OWASP Jakarta Tech Day Meetup 2017

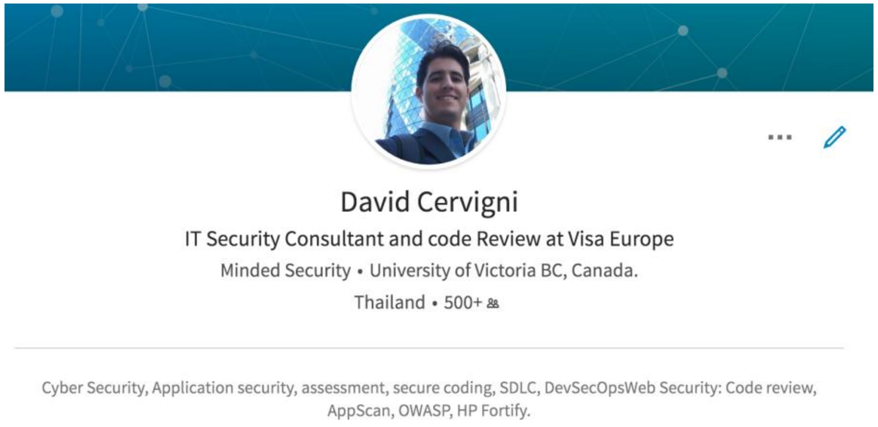
Client Side Security And Testing Tools

David Cervigni @ Minded Security

Agenda

- Short Intro
- Client side threats: Why important/difficult
- Examples: Dom XSS, HTTP Param Pollution
- Taint Analysis techniques
- BlueClosure
- Demo
- Questions…

Who am I

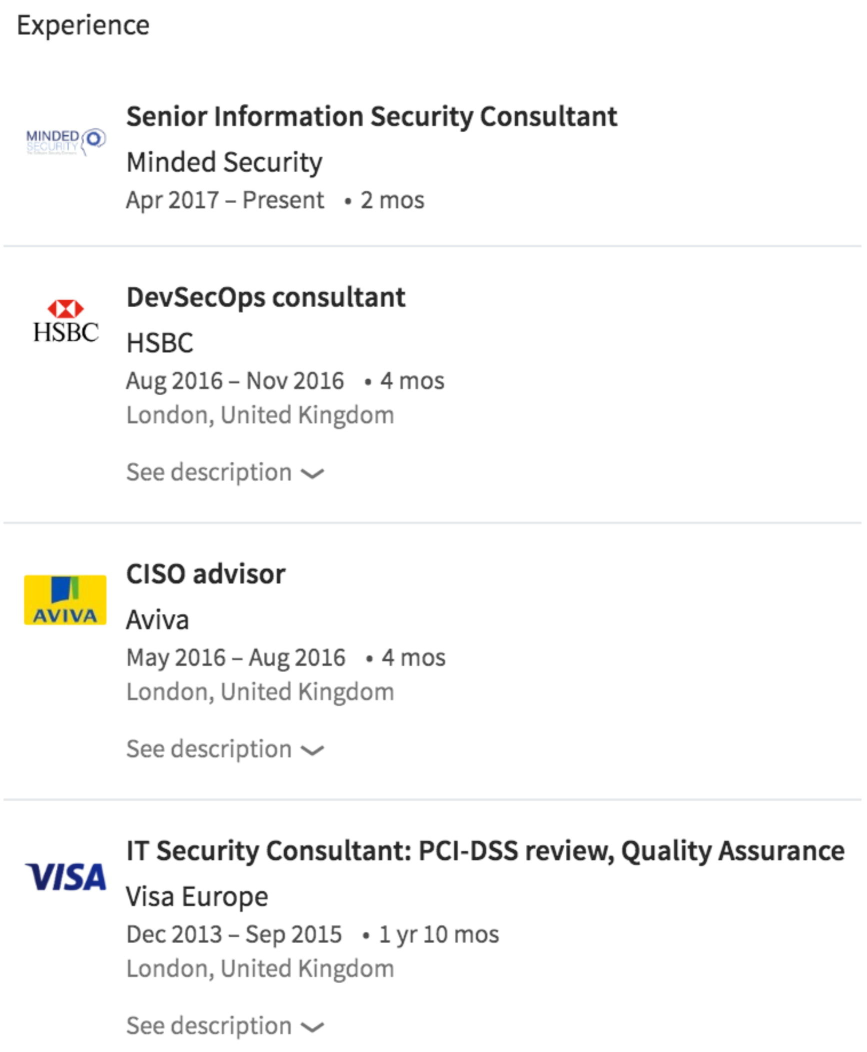

David Cervigni
IT Security Consultant and code Review at Visa Europe
Minded Security • University of Victoria BC, Canada.
Thailand • 500+

Cyber Security, Application security, assessment, secure coding, SDLC, DevSecOpsWeb Security: Code review, AppScan, OWASP, HP Fortify.

- 10+ yeas of development
- Software Security Enthusiast
- Securing SDLC
- Secure coding trainer
- […]
- Tango Dancing

Contacts:

- david.cervigni@mindedsecurity.com
- https://www.linkedin.com/in/david-cervigni-229569a/


Experience

Senior Information Security Consultant
Minded Security
Apr 2017 – Present • 2 mos

DevSecOps consultant
HSBC
Aug 2016 – Nov 2016 • 4 mos
London, United Kingdom
See description

CISO advisor
Aviva
May 2016 – Aug 2016 • 4 mos
London, United Kingdom
See description

IT Security Consultant: PCI-DSS review, Quality Assurance
Visa Europe
Dec 2013 – Sep 2015 • 1 yr 10 mos
London, United Kingdom
See description

Minded Security Services & Solutions

- **Secure Software Engineering Services:** helping mission critical businesses in the development of secure web and mobile applications and products from the inception to production cycle

A1 – Injection
A2 – Cross-Site Scripting (XSS)
A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object References
A5 – Cross-Site Request Forgery (CSRF)
t> A6 – Security Misconfiguration (NEW)
A7 – Insecure Cryptographic Storage
A8 – Failure to Restrict URL Access
A9 – Insufficient Transport Layer Protection
Unvalidated Redirects and Forwards (NEW)

- **JavaScript Vulnerability Detection Solutions:** innovative technology for automated detection of vulnerabilities in company owned client JavaScript that run on user’s browsers

Minded Security Global Customers

citi® NATO OTAN Google™ YAHOO!

ICBPI sogei FUTURO SEMPLICE ebay Yandex

UniCredit consip VISA DOWJONES

INTESA SANPAOLO MAGNETI MARELLI vodafone

Garanti eni criteo sky

Client Side Security Risks

Client-Side Security is concerned with the execution of code on the client, typically natively within a web browser or browser plugin. The execution of code on the client-side is distinct from executing on the server and returning the subsequent content.

The OWASP Testing Guide describes 12 major vulnerabilities that is possible to find analyzing th JavaScrip and HTML code of an application: https://www.owasp.org/index.php/Client_Side_Testing

DOM based Cross Site Scripting (OTG-CLIENT-001)

JavaScript Execution (OTG-CLIENT-002)

HTML Injection (OTG-CLIENT-003)

Client Side URL Redirect (OTG-CLIENT-004)

CSS Injection (OTG-CLIENT-005)

Client Side Resource Manipulation (OTG-CLIENT-006)

Cross Origin Resource Sharing (OTG-CLIENT-007)

Cross Site Flashing (OTG-CLIENT-008)

Clickjacking (OTG-CLIENT-009)

WebSockets (OTG-CLIENT-010)

Web Messaging (OTG-CLIENT-011)

Local Storage (OTG-CLIENT-012)

Why is always more **important**?

- SPA: Single Page Applications
- Mainly HTML & JavaScript (not anymore flash)
- Frameworks: Angular, React
- Third party libraries (JQuery and others)
- High degree of integration: portals

Why is always more **difficult**?

- Big codebases
- JavaScript is not easy to read: manual review
- Developing and Quality Assure for JavaScript and client components is DIFFICULT, time consuming and error prone.
- Classic approach with SCA (Static Code Analysis) leads to :

1. Too many false positives
2. Too many false negatives
3. Usually this is performed by QA, or periodically, distant in time from the developers writing the code…not the best practice: long time of detection and remediation, and high cost.

HTML Injection

```
<script>
    var userposition = location.href.indexOf("user=");
    var user = location.href.substring(userposition+5);
    document.getElementById("Welcome").innerHTML = " Hello, "+user;
</script>
```

Source

Sink

+

http://vulnerable.site/page.html?user=<img%20src="aaa"%20onerror=alert(1)>

=

<p id="Welcome">Hello, **<img src="aaa" onerror=alert(1)>**</p>

JavaScript Execution and DOM XSS

- These functionalities will interpret a string as JavaScript :

Arguments to eval, execScript, Function, setTimeout, setInterval

Assignments to src attribute of iframe or script tags.

Insecure usage of location.replace/assign.

Insecure assignments to location.

- Those functions can lead to:

JavaScript execution

HTTP Parameter Pollution (HPP)

- The term **Query String** is commonly used to refer to the part between the “?” and the end of the URI
- As defined in the **RFC 3986**, it is a series of fieldvalue pairs
- Pairs are separated by “&” or “;”
- The usage of semicolon is a W3C recommendation in order to avoid escaping
- **RFC 2396** defines two classes of characters:
  - Unreserved: a-z, A-Z, 0-9 and _ . ! ~ * ' ( )
  - Reserved: ; / ? : @ & = + $,

HTTP Parameter Pollution (HPP)

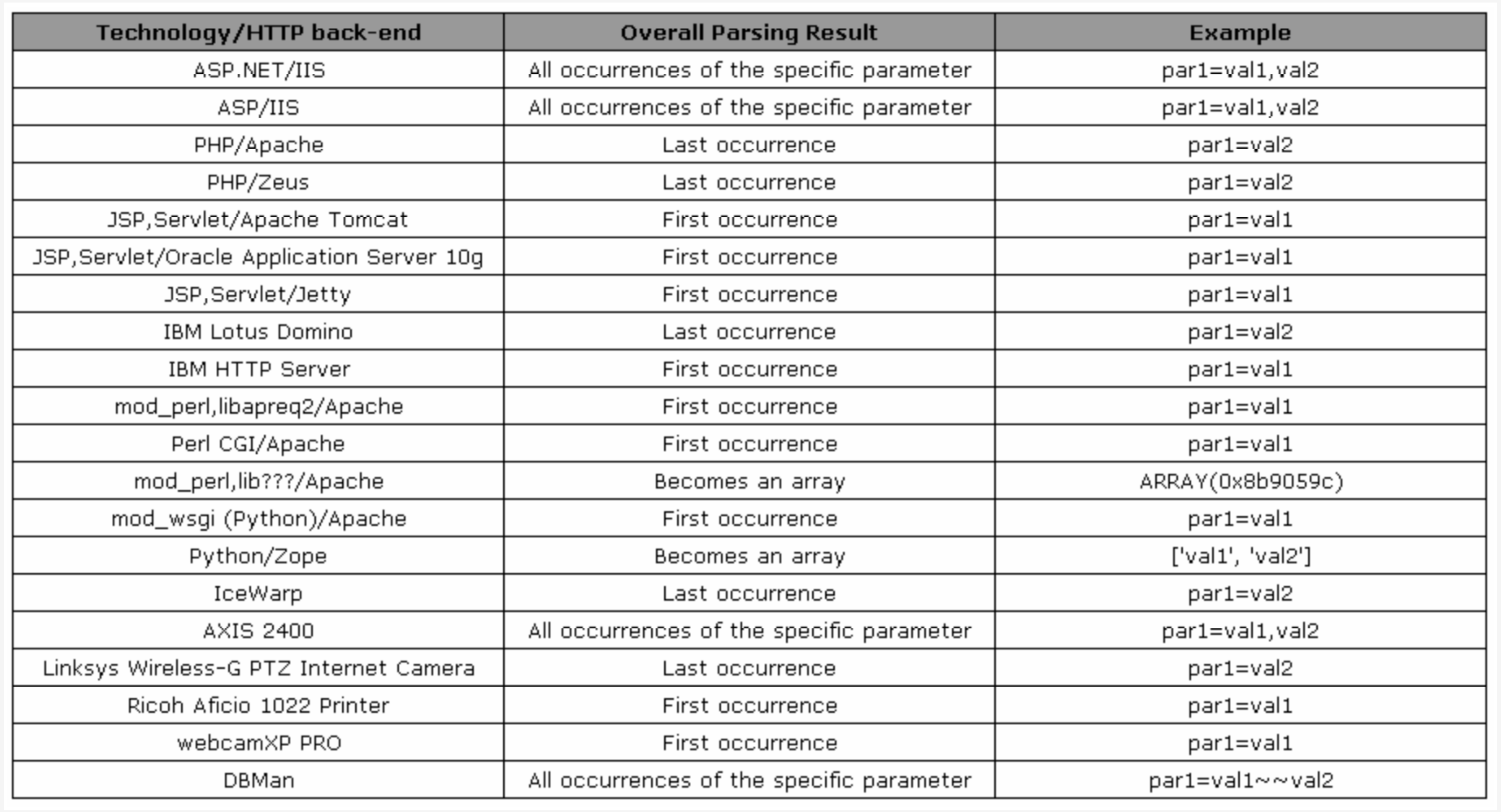
```
GET /foo?par1=val1&par2=val2 HTTP/1.1
User-Agent: Mozilla/5.0
Host: Host
Accept: */*
```

```
POST /foo HTTP/1.1
User-Agent: Mozilla/5.0
Host: Host
Accept: */*
Content-Length: 19

par1=val1&par2=val2c
```

```
POST /index.aspx?par=1&par=2 HTTP/1.1
User-Agent: Mozilla/5.0
Host: Host
Cookie: par=5; par=6
Content-Length: 19

par=3&par=4
```

ASP creates a vector:
1,2,3,4,5,6

Parameter Pollution – Server enumeration



Technology/HTTP back-end	Overall Parsing Result	Example
ASP.NET/IIS	All occurrences of the specific parameter	par1=val1,val2
ASP/IIS	All occurrences of the specific parameter	par1=val1,val2
PHP/Apache	Last occurrence	par1=val2
PHP/Zeus	Last occurrence	par1=val2
JSP,Servlet/Apache Tomcat	First occurrence	par1=val1
JSP,Servlet/Oracle Application Server 10g	First occurrence	par1=val1
JSP,Servlet/Jetty	First occurrence	par1=val1
IBM Lotus Domino	Last occurrence	par1=val2
IBM HTTP Server	First occurrence	par1=val1
mod_perl,libapreq2/Apache	First occurrence	par1=val1
Perl CGI/Apache	First occurrence	par1=val1
mod_perl,lib???/Apache	Becomes an array	ARRAY(0x8b9059c)
mod_wsgi (Python)/Apache	First occurrence	par1=val1
Python/Zope	Becomes an array	['val1', 'val2']
IceWarp	Last occurrence	par1=val2
AXIS 2400	All occurrences of the specific parameter	par1=val1,val2
Linksys Wireless-G PTZ Internet Camera	Last occurrence	par1=val2
Ricoh Aficio 1022 Printer	First occurrence	par1=val1
webcamXP PRO	First occurrence	par1=val1
DBMan	All occurrences of the specific parameter	par1=val1~~val2

MINDED SECURITY
The Software Security Company

HTTP Parameter Pollution (HPP)

Exploiting HPP vulnerabilities, it may be possible to:

- Override existing hardcoded HTTP parameters
- Modify the application behaviors
- Access and, potentially exploit, uncontrollable variables
- Bypass input validation checkpoints and **WAFs** rules

HTTP Parameter Pollution (HPP)

```
void private executeBackendRequest(HTTPRequest request){

String amount=request.getParameter("amount");
String beneficiary=request.getParameter("recipient");

HttpRequest("http://backendServer.com/servlet/actions","POST",
      "action=transfer&amount="+amount+"&recipient="+beneficiary);
}
```

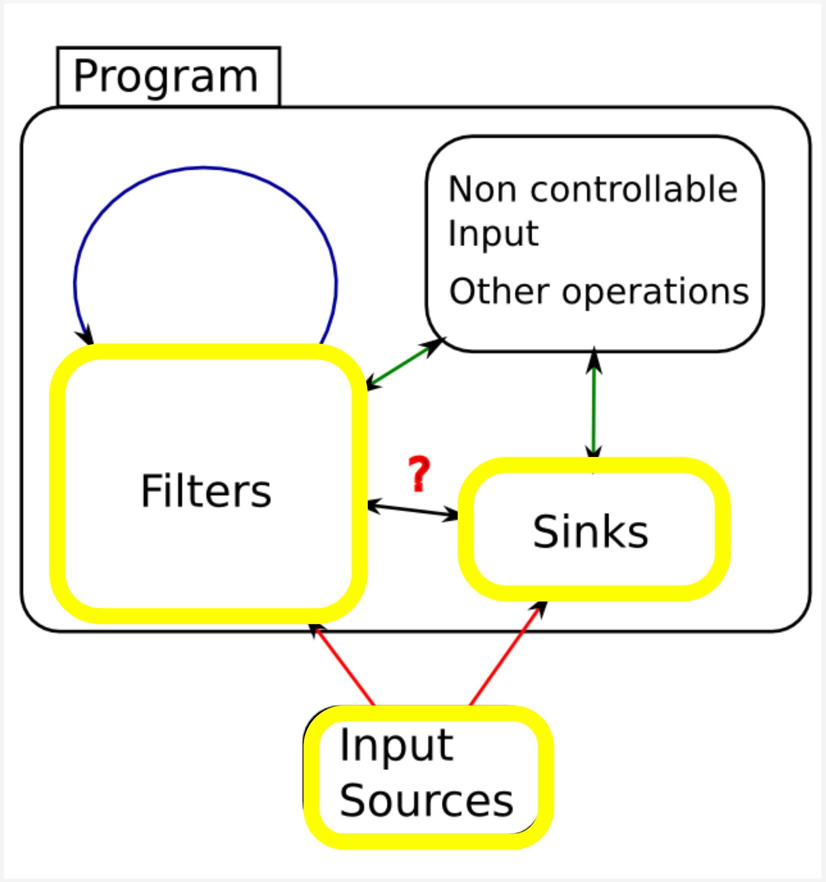
```
http://frontendHost.com/page?amount=1000&recipient=Mat%26action%3dwithdraw
```

⇩

```
action=transfer&amount=1000&recipient=Mat&action=withdraw
```

Code Flow and Taint analysis

- **Sources**: the input data that can be directly or indirectly controlled by an attacker.
- **Filters**: operations on Sources which change the content or check for specific structures/values.
- **Sinks**: potentially dangerous functions the can be abused to take advantage of some kind of exploitation.



MINDED SECURITY
The Software Security Company

Taint analysis

```
<script>
   var l = location.href;
   var user = l.substring(l.indexOf(“user”));          Tainted Source
   document.write(“Hello, ” + user);          Sink
</script>
```

The process of following the tainted value from source to sink is known as **Taint Propagation**.

Direct Input Sources: Location

Attacker controls all parts of a location except the victim hostname.

http://hostname/ path/to/page.ext/ PathInfo ?Query=String #Hash=value

❗ **He can force a user to visit a forged url address.**

DOM XSS Wiki:
http://code.google.com/p/domxsswiki/wiki/LocationSources

Indirect Input Sources: Cookies

Cookie value could have been instantiated somewhere else and retrieved on another page. Its value can be accessed/modified with:

- document.cookie:

```
<script>
   var cvalue = document.cookie;
   var cstart = cvalue.indexOf(“username=");
   cvalue = unescape(cvalue.substring(cstart+9, cstart+9+length));
   alert(“Welcome ” + cvalue);
</script>
```

! **The attacker could force a malicious cookie value**

JavaScript Security

The leading platform for JavaScript Security:

BlueClosure represents a JavaScript Security Platform for Developers, Auditors and Testers to identify and block Javascript flaws in your code.

JS Frameworks Support

BlueClosure can analyse any codebase written with JavaScript frameworks like Angular.js, jQuery, Meteor.js, React.js and many more.

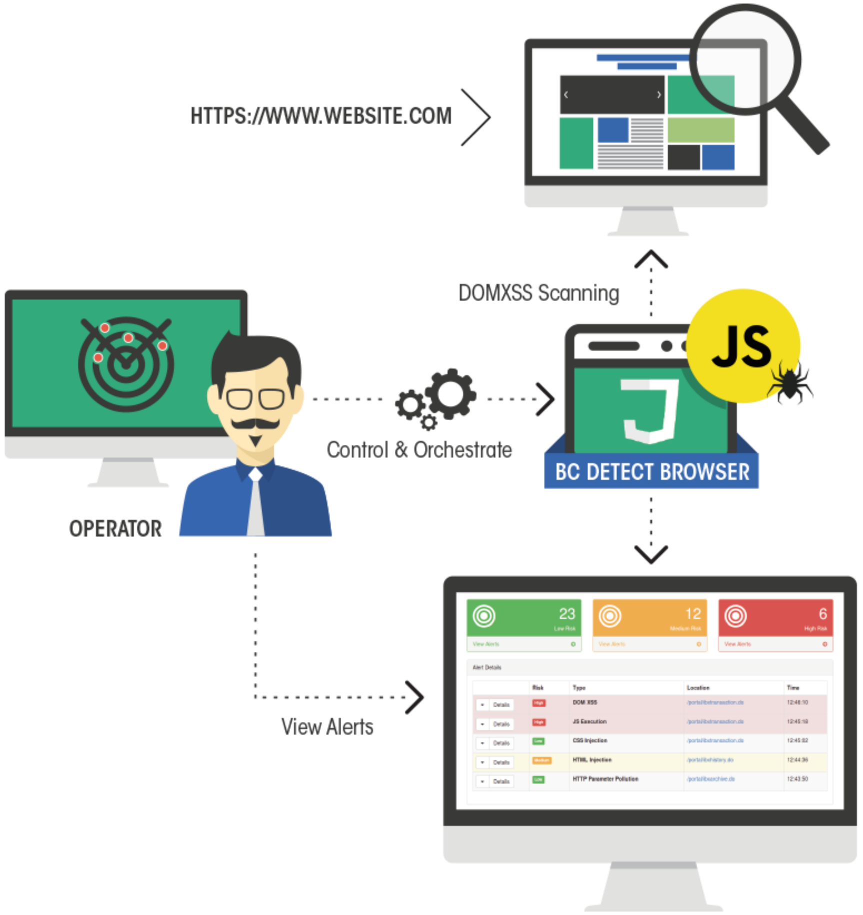
Realtime Dynamic Data Tainting

BlueClosure Detect uses an advanced Javascript Instrumentation engine to understand the code. By leveraging our proprietary technology the BC engine can inspect any code, no matter how obfuscated it is.

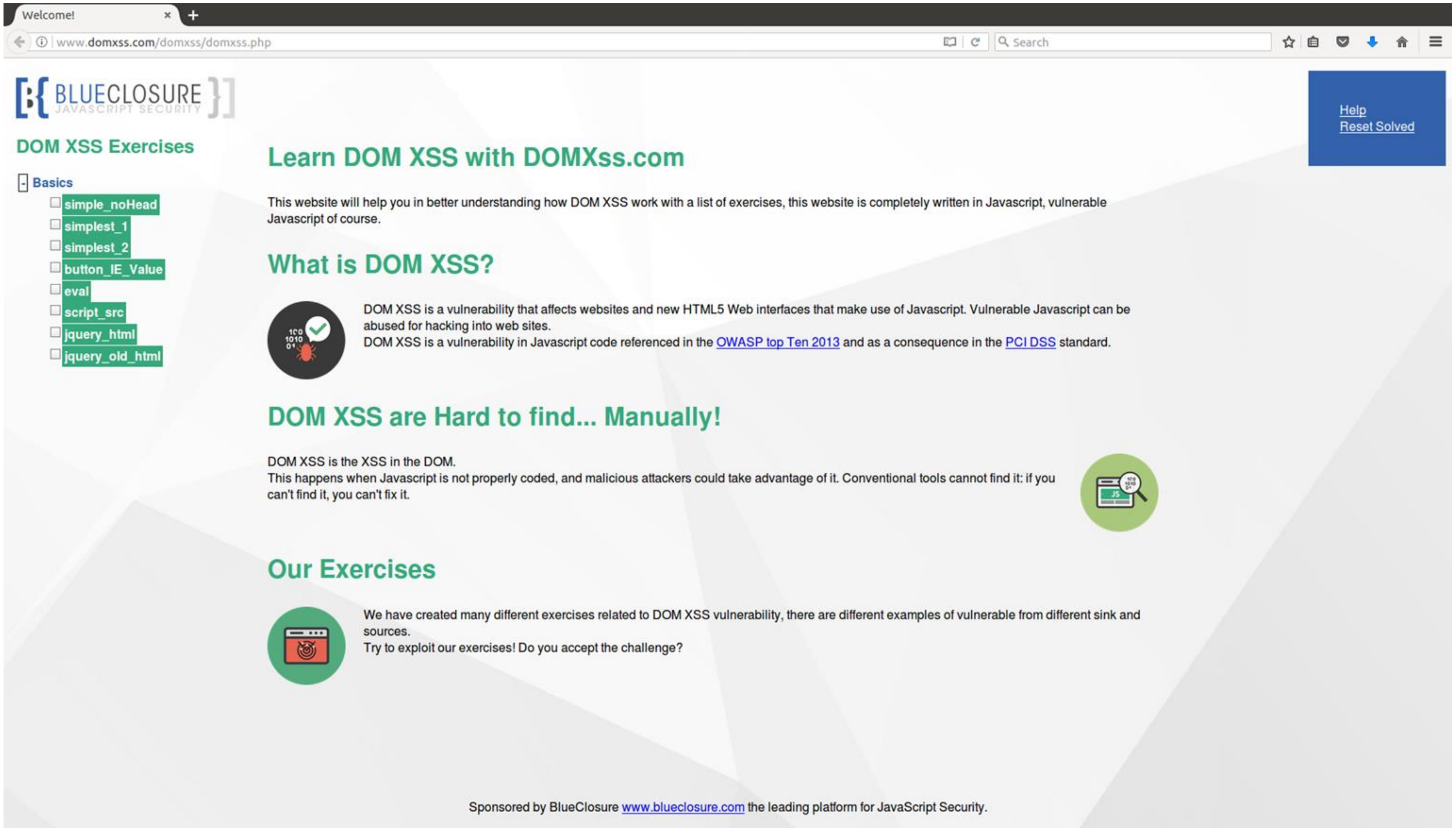
BlueClosure Detect

BC detect discovers JavaScript Flaws Before Anyone Else Does: BC Detect helps Companies to analyse and automatically discover Client Side Vulnerabilities thanks to its Hybrid IAST Engine together with the Smart Fuzzer module.

BC detect finds Client Side Vulnerabilities Easily: Dynamic execution flows, browser quirks, different interpreters: few of the many factors that add up to the inherent difficulty to pinpoint JavaScript security flaws. Conventional tools cannot find them.



Let's test it!



Welcome!

www.domxss.com/domxss/domxss.php

BLUECLOSURE JAVASCRIPT SECURITY

DOM XSS Exercises

- Basics
  - simple_noHead
  - simplest_1
  - simplest_2
  - button_IE_Value
  - eval
  - script_src
  - jquery_html
  - jquery_old_html

Help
Reset Solved

Learn DOM XSS with DOMXss.com

This website will help you in better understanding how DOM XSS work with a list of exercises, this website is completely written in Javascript, vulnerable Javascript of course.

What is DOM XSS?

DOM XSS is a vulnerability that affects websites and new HTML5 Web interfaces that make use of Javascript. Vulnerable Javascript can be abused for hacking into web sites.
DOM XSS is a vulnerability in Javascript code referenced in the OWASP top Ten 2013 and as a consequence in the PCI DSS standard.

DOM XSS are Hard to find... Manually!

DOM XSS is the XSS in the DOM.
This happens when Javascript is not properly coded, and malicious attackers could take advantage of it. Conventional tools cannot find it: if you can't find it, you can't fix it.

Our Exercises

We have created many different exercises related to DOM XSS vulnerability, there are different examples of vulnerable from different sink and sources.
Try to exploit our exercises! Do you accept the challenge?

Sponsored by BlueClosure www.blueclosure.com the leading platform for JavaScript Security.

Questions?

Minded Security Srl
Via Duca D'Aosta, 20
50129 Firenze - Italy
Mail: info@mindedsecurity.com

Minded Security UK Limited
One Canada Square Canary Wharf, E14 5AB
London - UK

Site: http://www.mindedsecurity.com
Blog: http://blog.mindedsecurity.com
Twitter: http://twitter.com/mindedsecurity
BlueClosure: http://www.blueclosure.com

Thanks!