



**afrog**

开发者: zan8in

A Security Tool for Bug Bounty, Pentest and Red Teaming.





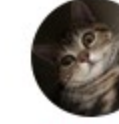


















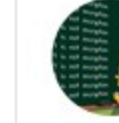




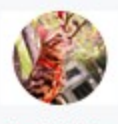




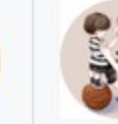
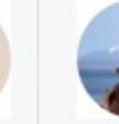








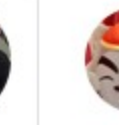
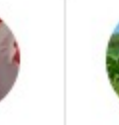







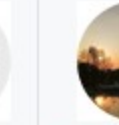
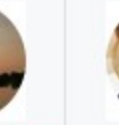


**A Security Tool for Bug Bounty, Pentest and Red Teaming**

Go@v2 v1.20 downloads 46k contributors 10 release v2.9.6 open issues 37 discussions 3 total

[Download](#) • [Wiki](#) • [Helper Function](#)

## PoC Contributors

 不动明王	 雪山	 White-hua	 123456	 ifofor	 Air	 执着	 purple-WL	 throat
 Secx	 冰河	 Sheen	 a16	 A1	 rainbow2972	 wuha0926	 茄子	 lei_sec
 G-H-Z	 wh1te	 清月	 york	 7eleven.eth	 Double...	 ICEY_	 lazy	 Lay0us
 m4sk	 沉默树人	 陈麻子	 leonardo-o1	 江湖人称魏...	 若兮风	 -sudo	 Cuez	 laohuan12138
 exp0l0zzz	 1derian	 CMDB-M	 li1u	 oxsonder	 Zhiliao	 段	 HuiTaiL	 Miracles666
 Observer	 黑熊	 TryA9ain	 fgz00	 Y3y1ng	 二大爷	 Wans	 海边的小米粥	 Wen

## What is afrog

afrog is a high-performance vulnerability scanner that is fast and stable. It supports user-defined PoC and comes with several built-in types, such as CVE, CNVD, default passwords, information disclosure, fingerprint identification, unauthorized access, arbitrary file reading, and command execution. With afrog, network security professionals can quickly validate and remediate vulnerabilities, which helps to enhance their security defense capabilities.

## Features

- ✓ Open source
- ✓ Fast, stable, with low false positives
- ✓ Detailed HTML vulnerability reports
- ✓ Customizable and stably updatable PoCs
- ✓ Active community exchange group

## Installation

### Prerequisites

- Go version 1.19 or higher.

you can install it with:

#### Binary

```
$ https://github.com/zan8in/afrog/releases
```

#### Github

```
$ git clone https://github.com/zan8in/afrog.git
```

```
$ cd afrog
$ go build cmd/afrog/main.go
$ ./afrog -h
```

## Go

```
$ go install -v github.com/zan8in/afrog/v2/cmd/afrog@latest
```

## Running afrog

By default, afrog scans all built-in PoCs, and if it finds any vulnerabilities, it automatically creates an HTML report with the date of the scan as the filename.

```
afrog -t https://example.com
```

### Warning occurs when running afrog

If you see an error message saying:

```
[ERR] ceye reverse service not set: /home/afrog/.config/afrog/afrog-config.yaml
```

it means you need to modify the [configuration file](#).

To execute a custom PoC directory, you can use the following command:

```
afrog -t https://example.com -P mypocs/
```

Use the command `-s keyword` to perform a fuzzy search on all PoCs and scan the search results. Multiple keywords can be used, separated by commas. For example: `-s weblogic,jboss`.

```
afrog -t https://example.com -s weblogic,jboss
```

Use the command `-S keyword` to scan vulnerabilities based on their severity level. Severity levels include: `info`, `low`, `medium`, `high`, and `critical`. For example, to only scan high and critical vulnerabilities, use the command `-S high,critical`.

```
afrog -t https://example.com -S high,critical
```

You can scan multiple URLs at the same time as well.

```
afrog -T urls.txt
```

## Configuration file

The first time you start afrog, it will automatically create a configuration file called `afrog-config.yaml`, which will be saved in the current user directory under `$HOME/.config/afrog/afrog-config.yaml`.

Here is an example config file:

```
reverse:
  ceye:
    api-key: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    domain: "xxxxxx.ceye.io"
  jndi:
    jndi_address: "x.x.x.x"
    ldap_port: "1389"
    api_port: "34567"
```

`reverse` is a reverse connection platform used to verify command execution vulnerabilities that cannot be echoed back. Currently, only ceye can be used for verification.

### Ceye Configuration

To obtain ceye, follow these steps:

- Go to the [ceye.io](#) website and register an account.
- Log in and go to the personal settings page.
- Copy the `domain` and `api-key` and correctly configure them in the `afrog-config.yaml` file.

### JNDI Configuration

The JNDI vulnerability refers to security vulnerabilities that exploit the JNDI (Java Naming and Directory Interface) functionality in Java applications. This type of vulnerability can lead to remote code execution or other security issues.

To obtain JNDI, follow these steps:

- To obtain the source code and compile the JAR file, please visit the official website [github.com/r00tSe7en/JNDIMonitor](#). Alternatively, you can go to the official afrog website [afrog/helper/jndi](#) to download the pre-compiled JAR file
- Upload the `JNDIMonitor-2.0.1-SNAPSHOT.jar` file to the server (such as a VPS server), and execute the following startup command:



```
java -jar ./JNDIMonitor-2.0.1-SNAPSHOT.jar -i 0.0.0.0 -l 1389 -p 3456
```

Below are example methods for writing POCs. [Please click to view.](#)

## Json Output (For developers)

### Json

Optional command: `-json -j`, Save the scan results to a JSON file. The JSON file includes the following contents by default: `target`, `fulltarget`, `id`, and `info`. The info field includes the following sub-fields: `name`, `author`, `severity`, `description`, and `reference`. If you want to save both `request` and `response` contents, please use the `-json-all` command parameter.

```
afrog -t https://example.com -json result.json
afrog -t https://example.com -j result.json
```

### Warning

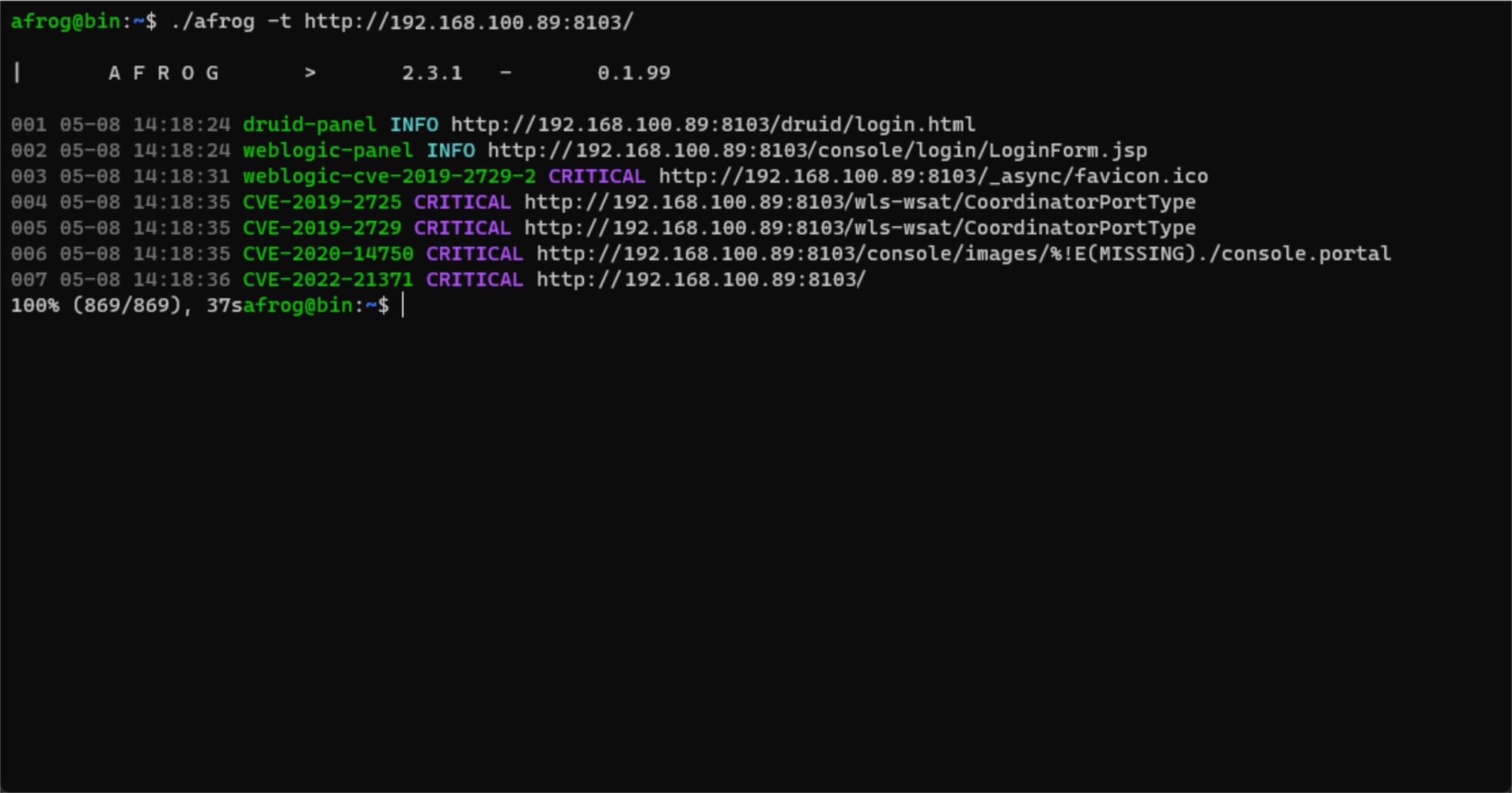
The content of the JSON file is updated in real time. However, there is an important note to keep in mind: before the scan is completed, if developers want to parse the file content, they need to add a `']` symbol to the end of the file by themselves, otherwise it will cause parsing errors. Of course, if you wait for the scan to complete before parsing the file, this issue will not occur.

### JsonAll

Optional command: `-json-all -ja`, The only difference between the `-json-all` and `-json` commands is that `-json-all` writes all vulnerability results, including `request` and `response`, to a JSON file.

```
afrog -t https://example.com -json-all result.json
afrog -t https://example.com -ja result.json
```

## Screenshot



## As Library

### Simple Example

Scan the website `http://example.com`

```
package main

import (
    "fmt"

    "github.com/zan8in/afrog/v2"
)

func main() {
    if err := afrog.NewScanner([]string{"http://example.com"}, afrog.Scanner{}); err != nil {
        fmt.Println(err.Error())
    },
```

```
}  
}
```

More examples:

- [Basic scanner](#)
- [Read URL batch scan from file](#)

## Discussion group

To join the afrog communication group on WeChat, please first add the afrog personal account and mark it as **afrog**. Then, you will be added to the group by the administrator.

