

uNGINXed

A tool that generates reports about misconfigurations in specified NGINX configuration files.

The reports may be in JSON format or PDF.

Installation Guide

The uNGINXed project relies on Poetry to manage Python dependencies. It is highly encouraged to perform all uNGINXed operations within Poetry's environment.

Follow [Poetry installation Guide](#)

Installing uNGINXed Dependencies


```
poetry install
```

unNGINXed Usage

The uNGINXed engine support scanning of NGINX configurations from the command line.

Command Line Report

```
poetry run python -m unginxed <NGINX Configuration Path> -sv
```



A static vulnerability scanner for NGINX Configuration

Signature for CRLF Injection
Improper usage of normalized URI variables \$uri and \$document_uri could allow an attacker to perform cross site scripting.

Line Number	Directive and Argument	Severity	Column Start	Column End
5	return 302 https://example.com\$uri	3	13	47

Reference URL:
<https://www.acunetix.com/vulnerabilities/web/crlf-injection-http-response-splitting-web-server/>

Signature for Missing Root Location
This could potentially leak useful information about the server installation to a remote, unauthenticated attacker.

Line Number	Directive and Argument	Severity	Column Start	Column End
2	http	2	1	5

Reference URL: <https://blog.detectify.com/2020/11/10/common-nginx-misconfigurations/>

.\examples\configs\crlf-injection.conf

Line No.	Configuration File
1	events{
2	http{
3	server {
4	location / {
5	return 302 https://example.com\$uri;
6	}
7	}
8	}

Filepath: .\examples\configs\crlf-injection.conf

You have been NGINXED! 2 directive flagged in your configuration

PDF Report Generation

```
poetry run python -m unginxed <NGINX Configuration Path> -o <output directory>
```

Report Generation With Command Line Report

```
poetry run python -m unginxed <NGINX Configuration Path> -svo <output directory>
```

Development for uNGINXed

Adding signatures

Signatures come in the form of python code.

In the sigs folder, create a new python file which contains a function named `matcher`. The function takes in an `NGINXConfig` object as a parameter, and should return a `Signature` object as a result. Use the `SignatureBuilder` class to build your signatures, as it abstracts the complicated logic away from creating the Signature.

Command line tool

Use the `tools/sigs.py` tool to create a signature python file which contains boilerplate to get you started.

Example usage (from `unginxed` folder):

```
poetry run python unginxed/tools/sigs.py create Alias LFI
```

This creates a file named `alias_lfi.py`, with the following boilerplate code:

```
from ..nginx_config import NginxConfig
from ..signature import Signature, SignatureBuilder

def matcher(config: NginxConfig) -> Signature:
    signature_builder = SignatureBuilder(config.raw).set_name('Alias LFI')
    \
        .set_reference_url('') \
        .set_description('') \
        .set_severity()

    # Your logic here.
    # Flag out directives using signature_builder.add_flagged(directive,
    config)

    return signature_builder.build()
```