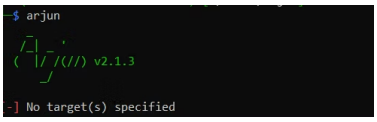
Hacker tools: Arjun – The parameter discovery tool

BY INTIGRITI · MAY 17, 2021 · LAST UPDATED ON MARCH 6, 2025

Time is money, and certainly when it comes to bug bounty! Good tools can help you find bugs before others do – but only if you know how to properly use them.
Today, we are reviewing a parameter discovery tool called Arjun.

Arjun is a command-line tool specifically designed to look for hidden HTTP parameters. Today's web applications have lots of parameters to make an application dynamic. Arjun will try to discover those parameters and give you a new set of endpoints to test on.

By default, Arjun makes use of a default wordlist but this can be modified by the user. It is a multi-threaded application, can handle rate limiting, allows input of custom headers, and most importantly, supports GET, POST, XML, and JSON methods.

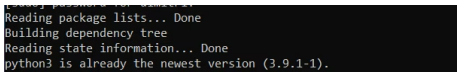


The installation

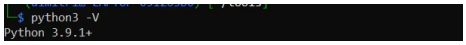
Installing python3:

Arjun needs **python 3.4** or higher to work properly. Check if you have the correct python version installed with *python3 -V.* If this version is not installed you can install it with the following command.

```
apt-get install python3
```




```
python -V
```



Installing Arjun:

Arjun is being actively developed by **s0md3v** at his Git https://github.com/s0md3v/Arjun. We will download and install the latest version from the git and go from there.

```
git clone https://github.com/s0md3v/Arjun
```

```
┌──$ git clone https://github.com/s0md3v/Arjun
Cloning into 'Arjun'...
remote: Enumerating objects: 556, done.
remote: Counting objects: 100% (120/120), done.
remote: Compressing objects: 100% (118/118), done.
remote: Total 556 (delta 73), reused 2 (delta 0), pack-reused 436
Receiving objects: 100% (556/556), 298.20 KiB | 1.06 MiB/s, done.
Resolving deltas: 100% (282/282), done.
```

Change the directory to Arjun and run the **setup.py** with the command below. If you encounter the error: *"ModuleNotFoundError: No module named 'setuptools'"* you first need to install python3-setuptools (*apt-get install python3-setuptools*).

```
python3 setup.py install
```

```
Moving dicttoxml-1.7.4-py3.9.egg to /usr/local/lib/python3.9/dist-packages
Adding dicttoxml 1.7.4 to easy-install.pth file

Installed /usr/local/lib/python3.9/dist-packages/dicttoxml-1.7.4-py3.9.egg
Searching for requests==2.25.1
Best match: requests 2.25.1
Adding requests 2.25.1 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Finished processing dependencies for arjun==2.1.3
```

Now that Arjun is installed we can use it by just typing "arjun". It should be located at **/usr/local/bin/arjun**. To check all the options we will use the (**-h**) flag.

```
┌──$ arjun -h
usage: arjun [-h] [-u URL] [-o JSON_FILE] [-oT TEXT_FILE] [-oB BURP_PORT] [-d DELAY] [-t THREADS] [-w WORDLIST]
             [-m METHOD] [-i [IMPORT_FILE]] [-T TIMEOUT] [-c CHUNKS] [-q] [--headers [HEADERS]] [--passive PASSIVE]
             [--stable] [--include INCLUDE]

optional arguments:
  -h, --help            show this help message and exit
  -u URL                target url
  -o JSON_FILE, -oJ JSON_FILE
                        path for json output file
  -oT TEXT_FILE         path for text output file
  -oB BURP_PORT         port for burp suite proxy
  -d DELAY              delay between requests
  -t THREADS            number of threads
  -w WORDLIST           wordlist path
  -m METHOD             request method: GET/POST/XML/JSON
  -i [IMPORT_FILE]      import targets from file
  -T TIMEOUT            http request timeout
  -c CHUNKS             chunk size/number of parameters to be sent at once
  -q                    quiet mode, no output
  --headers [HEADERS]   add headers
  --passive PASSIVE     collect parameter names from passive sources
  --stable              prefer stability over speed
  --include INCLUDE     include this data in every request
```

The Basics

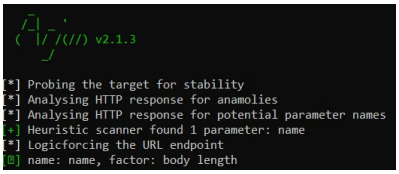
To understand Arjun, we need to know what HTTP parameters are. HTTP parameters are query strings that are part of a URL that accepts user input. For example:

```
http://server.com/page?id=1
```

When the server receives the request, it will process the query and return the page with "id" 1. In this example the "id" is the HTTP parameter. It's also possible there are multiple parameters, or even hidden parameters that are not known by the user. Here is were Arjun comes into play. The tool will try to discover unknown parameters by the use of a wordlist.

Lets start with the very basic command and providing a single URL with the (**-u**) flag.

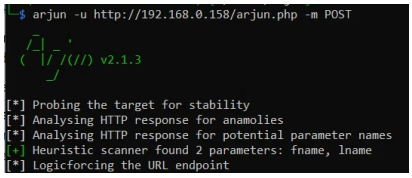
```
arjun -u http://server.com/page/
```



Arjun will analyze the page and will search for potential hidden HTTP parameters.

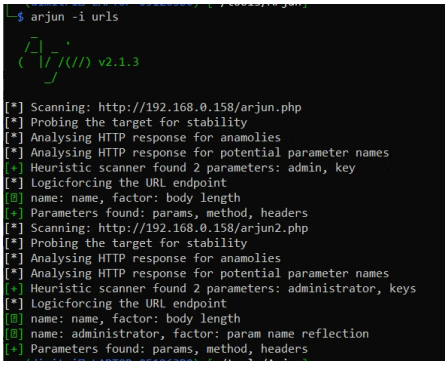
You can also specify the method by providing it to the Arjun with the (**-m**) parameter. Possible methods are: GET,POST,XML,JSON. By default it will use the GET method.

```
arjun -u http://server.com/page/ -m POST
```



More powerful is using a list of URLs that we feed Arjun with the (**-i**) flag.

```
arjun -i urls
```



Like all other tools, the power of finding things that are not found before is using your own custom lists. You can provide your own parameter list with the (**-w**) flag.

```
arjun -u http://server.com/page/ -w parameters.txt
```

Other interesting flags

Threading:

By default arjun will handle 2 threads, but we can increase this with the (**-t**) flag.

```
arjun -u http://server.com/page/ -t 20
```

Delaying between requests:

Some bounty programs require you to set a limit on your requests. With the (**-d**) flag we can set a delay on our requests. (**this is in seconds**)

```
arjun -u http://server.com/page/ -d 2
```

Saving to files:

You can save your results in 2 ways. One is in JSON format (**-o or -oJ**) and the other way is in text format (**-oT**)

```
Arjun -i urls -oJ results.json
```

Including data:

Arjun can detect parameters in a specified location when using JSON or XML method parameters by default by adding **$arjun$.** If you want to send specific data with every request you can do this with the (**–include**) flag.

```
Arjun -u http://server.com/page/ --include '{"api_key":"xxxxx"}'
OR
Arjun -u http://server.com -m XML --include='<?xml><root>$arjun$</root>'
```

Adding headers:

When we want to add specific headers, or test on a authenticated endpoint, we can make use of the (**–headers**) flag. If we don't provide a string after the headers flag, the default text editor will open where we can past our headers.

```
Arjun -u http://server.com/page/ --headers 'Cookie: PHPSESSID=xxxx'
```

Conclusion

Arjun is a tool that can expose hidden parameters, what will lead to a greater attack surface. You can easily chain this tool in your automation by providing it a list of URLs and send the output to a json file to be used by another tool. This was a short article describing the most useful flags of Arjun. I hope you enjoyed it and see you next time.