Hacker tools: BBRF – organizing your recon

BY ANNA HAMMOND · JUNE 22, 2021 · LAST UPDATED ON MARCH 6, 2025

Nice weather, lots of new programs on Intigriti, and another tool to discover. This week we will look at a tool created by one of Intigriti's top researchers. Like Honoki, you probably faced the overwhelming information coming in when doing recon. BBRF will help to organize your findings in a centralized way.

BBRF or Bug Bounty Reconnaissance Framework is a tool to organize your recon information in a centralized way. It exists of two parts, the server part, and the client part. The server part is a database that will hold all the information in a CouchDB filesystem, and the client part is a Python script interacting with this server. The power of this setup is that you can have multiple clients connecting to your central server. This will result in easy collaboration or hunting with multiple devices. I will show you how to set them up.

BBRF is created and recently shared with the community by Honoki. You can find his original blog post on https://honoki.net/2020/10/08/introducing-bbrf-yet-another-bug-bounty-reconnaissance-framework/

The installation

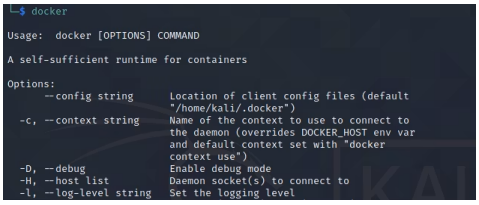
The Server Part:

As discussed above, BBRF consists of 2 parts. Here we will set up the server part with a docker image. Find a VPS or a local machine you want to install the BBRF server on. In this article, I will install it on a Kali VM. Be aware this Image will open up port 443.

Install docker:

If you didn't do this before, we first need to install Docker to run and configure our container.

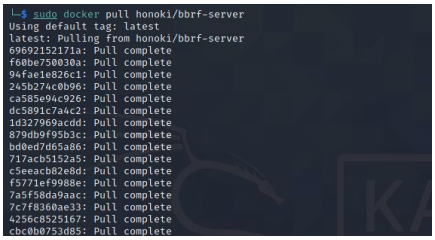
```
sudo apt-get install docker.io
docker
```



Getting BBRF:

Get the docker image from https://hub.docker.com/r/honoki/bbrf-server

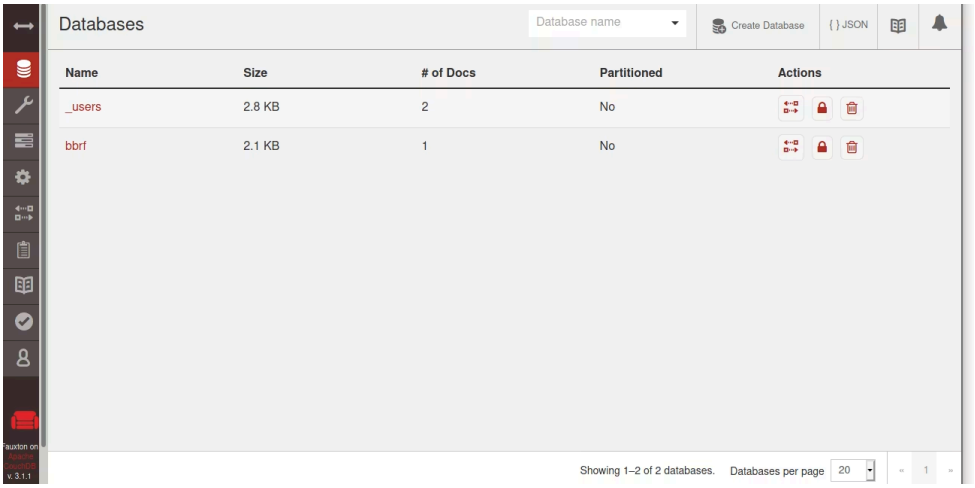
```
docker pull honoki/bbrf-server
```



Now that we have the latest docker image, we can configure and start the server. Fill in your desired information on <ADMIN-USER>, <PASSWORD> and <BBRF-PASSWORD>

```
sudo docker run -p 443:6984 -e COUCHDB_USER=<ADMIN-USER -e COUCHDB_PASSWORD=<PASSWORD> -e BBRF_PASSWORD=<BBRF-PASSWORD> honoki/bbrf-server
```

To verify if everything is working go to https://127.0.0.1/_utils/#database/bbrf/_all_docs. If you want a valid certificate you can replace the */etc/couchdb/cert* with your own.



The Client Part:

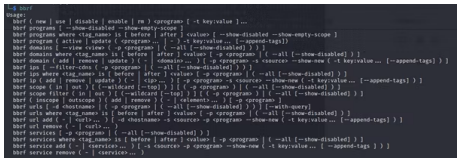
The server is online and running. Now we can start installing the client, and with this client, we will be able to add and retrieve data from our server. BBRF uses python3 to run, so this needs to be installed on your system.

The client can be installed with *pip* like described on the Github page https://github.com/honoki/bbrf-client/

```
sudo pip install bbrf
```

Then create the config file at ~/.bbrf/config.json and fill in the server settings. If you are using a self-signed certificate you need to set the flag "*ignore_ssl_errors*" to true.

```
{
   "username": "admin",
   "password": "<Server Password>",
   "couchdb": "https://127.0.0.1/bbrf",
   "slack_token": "<a slack token to receive notifications>",
   "discord_webhook": "<your discord webhook if you want one>",
   "ignore_ssl_errors": true
}
```



The Basics

Both the server and the client are running now and we can start using the framework. I will go over a couple of things to get you going. All information can also be found on Honoki's Github page https://github.com/honoki/bbrf-client.

First we need to create a program where we will store all the information for this program. We can also enable, disable or remove programs.

```
bbrf new Intigriti      # create new program
bbrf disable Intgriti      # disable program
bbrf enable Intigriti      # enable program
bbrf rm Intigriti     # remove program
```

We can list programs we created with 2 commands.

```
bbrf programs –show-disabled
bbrf programs –show-empty-scope
```

Now we can start using this program and add some data to it.

```
bbrf use Intigriti       # select a program
bbrf program active     # see what program is active
```

Every program has a scope. It is important to know what's allowed and what not. BBRF can run commands directly on programs by providing the (-p) flag. This way we don't need to use the "*use*" command first.

```
bbrf -p intigriti inscope add 'www.intigriti.com' 'www.intigriti.be'
bbrf -p intigriti outscope add 'blog.intigriti.com'
```

To view the scope you just created you can use one of the below commands. There are some filters we can use to list more specific. Use the (-p) flag to filter for a specific program.

```
bbrf scope in          # in-scope of current program
bbrf scope out         # out-scope of current program
bbrf scope in –wildcard        # list wildcard domains
bbrf scope in –all –show-disabled # all domains on every program
```

Adding domains:

We will now add domains manually, but this can also be done by piping to BBRF from another program.

```
bbrf domain add blah.intigriti.com one.intigrti.com
cat domains | bbrf -p Intigriti domains add –

bbrf domain remove <domain>
```

BBRF will check already added domains to prevent duplicates, and it will check the in and out of scope domains before adding them. To list the domains of programs we can use one of the following commands

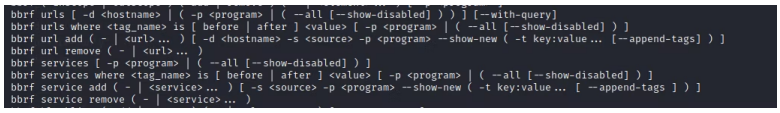
```
bbrf domains           # show domains of current program
bbrf -p Intigriti domains   # show domains of specific program
bbrf domains –all -show-disabled # show all domains
```

Adding IP's:

The same can be done for storing IP addresses.

```
bbrf ips         # list IP's
bbrf ip add 10.10.10.1     # add single IP
bbrf ip remove <IP>     # remove IP

# piping to BBRF
cat ips | bbrf -p Intigriti ip add - # add IP's from file
```

BBRF can also add URLs and services in the same way as adding domains or IP's. More on this in the advanced section.



Advanced features

Now that we have covered the basics and we can add all of our information in a centralized place, we can look at some more advanced features.

When we have everything set up and BBRF is integrated into your automation, you probably want to get notified if there are new assets discovered. BBRF allows integration with slack to send notifications. This has to be set up in the config file. To run a listener and send notifications you can run the below command.

```
bbrf listen &
```

The listener will also check for custom scripts on the following paths. With this you can build custom automation when certain actions occur.

- ~/.bbrf/hooks/ip/new/,
- ~/.bbrf/hooks/ip/update/,
- ~/.bbrf/hooks/domain/new/,
- ~/.bbrf/hooks/domain/update/,
- ~/.bbrf/hooks/url/new/,
- ~/.bbrf/hooks/url/update/,
- ~/.bbrf/hooks/service/new/,
- ~/.bbrf/hooks/service/update/,

A nice example from the Github page on how you can pipe results with BBRF is the command below. It will feed wildcard domains to subfinder and add new domains to the database.

```
bbrf scope in --wildcard --top | subfinder | bbrf domain add - --show-new
```

Domains:

There is also an option to add IPs to a domain. These IPs won't be added automatically to the IP list. If you want to add them to the IP list, you need to do that programmatically.

```
# following formats are accepted
<domain>:<ip>
<domain>:<ip>,<ip>,...
```

IP's:

The same can be done for IPs, the domains are not automatically added to the domains list. The following formats are accepted.

```
<ip>:<domain>
<ip>:<domain>,<domain>,...
```

To check the info you just added, we can query for the raw server data. This will result in a JSON output we can use in our program.

```
bbrf show te.intigriti.com
```

URL's:

BBRF can also store URLs you discovered with some extra info. There are 2 formats supported at the moment.

```
bbrf url add '<URL>'
bbrf url add '<URL:PORT>'
bbrf add '<URL> <STATUSCODE> <CONTENT LENGTH>'
```

The URL can also be relative, but then you need to provide the (-d) flag (domain).

```
bbrf urls        # list URLs
bbrf urls -d www.intigriti.com # retur urls of specific host
bbrf urls –all         # list all URLs from all programs
bbrf urls –with-query       # list urls with query string
```

Services:

One last type of data we can add is services data. To store services we need an IP and a service. This can't be a domain for example. The service can be a port or service name. The best is to combine this with tags so you can filter on them. Here some examples.

```
bbrf service add <IP>:80 -t host:localhost
bbrf service add <IP>:http -t protocol:tcp
```

Tags and searching:

We can set tags to our data with the (-t) flag. These tags are in the *key:value* format and can be anything you want. Using these tags makes it easier to search for specific data.

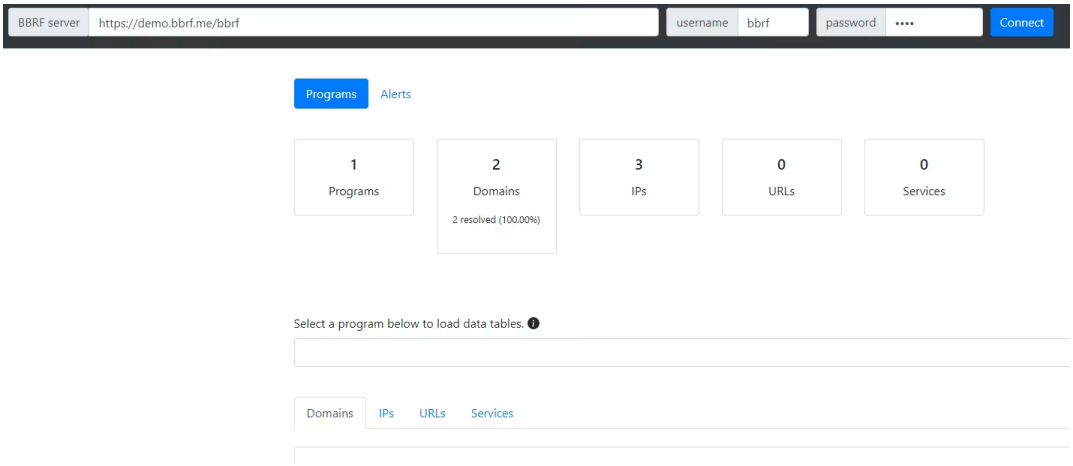
```
# add tag platform:Intigriti
bbrf domain add 'test.com' -t platform:Intigriti

#searching (this can also be done for scope, IP's, URLs and services)
bbrf domains where <KEY> is <VALUE>
bbrf domains where platform is Intigriti
```

Agents:

There is also the possibility to run separate programs called agents. These can be deployed on different machines, and execute various tasks. I won't discuss this further as it would make the article too long but you can find out more on https://github.com/honoki/bbrf-agents

Web GUI:

There is a web-based GUI available to view your gathered data. Surf to https://bbrf.me/ and fill in your server information or set up your own: https://github.com/honoki/bbrf-dashboard.



Conclusion

BBRF is a wonderful program for organizing and centralizing all your information. It can easily be integrated with your automation process and add the new discovered results to its database. Definitely worth checking out. Keep safe everybody and hope to see you on our next article.

REQUEST A DEMO

intigriti.com/demo

VISIT THE WEBSITE

intigriti.com

GET IN TOUCH

hello@intigriti.com