Web Application Hacking Lesson – More XSS Basics and Automated XSS scanning with ZAP

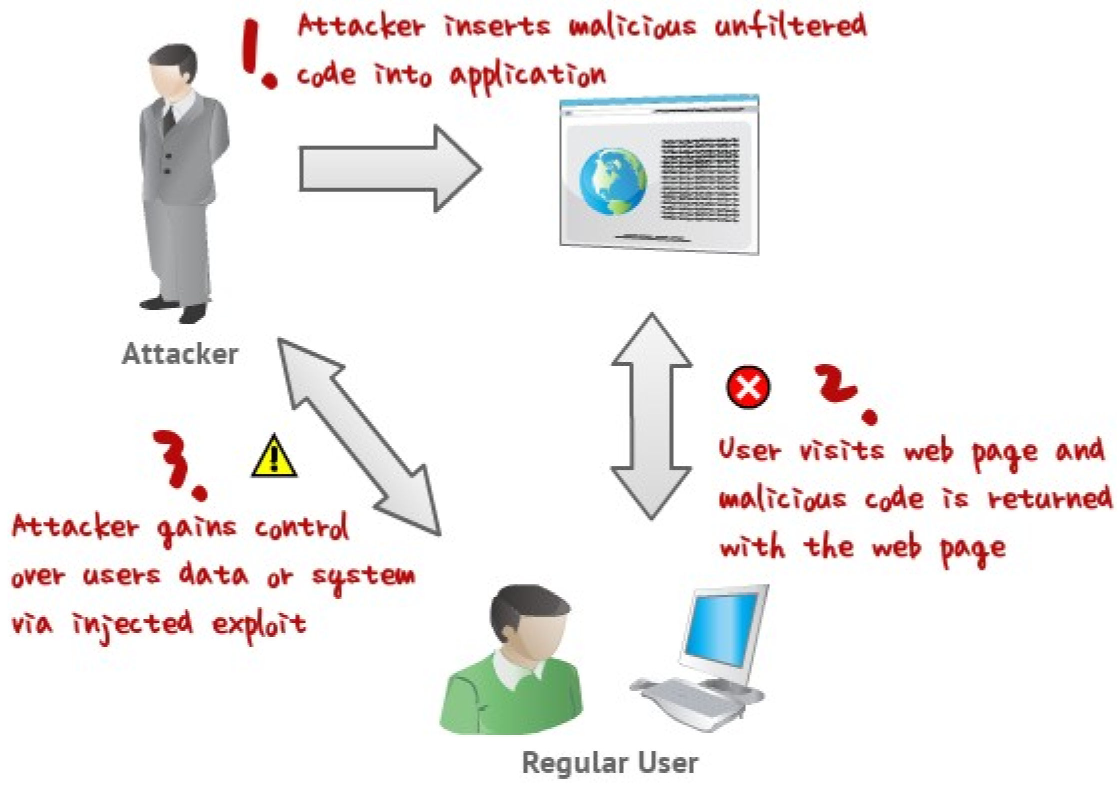
Lesson Objectives

- Review lecture on the Cross Site Scripting(below)
- Complete lab on Cross Site Scripting(below)

Lecture for Cross Site Scripting Introduction

Many people treat an XSS vulnerability as a low to medium risk vulnerability, when in reality it is a damaging attack that can lead to your users being compromised. SQL Injection is a more easily understood vulnerability, as it involves attacking a web application to extract data or modify the web apps back-end database.
An XSS attack involves compromising the users browser rather than the actual web application; keep in mind that the web application is still involved as it is where the attack will originate. So in a typical attack; the bad guy will leverage the web application to effectively launch a browser based attack back at an end user.

Attacker -> exploits web application -> web application delivers a malicious script to a normal users browser -> attacker now has the ability to control the users browser. This is bad for the user and bad for you if you manage the web application.


1. Attacker inserts malicious unfiltered code into application
Attacker
2. User visits web page and malicious code is returned with the web page
3. Attacker gains control over users data or system via injected exploit
Regular User

Some examples of the damage an XSS attack can cause:

- Redirect page to phishing sites, or fake login pages
- Steal the users cookies, allowing them access to other web applications with authenticated sessions
- Insert links to remotely hosted client side exploits within a html body; with the goal of installing malware on the system (key loggers, remote access tools)

These are the most common and dangerous attack outcomes, which typically lead to complete compromise of a users system or personal information.

How does XSS work?

The actual xss attack is formed by injecting unsanitised input into a web application. The input is usually in the form of javascript, that can be stored by the application and returned to other users when they visit the page. Thereby executing the javascript in the users browser.

There are different types of XSS attack and different exploitation points but this is a typical and easy to understand scenario.

How to prevent XSS

Sanitize the input, all user submitted input anywhere in an application must be treated as hostile and filtered. This should be done by the application code, but can also be performed by a web application firewall (WAF) such as mod_security. The most effective way to prevent this is to do both, use well coded applications and have a WAF or filtering as a second line of defense.

In addition there is a HTTP Header that can be used to leverage features in a users browser to prevent XSS attacks. This is the X-XSS-Protection HTTP Header.

Keep in mind that the malicious input could be executed from not only *script tags* but also the *body tag*, *image tags* and more. A browser can be quite forgiving even if the resulting html is broken, it still may execute the script.
This tutorial is aimed at those who need a basic understanding of cross site scripting. For further information take a look at the resources available on the OWASP web site.

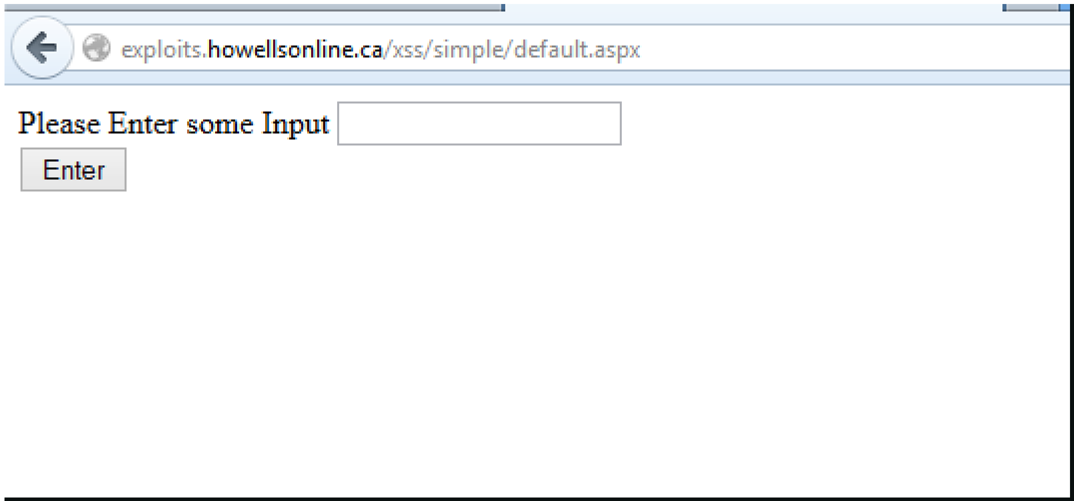
Lab for Cross Site Scripting Introduction

Lab setup

- **Instructions**:
    1. The target site is located here: http://exploits.howellsonline.ca/xss/simple/default.aspx
    2. Update Java on Backtrack 5:
       It's pretty simple really, just wanted to document it in case I ever have to do it again. You can download the latest tar.gz JDK package from Oracle's website http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html
    3. Note change file name to match whatever the current version of Java 7 that you download.

```
root@bt:# mv jdk-7u3-linux-i586.tar.gz /opt/java/
root@bt:# cd /opt/java
root@bt:/opt/java# tar xvfz jdk-7u3-linux-i586.tar.gz
root@bt:/opt/java# update-alternatives --install "/usr/bin/java" "java" "/opt/java/jdk1.7.0_03/bin/java" 1
root@bt:/opt/java# update-alternatives --set java /opt/java/jdk1.7.0_03/bin/java
root@bt:/opt/java# update-alternatives --install "/usr/bin/javac" "javac"
"/opt/java/jdk1.7.0_03/bin/javac" 1
root@bt:/opt/java# update-alternatives --set javac /opt/java/jdk1.7.0_03/bin/javac
root@bt:/opt/java# javac -version
javac 1.7.0_03
root@bt:/opt/java# java -version
java version "1.7.0_03"
Java(TM) SE Runtime Environment (build 1.7.0_03-b04)
Java HotSpot(TM) Client VM (build 22.1-b02, mixed mode)
root@bt:/opt/java#
```

4. Download the latest version of Zed Attack Proxy https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
5. 3 minute video on how to setup ZAP with Firefox http://www.youtube.com/watch?v=qZkxiRDRIAY

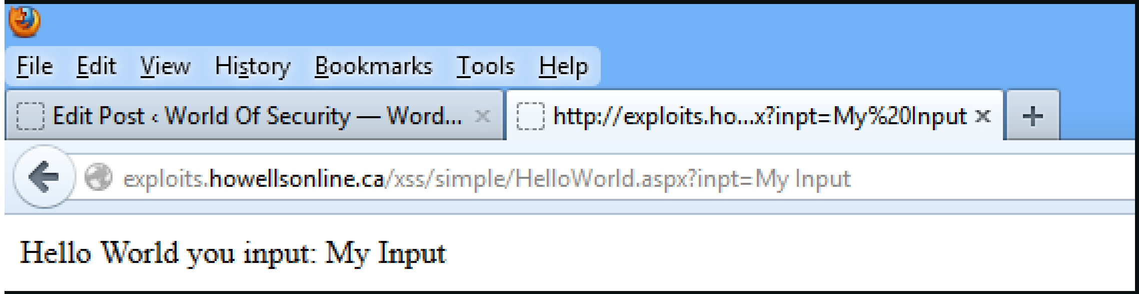


exploits.howellsonline.ca/xss/simple/default.aspx

Please Enter some Input

Enter

Here you have a basic input form, that takes some input which could actually be anything a user name, password whatever you want it to be. When the user behaves and the form works correctly as you can see, the input page redirects your input to a page that does some processing, in my case the processing page simple echos the output back out to you, as observed below.

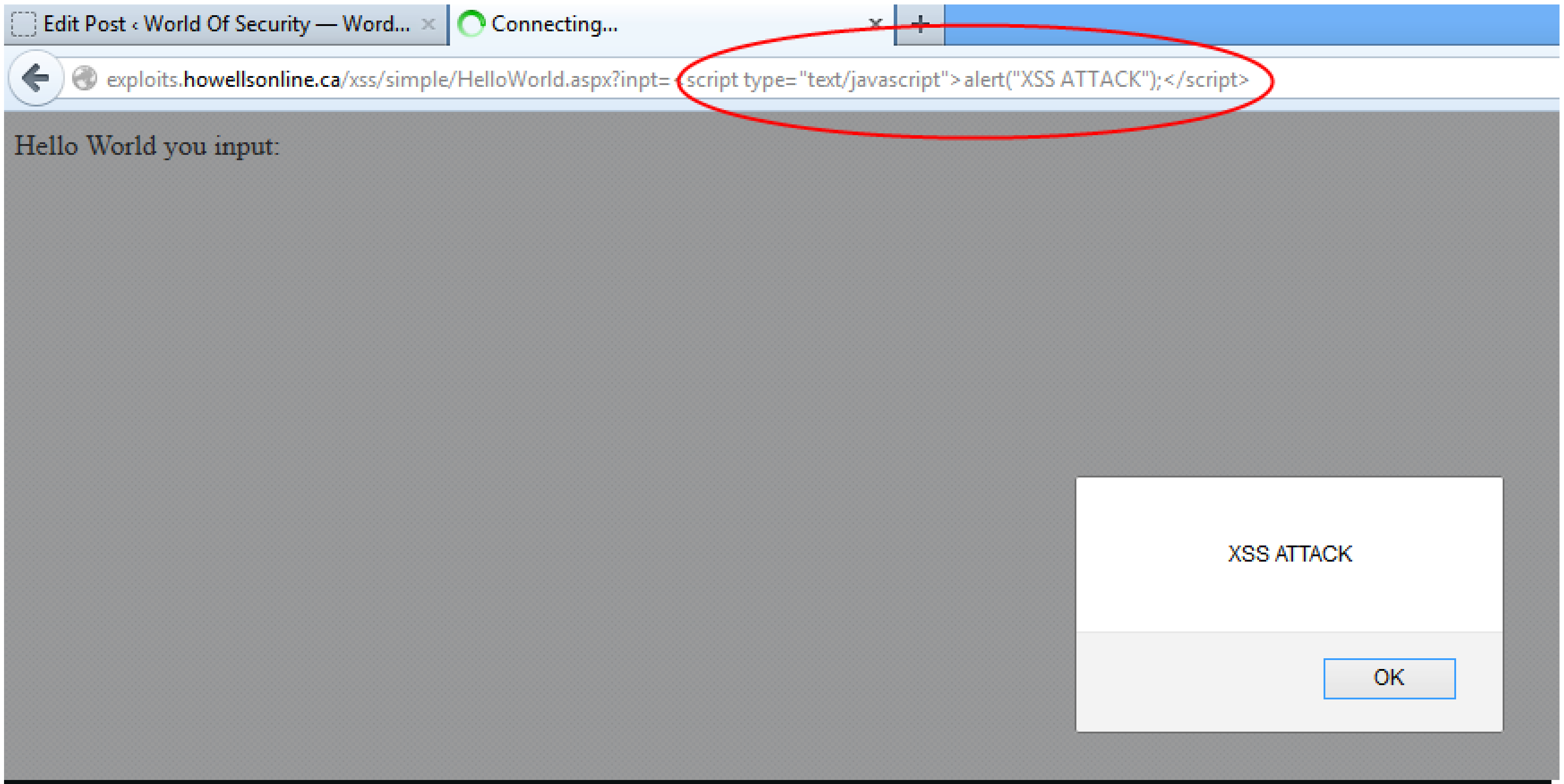


File Edit View History Bookmarks Tools Help
Edit Post ‹ World Of Security — Word...
http://exploits.ho...x?inpt=My%20Input

exploits.howellsonline.ca/xss/simple/HelloWorld.aspx?inpt=My Input

Hello World you input: My Input

This page has an xss vulnerability in it, immediately because all this page does is echo the data back it’s suspicious but not 100% obvious. The easiest way for me as a primitive attacker is to test my theory out, watch what happens when I test that theory.



exploits.howellsonline.ca/xss/simple/HelloWorld.aspx?inpt=<script type="text/javascript">alert("XSS ATTACK");</script>

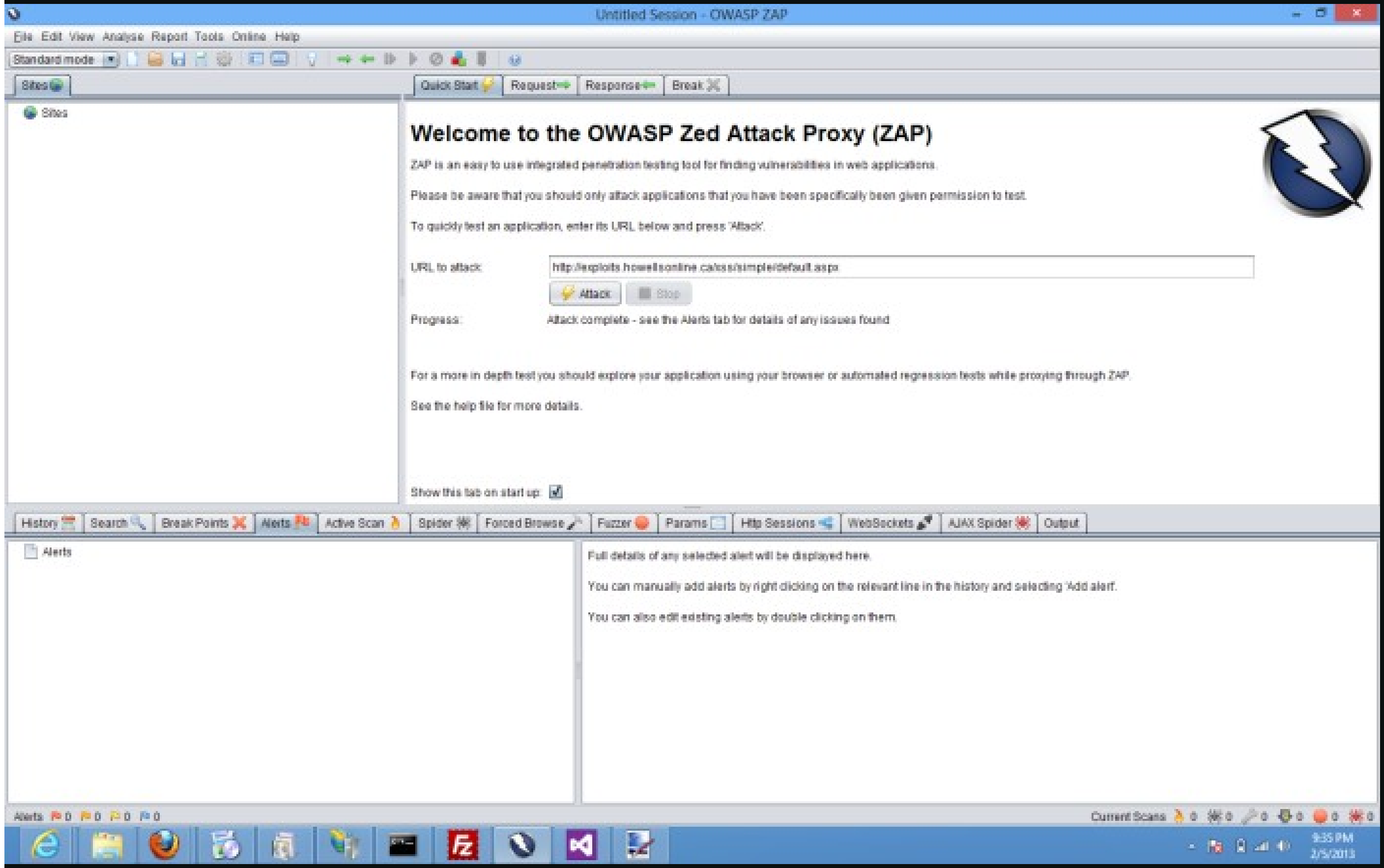
Hello World you input:

XSS ATTACK

As you can see with from the images above I attempt to test my theory with a little bit of JavaScript. Notice how in the second image that, my JavasSript is appearing as the value for the input parameter exactly as I entered it on the page before? This tells me I don't even have to waste my time going through the first page, I can immediately send out a link with whatever xss vulnerabilities I want to my potential victims, I just have to decide how I want to exploit the vulnerability I've found, whether it be through JavaScript, Html or some other mechanism. Fortunately in this case the input is not actually being stored in a file or a database so the attack will only attack one victim at a time, once they click on the link. If the input was stored anywhere! cookie, db, file, webservice & loaded every time that would be great for me as an attacker because I'd only have to attack the site once, and have my input recalled for every user who visited the site. It's worth noting that I don't actually have to keep the user on the page, my javascript could for a new window open or a window redirect. If that were the case I could navigate them away from the page every time without accomplishing the goal of page.

Another Detection Mechanism

I understand that in a large web app manually testing every form for a potential xss issue is unrealistic for qa and even more so for developers/engineers. I am a huge fan of the ZAP attack Proxy this is an intercepting proxy that is quite powerful.



Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://exploits.howellsonline.ca/xss/simple/default.aspx

Progress: Attack complete - see the Alerts tab for details of any issues found

For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

See the help file for more details.

Show this tab on start up: ✓

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

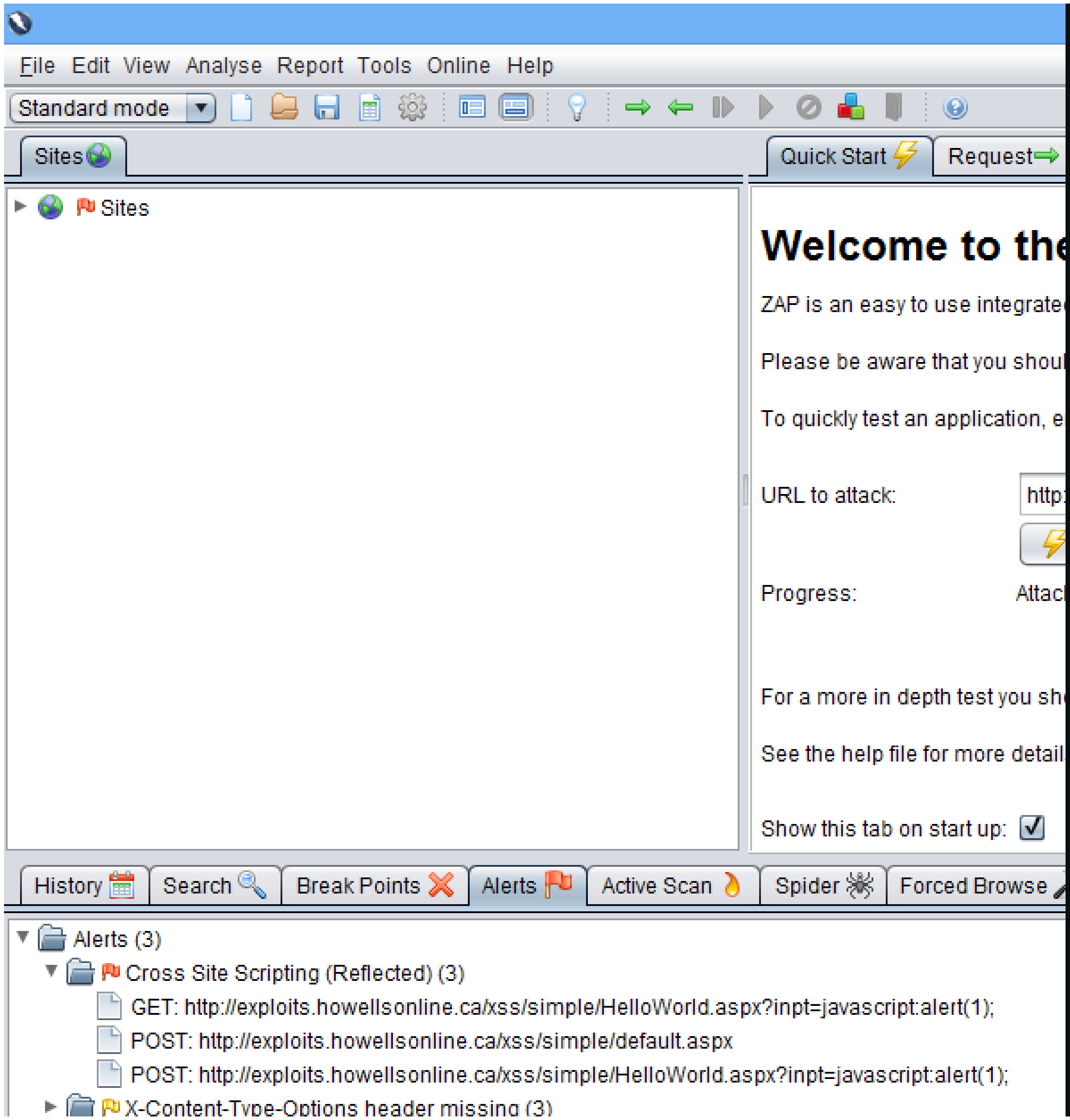
You can also edit existing alerts by double clicking on them.

Alerts 0 0 0 0

Current Scans 0 0 0 0 0 0

9:35 PM
2/5/2013

As you can see with the ZAP attack proxy. I fire it up, and enter the URL of the site that I want to scan (Attack). Zap will test the site against standard definitions and attack behaviors.



Welcome to the

ZAP is an easy to use integrate

Please be aware that you shoul

To quickly test an application, e

URL to attack: http:

Progress: Attac

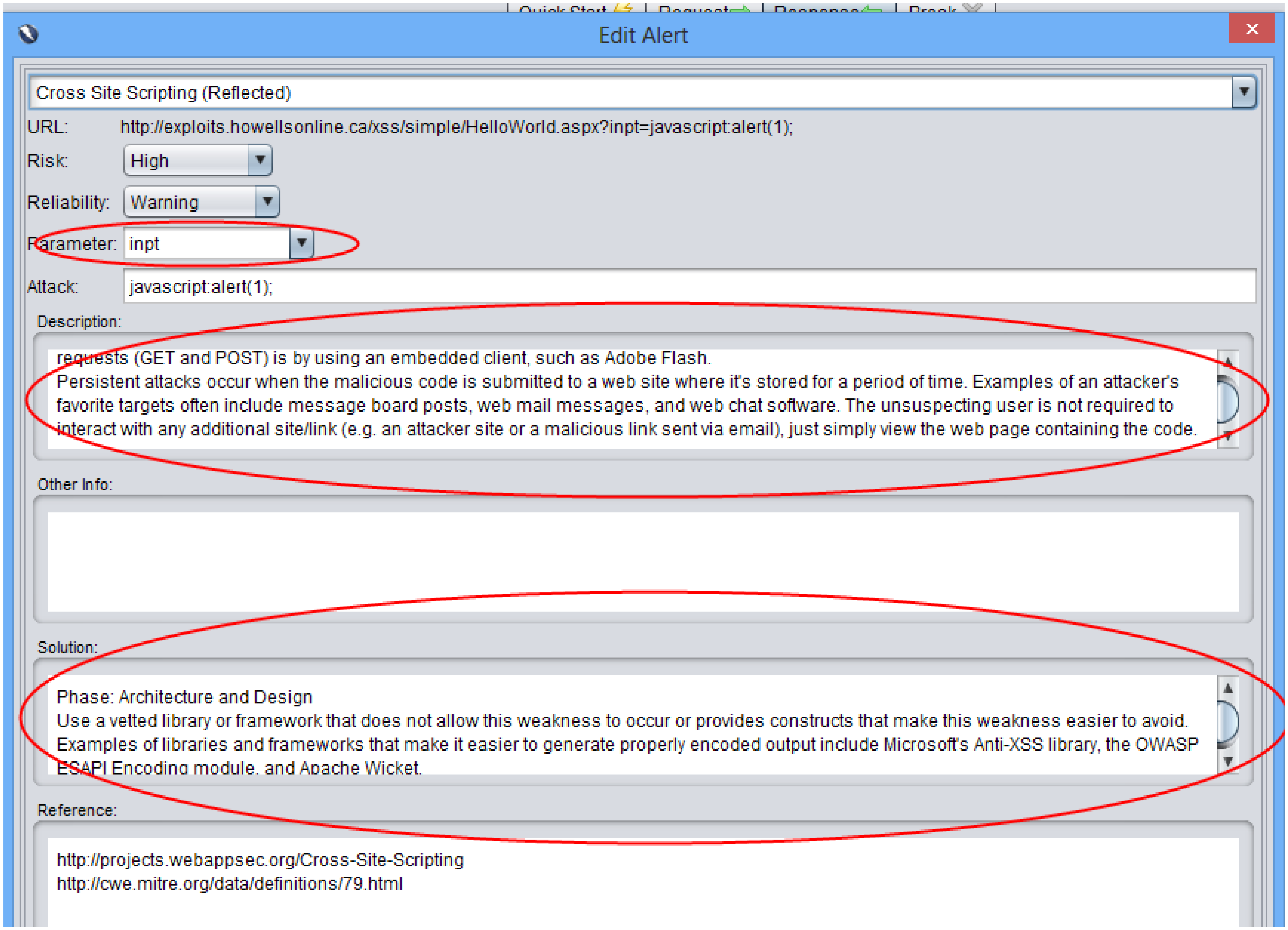
For a more in depth test you sh

See the help file for more detail

Show this tab on start up: ✓

Alerts (3)
Cross Site Scripting (Reflected) (3)
GET: http://exploits.howellsonline.ca/xss/simple/HelloWorld.aspx?inpt=javascript:alert(1);
POST: http://exploits.howellsonline.ca/xss/simple/default.aspx
POST: http://exploits.howellsonline.ca/xss/simple/HelloWorld.aspx?inpt=javascript:alert(1);
X-Content-Type-Options header missing (3)

When Zap is done it’s scan, anything that, proxy thinks is worth my attention shows up in the alerts window, As you can see I have some xss concerns here, 3 of them to be exact.



Cross Site Scripting (Reflected)

URL: http://exploits.howellsonline.ca/xss/simple/HelloWorld.aspx?inpt=javascript:alert(1);

Risk: High

Reliability: Warning

Parameter: inpt

Attack: javascript:alert(1);

Description:

requests (GET and POST) is by using an embedded client, such as Adobe Flash.
Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.

Other Info:

Solution:

Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Reference:

http://projects.webappsec.org/Cross-Site-Scripting
http://cwe.mitre.org/data/definitions/79.html

When I click on, one of the alerts I get a very detailed description of the issue, it tells me of the vulnerability the input parameter(s) that were vulnerable, and where I should have caught this issue to remediation it as well it provides me with some references for more info should I wish to follow up on the issue.

The Dangerous thing with the ZAP proxy is, as much as it can be used for beneficial purposes it can also be used for malicious purposes. An attacker is just as capable for downloading ZAP and scanning your website with it to find all the vulnerabilities that one can detect. The attacker now has a very good idea where to start hitting your site for potential xss vulnerabilities, and what URL's they can take advantage of.

A Remediation

One might think that a solution this problem would should be implemented on the first, input page and to scan the inputs, before the page is redirected, well they would be wrong! Remember how earlier I said, that i did not to process a request through the first input page, I could take advantage of the vulnerability directly on the 2nd page? That's where the solution needs to come from. As of .NET 4.0 Microsoft does a decent job with .NET projects to scan HTTP requests and block them if they contain potential xss issues.

I tend to not trust this approach because:

1. It can be turned off in the web.config
2. It can be turned off at the page level

This approach only works within the confines of a .NET Web app. The only safe way to re-mediate a xss scripting attack, is through diligent white listing of all inputs. That's a discussion for another time. In the mean time, would developers and QA please start testing their projects! It only takes a few minutes to run a ZAP attack scan, which can quite possibly save your firm & you a lot of trouble in the future! Tune in soon, as I dig deeper into more complex xss vulnerabilities, how they are created, how they are exposed and how to prevent them!

-

Proof of Lab

1. Proof of Lab
   - **Proof of Lab Instructions**:
     1. Do a <PrtScn>
     2. Paste into a word document
     3. Email me
   -

Questions:

1. Does XSS affect the browser or the web application?
2. Explain in your own words how XSS works?
3. What are three ways to prevent Cross Site Scripting in your web application?
4. Besides the script tags, name two other types of tags that XSS code could be executed?