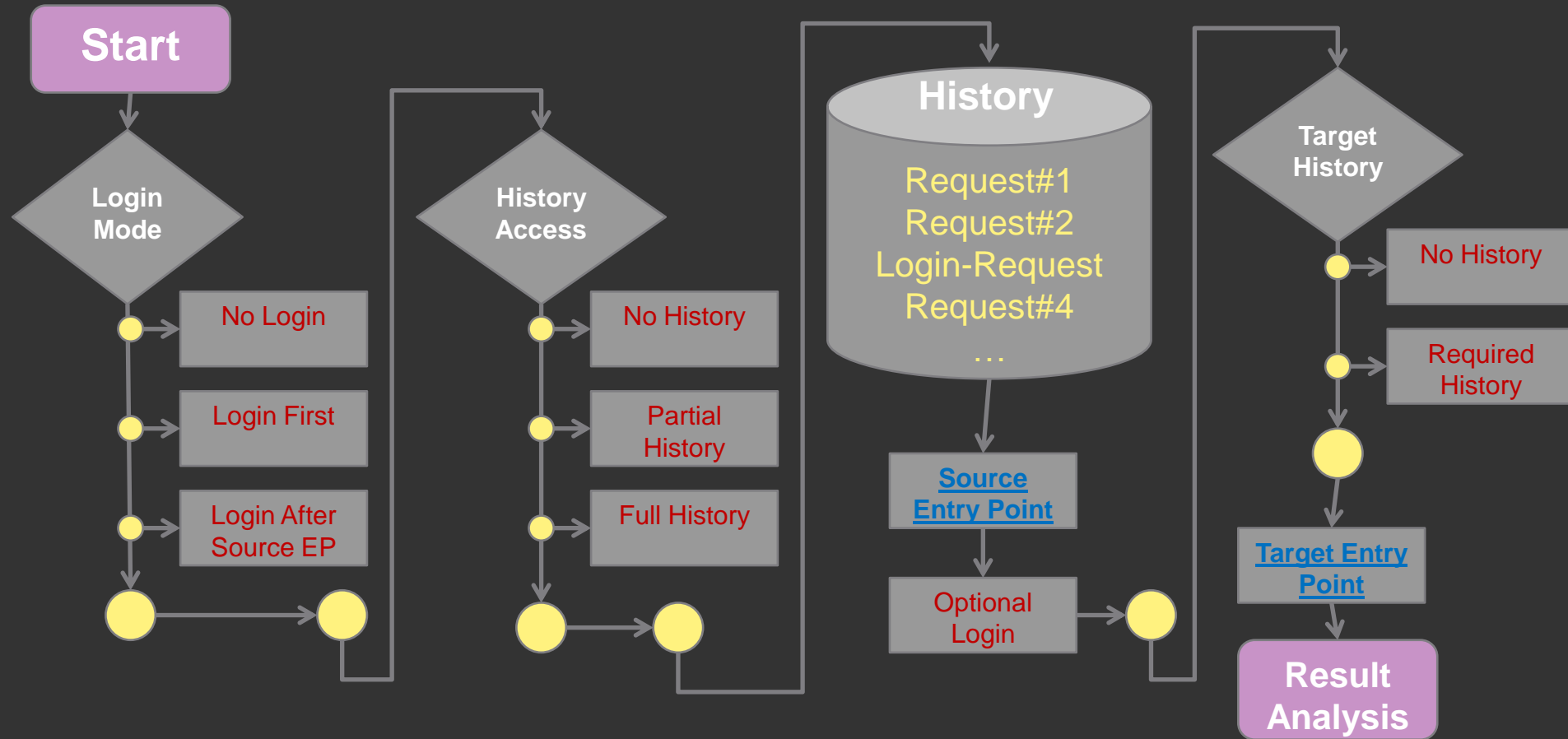


Divination Attacks



Exploring Different Paths of Execution

Behavior in Different Authentication Modes and History Perquisites



Gathering Information from Different Paths

- ▶ Generate as many different events as possible
- ▶ Analyze the behavior of entry points in each scenario
- ▶ Use a set of rules to decipher the possible reasons for each behavior
- ▶ Provide a clear visual representation of the interpreted information.
- ▶ Can be further analyzed using AI (thus, making the process produce scanner-like results) or manually provide the tester information that he wasn't aware of.

Which can lead to...



Process Visualization & Behavior Analysis

- ▶ Additional information can be obtained by analyzing behaviors (normal & abnormal).
- ▶ Each behavior can provide insight into server side processes, states and code.
- ▶ The process can be enhanced by actively causing a wide array of events (a.k.a active information disclosure).
- ▶ Massive active information disclosure can be used to gather information more effectively.

And also...



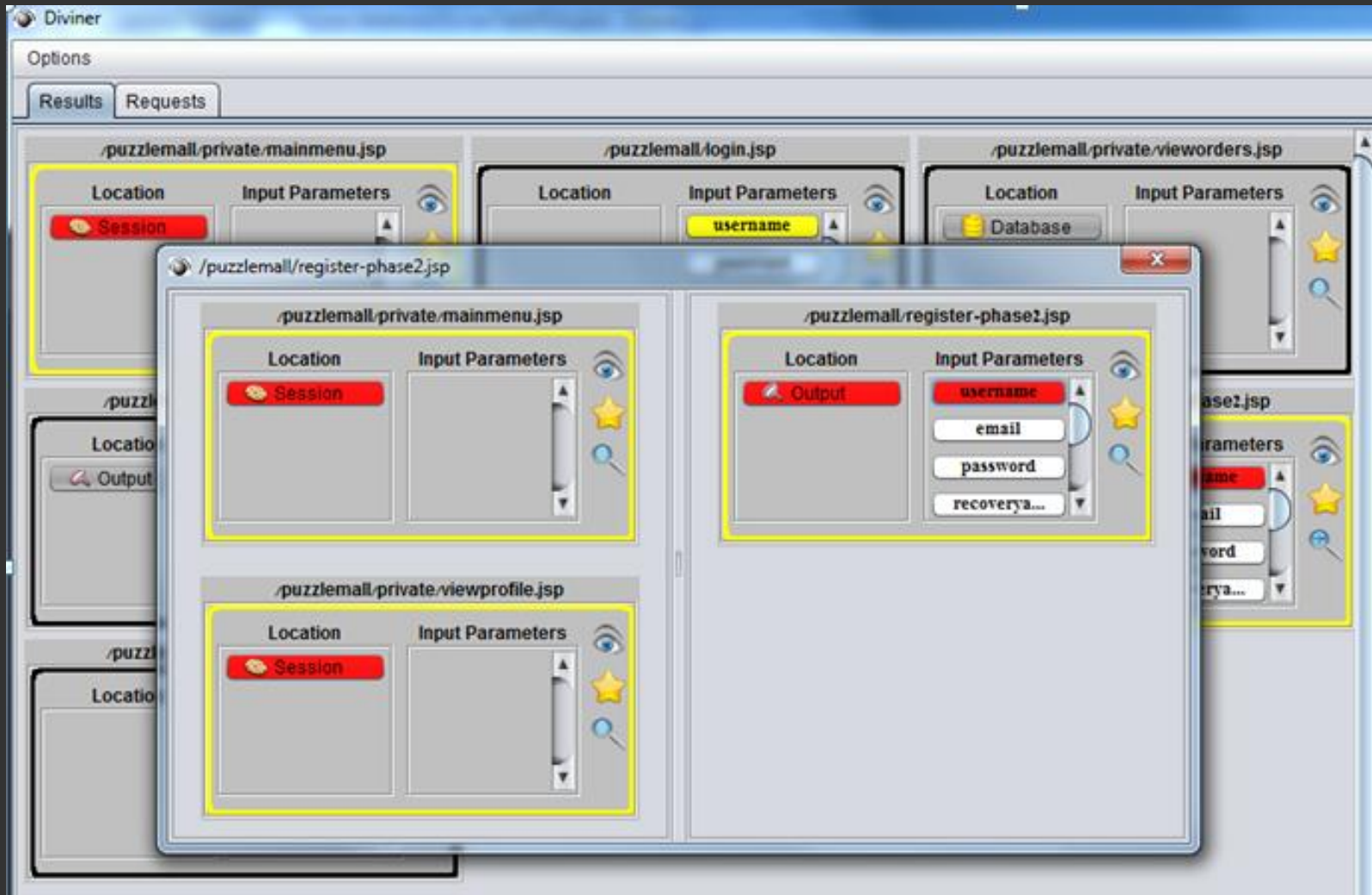
Initial Analysis and Structure Visualization

- ▶ Prominent behaviors and exceptional events can provide **leads for potential vulnerabilities**, without requiring exposure specific payloads (similar to a passive scanner).
- ▶ Can also be used to:
 - ▶ Identify the type of storage used to host each input (variable, session, database, etc).
 - ▶ Identify entry points that indirectly affect others.
 - ▶ Identify which input parameters are responsible for specific behaviors (using valid input, unlike scanners).
 - ▶ Identify different input parameters that affect identical server resources.
 - ▶ Work despite of IDS/IPS/WAF.



Entry Point Specific Cross Page Effect

Visual Entry Point Input-Output Correlation



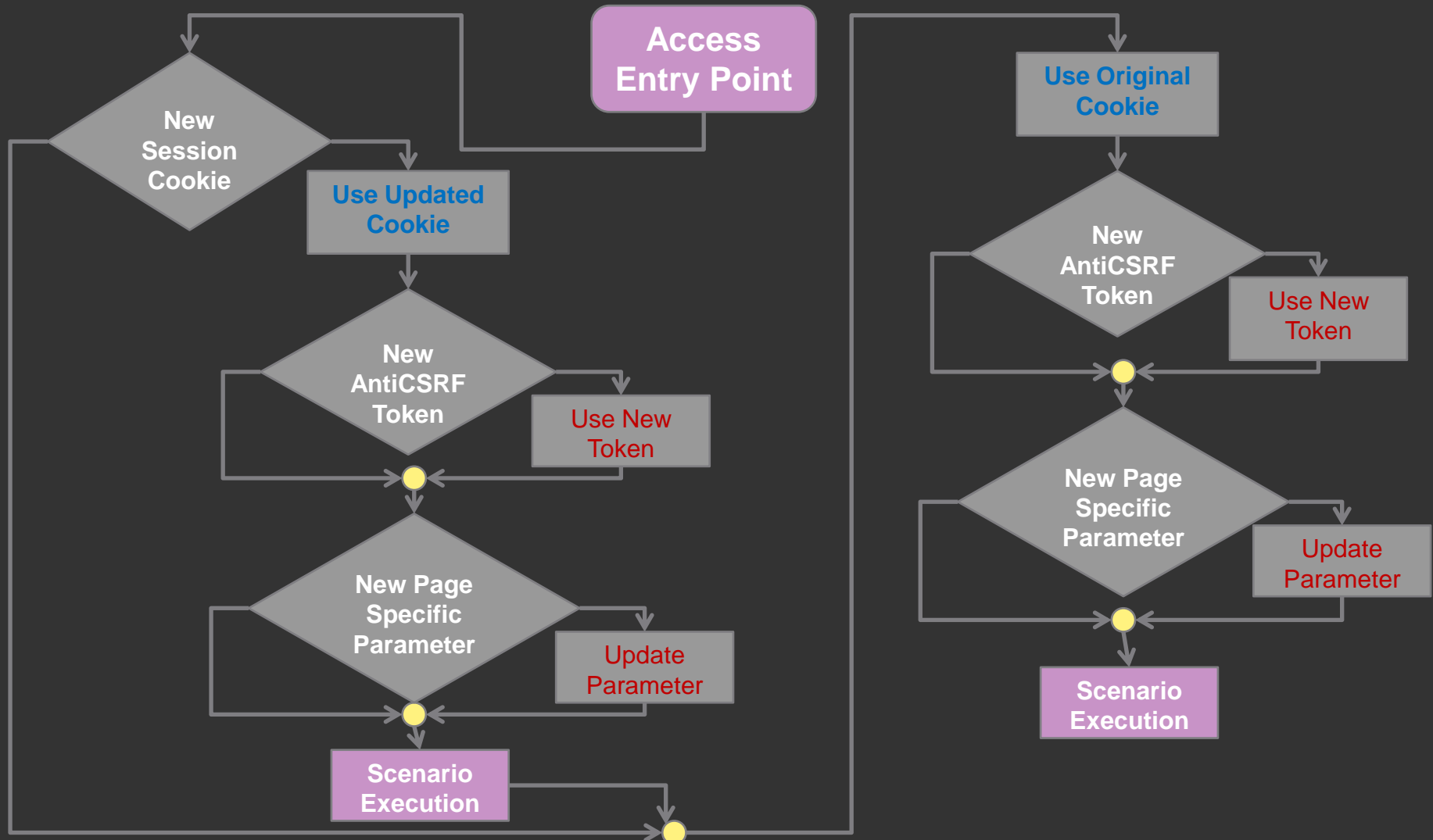
Primary & Secondary Conclusions

- ▶ Each behavior might be related to a potential security flaws, **but** can also indicate the existence of a specific process, line of code or restriction.
- ▶ The process can lead to secondary conclusions, such as:
 - ▶ Does session variables and identity tokens really expire, or continue to be associated with information, even when replaced?
 - ▶ Which entry point affects the behavior and content of other entry points, and how?
 - ▶ Where does the server store client originating inputs (variable/session/database/file)?
 - ▶ Which line of code in the server is responsible for the behavior?
 - ▶ Does input fields from different entry points affect identical server resources?



Exploring Different Paths of Execution, Cont.

Behavior With Different Session Cookies, Identifiers and Tokens



Source Code Divination Accuracy

ID	Behaviour Name
1	Input Reflected from Variable
2	Input Reflected from Session
3	Input Reflected from Database
4	Input Stored in Server Variable
5	Input Stored in Session Variable
6	Input Stored in Database Table
7	New Cookie Value
...	...



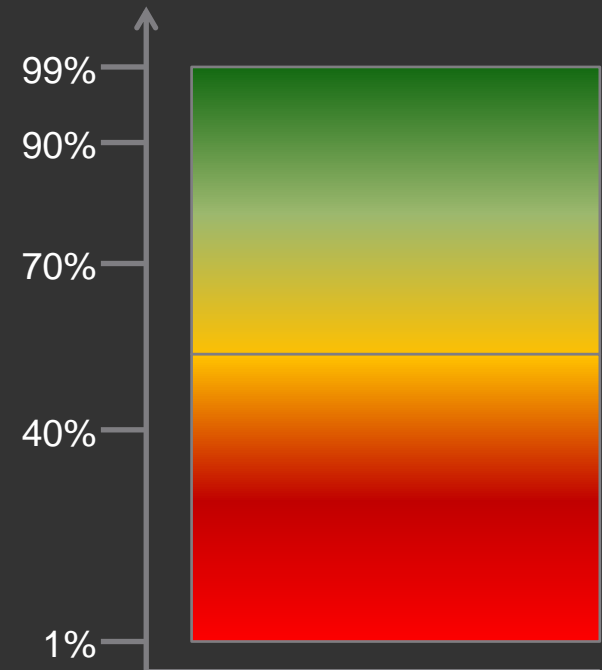
Source Code Divination Accuracy

ID	Code Description	JSP Code	ASP.Net Code	...
1	Read Input to Variable	String input\$\$1\$\$ = request.getParameter(##1##);	String input\$\$1\$\$ = Request["##1##"];	
2	Invalidate Session	session.invalidate();	Session.Abandon();	
3	New Session Identifier	request.getSession(true);	...	
4	New Cookie Value	Cookie cookie = new Cookie ("##1##",val); response.addCookie(cookie);	Response.Cookies("##1##").Value = "val";	
5	Get Database Connection	Class.forName(DriverClassName); Connection conn = DriverManager.getConnection(X);	SqlConnection conn = new SqlConnection(X);	
...



Source Code Divination Accuracy

Behavior ID	Code ID	Code Type	Rank	Default Probability
7	3	1	1010	50%
7	4	1	10040	70%
7	2	2	5550	40%
6	1	1	2010	90%
6	5	2	10000	80%
...

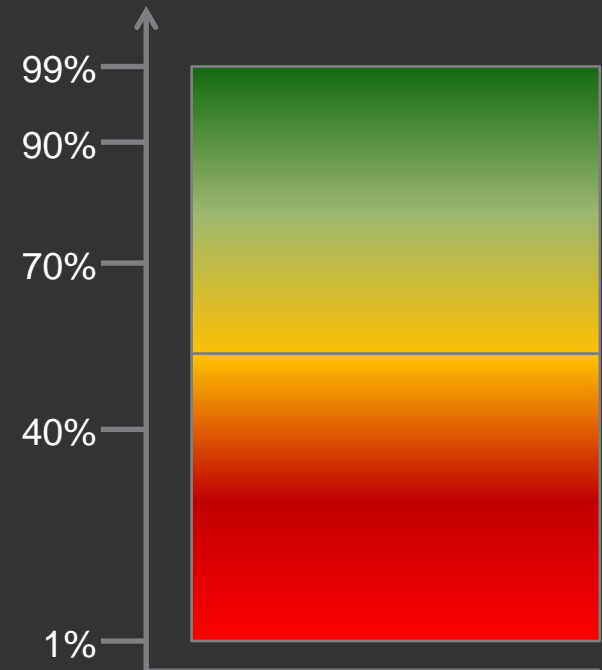


Verification Process and Probability

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%

Behavior ID	Code ID	Code Type	Rank	Current Probability
7	3	1	1010	70%
7	4	1	10040	60%
7	2	2	5550	80%
6	1	1	2010	90%
6	5	2	10000	80%
...



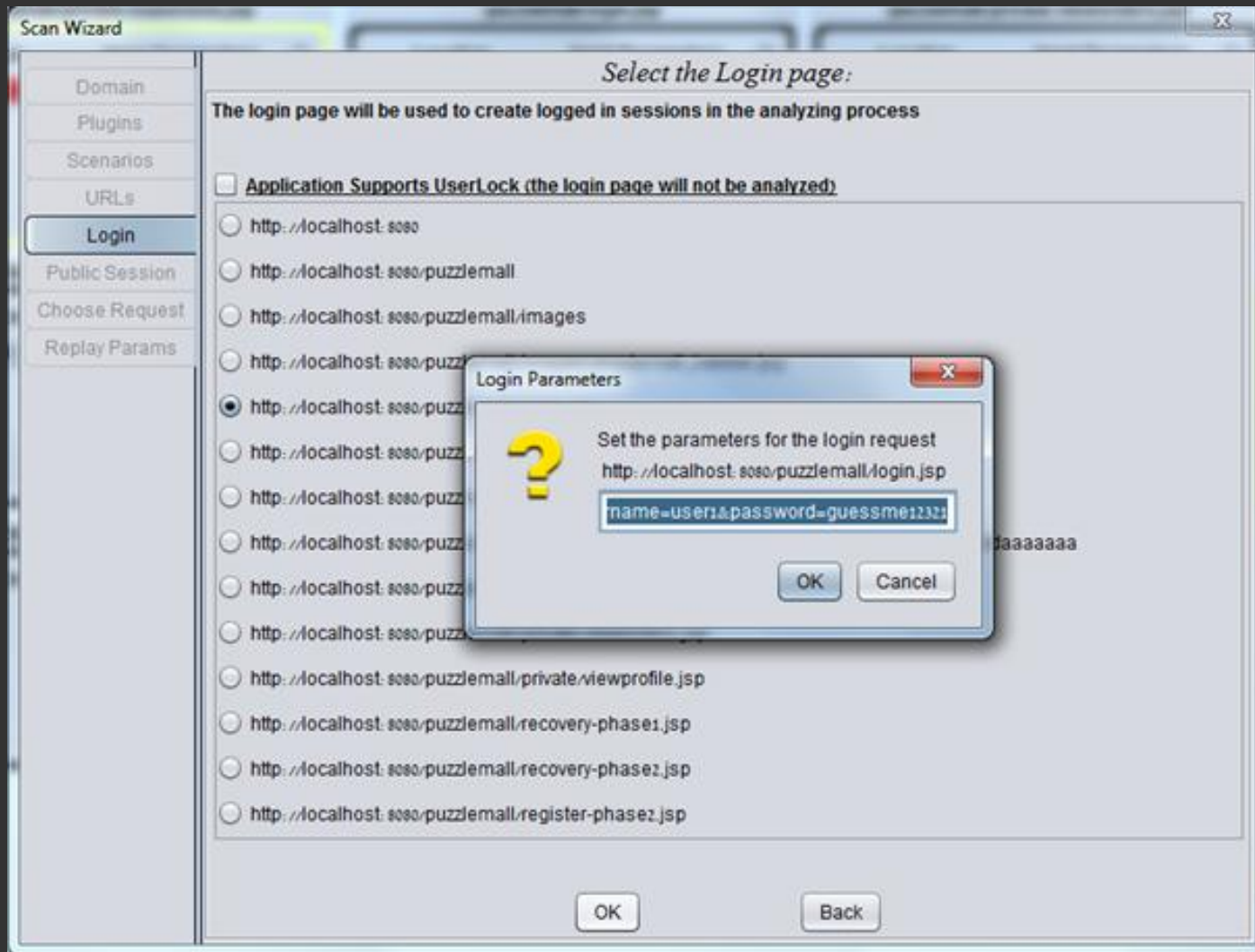
Diviner

A New ZAP Extension

Live Demo!



Divination Wizard – Record Login Scenario



Divination Wizard – Handle CSRF Barriers

Scan Wizard

Domain
Plugins
Scenarios
URLs
Login
Public Session
Choose Request
Replay Params

CSRF Tokens and Replayable Parameters:

The Replayable parameters' value is generated by the application and cannot be modified. These include anti-CSRF tokens and Viewstate.

The "Per Page" tokens are referring to parameters that may have different values for different pages, such as Viewstate.

The "Per Application" tokens are referring to parameters that normally have the same value across different pages

Add Token

Token name	Per page	Active
VIEWSTATE	<input checked="" type="checkbox"/>	<input type="checkbox"/>
EVENTTARGET	<input checked="" type="checkbox"/>	<input type="checkbox"/>
EVENTARGUMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>
LASTFOCUS	<input checked="" type="checkbox"/>	<input type="checkbox"/>
VIEWSTATEENCRYPTED	<input checked="" type="checkbox"/>	<input type="checkbox"/>
LASTFOCUS	<input checked="" type="checkbox"/>	<input type="checkbox"/>
VSTATE	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PREVIOUSPAGE	<input checked="" type="checkbox"/>	<input type="checkbox"/>
REQUESTDIGEST	<input checked="" type="checkbox"/>	<input type="checkbox"/>
anticsrf	<input type="checkbox"/>	<input type="checkbox"/>
CSRFToken	<input type="checkbox"/>	<input type="checkbox"/>
RequestVerificationToken	<input type="checkbox"/>	<input type="checkbox"/>

OK Back



Divination Wizard – Define Analysis Mode

Scan Wizard

Domain
Plugins
Scenarios
URLs
Login
Public Session
Choose Request
Replay Params

Scan scenarios:

At least one analyzing process and one history mode need to be selected

Analyzing Scenarios

- ☒ Login First
- ☒ Public Direct
- ☒ Login After

History Modes

- ☒ No History
- ☒ Partial History
- ☒ Full History

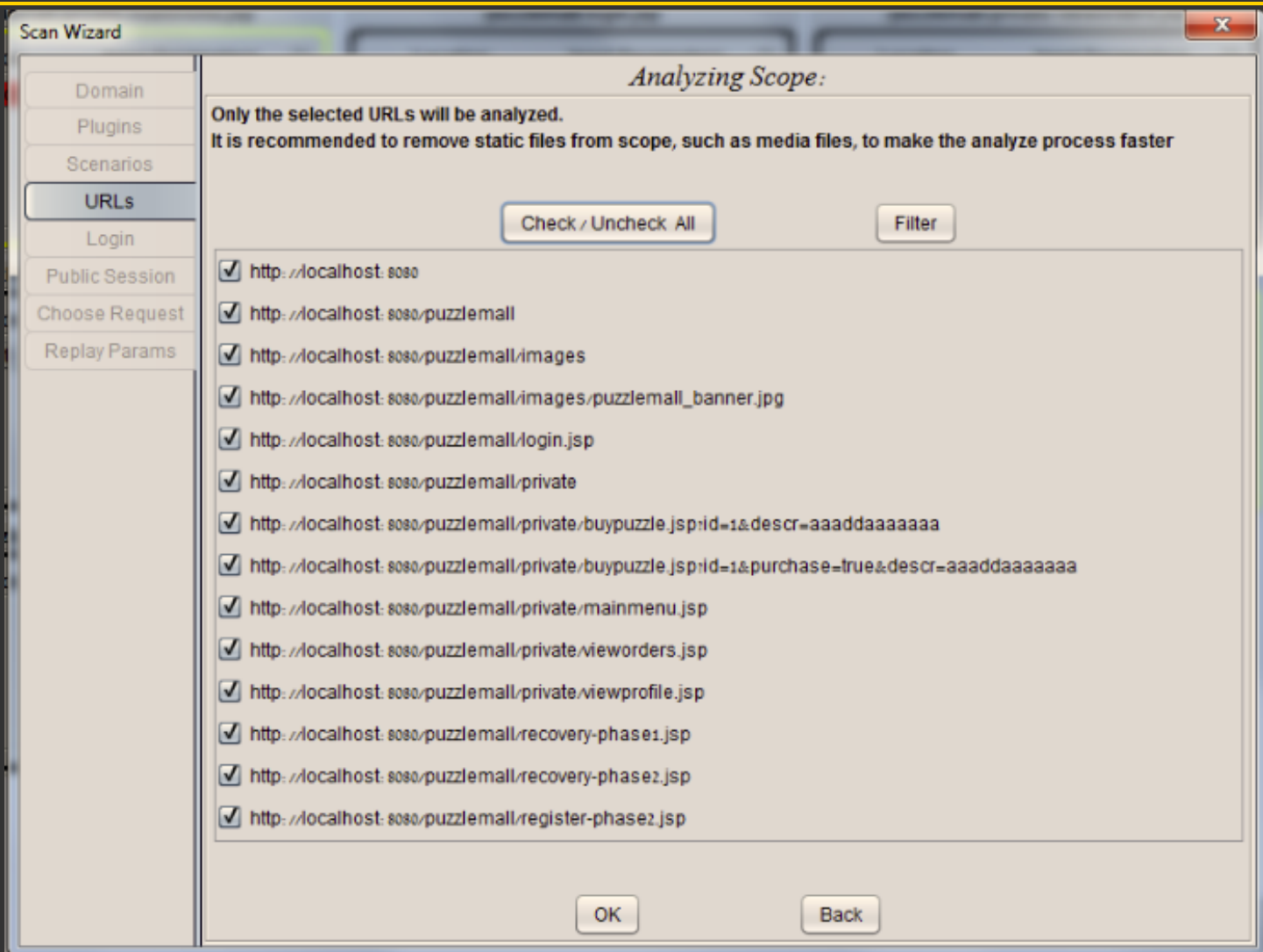
Verify Mode

- ☒ Safer and More Accurate scan

OK Back



Divination Wizard – Define Analysis Scope



Visual Penetration Testing & Payload Reuse

Diviner

Options

Results Requests

localhost

Update Domains Import Requests

/puzzlemall/register-phase2.jsp

Statistics	Input Parameters
Method: POST	email
Zap ID: 1	password
# Params: 6	recoverya...
Code: 200	recoveryq...
SSL: false	

/puzzlemall/images/puzzlemall_banner.jpg

Statistics	Input Parameters
Method: GET	
Zap ID: 4	
# Params: 0	
Code: 200	
SSL: false	

/puzzlemall/recovery-phase2.jsp

Statistics	Input Parameters
Method: POST	username
Zap ID: 6	
# Params: 1	
Code: 302	
SSL: false	

/puzzlemall/recovery-phase1.jsp

Statistics	Input Parameters
Method: GET	
Zap ID: 7	
# Params: 0	
Code: 200	
SSL: false	

/puzzlemall/private/mainmenu.jsp

Statistics	Input Parameters
Method: GET	
Zap ID: 8	
# Params: 0	
Code: 200	
SSL: false	

/puzzlemall/private/viewprofile.jsp

Statistics	Input Parameters
Method: GET	
Zap ID: 10	
# Params: 0	
Code: 200	
SSL: false	

/puzzlemall/private/buypuzzle.jsp

Statistics	Input Parameters
Method: GET	descr
Zap ID: 11	id
# Params: 2	
Code: 200	
SSL: false	

/puzzlemall/private/buypuzzle.jsp

Statistics	Input Parameters
Method: GET	descr
Zap ID: 12	id
# Params: 3	purchase
Code: 200	
SSL: false	

/puzzlemall/private/vieworders.jsp

Statistics	Input Parameters
Method: GET	
Zap ID: 13	
# Params: 0	
Code: 200	
SSL: false	

/puzzlemall/login.jsp

Statistics	Input Parameters
Method: POST	password
Zap ID: 14	username
# Params: 2	
Code: 200	

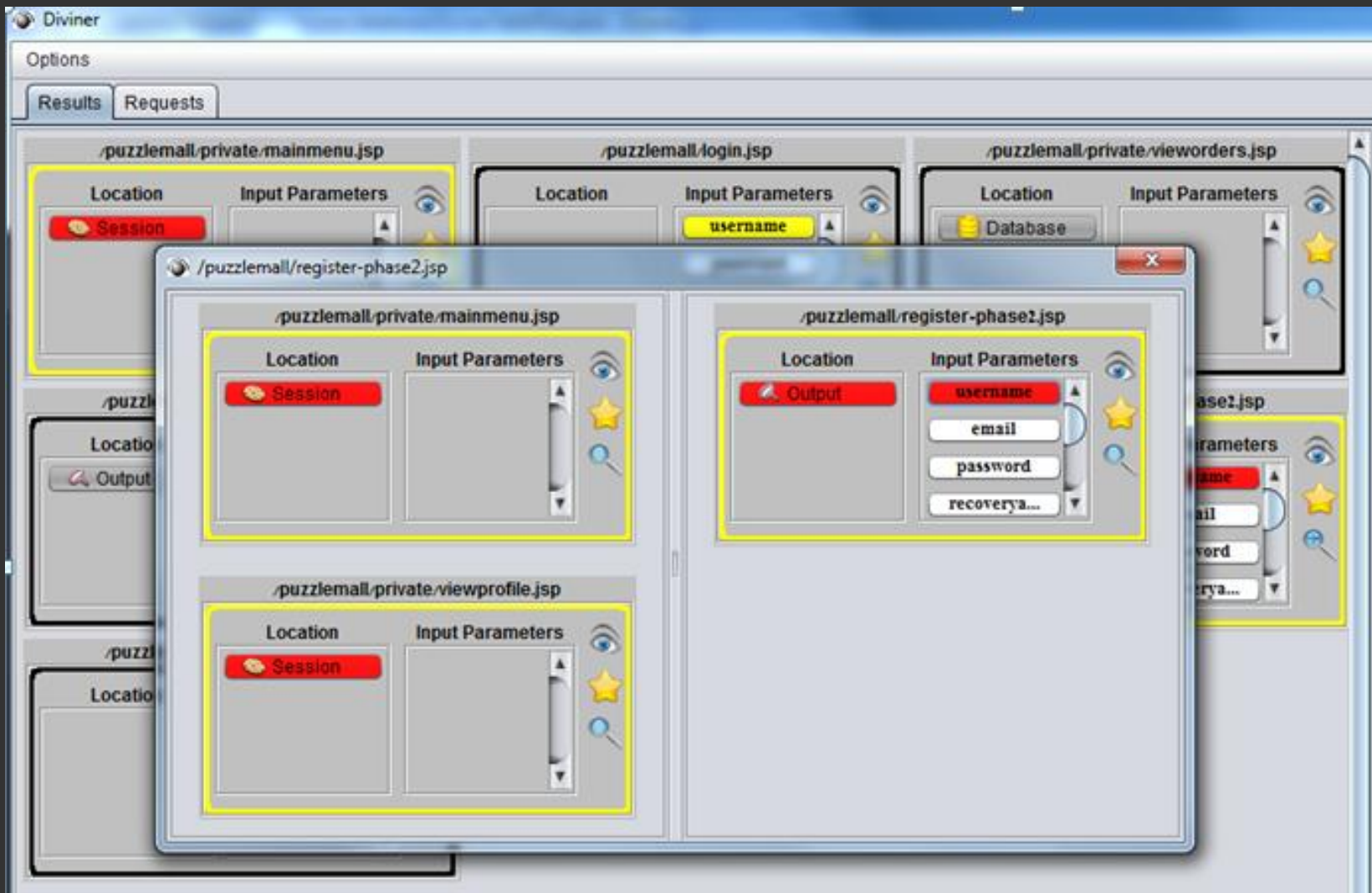
Tasks / Leads

- XSS
- SQL Injection
- Session Puzzle

Advisor



Visual Entry Point Input - Output Correlation



Entry Point Structure & Source Visualization

The screenshot displays the Diviner application interface. On the left, a sidebar contains a tree view with nodes for 'Location' and 'Session'. The main window is titled 'Clairvoyance - source code divination' and features tabs for 'JSP', 'ASP', and 'ASP.NET'. The selected 'JSP' tab shows the source code for the file `/puzzlemall/private/vieworders.jsp`. The code is displayed in a list of lines, each preceded by a red '70%' marker. The code includes SQL queries and database operations. On the right side of the main window, there are two buttons: 'Deep Analysis' and 'Show Path'. The application also has a top menu bar with 'Options', 'Results', and 'Requests' tabs, and a right sidebar with 'Tasks / Leads' and 'XSS' sections.

```
70% PreparedStatement stmt16 = conn.prepareStatement("SELECT sourceField16, [FieldCollection] FROM tabel16[Table Collection] WHERE [criteria]");  
70% PreparedStatement stmt4 = conn.prepareStatement("SELECT sourceField4, [FieldCollection] FROM tabel4[Table Collection] WHERE [criteria]");  
70% ResultSet rs16 = stmt16.executeQuery();  
70% ResultSet rs4 = stmt4.executeQuery();  
70% if (rs16.next())  
70% { out.println(rs.getString(1)); }  
70% if (rs4.next())  
70% { out.println(rs.getString(1)); }
```



Source/Target Entry Points Code Correlation

The screenshot displays the Diviner application interface. The main window, titled "Clairvoyance - source code divination", shows a list of code snippets with their correlation percentages. The snippets are for the file `/puzzlemall/private/buypuzzle.jsp`. The correlation percentages are: 80%, 80%, 70%, 70%, 70%, 70%, 70%, 70%, and 90%.

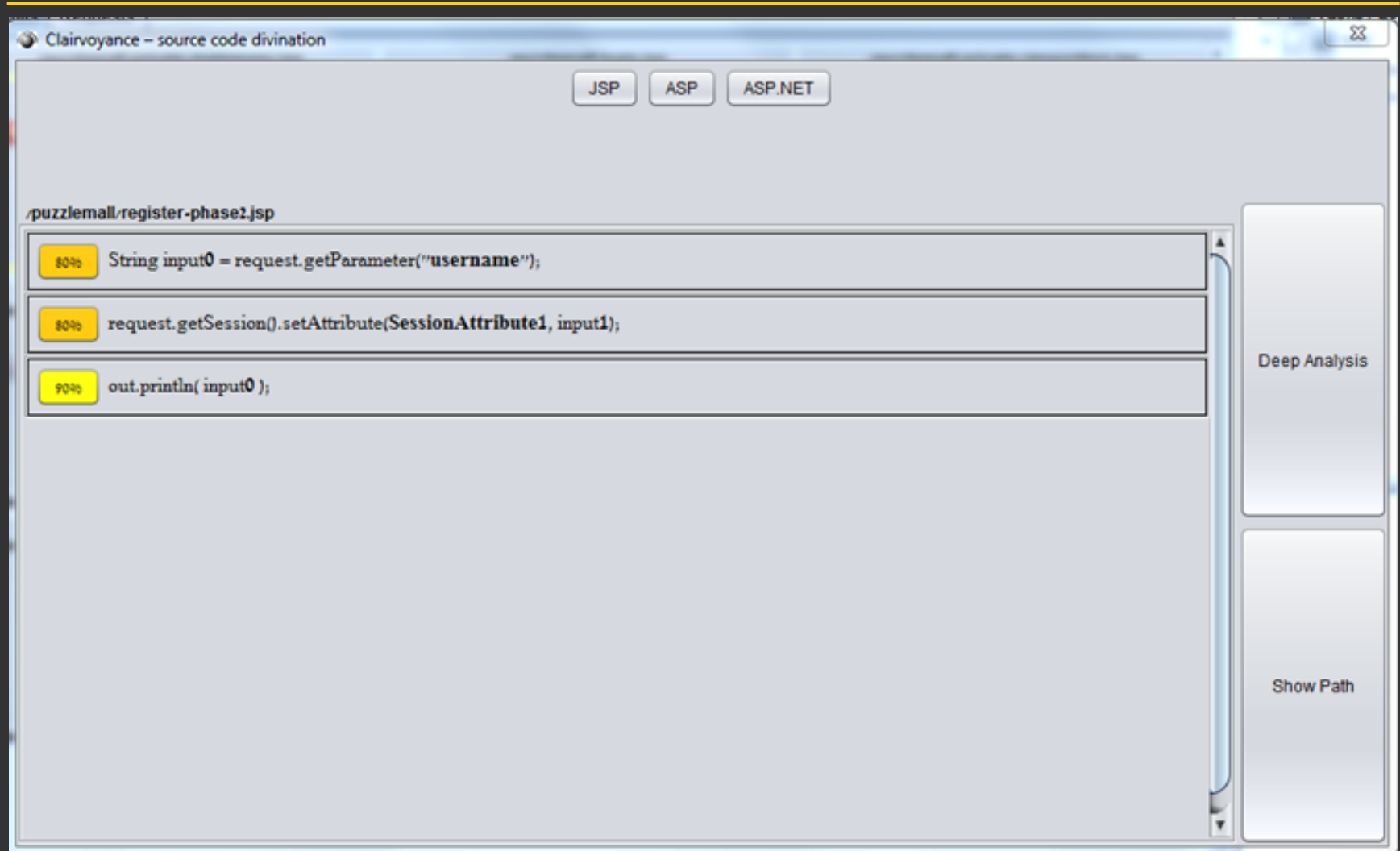
On the right side of the window, there are two buttons: "Deep Analysis" and "Show Path".

The background shows the Diviner application's main interface with tabs for "Options", "Results", and "Requests". The "Results" tab is active, showing a list of results for the file `/puzzlemall/private/buypuzzle.jsp`. The "Location" column shows the file path, and the "Input" column shows the input type (e.g., "Session", "Output").

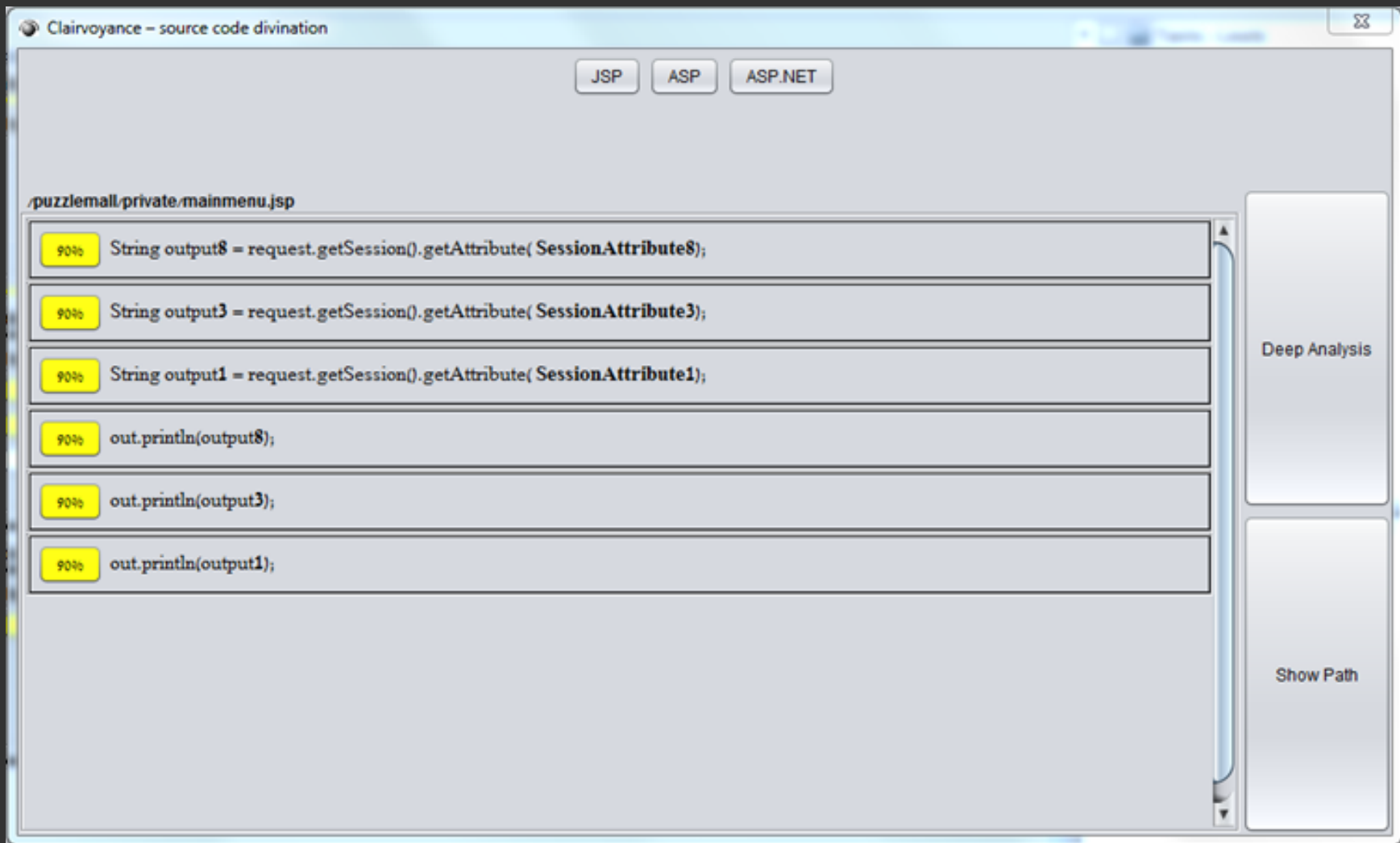
Correlation	Code Snippet
80%	<code>String input16 = request.getParameter("id");</code>
80%	<code>String input4 = request.getParameter("deser");</code>
70%	<code>connection conn = DriverManager.getConnection("[connection-string]");</code>
70%	<code>PreparedStatement Sqlstatement16 = conn.PreparedStatement("UPDATE tabel16 SET target_field16 = ? WHERE [conditions]");</code>
70%	<code>PreparedStatement Sqlstatement4 = conn.PreparedStatement("UPDATE tabel4 SET target_field4 = ? WHERE [conditions]");</code>
70%	<code>Sqlstatement16.setString(1, input16);</code>
70%	<code>Sqlstatement4.setString(1, input4);</code>
70%	<code>Sqlstatement16.executeUpdate();</code>
70%	<code>Sqlstatement4.executeUpdate();</code>
90%	<code>out.println(input4);</code>



Detect Indirect Attack Vectors – Source Page



Detect Indirect Attack Vectors – Target Page



The screenshot displays the Clairvoyance tool interface, titled "Clairvoyance – source code divination". It features three tabs: "JSP", "ASP", and "ASP.NET", with "JSP" currently selected. The main area shows the source code for the file `/puzzlemall/private/mainmenu.jsp`. The code is presented in a list of six lines, each preceded by a yellow box indicating a 90% confidence level. The code lines are:

- `String output8 = request.getSession().getAttribute(SessionAttribute8);`
- `String output3 = request.getSession().getAttribute(SessionAttribute3);`
- `String output1 = request.getSession().getAttribute(SessionAttribute1);`
- `out.println(output8);`
- `out.println(output3);`
- `out.println(output1);`

On the right side of the interface, there are two buttons: "Deep Analysis" and "Show Path".



Support Different Technologies

The screenshot displays the Diviner application interface. The main window is titled "Options" and has tabs for "Results" and "Requests". The "Requests" tab is active, showing a list of requests for the domain "localhost". Each request is represented by a card with the following information:

- URL: /testApp/Account/LoginPage.aspx, /testApp/Reflection.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/ViewSessionFromFile.aspx, /testApp/ViewSession.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Styles/Site.css
- Statistics: Method (POST, GET), Zap ID, # Params, Code, SSL
- Input Parameters: _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT...

On the right side of the interface, there is a sidebar with a tree view showing tasks and leads:

- Tasks / Leads
 - XSS
 - SQL Injection
 - Session Puzzle

Below the tree view is an "Advisor" section, which is currently empty.



Support Different Technologies

The screenshot displays the Diviner application interface. The main window is titled "Options" and has tabs for "Results" and "Requests". The "Requests" tab is active, showing a list of requests for the domain "localhost". Each request is represented by a card with the following details:

- URL: /testApp/Account/LoginPage.aspx, /testApp/Reflection.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/ViewSessionFromFile.aspx, /testApp/ViewSession.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Default.aspx, /testApp/Styles/Site.css
- Statistics: Method (POST, GET), Zap ID, # Params, Code, SSL
- Input Parameters: _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT..., _EVENT...

On the right side of the interface, there is a sidebar with a tree view showing tasks and leads:

- Tasks / Leads
 - XSS
 - SQL Injection
 - Session Puzzle

Below the tree view is an "Advisor" section, which is currently empty.



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



Source Code 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



Source Code Divination – Code Prediction

- ▶ 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

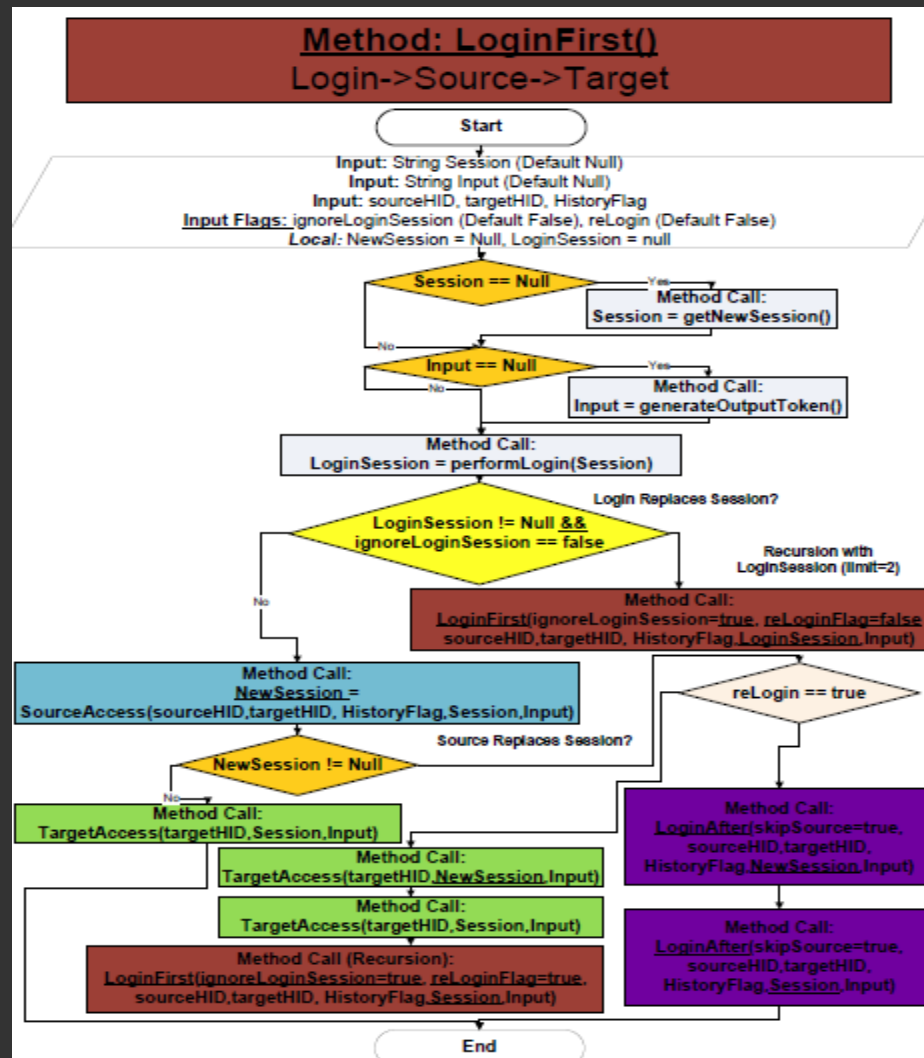


Source Code Divination – Code Prediction

- ▶ 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



Twists & Turns



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.

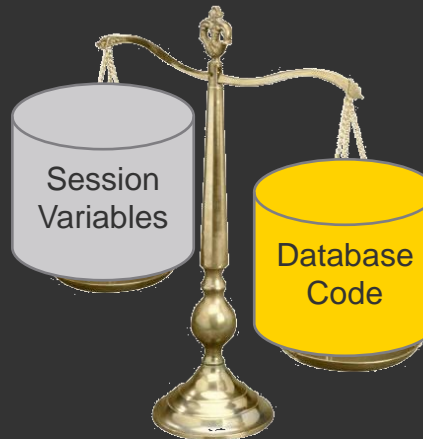
```
session.setAttribute(  
    SessionConstants.USERNAME_VARIABLE,  
    username);  
.  
.  
.  
.  
.  
.  
session.invalidate(); //invalidate session, erase all variables
```

Productive Latency

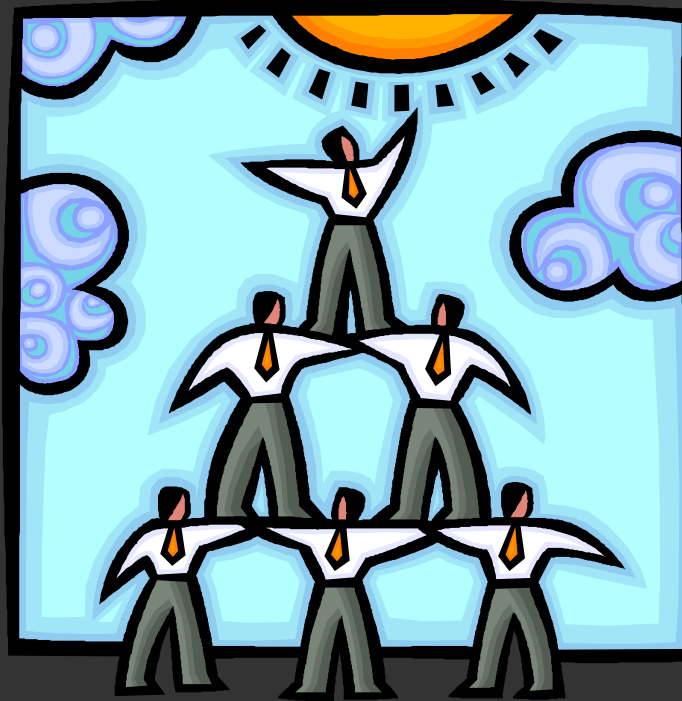


Productive Latency Rules

- ▶ 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, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa!

```
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 {
    ...
}
```



Occupying Connections to Increase Latency

- ▶ 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 then 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.

```
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
 - ▶ Home: <http://code.google.com/p/payload-manager/>
- ▶ OWASP ZAP Proxy:
 - ▶ <http://code.google.com/p/zaproxy/>



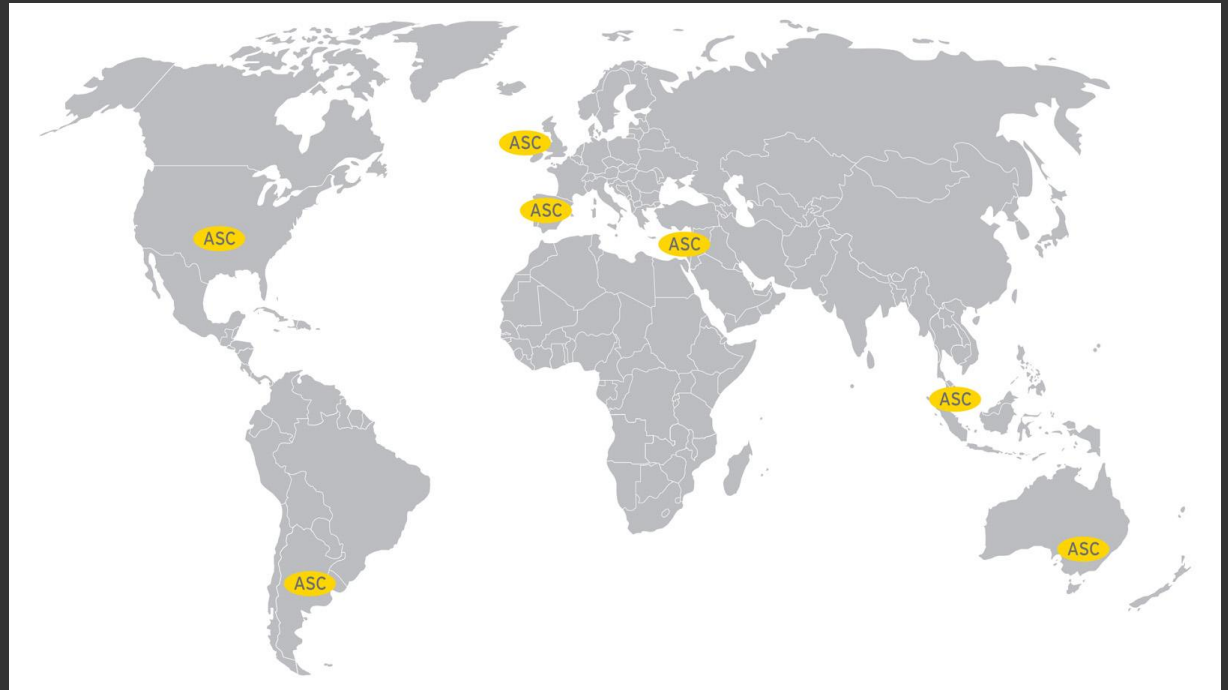
Acknowledgments

- ▶ **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
- ▶ EMEA
 - ▶ Dublin
 - ▶ Barcelona
- ▶ Asia Pacific
 - ▶ Singapore
 - ▶ Melbourne



Ernst & Young

Assurance | Tax | Transactions | Advisory

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](https://twitter.com/#!/Secure_ET))

