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

ZERO NIGHTS

About Me

- **Addictions**

About Me

- Security Tools Collector / Addict

About Me

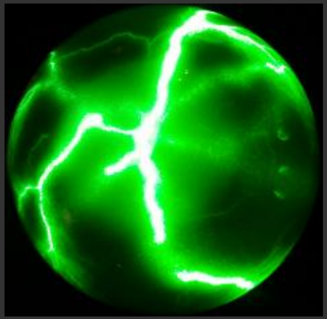
- Law of Familiarity

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+
  - 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)



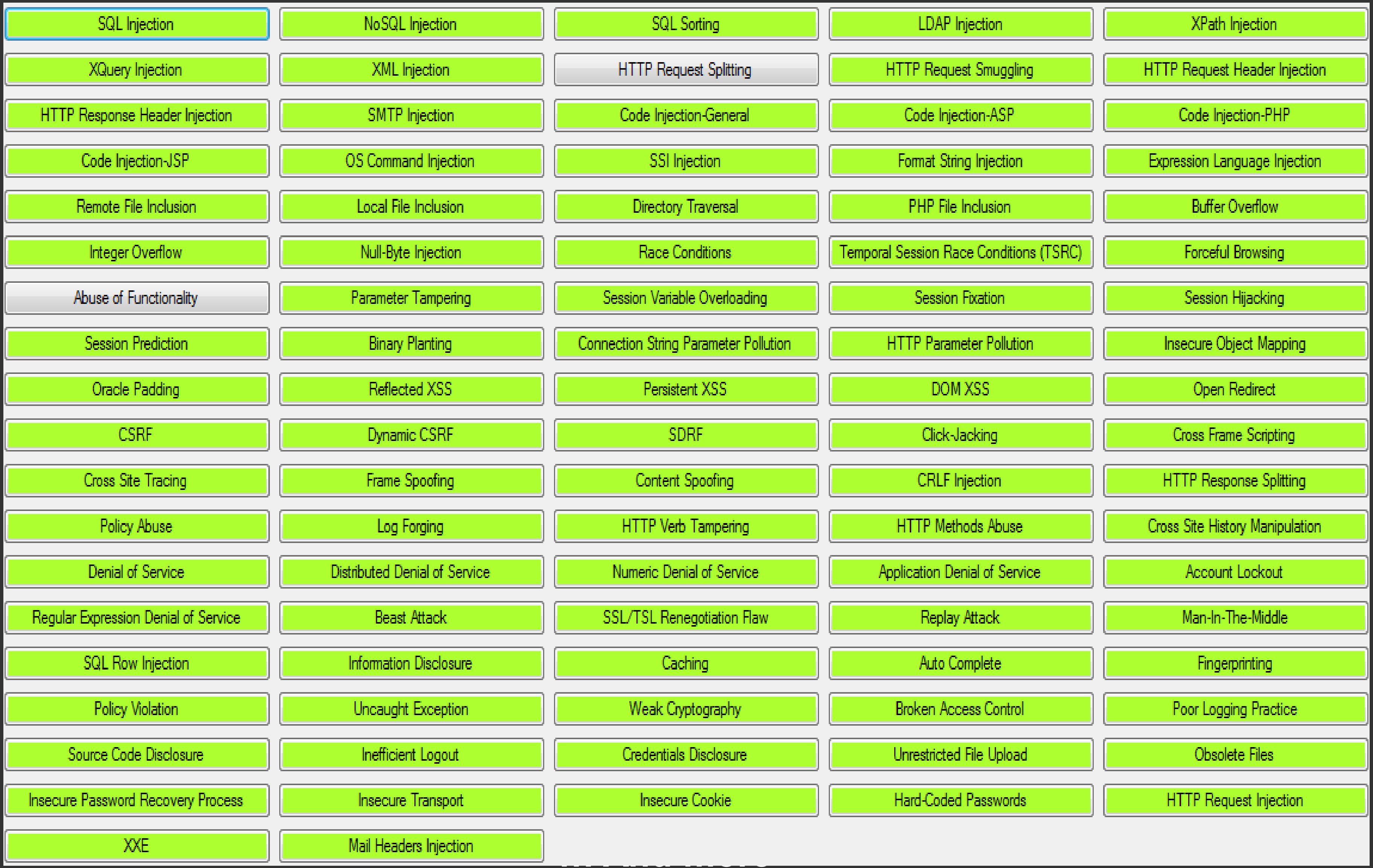
The Problem

The numerous tasks of manual penetration testing

Exploitation
Google Hacking
Attack Surface Discovery
...
Fingerprinting
...
Test Manually
Vulnerability Scanning

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.)

#tests * #pages * #params

100 3

=

6,000 tests

The Limited Time Frame (Cont.)

#tests * #pages * #params

100 100 3

=

30,000 tests

The Limited Time Frame (Cont.)

!!!30,000

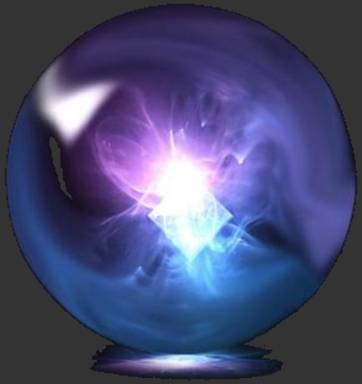
The Limited Time Frame, Potential Solutions

- 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

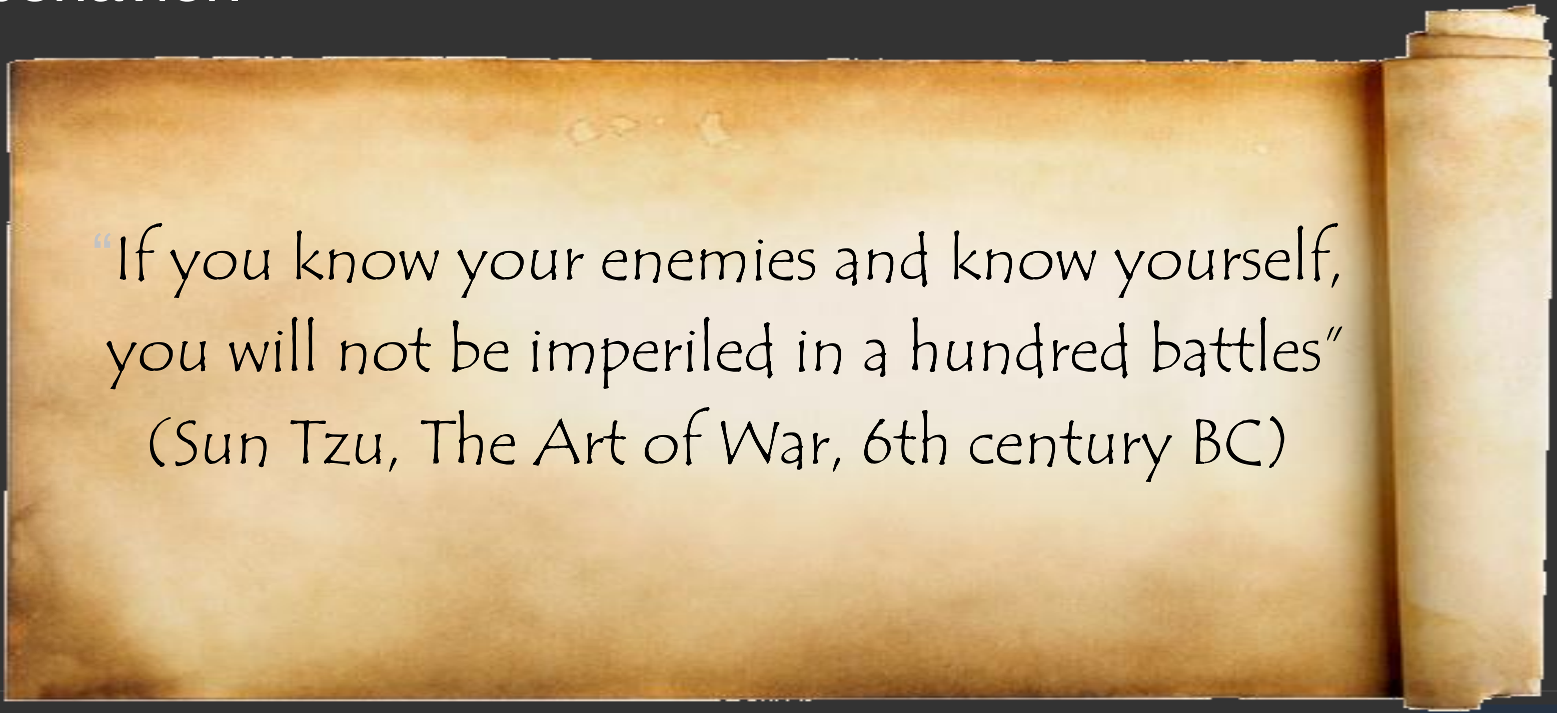
The Art of War:

Information Gathering



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)

Passive Information Gathering

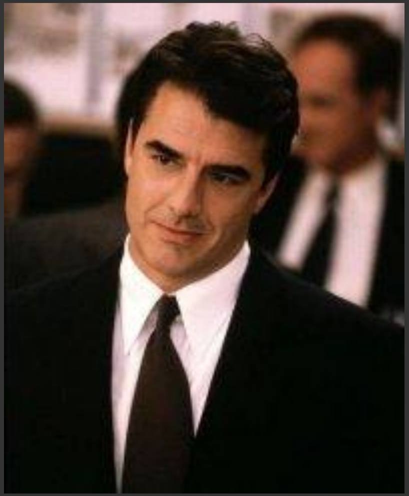
- Dictionary term: “accepting or allowing what happens or what others do, without active response or resistance.”
- Application-level passive analysis is performed using techniques such as:
  - Google hacking
  - Entry point mapping
  - Content analysis tools:
    - **Watcher**, ZAP, WebFight , Etc.
  - 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.

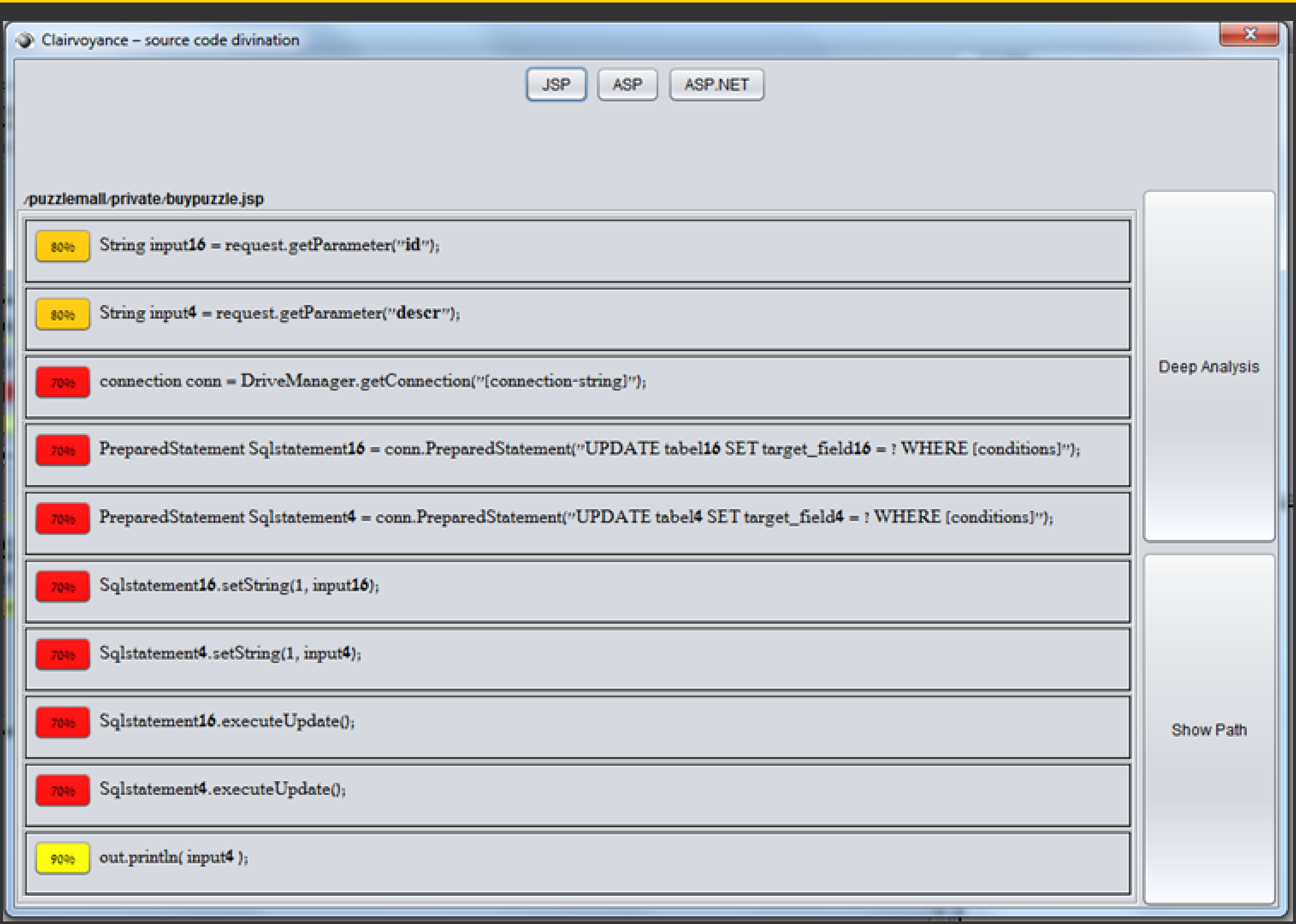
Security by Obscurity – Officially Dead?

- 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

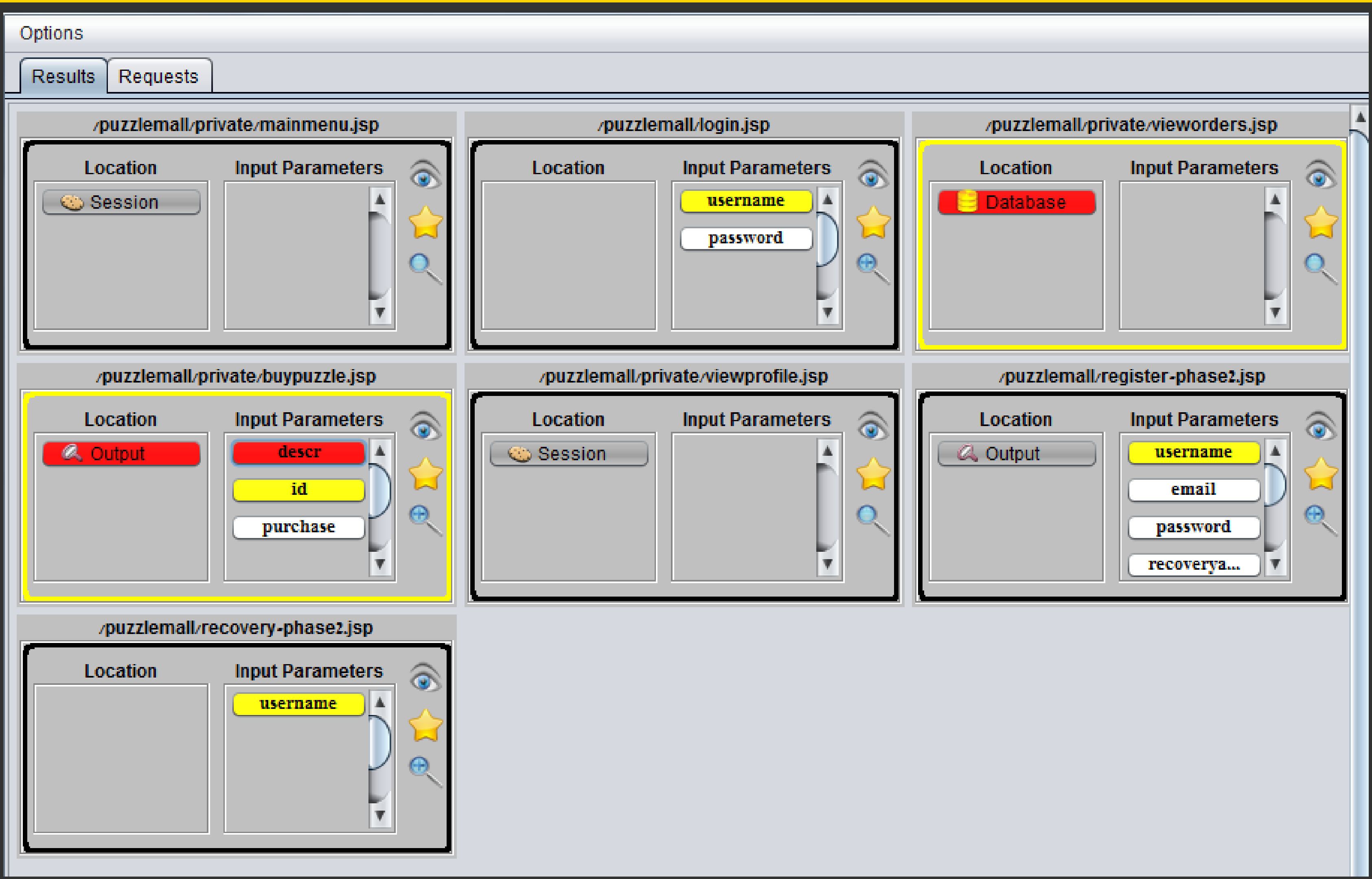


/puzzlemall/private/buypuzzle.jsp

```
80%  String input16 = request.getParameter("id");
80%  String input4 = request.getParameter("descr");
70%  connection conn = DriveManager.getConnection("[connection-string]");
70%  PreparedStatement Sqlstatement16 = conn.PreparedStatement("UPDATE tabel16 SET target_field16 = ? WHERE [conditions]");
70%  PreparedStatement Sqlstatement4 = conn.PreparedStatement("UPDATE tabel4 SET target_field4 = ? WHERE [conditions]");
70%  Sqlstatement16.setString(1, input16);
70%  Sqlstatement4.setString(1, input4);
70%  Sqlstatement16.executeUpdate();
70%  Sqlstatement4.executeUpdate();
90%  out.println( input4 );
```

Direct & Indirect Cross Entry Point Effect

Visual Entry Point Input-Output Correlation



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

/puzzlemall/login.jsp
Location:
Input Parameters: username, password

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

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

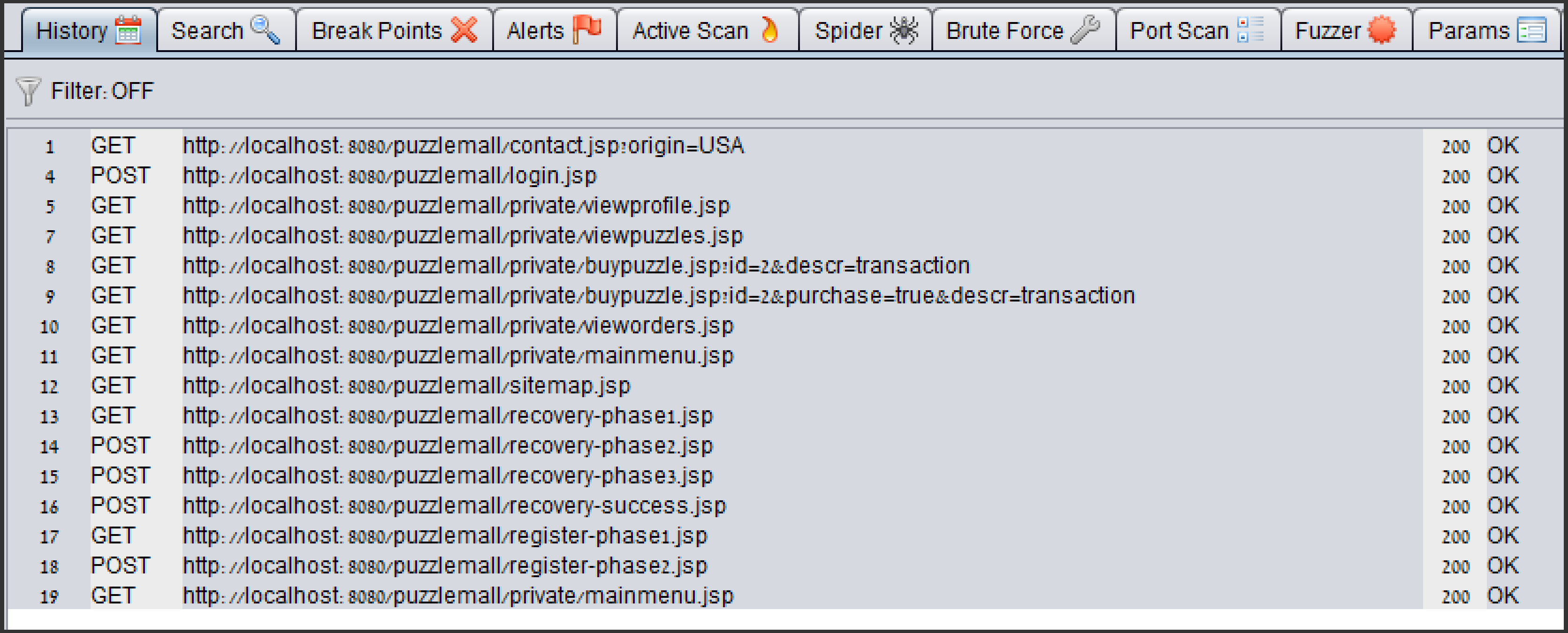
/puzzlemall/private/viewprofile.jsp
Location: Session
Input Parameters:

/puzzlemall/register-phase2.jsp
Location: Output
Input Parameters: username, email, password, recoverya...

/puzzlemall/recovery-phase2.jsp
Location:
Input Parameters: username

Divination Attacks

ZAP's Request History

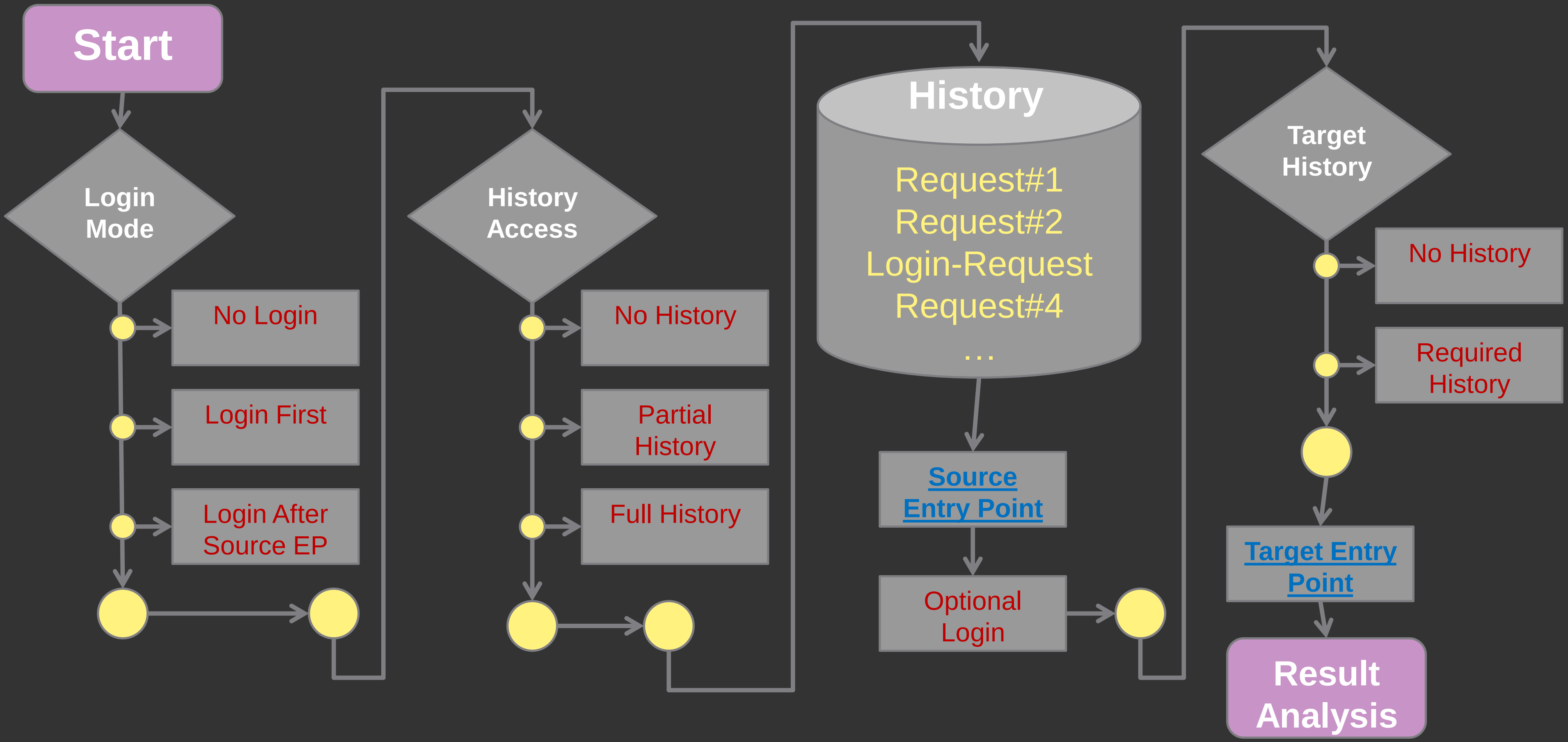


Filter: OFF

1	GET	http://localhost:8080/puzzlemall/contact.jsp?origin=USA	200	OK
4	POST	http://localhost:8080/puzzlemall/login.jsp	200	OK
5	GET	http://localhost:8080/puzzlemall/private/viewprofile.jsp	200	OK
7	GET	http://localhost:8080/puzzlemall/private/viewpuzzles.jsp	200	OK
8	GET	http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=2&descr=transaction	200	OK
9	GET	http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=2&purchase=true&descr=transaction	200	OK
10	GET	http://localhost:8080/puzzlemall/private/vieworders.jsp	200	OK
11	GET	http://localhost:8080/puzzlemall/private/mainmenu.jsp	200	OK
12	GET	http://localhost:8080/puzzlemall/sitemap.jsp	200	OK
13	GET	http://localhost:8080/puzzlemall/recovery-phase1.jsp	200	OK
14	POST	http://localhost:8080/puzzlemall/recovery-phase2.jsp	200	OK
15	POST	http://localhost:8080/puzzlemall/recovery-phase3.jsp	200	OK
16	POST	http://localhost:8080/puzzlemall/recovery-success.jsp	200	OK
17	GET	http://localhost:8080/puzzlemall/register-phase1.jsp	200	OK
18	POST	http://localhost:8080/puzzlemall/register-phase2.jsp	200	OK
19	GET	http://localhost:8080/puzzlemall/private/mainmenu.jsp	200	OK

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

Target History

No History

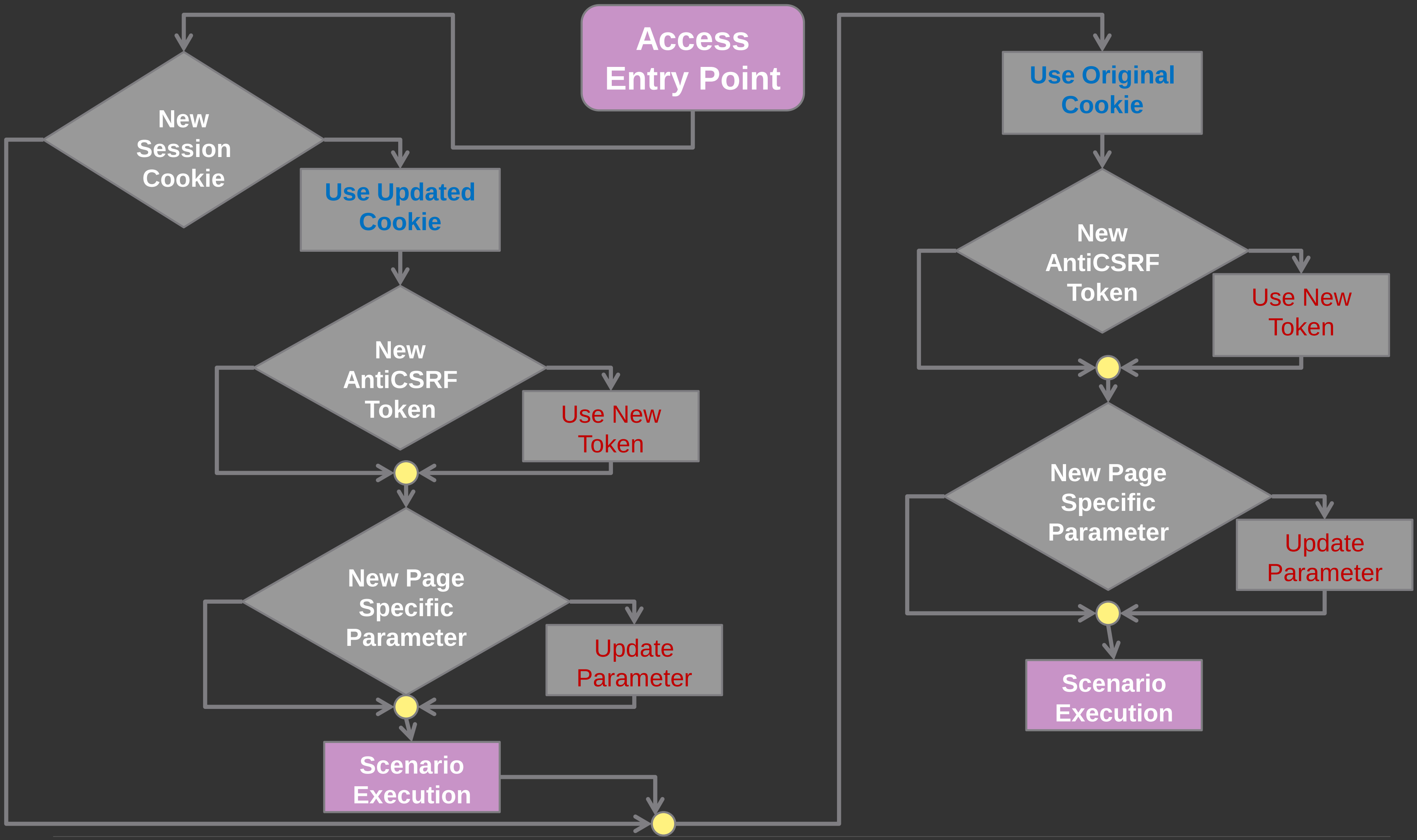
Required History

Target Entry Point

Result Analysis

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);	…	

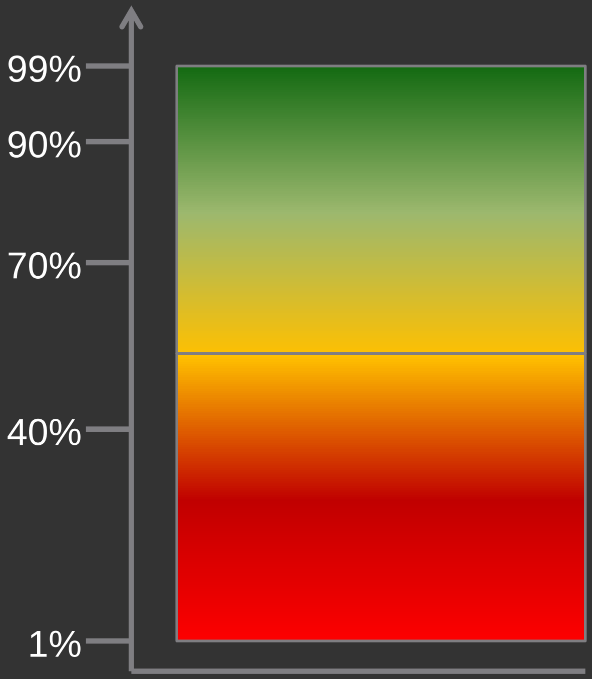
4	New Cookie Value	Cookie cookie = new Cookie ("##1##",val);
		response.addCookie(cookie);
			Response.Cookies("##1##").Value = "val";	

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%
…	…	…	…	…

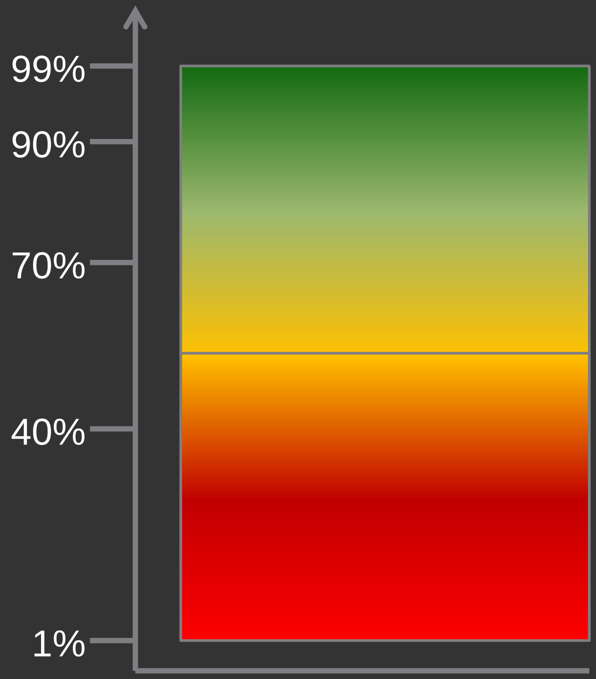


Verification Process and Probability

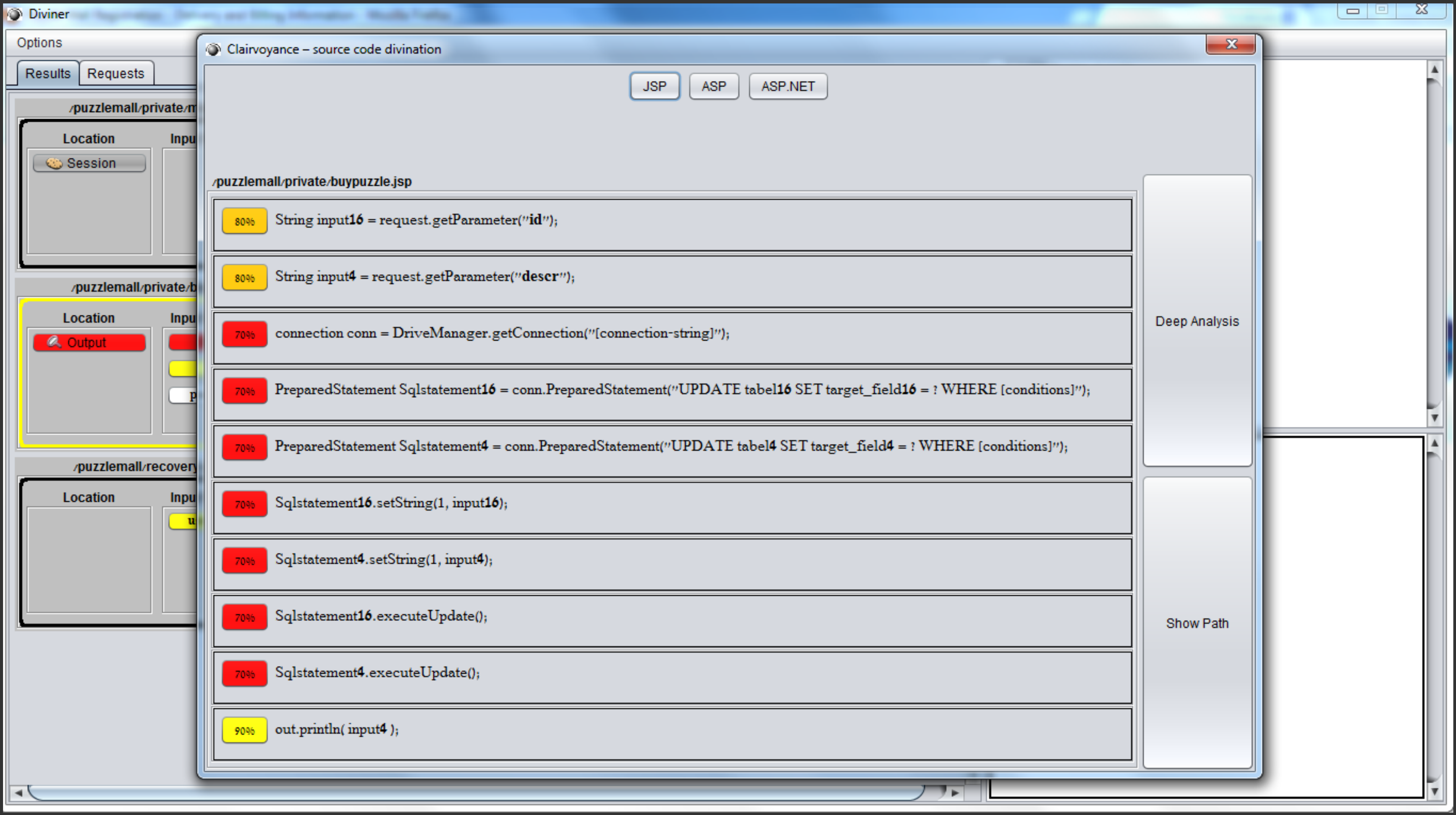
For **each** unique entry point / request, the probability for the existence of specific lines of code is adjusted according to the results of various behavior specific confirmation processes.

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%
...	...	...	...	...



Source/Target Entry Points Code Correlation



/puzzlemall/private/buypuzzle.jsp

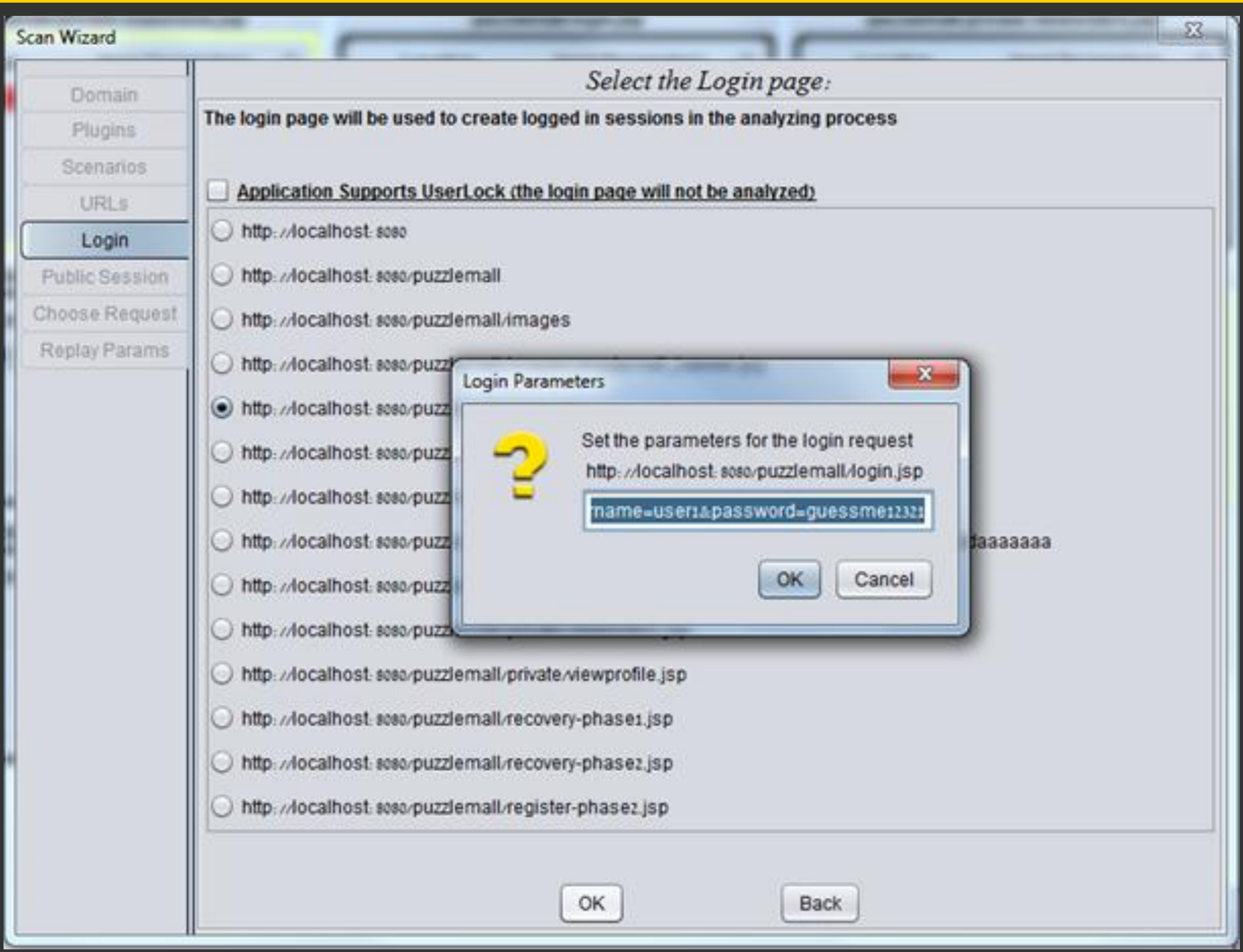
```
80%  String input16 = request.getParameter("id");
80%  String input4 = request.getParameter("descr");
70%  connection conn = DriveManager.getConnection("[connection-string]");
70%  PreparedStatement Sqlstatement16 = conn.PreparedStatement("UPDATE tabel16 SET target_field16 = ? WHERE [conditions]");
70%  PreparedStatement Sqlstatement4 = conn.PreparedStatement("UPDATE tabel4 SET target_field4 = ? WHERE [conditions]");
70%  Sqlstatement16.setString(1, input16);
70%  Sqlstatement4.setString(1, input4);
70%  Sqlstatement16.executeUpdate();
70%  Sqlstatement4.executeUpdate();
90%  out.println( input4 );
```

Diviner

A New ZAP Extension

Live Demo!

Divination Wizard – Record Login Scenario



Select the Login page:

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)

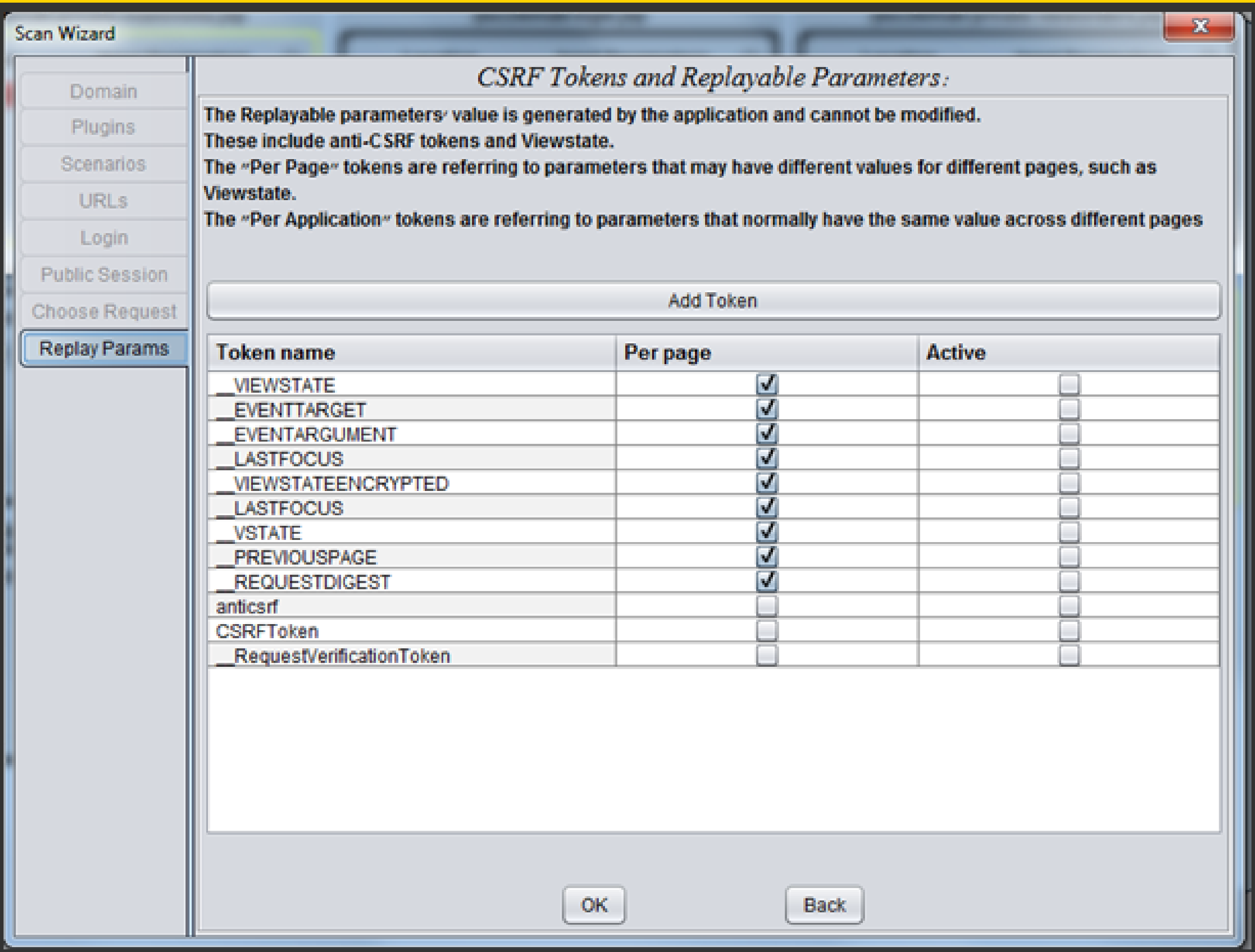
○ http://localhost:8080
○ http://localhost:8080/puzzlemall
○ http://localhost:8080/puzzlemall/images
○ http://localhost:8080/puzz
● http://localhost:8080/puzz
○ http://localhost:8080/puzz
○ http://localhost:8080/puzz
○ http://localhost:8080/puzz ... daaaaaaa
○ http://localhost:8080/puzz
○ http://localhost:8080/puzz
○ 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

Login Parameters

Set the parameters for the login request
http://localhost:8080/puzzlemall/login.jsp

name=user1&password=guessme12321

Divination Wizard – Handle CSRF Barriers

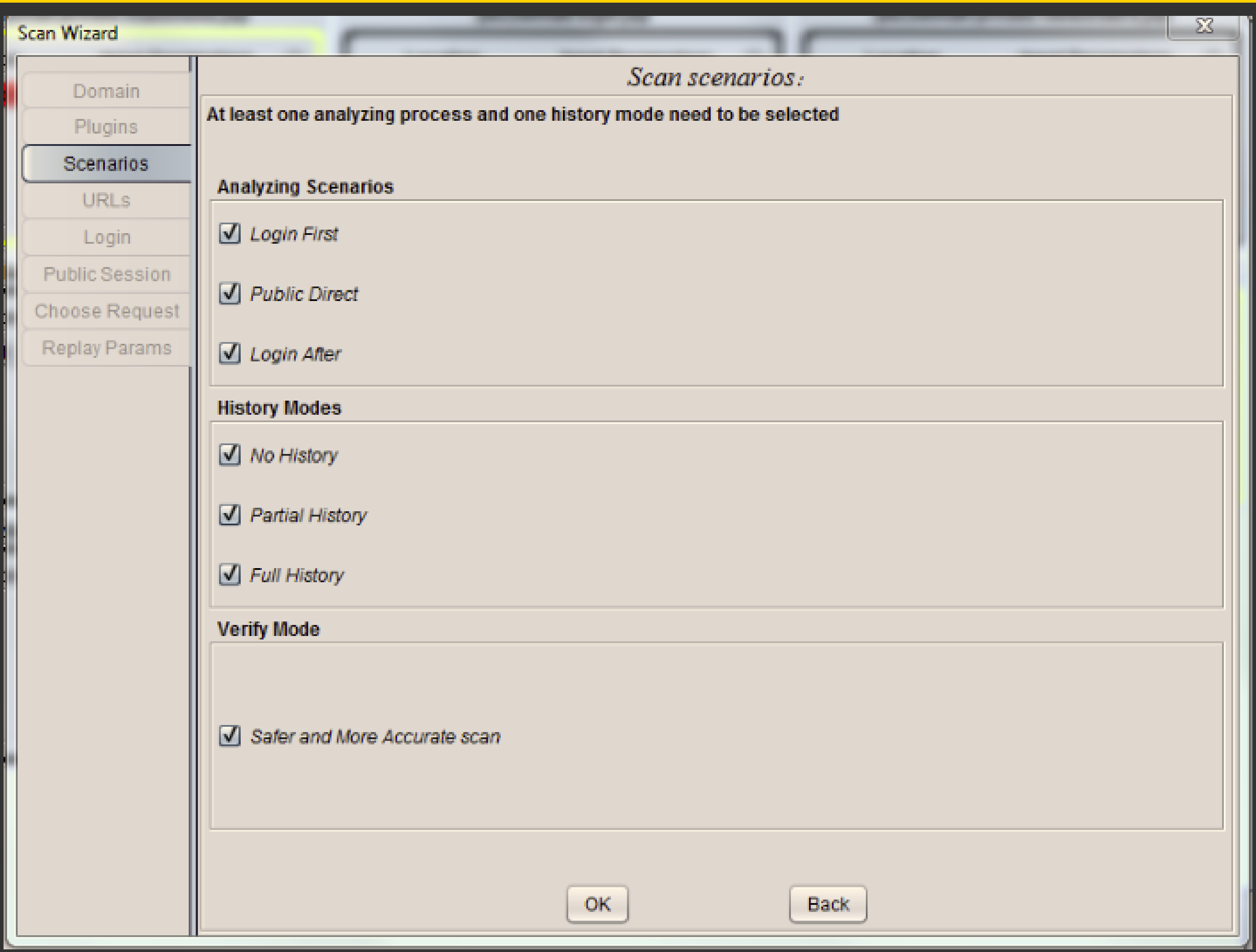


CSRF Tokens and Replayable Parameters:

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

Token name	Per page	Active
__VIEWSTATE	☑	□
__EVENTTARGET	☑	□
__EVENTARGUMENT	☑	□
__LASTFOCUS	☑	□
__VIEWSTATEENCRYPTED	☑	□
__LASTFOCUS	☑	□
__VSTATE	☑	□
__PREVIOUSPAGE	☑	□
__REQUESTDIGEST	☑	□
anticsrf	□	□
CSRFToken	□	□
__RequestVerificationToken	□	□

Divination Wizard – Define Analysis Mode



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*

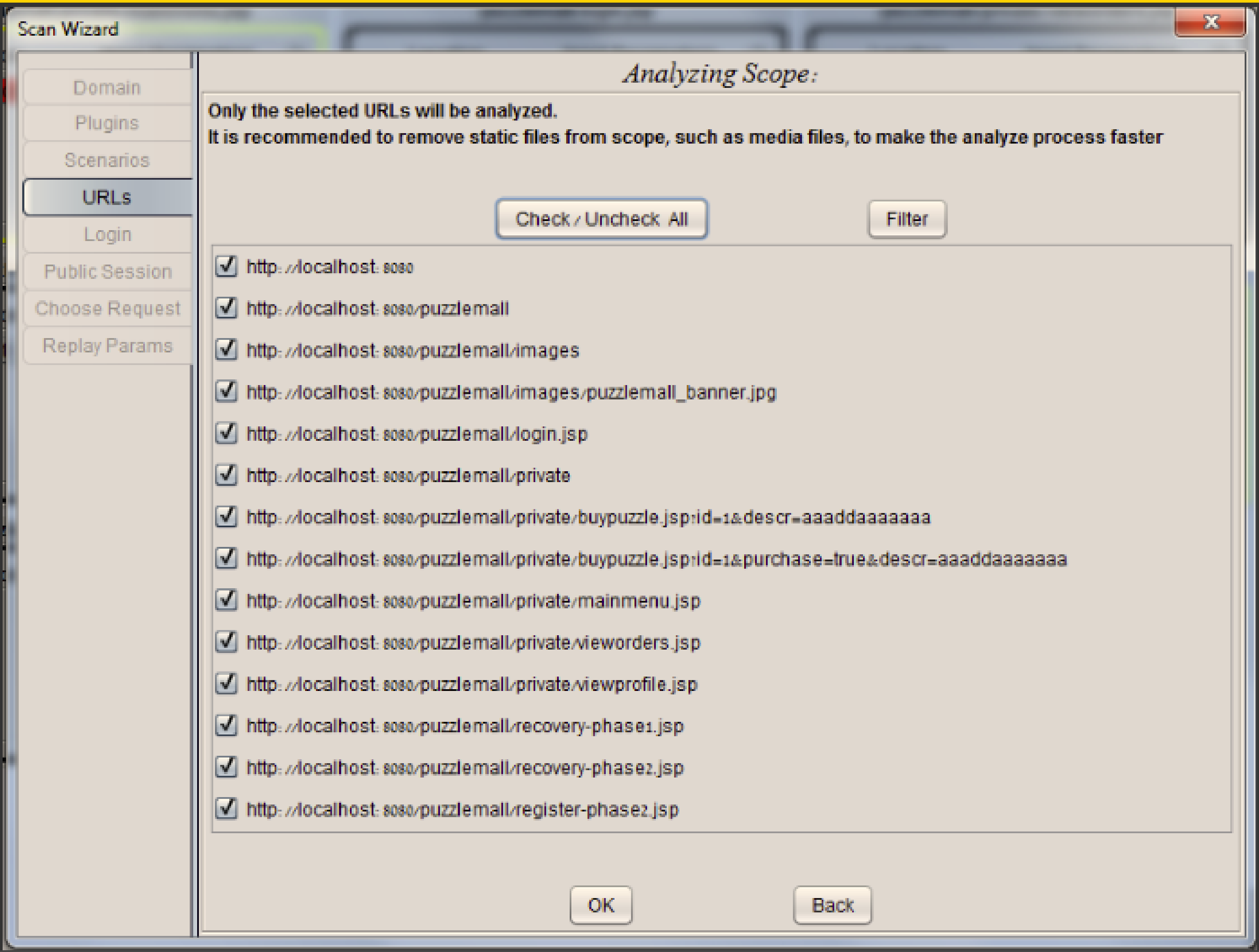
☑ *Partial History*

☑ *Full History*

Verify Mode

☑ *Safer and More Accurate scan*

Divination Wizard – Define Analysis Scope

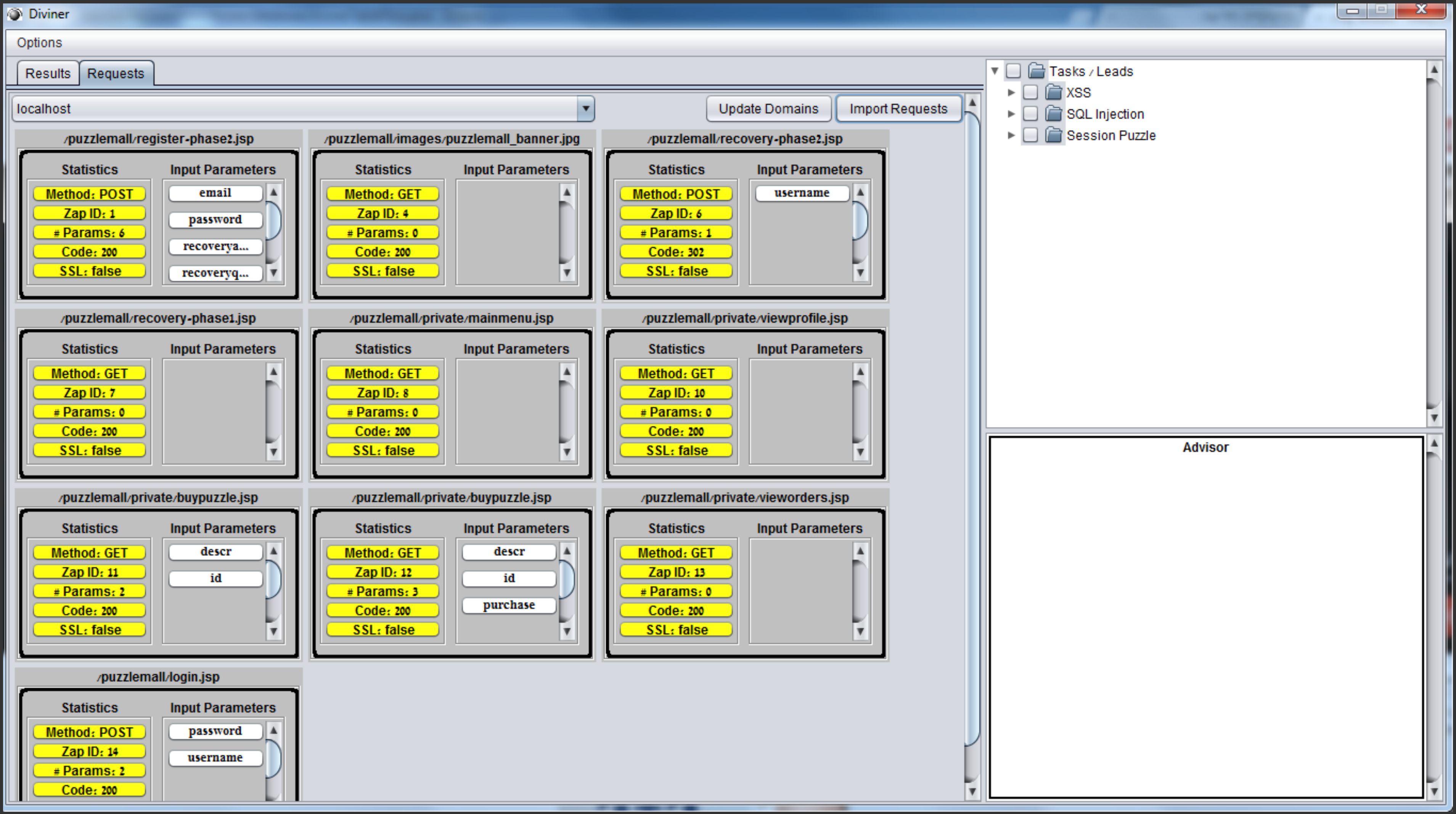


Analyzing Scope:

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/images/puzzlemall_banner.jpg
- ☑ http://localhost:8080/puzzlemall/login.jsp
- ☑ http://localhost:8080/puzzlemall/private
- ☑ http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=1&descr=aaaddaaaaaaa
- ☑ http://localhost:8080/puzzlemall/private/buypuzzle.jsp?id=1&purchase=true&descr=aaaddaaaaaaa
- ☑ 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

Visual Penetration Testing & Payload Reuse



Diviner

Options

Results | Requests

localhost ▾

Update Domains | Import Requests

/puzzlemall/register-phase2.jsp
Statistics: Method: POST, Zap ID: 1, # Params: 6, Code: 200, SSL: false
Input Parameters: email, password, recoverya..., recoveryq...

/puzzlemall/images/puzzlemall_banner.jpg
Statistics: Method: GET, Zap ID: 4, # Params: 0, Code: 200, SSL: false
Input Parameters

/puzzlemall/recovery-phase2.jsp
Statistics: Method: POST, Zap ID: 6, # Params: 1, Code: 302, SSL: false
Input Parameters: username

/puzzlemall/recovery-phase1.jsp
Statistics: Method: GET, Zap ID: 7, # Params: 0, Code: 200, SSL: false
Input Parameters

/puzzlemall/private/mainmenu.jsp
Statistics: Method: GET, Zap ID: 8, # Params: 0, Code: 200, SSL: false
Input Parameters

/puzzlemall/private/viewprofile.jsp
Statistics: Method: GET, Zap ID: 10, # Params: 0, Code: 200, SSL: false
Input Parameters

/puzzlemall/private/buypuzzle.jsp
Statistics: Method: GET, Zap ID: 11, # Params: 2, Code: 200, SSL: false
Input Parameters: descr, id

/puzzlemall/private/buypuzzle.jsp
Statistics: Method: GET, Zap ID: 12, # Params: 3, Code: 200, SSL: false
Input Parameters: descr, id, purchase

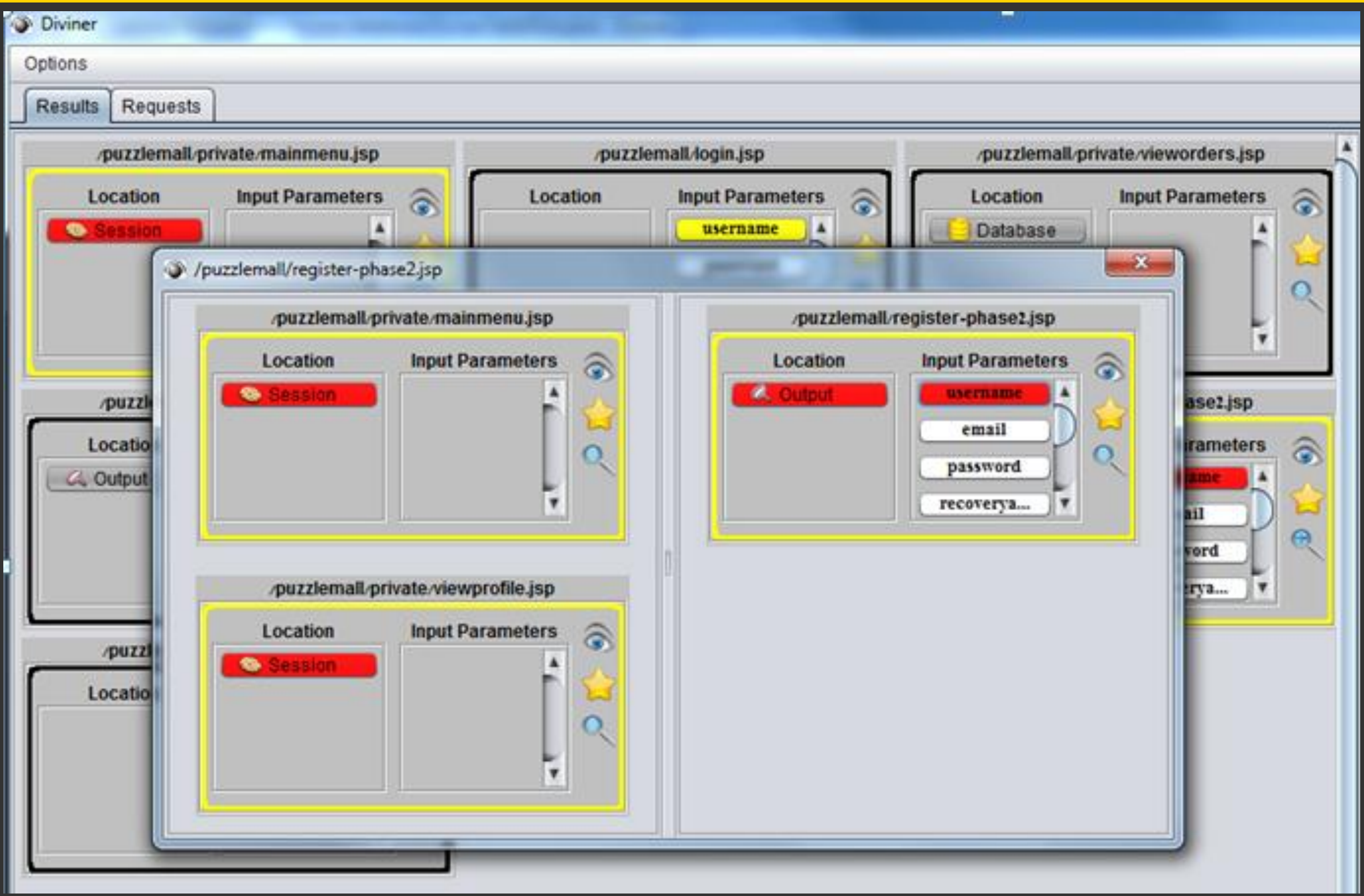
/puzzlemall/private/vieworders.jsp
Statistics: Method: GET, Zap ID: 13, # Params: 0, Code: 200, SSL: false
Input Parameters

/puzzlemall/login.jsp
Statistics: Method: POST, Zap ID: 14, # Params: 2, Code: 200
Input Parameters: password, username

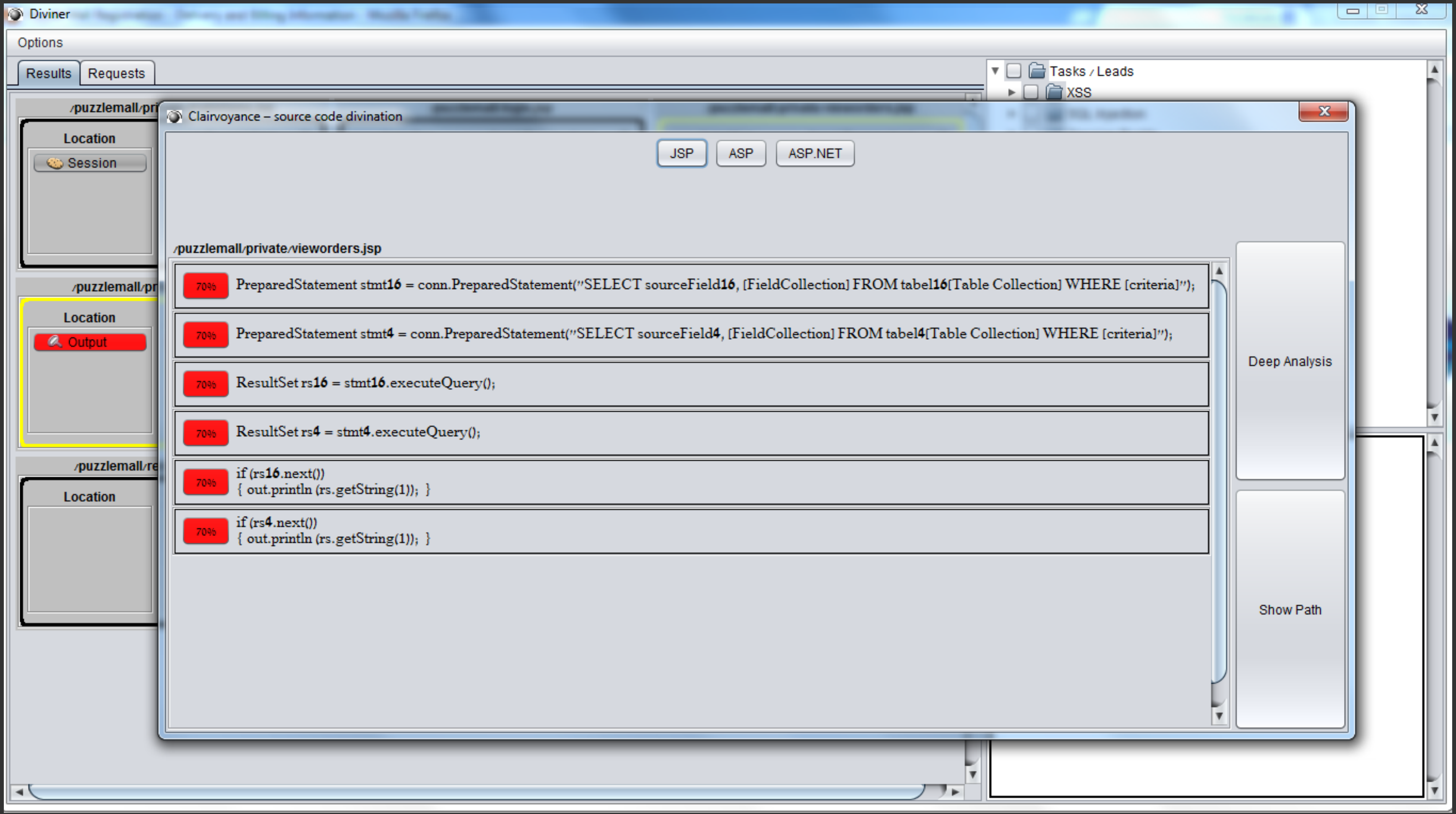
▼ □ Tasks / Leads
▶ □ XSS
▶ □ SQL Injection
▶ □ Session Puzzle

Advisor

Visual Entry Point Input - Output Correlation


Diviner
Options
Results
Requests
/puzzlemall/private/mainmenu.jsp
/puzzlemall/login.jsp
/puzzlemall/private/vieworders.jsp
Location
Input Parameters
Session
username
Database
Output
/puzzlemall/register-phase2.jsp
/puzzlemall/private/mainmenu.jsp
/puzzlemall/register-phase2.jsp
/puzzlemall/private/viewprofile.jsp
email
password
recoverya...

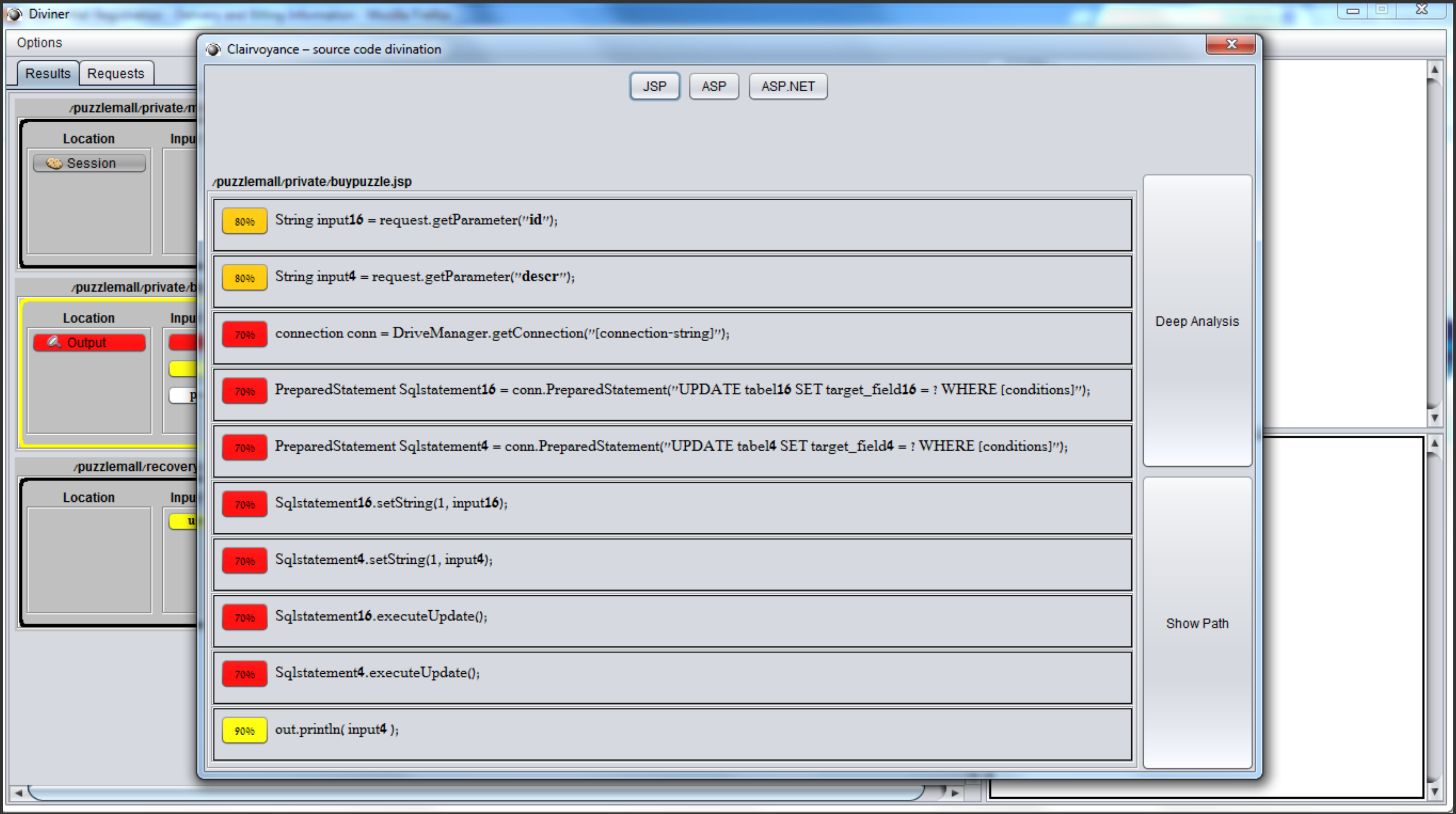
Entry Point Structure & Source Visualization



/puzzlemall/private/vieworders.jsp

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

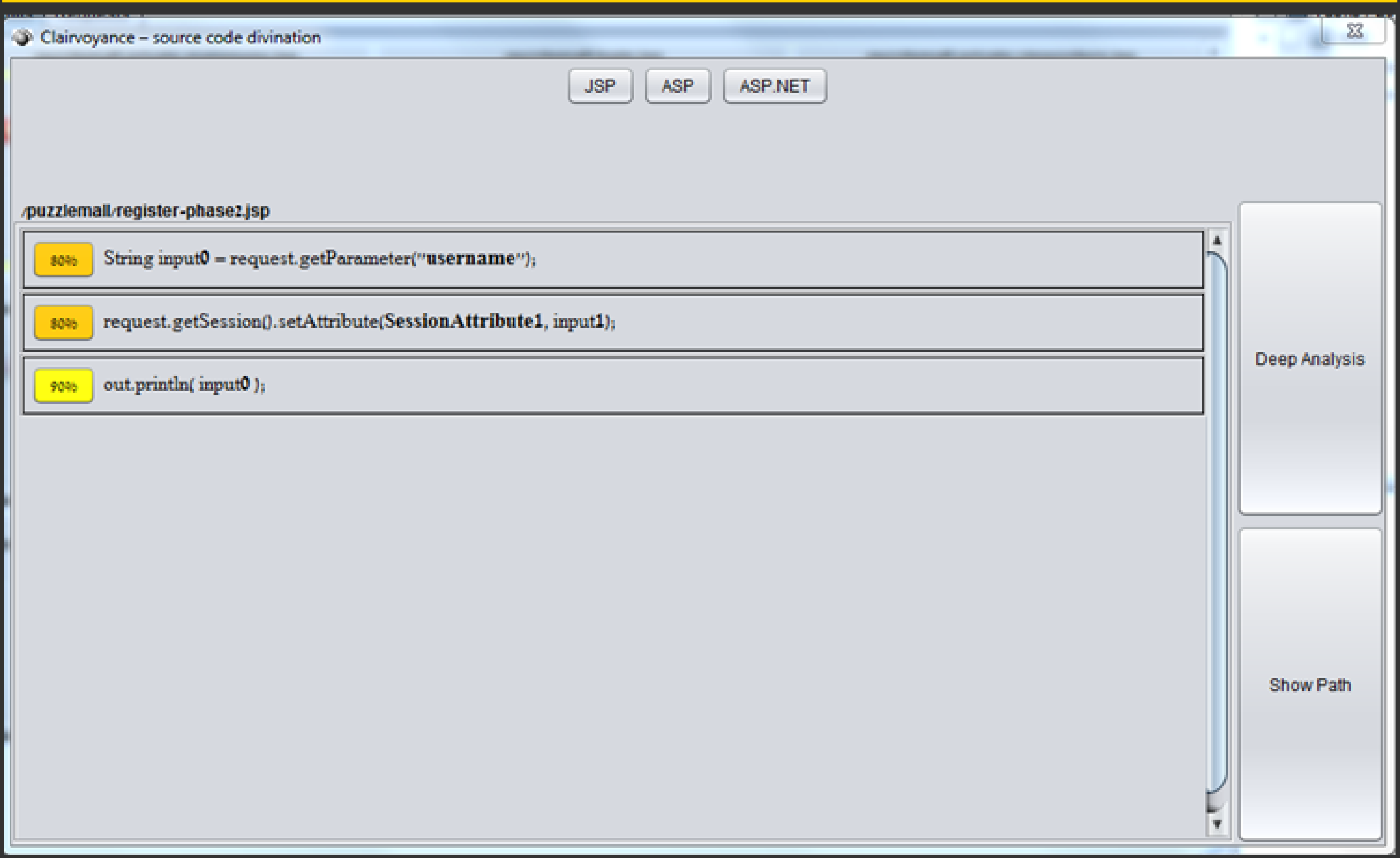
Source/Target Entry Points Code Correlation



/puzzlemall/private/buypuzzle.jsp

```
80%  String input16 = request.getParameter("id");
80%  String input4 = request.getParameter("descr");
70%  connection conn = DriveManager.getConnection("[connection-string]");
70%  PreparedStatement Sqlstatement16 = conn.PreparedStatement("UPDATE tabel16 SET target_field16 = ? WHERE [conditions]");
70%  PreparedStatement Sqlstatement4 = conn.PreparedStatement("UPDATE tabel4 SET target_field4 = ? WHERE [conditions]");
70%  Sqlstatement16.setString(1, input16);
70%  Sqlstatement4.setString(1, input4);
70%  Sqlstatement16.executeUpdate();
70%  Sqlstatement4.executeUpdate();
90%  out.println( input4 );
```

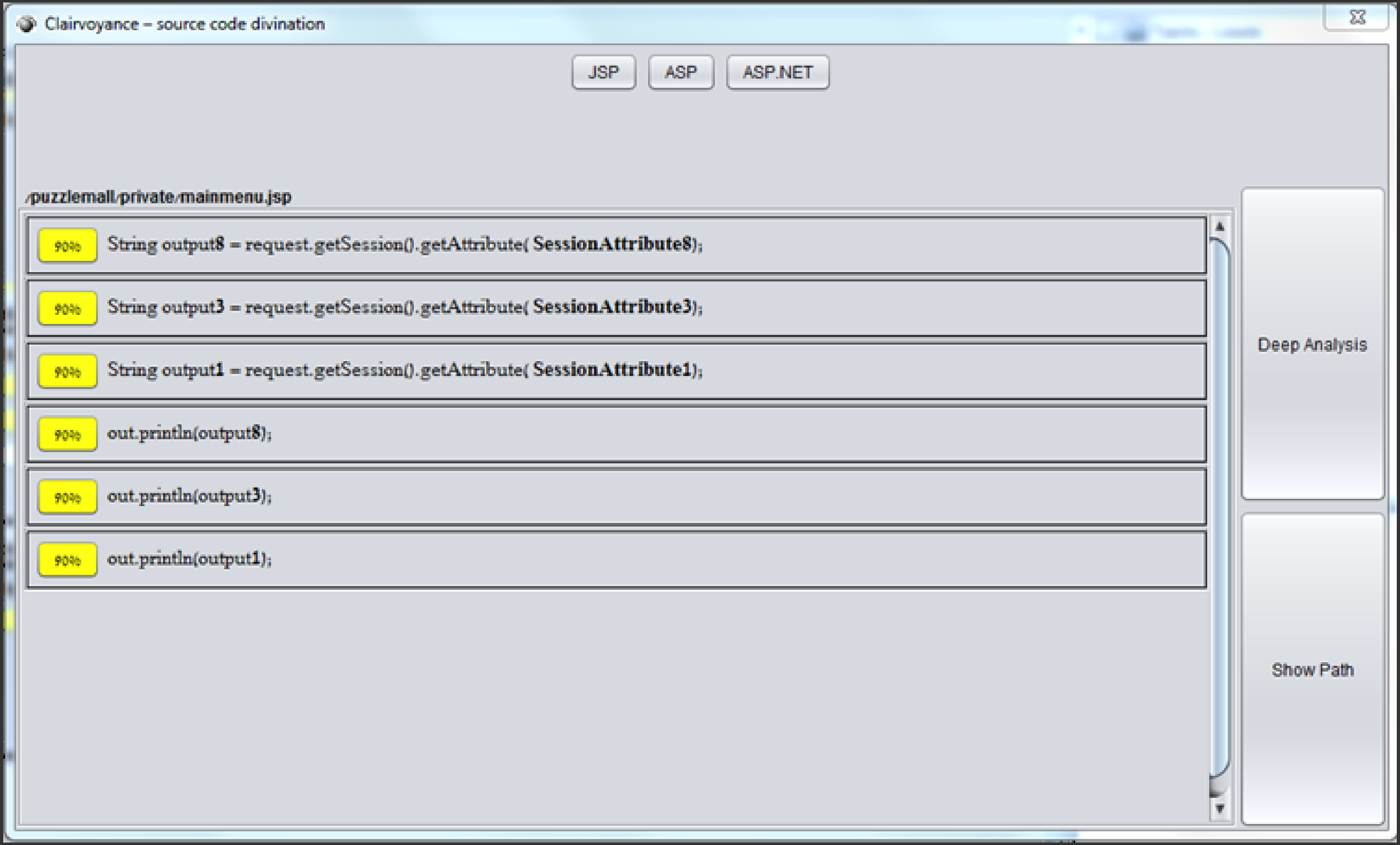
Detect Indirect Attack Vectors – Source Page



/puzzlemall/register-phase2.jsp

```
String input0 = request.getParameter("username");
request.getSession().setAttribute(SessionAttribute1, input1);
out.println( input0 );
```

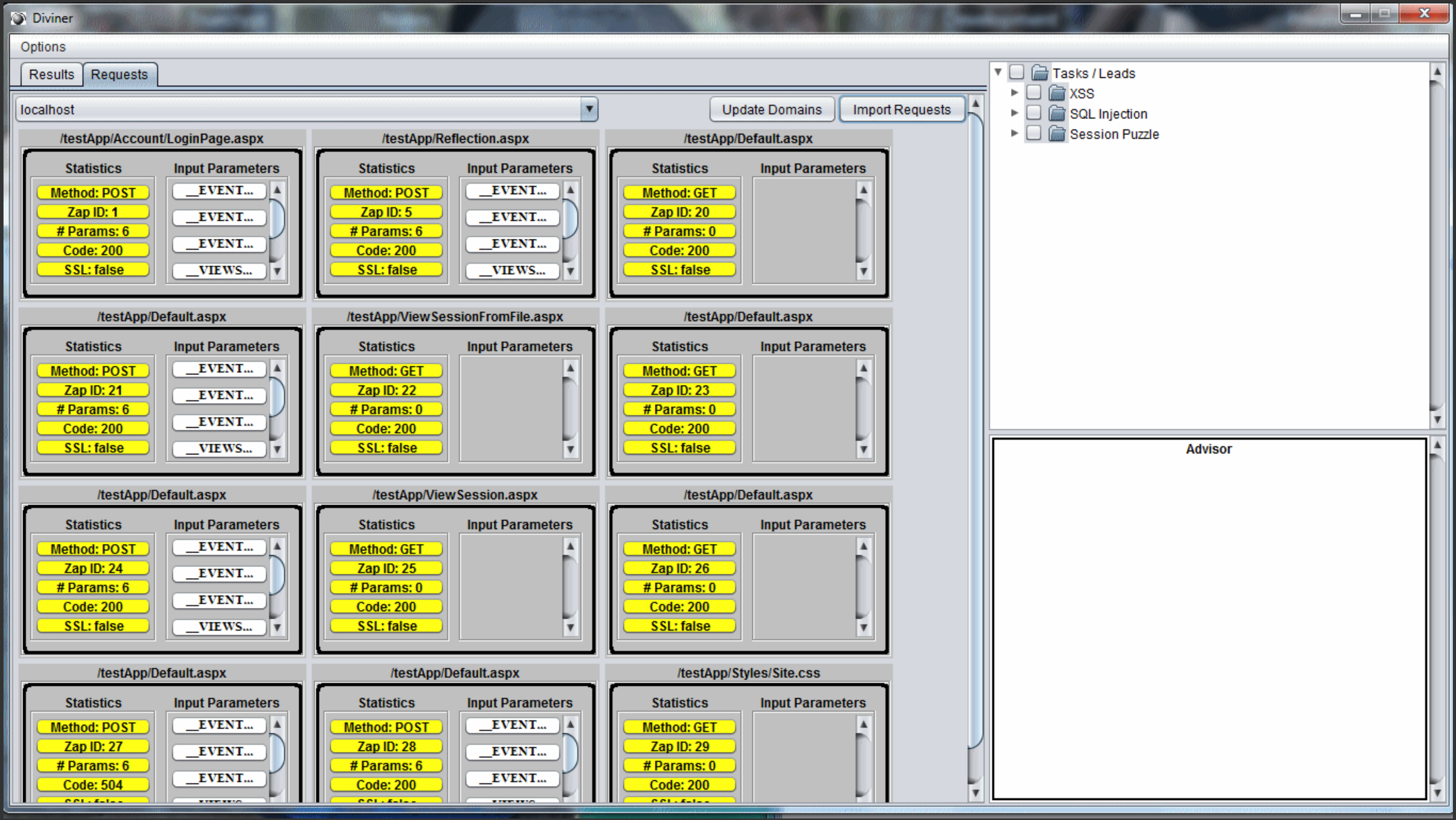
Detect Indirect Attack Vectors – Target Page



/puzzlemall/private/mainmenu.jsp

```
90%  String output8 = request.getSession().getAttribute( SessionAttribute8);
90%  String output3 = request.getSession().getAttribute( SessionAttribute3);
90%  String output1 = request.getSession().getAttribute( SessionAttribute1);
90%  out.println(output8);
90%  out.println(output3);
90%  out.println(output1);
```

Support Different Technologies



localhost

/testApp/Account/LoginPage.aspx
Statistics: Method: POST, Zap ID: 1, # Params: 6, Code: 200, SSL: false
Input Parameters: __EVENT…, __EVENT…, __EVENT…, __VIEWS…

/testApp/Reflection.aspx
Statistics: Method: POST, Zap ID: 5, # Params: 6, Code: 200, SSL: false
Input Parameters: __EVENT…, __EVENT…, __EVENT…, __VIEWS…

/testApp/Default.aspx
Statistics: Method: GET, Zap ID: 20, # Params: 0, Code: 200, SSL: false
Input Parameters

/testApp/Default.aspx
Statistics: Method: POST, Zap ID: 21, # Params: 6, Code: 200, SSL: false
Input Parameters: __EVENT…, __EVENT…, __EVENT…, __VIEWS…

/testApp/ViewSessionFromFile.aspx
Statistics: Method: GET, Zap ID: 22, # Params: 0, Code: 200, SSL: false
Input Parameters

/testApp/Default.aspx
Statistics: Method: GET, Zap ID: 23, # Params: 0, Code: 200, SSL: false
Input Parameters

/testApp/Default.aspx
Statistics: Method: POST, Zap ID: 24, # Params: 6, Code: 200, SSL: false
Input Parameters: __EVENT…, __EVENT…, __EVENT…, __VIEWS…

/testApp/ViewSession.aspx
Statistics: Method: GET, Zap ID: 25, # Params: 0, Code: 200, SSL: false
Input Parameters

/testApp/Default.aspx
Statistics: Method: GET, Zap ID: 26, # Params: 0, Code: 200, SSL: false
Input Parameters

/testApp/Default.aspx
Statistics: Method: POST, Zap ID: 27, # Params: 6, Code: 504
Input Parameters: __EVENT…, __EVENT…, __EVENT…

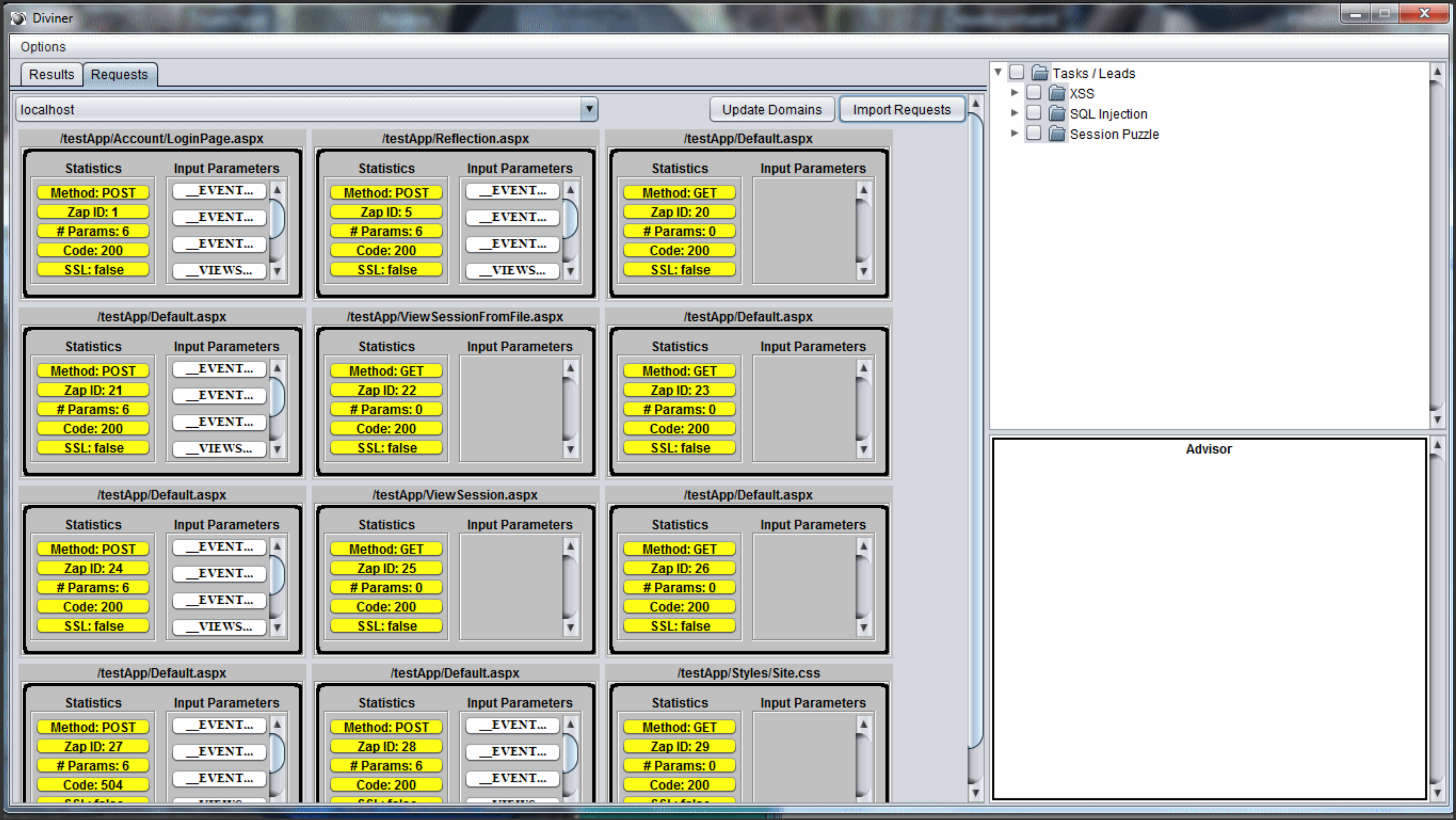
/testApp/Default.aspx
Statistics: Method: POST, Zap ID: 28, # Params: 6, Code: 200
Input Parameters: __EVENT…, __EVENT…, __EVENT…

/testApp/Styles/Site.css
Statistics: Method: GET, Zap ID: 29, # Params: 0, Code: 200
Input Parameters

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

Advisor

Support Different Technologies



Diviner

Options

Results | Requests

localhost ▾ | Update Domains | Import Requests

/testApp/Account/LoginPage.aspx
Statistics | Input Parameters
Method: POST | __EVENT...
Zap ID: 1 | __EVENT...
Params: 6 | __EVENT...
Code: 200 | __VIEWS...
SSL: false

/testApp/Reflection.aspx
Statistics | Input Parameters
Method: POST | __EVENT...
Zap ID: 5 | __EVENT...
Params: 6 | __EVENT...
Code: 200 | __VIEWS...
SSL: false

/testApp/Default.aspx
Statistics | Input Parameters
Method: GET
Zap ID: 20
Params: 0
Code: 200
SSL: false

/testApp/Default.aspx
Statistics | Input Parameters
Method: POST | __EVENT...
Zap ID: 21 | __EVENT...
Params: 6 | __EVENT...
Code: 200 | __VIEWS...
SSL: false

/testApp/ViewSessionFromFile.aspx
Statistics | Input Parameters
Method: GET
Zap ID: 22
Params: 0
Code: 200
SSL: false

/testApp/Default.aspx
Statistics | Input Parameters
Method: GET
Zap ID: 23
Params: 0
Code: 200
SSL: false

/testApp/Default.aspx
Statistics | Input Parameters
Method: POST | __EVENT...
Zap ID: 24 | __EVENT...
Params: 6 | __EVENT...
Code: 200 | __VIEWS...
SSL: false

/testApp/ViewSession.aspx
Statistics | Input Parameters
Method: GET
Zap ID: 25
Params: 0
Code: 200
SSL: false

/testApp/Default.aspx
Statistics | Input Parameters
Method: GET
Zap ID: 26
Params: 0
Code: 200
SSL: false

/testApp/Default.aspx
Statistics | Input Parameters
Method: POST | __EVENT...
Zap ID: 27 | __EVENT...
Params: 6 | __EVENT...
Code: 504

/testApp/Default.aspx
Statistics | Input Parameters
Method: POST | __EVENT...
Zap ID: 28 | __EVENT...
Params: 6 | __EVENT...
Code: 200

/testApp/Styles/Site.css
Statistics | Input Parameters
Method: GET
Zap ID: 29
Params: 0
Code: 200

▾ □ Tasks / Leads
▸ □ XSS
▸ □ SQL Injection
▸ □ Session Puzzle

Advisor

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

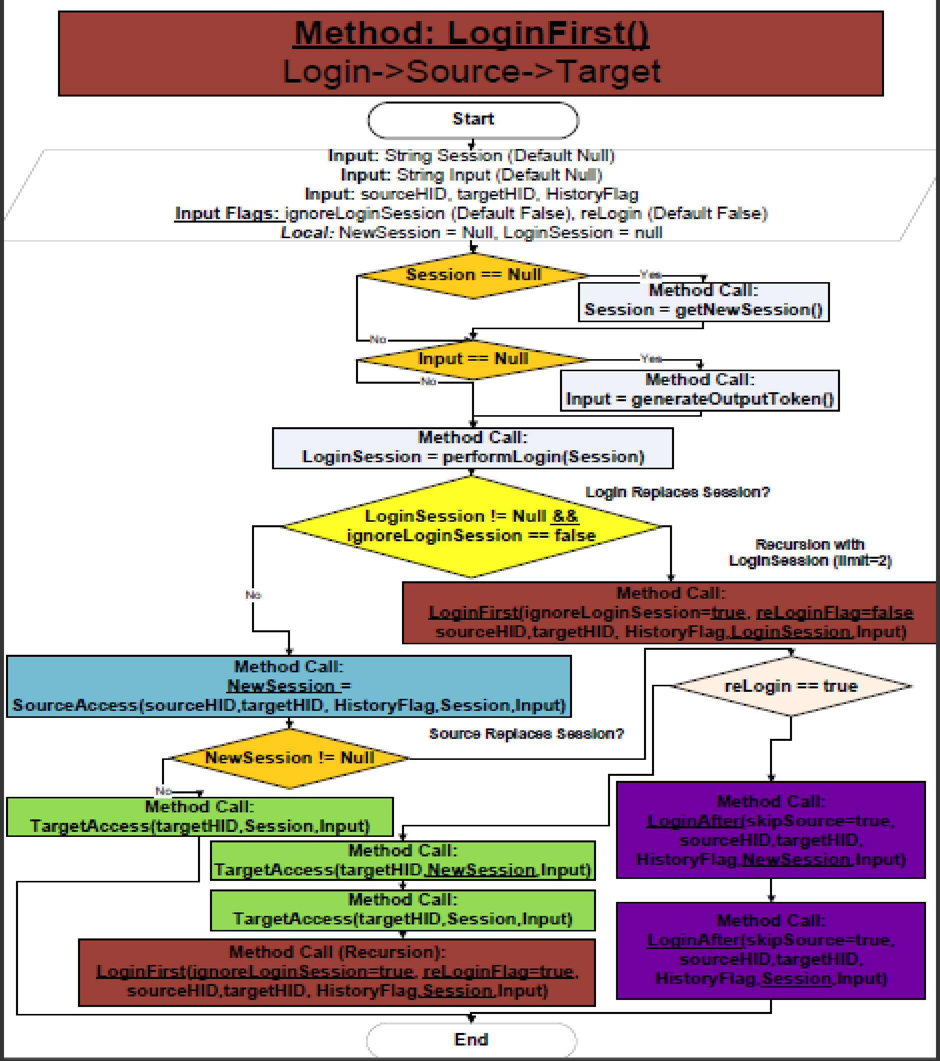
Source Code Divination – Code Prediction

- 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



Method: LoginFirst()

Login->Source->Target

Start

Input: String Session (Default Null)
Input: String Input (Default Null)
Input: sourceHID, targetHID, HistoryFlag
Input Flags: ignoreLoginSession (Default False), reLogin (Default False)
Local: NewSession = Null, LoginSession = null

Session == Null

Yes

Method Call:
Session = getNewSession()

No

Input == Null

Yes

Method Call:
Input = generateOutputToken()

No

Method Call:
LoginSession = performLogin(Session)

Login Replaces Session?

LoginSession != Null && ignoreLoginSession == false

Recursion with LoginSession (limit=2)

Method Call:
LoginFirst(ignoreLoginSession=true, reLoginFlag=false sourceHID,targetHID, HistoryFlag,LoginSession,Input)

No

Method Call:
NewSession =
SourceAccess(sourceHID,targetHID, HistoryFlag,Session,Input)

reLogin == true

Source Replaces Session?

NewSession != Null

No

Method Call:
TargetAccess(targetHID,Session,Input)

Method Call:
TargetAccess(targetHID,NewSession,Input)

Method Call:
TargetAccess(targetHID,Session,Input)

Method Call (Recursion):
LoginFirst(ignoreLoginSession=true, reLoginFlag=true, sourceHID,targetHID, HistoryFlag,Session,Input)

Method Call:
LoginAfter(skipSource=true, sourceHID,targetHID, HistoryFlag,NewSession,Input)

Method Call:
LoginAfter(skipSource=true, sourceHID,targetHID, HistoryFlag,Session,Input)

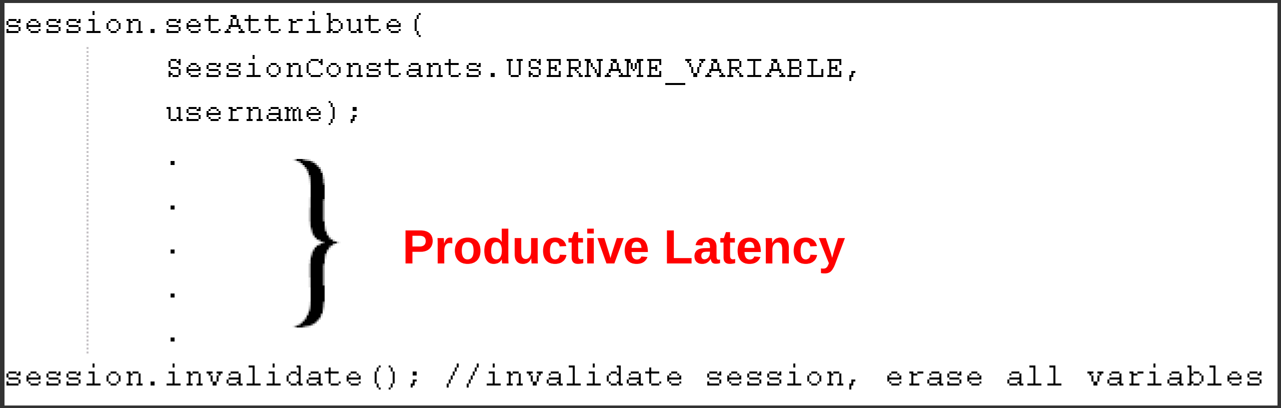
End

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.

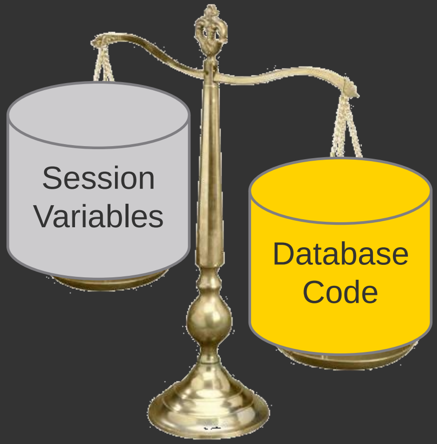


```
session.setAttribute(
        SessionConstants.USERNAME_VARIABLE,
        username);
        .
        .
        .
        .
        .
session.invalidate(); //invalidate session, erase all variables
```

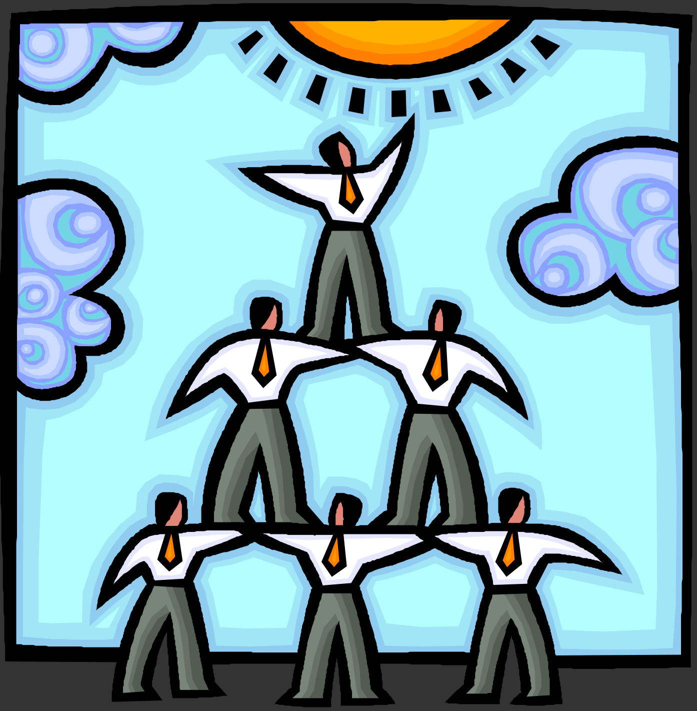
Productive Latency

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

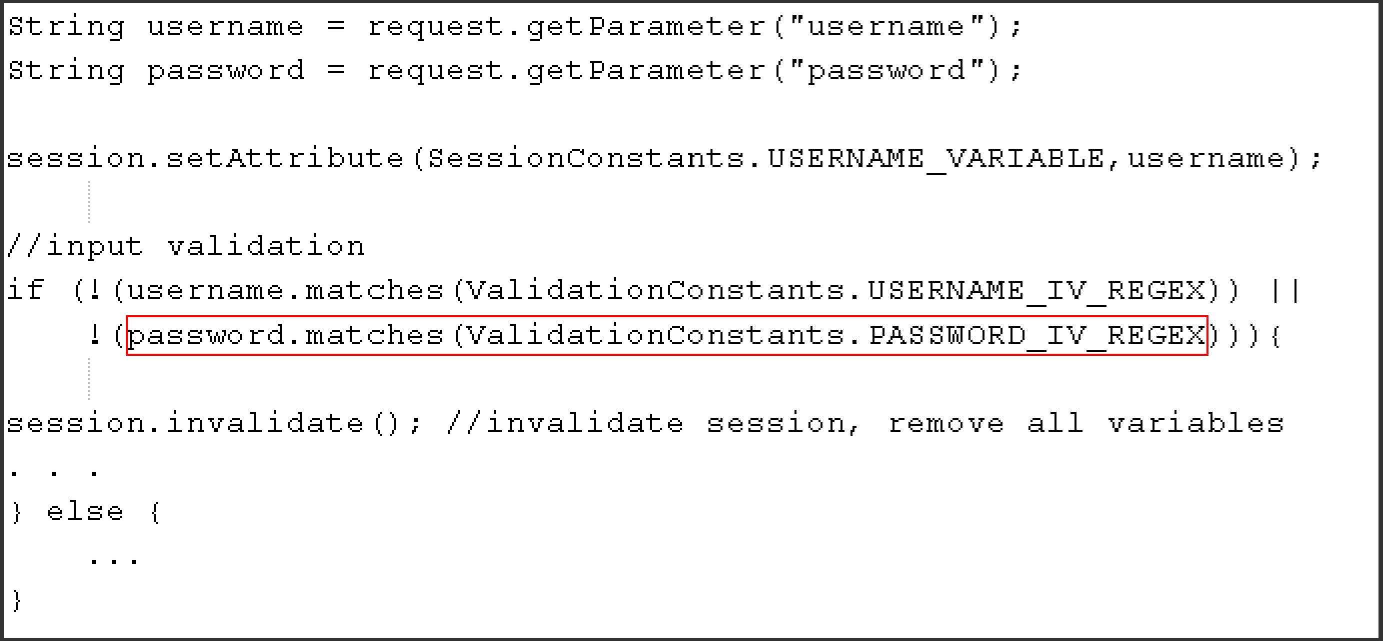


Layer Targeted Denial Of Service

- 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, aaaaaaaaaaaaaaaaaaaaaaaaaaa!



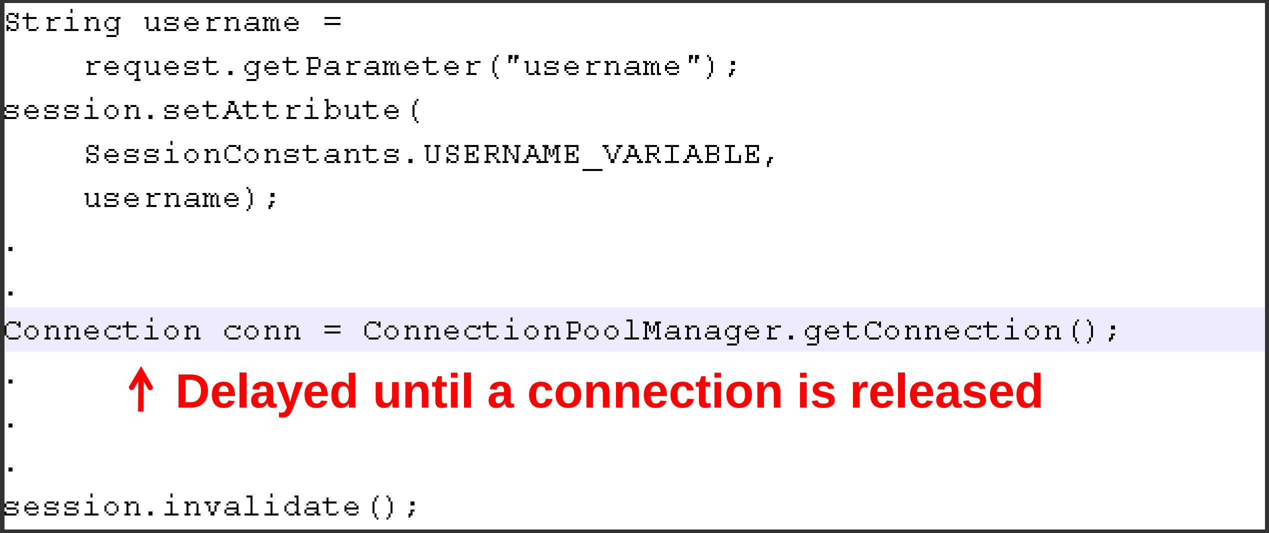
```
String username = request.getParameter("username");
String password = request.getParameter("password");

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();
.
.
.
session.invalidate();
```

↑ Delayed until a connection is released

And Finally...

Additional Resources

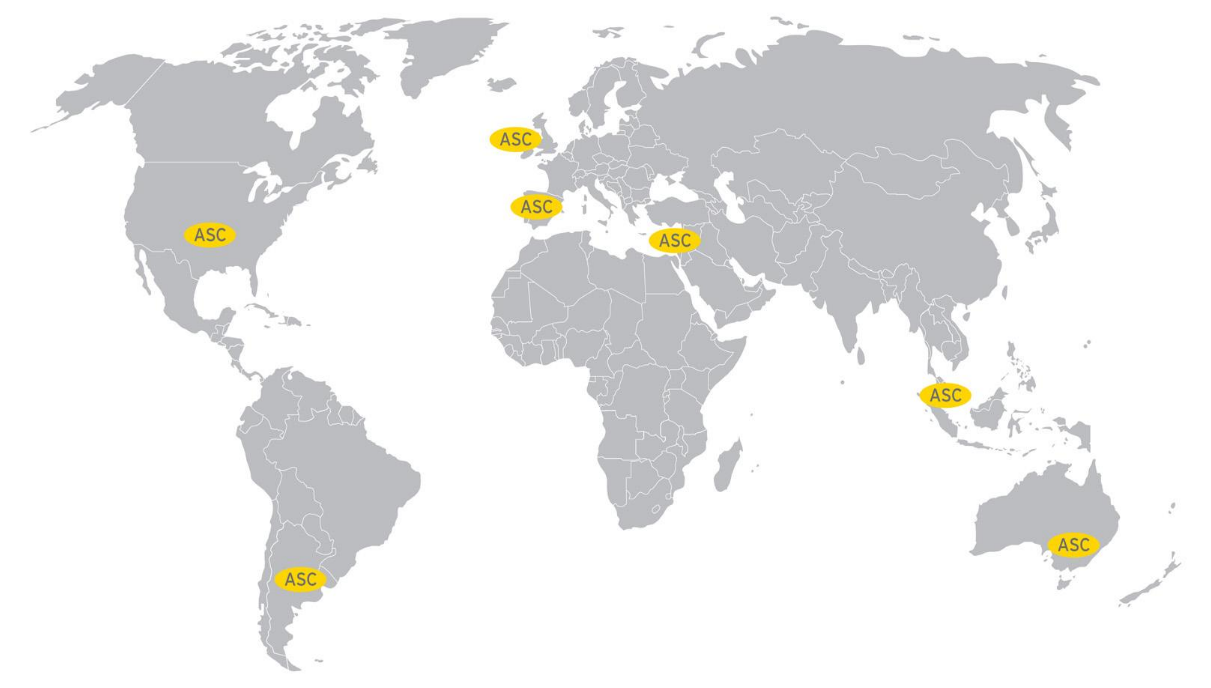
- Diviner Homepage (ZAP 1.4+ Extension)
  - http://code.google.com/p/diviner/
    - Structure and input/output flow **visualization**
    - Source code & memory structure **divination**
    - Advisor and task list manager
    - Payload manager integrated with ZAP repeater
- Payload Manager .Net
  - External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

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)