



Hacker tools: Nuclei, a YAML based vulnerability scanner

BY ANNA HAMMOND · MAY 10, 2021 · LAST UPDATED ON JUNE 13, 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.

We will be reviewing some of our favourite open-source tools and providing you with some tips and tricks on how to use them. Today we will review Nuclei, the community-powered vulnerability scanner.

Like our previous review of [FFuF](#), Nuclei is also written in Go.

More and more security tools migrate to Golang for fast processing.

Nuclei is a fast open-source vulnerability scanner that is configurable with templates. This makes it possible to look for one type of vulnerability across a large number of hosts.

Nuclei can scan a variety of protocols like TCP, HTTP, DNS ... to find specific vulnerabilities. The templates are YAML-based for easy and fast configuration.

In general, it is a template-based scanner that you can customize to your needs. The tool is being maintained by the project-discovery team <https://nuclei.projectdiscovery.io>

```

$ ./nuclei -l targetlist -t templates/network/unauth-ftp.yaml

nuclei v2.3.5
projectdiscovery.io

[INF] Loading templates...
[INF] [unauth-ftp] FTP Anonymous Login (@Celesian (@C3l3s14n )) [medium]
[INF] Loading workflows...
[INF] Using 1 rules (1 templates, 0 workflows)
[INF] No results found. Better luck next time!

```

The installation

Installing GoLang:

Before we can start using Nuclei, we need to install Go. Like our previous article, this can easily be done with a packet manager. We will use “apt” to install Golang. If you want to install it from the website, check out <https://golang.org>

```
sudo apt install golang
```

```

$ sudo apt install golang
Reading package lists... Done
Building dependency tree
Reading state information... Done
golang is already the newest version (2:1.15-1).
0 upgraded, 0 newly installed, 0 to remove and 131 not upgraded.

```

Installing Nuclei:

For reference, more information and documentation can be found on the Github of project discovery or on their website. <https://nuclei.projectdiscovery.io> and <https://github.com/projectdiscovery/nuclei>.

We will use Go to install the tool, but feel free to build it from source.

```
GO111MODULE=on go get -v github.com/projectdiscovery/nuclei/v2/cmd/nuclei
```

To verify your installation, go to your Go directory `$HOME/go/bin/` and execute `./nuclei -v`

If you want it to be available on every directory, you need to add `go/bin` to your `PATH` variable.

```
$ ./nuclei -v
nuclei
v2.3.5
projectdiscovery.io
[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[FTL] Program exiting: no template/templates provided
```

Downloading some templates:

As we mentioned before, Nuclei runs on templates. By default no templates are available, and we need to download them. All the templates from the community can be found on the project-discovery Git. (<https://github.com/projectdiscovery/nuclei-templates>).

These templates are made by the community and are a great starting point.

By default, the templates will be stored in `$HOME/nuclei-templates`

```
./nuclei -update-templates
```

```
Nuclei Templates v8.2.3 Changelog
-----+-----+-----+
| TOTAL | ADDED | REMOVED |
|-----+-----+-----+
| 1038 | 1038 | 0 |
+-----+-----+
[INF] Successfully downloaded nuclei templates (v8.2.3). Enjoy!
```

The Basics

We are ready to use Nuclei and its community created templates. Let's explore the basics. To view all options you need to run nuclei with the `(-h)` flag. We are going to explain a couple to get you started.

```
-$ ./nuclei -h
Nuclei is a fast tool for configurable targeted scanning
based on templates offering massive extensibility and ease of use.

Usage:
  ./nuclei [flags]

Flags:
  -H, -header value          Custom Header.
  -biid, -burp-collaborator-biid string  Burp Collaborator BIID
  -bs, -bulk-size int        Maximum Number of hosts analyzed in parallel per template (default 25)
  -c, -concurrency int      Maximum Number of templates executed in parallel (default 10)
  -config string            Nuclei configuration file
  -de, -disk-export string  Directory on disk to export reports in markdown to
  -debug                   Debugging request and responses
  -debug-req              Debugging request
  -debug-resp            Debugging response
  -et, -exclude value     Templates to exclude, supports single and multiple templates using directory.
  -etags, -exclude-tags value  Exclude templates with the provided tags
  -headless              Enable headless browser based templates support
  -impact, -severity value  Templates to run based on severity, supports single and multiple severity.
  -irr, -include-rr      Write requests/responses for matches in JSON output
  -json                  Write json output to files
  -l, -list string       List of URLs to run templates on
  -metrics               Expose nuclei metrics on a port
  -metrics-port int     Port to expose nuclei metrics on (default 9092)
  -nc, -no-color         Disable colors in output
  -nt, -new-templates   Only run newly added templates
  -nm, -no-meta         Don't display metadata for the matches
  -o, -output string    File to write output to (optional)
  -page-timeout int     Seconds to wait for each page in headless (default 20)
  -passive              Enable Passive HTTP response processing mode
  -project              Use a project folder to avoid sending same request multiple times
  -project-path string  Use a user defined project folder, temporary folder is used if not specified but enabled
```

The very basic is providing a list of targets (-l) and a template (-t) that needs to be checked on these targets.

```
./nuclei -l <target-list> -t <template-path>
```

Many Bug-Bounty programs requires you to identify the HTTP traffic you make, this can be achieved by setting custom header using config file at \$HOME/.config/nuclei/config.yaml or with the (-H) flag.

```
./nuclei -H 'Your-Custom-Header' -l <target-list> -t <template-path>
```

Using Templates:

Interesting to know is that Nuclei support standard input (STDIN) for its target list. This makes it easier in chaining multiple tools.

```
cat <target-list> | ./nuclei -t <template-path>
```

To save results to a file you can make use of the (-o) flag, as in most command line tools.

```
cat <target-list> | ./nuclei -t <template-path> -o results.txt
```

If you like to run multiple templates against the target list, you can make use of providing a directory instead of a file. Nuclei will process all .yaml files in the directory. If you want to select templates from multiple directories, you can add (-t) flags for each template or directory.

```
./nuclei -l <target-list> -t templates/http/ -t templated/ftp/ -o results.txt
```

You can control the number of hosts and number of templates that are being processed in parallel by using the (-bs max nr hosts default 25) and (-c max nr templates default 10) flags.

```
./nuclei -l <target-list> -t templates/http/ -bs 30 -c 50 -o results.txt
```

Working with TAGs:

```
-tags value          Tags to execute templates for
```

Tags are a collection of templates that can be used for template execution with or without the need for the (-t) flag. If the (-t) flag is used with tags, the tags will be applied on the particular template directory, otherwise, it will run all the templates with matched tags from the default template download location `$HOME/nuclei-templates/`.

Some common tags are:

```
cve, rce, lfi, xss, network, logs, config, ssrf
```

If you want to run a tag on a specific template directory, you can use the (-t) flag.

```
./nuclei -tags rce -t <my-templates> -l <target-list>
```

And as a last example, running multiple tags on your target list.

```
./nuclei -tags rce,cve,config -t <my-templates> -l <target-list>
```

Creating Templates and Workflows

Now the more interesting part. Custom templates and creating your own is the power of nuclei. When you want to find valid bugs with nuclei you need to create your own templates and workflows. A full reference at: <https://nuclei.projectdiscovery.io/templating-guide>

```
id: dowa-default-login
info:
  name: DOWA Default Login
  author: pdteam
  severity: critical

requests:
  - raw:
    - |
      GET /login.php HTTP/1.1
      Host: {{Hostname}}
      Accept-Language: en-gb,en-us;q=0.9,en;q=0.8
      Connection: close

    - |
      POST /login.php HTTP/1.1
      Host: {{Hostname}}
      Content-Type: application/x-www-form-urlencoded
      Cookie: PHPSESSID={{session}}; security=low
      Connection: close

      username=admin&password=password&login=Login&user_token={{token}}
```

Creating your custom template:

The templates are written in YAML for easy to configure and read purposes. This file is broken up in sections that we are going to discuss here.

The first in each template is the "id". This is a unique string to identify your template. Keep in mind this cannot contain spaces.

```
id: my-first-template
```

Next, we have the info block. This will provide more information on the template and what it is used for. Previously we discussed tags, which we can define in the info block.

```
id: my-first-template

info:
  name: My First Template
  author: Intigriti
  severity: medium
  description: Searches for bugs.
  tags: config
```

Now we have the base of our template and can fill it up with things it needs to do. The next block depends on what you want to check for. Some blocks are requests, headless, network, file, dns. For our article, we will keep it simple and use the *requests* block.

In the *requests* block, we need to define some parameters like the method we want to use, the path we want to check, and matches that need to be checked. For a full reference of all parameters and options go to: <https://nuclei.projectdiscovery.io/templating-guide>

```
id: my-first-template

info:
  name: My First Template
  author: Intigriti
  severity: medium
  description: Searches for bugs.
  tags: config

requests:
  - method: GET
    path:
      - '{{BaseURL}}/secret/login'
    matchers:
      - type: word
        words:
          - "admin_content"
```

This is a basic custom template we can use in our automation that checks for the */secret/login* path on each target.

Creating a workflow:

Workflows allow you to define an execution sequence for templates. The templates will be run on the defined conditions. To make use of workflows we use the **(-w)** flag.

Workflows can be defined with workflows attribute, following the template or sub-templates to execute.

```
workflows:
```

- template: files/my-first-template.yaml
- template: files/find-xss.yaml

The real power of Nuclei automation comes with the use of conditional workflows. This way we can search if we find technology and then run sub-templates to see if the technology is vulnerable.

```
workflows:
```

- template: technologies/tech-detect.yaml
- ```
 matchers:
```
- name: vbulletin
- ```
    subtemplates:
```
- template: exploits/vbulletin-exp1.yaml
 - template: exploits/vbulletin-exp2.yaml
- name: jboss
- ```
 subtemplates:
```
- template: exploits/jboss-exp1.yaml
  - template: exploits/jboss-exp2.yaml

In the above example we look for technologies, and if there are found, we will run templates on the found technologies.

## Conclusion

That's all for today. We covered the basics of Nuclei to get you started.

Nuclei is a powerful scanner that you can customize to your needs to find your secret bugs on multiple targets. The speed and accuracy when you use custom workflows are amazing. The options are endless, Nuclei will bring great value to your bug-hunting tools.

Happy hunting!

**REQUEST A DEMO**

[intigriti.com/demo](https://intigriti.com/demo)

**VISIT THE WEBSITE**

[intigriti.com](https://intigriti.com)

**GET IN TOUCH**

[hello@intigriti.com](mailto:hello@intigriti.com)