OWASP

Open Web Application Security Project

# O2 Project ALPHA

## OWASP Foundation

# O2 Web Scripting

**O2 is your one stop shop for everything automation.** I have only touched the surface of what Dinis has built into O2 but I highly recommend it as a very fast way to write automation scripts, create demo's but for what I use it for is penetration testing. This is supposed to be a cheat sheet to come back to as a reference when using O2 but it might help people learn the tool aswell.

### If you like C# You'll love this....

Simply put O2 scripting using Wathin which is an scripting library in C#. This means that if you know C# you'll have no problems getting to grips with the language. I didn't know C#, I knew Java, but its so incredibly straight forward it really shouldn't be a problem for anyone who has some coding knowledge.

The appendix is slowly filling, archiving some of the links to the examples written by Dinis.

# Contents

# Starting a New Script

Go to Contents

When beginning a new script the first thing you need to do is bring inject an instance of a web browser into O2. To do this we create a panel within O2, then insert an instance of Internet Explorer into that:

```
var topPanel = panel.clear().add_Panel();
var ie = topPanel.add_IE();
```

If you dont want to inject into the O2 window, you can create a new window. This isn't very convenient when designing your script but does come in handy later:

```
var topPanel = "WINDOW TITLE".popupWindow();
var ie = topPanel.add_IE();
```

Every script will need some basic libraries to work. Libraries can go anywhere in the script, the easiest and most convenient place to put them is right at the bottom.

```
//using FluentSharp.Watin;
//O2Ref:FluentSharp.Watin.dll
//O2Ref:WatiN.Core.dll
```

Lastly, it is necessary to know how to load the webpage we want to automate:

```
ie.open("http://www.google.com");
```

So a very first script could look like:

```
var topPanel = panel.clear().add_Panel();
var ie = topPanel.add_IE();
ie.open("http://www.google.com");

//using FluentSharp.Watin;
//O2Ref:FluentSharp.Watin.dll
//O2Ref:WatiN.Core.dll
```

## Links & URLs

Go to Contents

Links can be clicked to go to different areas of the site. First we need to detect them on the page. This is a little bit of an art as you will need to be able to look at the source code for the site. I tend to do this from another browser as generally the source code viewer is better than the IE one. The other way is to get O2 to pass back all the links on the page.

```
return ie.links();
```

That will put all the links into a **list** in the debugging area. By clicking on one you can see its properties.

If you run the above command in the above script, you would get one link in the list called drive - for Google's Drive. By clicking on it, you can get its properties - including its ID, "gb_25". If a link has an ID you can click that link like this:

```
ie.link("gb_25").click();
```

However, not always do links have ID's so one of the easiest ways of clicking on links like this is with

```
ie.links().where((link)=>link.url().contains("drive")).first().click();
```

That may seem long and confusing. The first thing to do is find a unique bit of the link that will allow the script to know which link we are talking about. The link for Google Drive is

    https://drive.google.com/?tab=wo

```
ie.links();
```

creates a list of all the links

```
.where((link)
```

Is saying I want the items in the list, that are of the type **link**

```
url().contains("drive"))
```

Where the url of those links contains the keyword "drive"

```
.first().click()
```

Finally, get the first one in the list that matches those criteria and click it. There is also

```
second()
third()
```

If you want further than that you can do

```
.contains("drive"))[i].click();
```

where i is the number of the link that matches that criteria. Remember - in computer world the first element in a list is the zeroth element i.e i=0.

To get a page's url:

```
return ie.url();
```

or

var webaddress = ie.url(); return webaddress.str();

## Forms

Go to Contents

Forms are everywhere. To really script a page, it is necessary to be able to fill them out. The first thing to deal with are input fields such as google's search field. You can fill out a field based on knowing its name. To find out what fields are on the page:

return ie.fields();

We can look at the field properties to find out what they are called, however if we look at the source of Google's homepage (I recommend using firebug), there is this line:

<input id="gbqfq" class="gbqfif" name="q" type="text" autocomplete="off" value=""

Its longer that that, but thats the beginning and important bit.

To input text into it using its name, which we can see is "q"

```
ie.form("q").value("Dinis Cruz");
```

Don't forget that hidden fields are still fields, and if you need to set the value of a hidden field, you can do it in the same way.

Next we need to be able to perform the search. To do this we click on the Google Search button.

## Buttons

Go to Contents

To see what buttons there are:

```
return ie.buttons();
```

You can click on a button with

```
ie.button("button name").click();
```

However, these days there is a lot of javascript on the web. This means that a lot of the elements are not straight forward for elements. And infact that is the case with Google's search button. This means that there are different way's of performing the same thing when designing a wesbite, and therefore we need different ways of automating these things so we can cope with the entire web, rather than just little bits of it.

So if the above doesn't work, don't fret. We just need to bring some jquery controls into our script:

```
        ie.inject_jQuery();
        ie.invokeEval("jQuery('form').submit()");
```

The first line injects jQuery into the webbrowser, the second line uses jQuery to submit the form. This is definitely not a bad way of doing it at all, and probably when automating something like Google, quite a good way of doing it. However from a viewer's perspective, you don't really get to see what's going on - sometimes thats a problem. My personal favourite then, is to inject a button into the page that will submit the form. Our own personal button.

## Fields

Go to Contents

Find where you want to inject it - we know that, we want to inject it near the search box, which we also know is called "q". Then:

```
ie.field("fieldName").injectHtml_afterEnd("<input type='submit' value='Submit'>");
ie.button("Submit").click();
```

It could also be an element you inject after:

```
ie.element("elementName").injectHtml_afterEnd("<input type='submit' value='Submit'>");
ie.button("Submit").click();
```

There is also "injectHtml_beforeEnd(...)"

Note: To use any HTML injection you need to include the library

```
//O2Ref:Microsoft.mshtml.dll
```

It can take a bit of trial and error to get the button to inject to exactly where you want it, but its a good way of doing it, and also is a good lesson at this stage.

You can inject anything into the page that is valid HTML. Remember that if you are trying to inject so that you can submit a form with a form submit button, it needs to be inside the

tags.

If you want to click on a different type of element, say a checkbox or radio button, treat it as an HTML element:

```
ie.elements("INPUT").where((input)=>input.Name=="Submit").first().Click();
```

At this point its probably a good idea to say there is autocomplete. If you start typing after putting down a "." it will suggest the things you can do. Try writing:

```
ie.
```

and wait. A drop down will appear, as you start typing it will refine the options, This is a good way of finding out what's available to you.

## Grabbing text from the page for analysis or...whatever.

Go to Contents

Sometimes a decision needs to be made based on whether something has appeared on the page. I wrote a script that Google picked up as a potential bot therefore redirected it to a Captcha. When my script loaded the page it checked to see whether:

> To continue, please type the characters below:

appeared anywhere on the page

```
var captchaFound = ie.html().lines().where((line)=>line.Contains("To continue, please type the characters below:")).first();
```

Then I checked whether captchaFound was null. If it was, all was good, if it wasn't, Google thought my script was a bot (well done Google). The script then asked the user to enter the captcha value, and it carried on.

As another example, I wanted to get someone's **name** and **job** out of LinkedIn from the profile page. by checking the source code (again I recommend firebug) I could extract the text between some elements:

```
var jobTitle = ie.html().lines().where((line)=>line.Contains("headline-title title")).first().subString_After(">").subString_Before("<");
```

Its **C#** remember so for the name, I could use the page's url and split it out:

```
var data = ie.geturl();
var splitAddress = data.split("/");
var name = splitAddress[4].split("-");

return name[0] + " " + name[1];
```

## Injecting into the page

Go to Contents

Often its nice to be able to change a value on the page, or add a text field or whatever. I went over some of this earlier with submitting forms - I injected a button into the page to submit the form. One example of this is Dinis' JPetStore hack. He hacks the page so that the attacker can input how

much he'd like to pay for his fish into a nice text field. This is a little forced as an attacker probably wouldn't go to such lengths, but from the point of view of a demonstration this is great as it shows a project manager a very straight forward and easy to do, hack.

```
ie.field("fieldName").injectHtml_afterEnd("ANY HTML/JS here!");
```

Remember to include the library

```
//O2Ref:Microsoft.mshtml.dll
```

### Posting Data

Go to Contents

If you want to bypass controlling the site completely, and just posting data using the POST method you can do

```
var url = "http://theurl.com";
var postData = "{PUT POST DATA HERE}":
url.POST(postData);
```

The GET method is straight forward as you can just ie.open(""); the link and place variables into the correct part of the URL

```
variable="Dinis+Cruz" //my google search
page2=10;//google displays 10 links a page
ie.open("https://www.google.co.uk/?gws_rd=cr&ei=OXVJUvH2A6Ku7AboqYH4BQ#q="+variable+"&safe=off&start="+page2);
```

### Scrolling

Go to Contents

If you are using the flashing technique to show a user where the script is acting, then you will want to be able to scroll that part of the page into view

```
ie.link("hello").scrollIntoView().flash().click();
```

Remember you can scroll any element into view.

### Interface and interaction with a user

Go to Contents

When making an application that you might use more than once, its a pain to have to edit the code each time you want to make a change. Another example was I was working on a project and wanted to send the demonstration of the hack to the customer. I wanted a "commentary" as the

hack took place. A nice way of doing this was having text across the top explaining what was currently happening.

## Action Panels

Go to Contents

To create an "area" at the top of the panel to display text/buttons/textboxes, add an actionPanel:

```
var actionPanel = topPanel.insert_Above(40);
```

That adds an actionPanel 40 pixels high.

## Labels

Go to Contents

Then to add a label to display text:

```
actionPanel.add_Label("Displaying some text").size(20);
```

You can clear the actionPanel with:

```
actionPanel.clear();
```

## Text Boxes and Buttons

Go to Contents

A more advanced actionPanel may contain a textbox and a button. One option here is the:

```
  actionPanel.add_LabelAndTextAndButton("Type Here:",
  "initial text",
  "Button Text",
  actionToCall);
```

See Functions and Actions to understand how to get the clicking of the button to call an action and perform a task.

## Datagrids

Go to Contents

To display data back to a user a datagrid is often a good bet. Infact I dont think I've written a script where I didnt use one. Add it after adding the ie window to the panel

```
var dataGridView = topPanel.insert_Right("Results").add_DataGridView();
```

Then add the columns you want

```
dataGridView.add_Columns("Column 1", "Column 2", "Column 3", "Column 4");
```

Adding rows to the datagrid is easy

```
dataGridView.add_Row("C1 data", "C2 data", "C3 data", "C4 data");
```

You can clear the datagrid with

```
dataGridView.remove_Rows();
```

## Delays

Go to Contents

If you want to slow a demonstration down, then there are two useful things to know. The first is just a straightforward delay

```
1000.wait();
```

That will delay for 1 second. You can set any number of milliseconds to delay for though.

Another is getting the field/button or whatever O2 is doing to flash a few times before clicking or moving on to the next task - for instance

```
ie.link("hello").flash().click();
```

That will find the link called hello, flash it yellow a couple of times and then click it. You can piggy back as many flashes as you like before clicking.

```
ie.button("button").flash().click();
```

Just another example....

## Saving Out data

Go to Contents

If you want to keep the results then you can store them to a list and save the list of data to a temporary text file that you can post process.

Create a list

```
 var allData = new List<string>();
```

add to the list

```
allData.add(stringToStore.ToString());
```

Then save it out, and if you wish, display it

```
    var savedData = allData.sort().save();
    savedData.showInCodeViewer();
```

## Reading From Files

Go to Contents

TBC... It is possible to read from CSV etc. I need to write this up

## Actions and Functions

Go to Contents

Actions and functions differ in the fact that an action doesn't return any data. To create an action:

```
Action myAction =
()=>{
  //Action code here
};
```

That can be called whenever its needed by calling

```
myAction();
```

However you can pass an action variables like this

```
Action<String> myAction =
  (input)=>{
            //Do something with variable input
      };
```

Note we have stated that input with be of type string. It could be an int or a float or a boolean. Call it in a similar way

```
myAction("hello");
```

A function can return data

```
Func<string,string,bool> myFunction =
(firstname, sirname)=>{
  //function code here...
  return firstname == sirname ? true : false;
  };
```

This function takes two strings and returns a boolean. In this case it checks whether the firstname matches the sirname and if so, returns true, otherwise returns false. Call it like

```
var names = myFunction("Dinis", "Cruz");
```

now the variable names will equal the result of calling that function

### Display a message across the IE window

Go to Contents

Just try it - easiest way to understand what it does

```
ie.showMessage("This is a message!");
```

### Exporting to a standalone product

Go to Contents

TBC... For demos and applications they can be exported to .exe files. I need to write this stuff

## Example code snippets

Go to Contents

These are examples, but also useful to remember how to do things

I wrote this to detect the captcha from google, and if so ask the user in a message popup box to enter the captcha

```
    Action<String> detectCaptcha =
  (captchaInput) =>
  {
    if (captchaInput.isNotNull())
    {
      var answer = "Enter the captcha here. Be very careful when reading the captcha".askUser();
      while (!answer.valid())
      {
        answer = "Enter the captcha here".askUser();
      }
      ie.field("captcha").value(answer);
      ie.button("Submit").click();
    }
  };
```

Call it like so

```
  var captchaFound = ie.html().lines().where((line) => line.Contains("To continue, please type the characters below:")).first();
  detectCaptcha(captchaFound);
```

This is a function that asks how many of googles pages you would like to browse through. It isn't very meaningful on its own, but the idea was that the script would look for google results that

contained the word **linkedin**. I wanted the user to input how many of the results pages the script should scan through. It also demonstrates using pop ups and returning integers.

```
      Func<int> numPages =
      () =>
      {
          var pages = "Have you checked how many pages there are for this search?".askUser();
          int passInt = 0;
          while (!int.TryParse(pages, out passInt))
          {
              pages = "Have you checked how many pages there are for this search? - thats not a valid
integer".askUser();
          }
          return passInt;
      };
```

Scrolling through a list

```
   var allData = new List<string>(
      foreach (var data in allData.sort())
     {
       ie.open(data); //assuming this list stores urls
   }
```

# Appendix

Go to Contents

Links to some of Dinis' tutorials for using O2.

- Demonstrates automating a website for login
- Automating WebGoat
  - Demonstrates Selecting from Dropdowns
  - Filling in forms
  - Clicking buttons
  - Flashing fields for demonstrations
- More explanation into packaging an O2 Script into an executable
- JPetStore
  - JPetStore is an example of a Spring web application. Spring has an inherent vulnerability.
  - You can download the example web app and everything you need here
  - This is an example of exploiting the vulnerability in the Spring Framework using O2


5 800099 195442