Introducing the BeanShell in OWASP ZAP

Introduction

The BeanShell is an interactive Java shell that can be used to execute BeanShell scripts. These scripts are a simplified form of Java that use many elements from Java syntax, but in a simpler scripting format. All Java code is also valid BeanShell code.
BeanShell integration in OWASP ZAP enables you to write scripts using the ZAP functions and data set. This can be a very powerful feature for analyzing web applications.

BeanShell Console

The console is started from the *Tools* menu, and contains a split screen where the top half is the interactive BeanShell console and the bottom half is a simple text editor. For complex scripts, you're encouraged to use a Java editor. Scripts can be loaded, saved and evaluated from the editor.
When the BeanShell starts a number of objects from ZAP are available for use, namely:

- the *Model* singleton, through the object named ***model***
- the *SiteTree* tree of current sites through the ***sites*** object
- an instance of *HttpSender* through the ***sender*** object

Notice that the BeanShell is loosely typed. Therefore, it is not necessary to declare variables before using them – this makes scripts a bit more concise than regular Java. But of course, if you did want to define the type you can.

Using the Site Map

Let's start with something useful and typical: Iterate through all the site nodes and check for the existence of a file. A script has already been created that accomplishes this, choose *Load* and select the *example.tree.bsh* file. Before clicking *Evaluate*, first browse to a site through ZAP to populate the tree:

Now click on evaluate to execute the script that's in the editor. If there are no errors, then you can now start using the object defined in the script by issuing these commands:

t = Tree();

Which constructs a new Tree object and assigns a reference to *t*.

t.find(sites.getRoot(), "index.html");

Call the *find* method on t, which takes a *SiteNode* as the first argument and a resource to find as the second. In this case, the method will iterate through all the nodes in the tree, because we started at the root, and will find index.html files.

Instead of iterating through all the nodes, we could choose to start a specific node by using the *findChild* method e.g.:

This should give you some idea of the power of the BeanShell in ZAP. But to fully exploit it, you'll need to familiarize yourself with the internal API and the BeanShell's features. The BeanShell has been setup so as to allow full access to *all* internal objects and methods – even private ones.

Simple HTTP Request

In the next example, we create and send an HTTP request directly in the interactive console:

To fully utilize the power of the BeanShell, you should familiarize yourself with ZAP's internals. The *sender* object is the same instance as is used by the *Manual Request Editor* and will therefore automatically use proxy settings set in the ZAP configuration.

TODO: POST example

Tips

Use the *unset(String)* command to unset any declared variables, methods or objects. This is useful if you want to replace a method declaration in the current namespace. Note that the command takes a String argument, not an Object, so to unset the *t* object we used above, it should be: *unset("t");* and not *unset(t);*

Original document by: Stephen de Vries