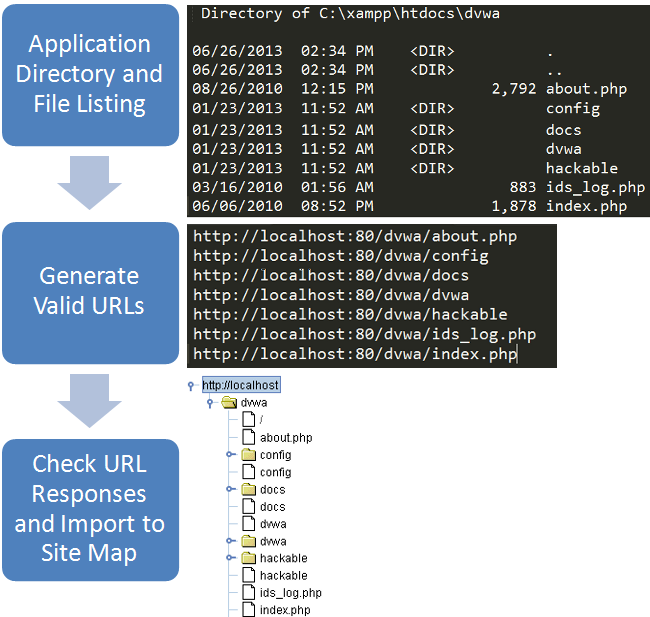
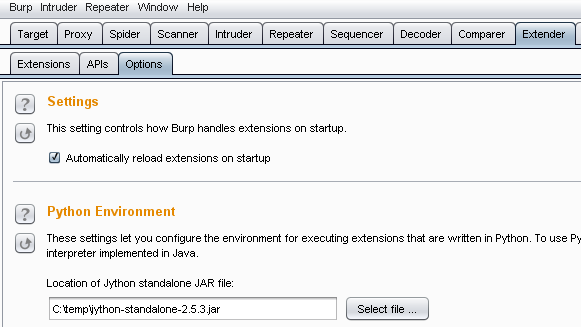
While conducting application penetration tests it’s sometimes necessary to request specific information from the application owner or client. As a pen tester it can be extremely beneficial to perform a test with a full directory and file listing of the application, which sometimes can be difficult to acquire.

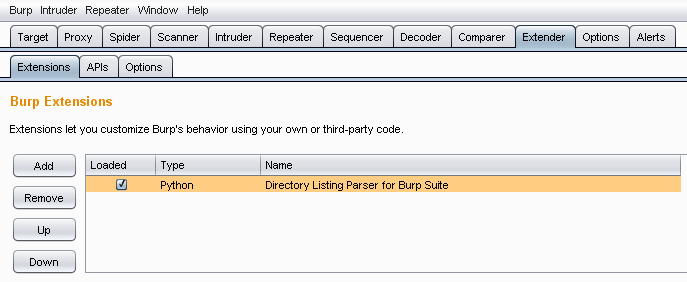
So let’s assume all clients are perfect and provide a full directory and file listing of their application (made you laugh!) but what do we do with it? My process usually involves manually looking over everything trying to find keywords which jump out… I just might like to take a look at adminpassword.txt. Depending on the size of the application I may attempt to reach every file but usually this is not an efficient use of time. I wanted to create a quick and easy process for dealing with directory and file listings so I created a [Burp Suite extension](http://portswigger.net/burp/extender/) which will do a lot of the work for me.

My Burp extension contains two main features. The first feature is the ability to parse the listing file and generate a list of valid URLs to request each resource. The second feature is generating a request for each URL and importing the valid request/response pairs into [Burp’s Target Site Map](http://portswigger.net/burp/help/target_sitemap.html). Why is having a full site map helpful? We now have the ability to see the entire structure of the application, search within all valid responses, conduct manual testing or an active scan on ALL accessible resources, and much more. The process flow looks like this:

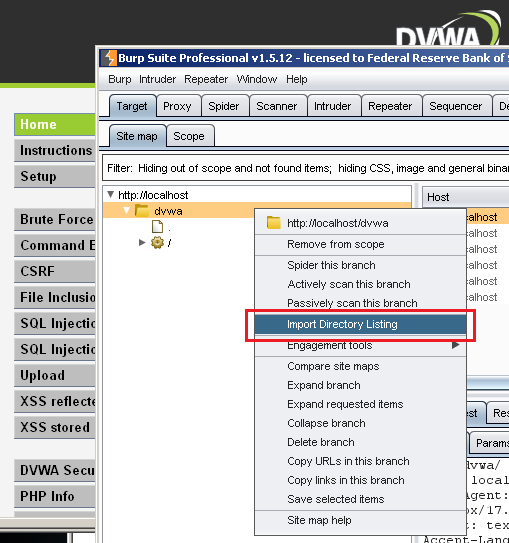


The Burp extension is written in python so a standalone jython jar will be needed to run it.



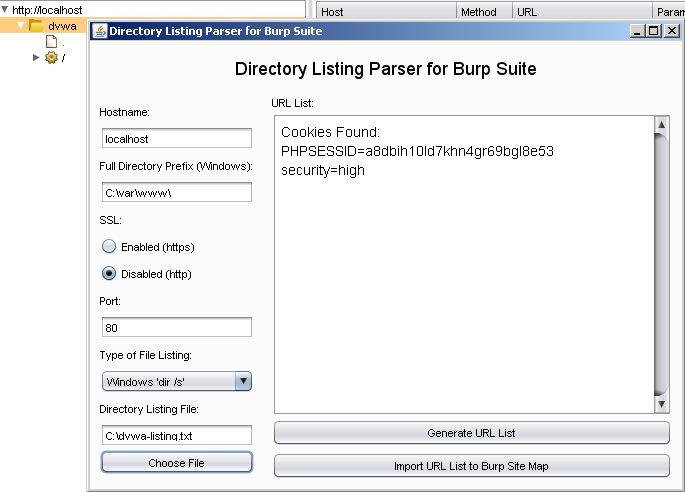


After loading the extension you will have an option in the context menu to “Import Directory Listing”:



A GUI will appear for the extension. Fields such as hostname, SSL, and port will automatically populate depending on the request or response the menu option was originally invoked from. Cookies will also be displayed and used in any requests the extension makes. This feature makes it easy to compare site maps of two application user roles (based on varying session information such as cookies) to determine if each role has the correct access.

In this example I have selected the “Import Directory Listing” menu option on the [DVWA](http://www.dvwa.co.uk/) web application which is running on my local machine. Fill out all options on the left side of the GUI including hostname, full directory path (windows only) which specifies where the root of the web application sits, SSL, port, file listing type, and path to listing file.



**Directory and File Listing – File Format and Parsing**

On a Windows XP machine, I used the ‘dir /s’ command in cmd.exe which displays all files from the current directory and all subdirectories. If the application is sitting on a Windows platform this is a very common command used for directory and file listings. The full directory and file listing for the DVWA web application (selected in the above image as C:\dvwa-listing.txt) looks like this:

Directory of C:\xampp\htdocs\dvwa

06/26/2013 02:34 PM <DIR> .

06/26/2013 02:34 PM <DIR> ..

08/26/2010 12:15 PM 2,792 about.php

06/06/2010 08:55 PM 5,066 CHANGELOG.txt

01/23/2013 11:52 AM <DIR> config

03/16/2010 01:56 AM 33,107 COPYING.txt

01/23/2013 11:52 AM <DIR> docs

01/23/2013 11:52 AM <DIR> dvwa

The parser is also capable of parsing directory listing files from various linux commands such as ‘ls -lR’ and ‘ls –R’. Examples of the format are as follows:

ls –lR:

.:

total 124

-rw-rw-r-- 1 user user 2792 Aug 26 2010 about.php

-rw-rw-r-- 1 user user 5066 Jun 6 2010 CHANGELOG.txt

drwxrwxr-x 2 user user 4096 Jul 1 16:52 config

-rw-rw-r-- 1 user user 33107 Mar 16 2010 COPYING.txt

drwxrwxr-x 2 user user 4096 Jul 1 16:52 docs

drwxrwxr-x 6 user user 4096 Jul 1 16:52 dvwa

ls –R:

.:

about.php

CHANGELOG.txt

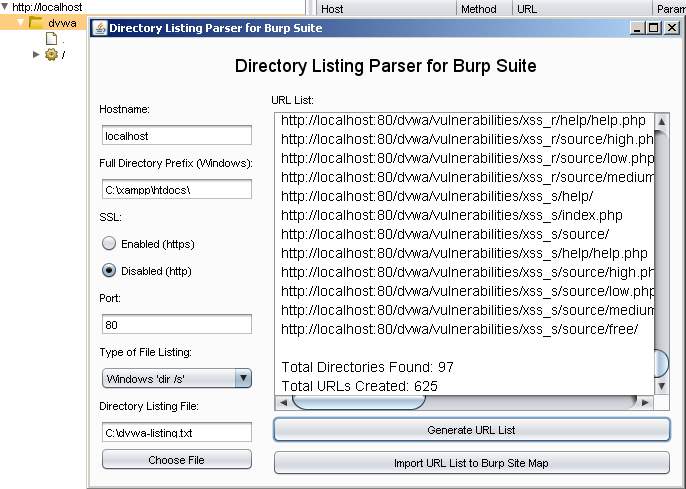
config

COPYING.txt

docs

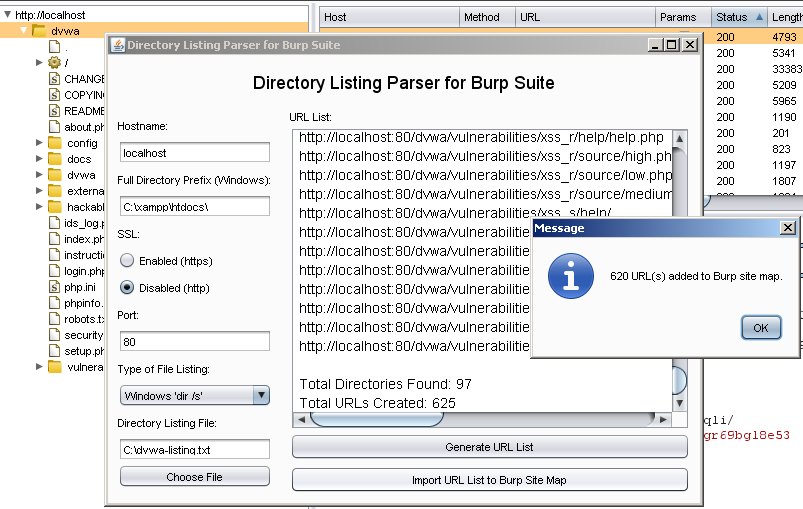
dvwa

Now that we have a directory listing for our application, let’s parse out all of the directories and files and create valid URLs. Click the “Generate URL List” button to start the parsing. Depending on the size of the application this could take a little while. ***Tip:*** If the generated URLs don’t look correct you can modify the fields and regenerate the list or copy the list and use your own text editor to make changes. (Please e-mail [SmeegeSec@gmail.com](mailto:SmeegeSec@gmail.com) with any parsing issues or suggestions)



A text area is populated with the URLs and the total count of directories and files processed.

At this point we have a few options. We can take our list of URLs and use them in [Burp’s Intruder](http://www.portswigger.net/burp/intruder.html). It would be very easy to remove the protocol, hostname, and port from each URL in the list within a text editor and use just the path of each resource in Intruder. We could then look to see which resources we are able to reach by analyzing status codes and content length. A second option is built into the extension via the “Import URL List to Burp Site Map” button. This button **makes a request to each URL** in the list (with any cookie information, if it was found) and if a valid response is returned, will add the request and response to Burp’s Site Map. This functionality was one of the main features I wanted to implement.



Done! A message dialog tells the user how many URLs were valid and imported into the site map. In the above image you can see a full site map and proxy history. With a full site map we are now ahead of the game. If you have multiple testers testing an application you can save the state in Burp and distribute it to save time, almost completely bypassing the discovery phase.

***Note:*** So far the parsing does not consider virtual directories or different URL mappings from different web frameworks. Future updates may include parsing of mapping files such as ASP.NET’s web.config and Java’s web.xml.

***Tip:*** Running a plugin multiple times or multiple plugins at a time may require increased PermGen, an example to increase the max when launching Burp would be:

java -XX:MaxPermSize=1G -jar burp.jar

Also, please provide feedback if you use this extension. With many different output formats for directory and file listings it can be difficult to write a dynamic parser which always works. If you have a listing file which is not being properly parsed please contact me. Thanks!