GreedyBear

Matteo Lodi

Sep 06, 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

The project goal is to extract data of the attacks detected by a TPOT or a cluster of them and to generate some feeds that can be used to prevent and detect attacks.

Official announcement here.

1.1 Public feeds

There are public feeds provided by The Honeynet Project in this site: greedybear.honeynet.org. Example

To check all the available feeds, Please refer to our usage guide

Please do not perform too many requests to extract feeds or you will be banned.

If you want to be updated regularly, please download the feeds only once every 10 minutes (this is the time between each internal update).

CHAPTER TWO

INSTALLATION

Start by cloning the project

```
# clone the Greedybear project repository
git clone https://github.com/honeynet/GreedyBear
cd GreedyBear/

# construct environment files from templates
cp .env_template .env
cd docker/
cp env_file_template env_file
cp env_file_postgres_template env_file_postgres
cd ..
cp frontend/public/env_template.js frontend/public/env.js
```

Now you can start by building the image using docker-compose and run the project.

```
# build the image locally
docker-compose build

# start the app
docker-compose up

# now the app is running on http://localhost:80

# shut down the application
docker-compose down
```

Note: To create a superuser run the following:

```
docker exec -ti greedybear_uwsgi python3 manage.py createsuperuser
```

The app administrator can enable/disable the extraction of source IPs for specific honeypots from the Django Admin. This is used for honeypots that are not specifically implemented to extract additional information (so not Log4Pot and Cowrie).

Note that GreedyBear *needs* a running instance of ElasticSearch of a TPoT to function. If you don't have one, you can make the following changes to make GreeyBear spin up it's own ElasticSearch and Kibana instances. (…Care! This option would require enough RAM to run the additional containers. Suggested is >=16GB):

1. In `docker/env_file`, set the variable `ELASTIC_ENDPOINT` to `http://elasticsearch:9200`.
2. Add `:docker/elasticsearch.yml` to the last defined `COMPOSE_FILE` variable or uncomment the `# local development with elasticsearch container` block in `.env` file.

2.1 Environment configuration

In the `env_file`, configure different variables as explained below.

Required variable to set:

- `DEFAULT_FROM_EMAIL`: email address used for automated correspondence from the site manager (example: `noreply@mydomain.com`)
- `DEFAULT_EMAIL`: email address used for correspondence with users (example: `info@mydomain.com`)
- `EMAIL_HOST`: the host to use for sending email with SMTP
- `EMAIL_HOST_USER`: username to use for the SMTP server defined in EMAIL_HOST
- `EMAIL_HOST_PASSWORD`: password to use for the SMTP server defined in EMAIL_HOST. This setting is used in conjunction with EMAIL_HOST_USER when authenticating to the SMTP server.
- `EMAIL_PORT`: port to use for the SMTP server defined in EMAIL_HOST.
- `EMAIL_USE_TLS`: whether to use an explicit TLS (secure) connection when talking to the SMTP server, generally used on port 587.
- `EMAIL_USE_SSL`: whether to use an implicit TLS (secure) connection when talking to the SMTP server, generally used on port 465.

Optional configuration:

- `SLACK_TOKEN`: Slack token of your Slack application that will be used to send/receive notifications
- `DEFAULT_SLACK_CHANNEL`: ID of the Slack channel you want to post the message to

2.1.1 Recaptcha configuration

The Registration Page and the Login Page contain a Recaptcha form from Google. By default, that Recaptcha is not configured and is not shown. If your intention is to publish GreedyBear as a Service you should configure different variables as explained below.

In the `frontend/public/env.js` set the variable:

- `RECAPTCHA_SITEKEY`: Recaptcha Key for your site

In the `docker/env_file` set the variables:

- `RECAPTCHA_SECRET_KEY_GB_LOCAL`: your recaptcha secret key internal deployment
- `RECAPTCHA_SECRET_KEY_GB_PUBLIC`: your recaptcha secret key for public deployment

In that case, you would need to *re-build* the application to have the changes properly reflected.

2.2 Update and Re-build

2.2.1 Rebuilding the project / Creating custom docker build

If you make some code changes and you like to rebuild the project, follow these steps:

1. Be sure that your `.env` file has a `COMPOSE_FILE` variable which mounts the `docker/local.override.yml` compose file.
2. `docker-compose build` to build the new docker image.

3. Start the containers with `docker-compose up`.

2.2.2 Update to the most recent version

To update the project with the most recent available code you have to follow these steps:

```
$ cd <your_greedy_bear_directory> # go into the project directory
$ git pull # pull new repository changes
$ docker pull intelowlproject/greedybear:prod # pull new docker images
$ docker-compose down # stop and destroy the currently running GreedyBear containers
$ docker-compose up # restart the GreedyBear application
```

2.3 Installer for TPot Instance

The file 'installer_on_tpot.sh' allows the automatic installation of Greedybear on an existing TPot instance. You can choose the type of Greedybear you want to install (http, https or local). The installer will either clone Greedybear to '/opt/GreedyBear' or if Greedybear exists on your system you need to input the absolute path to the existing Greedybear folder. It will prompt you for the necessary information/secrets needed.

Example: `sudo ./installer.sh --type=http --folder=/opt/GreedyBear`

CHAPTER THREE

USAGE

3.1 User management

3.1.1 Registration

Since Greedybear v1.1.0 we added a Registration Page that can be used to manage Registration requests when providing GreedyBear as a Service.

After an user registration, an email is sent to the user to verify their email address. If necessary, there are buttons on the login page to resend the verification email and to reset the password.

Once the user has verified their email, they would be manually vetted before being allowed to use the GreedyBear platform. The registration requests would be handled in the Django Admin page by admins. If you have GreedyBear deployed on an AWS instance you can use the SES service.

In a development environment the emails that would be sent are written to the standard output.

3.1.2 Recaptcha configuration

The Registration Page contains a Recaptcha form from Google. By default, that Recaptcha is not configured and is not shown. If your intention is to publish GreedyBear as a Service you should *configure the Recaptcha*.

3.1.3 Amazon SES

If you like, you could use Amazon SES for sending automated emails.

First, you need to configure the environment variable `AWS_SES` to `True` to enable it. Then you have to add some credentials for AWS: if you have GreedyBear deployed on the AWS infrastructure, you can use IAM credentials: to allow that just set `AWS_IAM_ACCESS` to `True`. If that is not the case, you have to set both `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.

Additionally, if you are not using the default AWS region of us-east-1, you need to specify your `AWS_REGION`. You can customize the AWS Region location of you services by changing the environment variable `AWS_REGION`. Default is `eu-central-1`.

3.2 Feeds

GreedyBear is created with the aim to collect the information from the TPOTs and generate some actionable feeds, so that they can be easily accessible and act as valuable information to prevent and detect attacks.

The feeds are reachable through the following URL:

```
https://<greedybear_site>/api/feeds/<feed_type>/<attack_type>/<age>.<format>
```

The available feed_type are:

- `log4j`: attacks detected from the Log4pot.
- `cowrie`: attacks detected from the Cowrie Honeypot.
- `all`: get all types at once
- The following honeypot feeds exist (for extraction of (only) the source IPs):
  - `heralding`
  - `ciscoasa`
  - `honeytrap`
  - `dionaea`
  - `conpot`
  - `adbhoney`
  - `tanner`
  - `citrixhoneypot`
  - `mailoney`
  - `ipphoney`
  - `ddospot`
  - `elasticpot`
  - `dicompot`
  - `redishoneypot`
  - `sentrypeer`
  - `glutton`

The available attack_type are:

- `scanner`: IP addresses captured by the honeypots while performing attacks
- `payload_request`: IP addresses and domains extracted from payloads that would have been executed after a speficic attack would have been successful
- `all`: get all types at once

The available age are:

- `recent`: most recent IOCs seen in the last 3 days
- `persistent`: these IOCs are the ones that were seen regularly by the honeypots. This feeds will start empty once no prior data was collected and will become bigger over time.

The available formats are:

- `txt`: plain text (just one line for each IOC)
- `csv`: CSV-like file (just one line for each IOC)
- `json`: JSON file with additional information regarding the IOCs

Check the Redoc specification or the to get all the details about how to use the available APIs.

3.3 Enrichment

GreedyBear provides an easy-to-query API to get the information available in GB regarding the queried observable (domain or IP address).

```
https://<greedybear_site>/api/enrichment?query=<observable>
```

This "Enrichment" API is protected through authentication. Please reach out Matteo Lodi or another member of The Honeynet Project if you are interested in gain access to this API.

If you would like to leverage this API without the need of writing even a line of code and together with a lot of other awesome tools, consider using IntelOwl.

CHAPTER FOUR

CONTRIBUTE

4.1 General Guidance

Please refer to IntelOwl Documentation for everything missing here.

4.2 Rules

GreedyBear welcomes contributors from anywhere and from any kind of education or skill level. We strive to create a community of developers that is welcoming, friendly and right.

For this reason it is important to follow some easy rules based on a simple but important concept: **Respect**.

- Before starting to work on an issue, you need to get the approval of one of the maintainers. Therefore please ask to be assigned to an issue. If you do not that but you still raise a PR for that issue, your PR can be rejected. This is a form of respect for both the maintainers and the other contributors who could have already started to work on the same problem.
- When you ask to be assigned to an issue, it means that you are ready to work on it. When you get assigned, take the lock and then you disappear, you are not respecting the maintainers and the other contributors who could be able to work on that. So, after having been assigned, you have a week of time to deliver your first *draft* PR. After that time has passed without any notice, you will be unassigned.
- Before asking questions regarding how the project works, please read *through all the documentation* and install the project on your own local machine to try it and understand how it basically works. This is a form of respect to the maintainers.
- Once you started working on an issue and you have some work to share and discuss with us, please raise a draft PR early with incomplete changes. This way you can continue working on the same and we can track your progress and actively review and help. This is a form of respect to you and to the maintainers.
- When creating a PR, please read through the sections that you will find in the PR template and compile it appropriately. If you do not, your PR can be rejected. This is a form of respect to the maintainers.

4.3 Code Style

Keeping to a consistent code style throughout the project makes it easier to contribute and collaborate. We make use of psf/black and isort for code formatting and flake8 for style guides.

4.4 How to start (Setup project and development instance)

To start with the development setup, make sure you go through all the steps in Installation Guide and properly installed it.

Please create a new branch based on the **develop** branch that contains the most recent changes. This is mandatory.

```
git checkout -b myfeature develop
```

Then we strongly suggest to configure pre-commit to force linters on every commits you perform:

```
# create virtualenv to host pre-commit installation
python3 -m venv venv
source venv/bin/activate
# from the project base directory
pip install pre-commit
pre-commit install -c .github/.pre-commit-config.yaml
```

Remember that whenever you make changes, you need to rebuild the docker image to see the reflected changes.

4.4.1 NOTE about documentation:

If you made any changes to an existing model/serializer/view, please run the following command to generate a new version of the API schema and docs:

```
docker exec -it greedybear_uwsgi python manage.py spectacular --file docs/source/schema.
↪yml && make html
```

4.4.2 Frontend

To start the frontend in "develop" mode, you can execute the startup npm script within the folder `frontend`:

```
cd frontend/
# Install
npm i
# Start
DANGEROUSLY_DISABLE_HOST_CHECK=true npm start
# See https://create-react-app.dev/docs/proxying-api-requests-in-development/#invalid-
↪host-header-errors-after-configuring-proxy for why we use that flag in development mode
```

Most of the time you would need to test the changes you made together with the backend. In that case, you would need to run the backend locally too:

```
docker-compose up
```

Certego-UI

The GreedyBear Frontend is tightly linked to the `certego-ui` library. Most of the React components are imported from there. Because of this, it may happen that, during development, you would need to work on that library too. To install the `certego-ui` library, please take a look to npm link and remember to start certego-ui without installing peer dependencies (to avoid conflicts with GreedyBear dependencies):

```
git clone https://github.com/certego/certego-ui.git
# change directory to the folder where you have the cloned the library
cd certego-ui/
# install, without peer deps (to use packages of GreedyBear)
npm i --legacy-peer-deps
# create link to the project (this will globally install this package)
sudo npm link
# compile the library
npm start
```

Then, open another command line tab, create a link in the `frontend` to the `certego-ui` and re-install and re-start the frontend application (see previous section):

```
cd frontend/
npm link @certego/certego-ui
```

This trick will allow you to see reflected every changes you make in the `certego-ui` directly in the running `frontend` application.

Example application

The `certego-ui` application comes with an example project that showcases the components that you can re-use and import to other projects, like GreedyBear:

```
# To have the Example application working correctly, be sure to have installed `certego-
↪ui` *without* the `--legacy-peer-deps` option and having it started in another command␣
↪line
cd certego-ui/
npm i
npm start
# go to another tab
cd certego-ui/example/
npm i
npm start
```

4.5 Create a pull request

4.5.1 Remember!!!

Please create pull requests only for the branch **develop**. That code will be pushed to master only on a new release.

Also remember to pull the most recent changes available in the **develop** branch before submitting your PR. If your PR has merge conflicts caused by this behavior, it won't be accepted.

4.5.2 Tests

Backend

Install testing requirements

You have to install `pre-commit` to have your code adjusted and fixed with the available linters:

```
pip install pre-commit
pre-commit install -c .github/.pre-commit-config.yaml
```

Once done that, you won't have to think about linters anymore.

Run all tests

```
docker exec greedybear_uwsgi python3 manage.py test
```

Frontend

All the frontend tests must be run from the folder `frontend`. The tests can contain log messages, you can suppress then with the environment variable `SUPPRESS_JEST_LOG=True`.

Run all tests

```
npm test
```

Run a specific component tests

```
npm test -- -t <componentPath>
// example
npm test tests/components/auth/Login.test.jsx
```

Run a specific test

```
npm test -- -t '<describeString> <testString>'
// example
npm test -- -t "Login component User login"
```

if you get any errors, fix them. Once you make sure that everything is working fine, please squash all of our commits into a single one and finally create a pull request.

CHAPTER FIVE

GREEDYBEAR API DOCS

`GET /api/apiaccess`

Durin's APIAccessTokenView.

- `GET` -> get token-client pair info
- `POST` -> create and get token-client pair info
- `DELETE` -> delete existing API access token

New in version 1.0.0.

Status Codes

- 200 OK –

`POST /api/apiaccess`

Durin's APIAccessTokenView.

- `GET` -> get token-client pair info
- `POST` -> create and get token-client pair info
- `DELETE` -> delete existing API access token

New in version 1.0.0.

Status Codes

- 200 OK –

`DELETE /api/apiaccess`

Durin's APIAccessTokenView.

- `GET` -> get token-client pair info
- `POST` -> create and get token-client pair info
- `DELETE` -> delete existing API access token

New in version 1.0.0.

Status Codes

- 204 No Content – No response body

`POST /api/auth/login`

Extends `durin.views.LoginView`.

Responds with `Set-cookie` header and empty response data.

Status Codes

- 200 OK – No response body

`POST /api/auth/logout`

Extends `durin.views.LogoutView`.

Responds with `Set-cookie` header and empty response data.

Status Codes

- 200 OK – No response body

`GET /api/authentication`

Status Codes

- 200 OK – No response body

`GET /api/enrichment`

Request if a specific observable (domain or IP address) has been listed by GreedyBear

Status Codes

- 200 OK –

`GET /api/feeds/{feed_type}/{attack_type}/{age}.{format_}`

Extract Structured IOC Feeds from GreedyBear

Parameters

- **age** (*string*) –
- **attack_type** (*string*) –
- **feed_type** (*string*) –
- **format** (*string*) –

Status Codes

- 200 OK – No response body

`GET /api/me/access`

Returns user's access information.

Status Codes

- 200 OK –
- 500 Internal Server Error –

`GET /api/sessions`

Durin's TokenSessionsViewSet.

- Returns list of active sessions of authed user.
- Only `list()` and `delete()` operations.

New in version 1.0.0.

Status Codes

- 200 OK –

`DELETE /api/sessions/{id}`

Durin's TokenSessionsViewSet.

- Returns list of active sessions of authed user.
- Only `list()` and `delete()` operations.

New in version 1.0.0.

Parameters

- **`id`** (*`integer`*) – A unique integer value identifying this auth token.

Status Codes

- 204 No Content – No response body

`GET /api/statistics/{id}/enrichment`

Parameters

- **`id`** (*`string`*) –

Status Codes

- 200 OK – No response body

`GET /api/statistics/{id}/feeds`

Parameters

- **`id`** (*`string`*) –

Status Codes

- 200 OK – No response body

`GET /api/statistics/feeds_types`

Status Codes

- 200 OK – No response body

CHAPTER

SIX

GREEDYBEAR API REDOC

HTTP ROUTING TABLE

/api