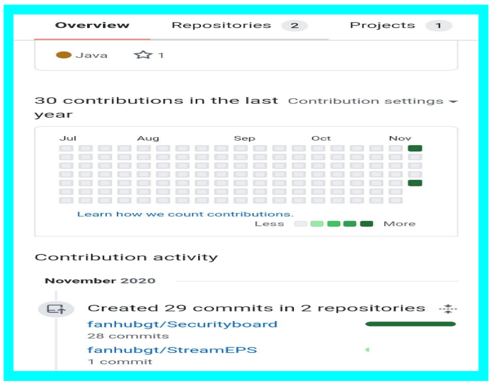
A Security-Board Director Research Project at Github

Frank Appiah

November 19, 2020

A Security-board Director Research Project at Github.



30 contributions in the last year

Contribution activity

November 2020

Created 29 commits in 2 repositories

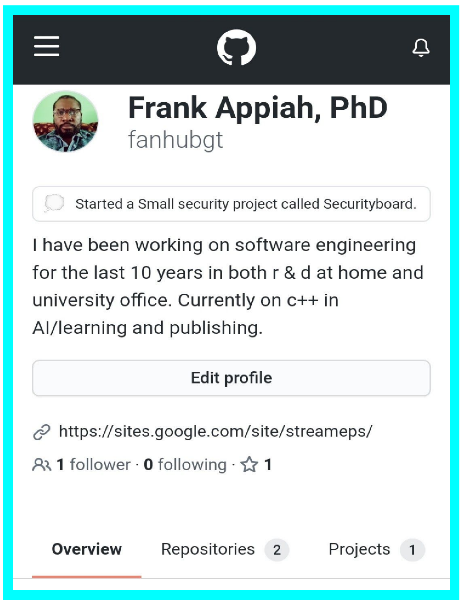
fanhubgt/Securityboard
28 commits

fanhubgt/StreamEPS
1 commit

Prof Frank Appiah AKA FAEng PhD
Affiliate: King's College London, Strand, London, England, UK.
Email:appiahnsiahfrank@gmail.com
12.2020.11.

- **Abstract. A small prolog program for security board director used to make countermeasure check on vulnerability prevention is hosted at Github with name *fanhubgt stroke Securityboard* is the main focus of this article.**

Keywords. Repository, Prolog, logic program, security board, vulnerability prevention.



Frank Appiah, PhD
fanhubgt

Started a Small security project called Securityboard.

I have been working on software engineering for the last 10 years in both r & d at home and university office. Currently on c++ in AI/learning and publishing.

https://sites.google.com/site/streameps/
1 follower · 0 following · 1

1 Introduction.

Github is how people build software. They're supporting a community where more than 50 million* people learn, share, and work together to build software.

Going deeper with Github as a developer is by integrating with their APIs, customizing your GitHub workflow, and building and sharing apps with the community.

Overview→

Learn about GitHub's APIs, secure your deployments, and join GitHub's Developer Program.

Webhooks and events→

You can set up, test, and secure webhooks so your integrations can subscribe and react to events on GitHub.Webhooks allow you to build or set up integrations, such as GitHub Apps or OAuth Apps, which subscribe to certain events on GitHub.com. When one of those events is triggered, we'll send a HTTP POST payload to the webhook's configured URL. Webhooks can be used to update an external issue tracker, trigger CI builds, update a backup mirror, or even deploy to your production server. You're only limited by your imagination.

Webhooks can be installed on an organization, a specific repository, or a GitHub App. Once installed, the webhook will be sent each time one or more subscribed events occurs.
You can create up to 20 webhooks for each event on each installation target (specific organization or specific repository).

Events

When configuring a webhook, you can use the UI or API to choose which events will send you payloads. Only subscribing to the specific events you plan on handling limits the number of HTTP requests to your server. You can also subscribe to all current and future events. By default, webhooks are only subscribed to the push event. You can change the list of subscribed events anytime.

Each event corresponds to a certain set of actions that can happen to your organization and/or repository. For example, if you subscribe to the issues event you'll receive detailed payloads every time an issue is opened, closed, labeled, etc.

See "Webhook event payloads" for the list of available webhook events and their payloads.

Ping event

When you create a new webhook, we'll send you a simple ping event to let you know you've set up the webhook correctly. This event isn't stored so it isn't retrievable via the Events API endpoint.

For more information about the ping event webhook payload, see the ping event.

Apps→
You can automate and streamline your workflow by building your own apps.

GitHub Marketplace→
List your tools in GitHub Marketplace for developers to use or purchase.

2 Research on Security-board Director

There are 3 main types of logic programming the same interpreter with methods like forward-backward chaining, case-based approach and predicate sentences on facts.

1. **director.pl** uses a forward-backward chaining method. This method has each rule having different head name and each rule calls the other after execution in chain with the main rule.
After each differential rule calling a different headrule is invoked until the last rule is runs. Then the main, first headrule which is invoked by the Prolog Interpreter is called again in the process. As result causing all the headrules to be chained.

2. **directorcase.pl** is based on case approach. Each head rule has the same name as the other but with a differential increments of integer value starting from 1 to the the total number of headrules desired. Here, we are looking at 12 in total. The Heads has the name cdd(1) to cdd(12). After each similar headrule runs, then the main is invoked again to bring a menu selection for any other head rule in any order.

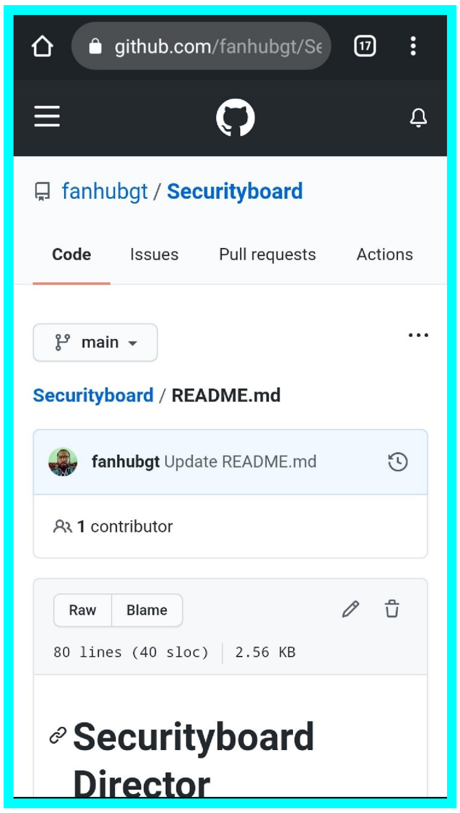
Here the choices on selection are 1, 4, 7, 6. It runs as follows:

Cdd(1) can run after main menu shows up.
Cdd(4) can run after main menu shows up again.
Cdd(7) can run after main menu shows up.
Cdd(6) can run after main menu runs again.

3. **assertcounter.pl** is based on consult approach using a fact loader of predicate sentences. Consult is a builtin rule of Prolog.

Using consult(assertcounter.pl), it will load the facts in the file into its database.

After, writing on the prompt of the Prolog Interpreter will invoke a Yes or no response in checking for assert of fact in the database. If yes then a fact is checked as member of the database.


github.com/fanhubgt/Se

fanhubgt / Securityboard

Code Issues Pull requests Actions

main

Securityboard / README.md

fanhubgt Update README.md

1 contributor

Raw Blame

80 lines (40 sloc) | 2.56 KB

Securityboard
Director

```
?- cm_check(invalid,info).
yes

?- cm_check(intern,replacement).
yes

? - cm_check(passby,riot).
yes

?- cm_check(access,system_codes).
yes

?- cm_check(unlawful,entry).
yes
```

This is a typical way of getting yes responses from the prompt of Prolog execution.

Now no responses.

```
?- cm_check(interns,replacements).
no
```

%made arguments Plural throw no response.

```
? - cm_check(passthrough,riots).
no
```

% passby is now pass-through, also riot is riots.

```
?- cm_check(access,
error throws
```

% incomplete with closing brackets and missing second argument.

# Further Reading

Search for author name Frank Appiah at easychair.org or Google scholar.

Four published articles are on display at the site.

The first is on security reasoning and the other three on the Prolog files

(1) Appiah, Frank. "Security Controls or Countermeasures: Vunerabilities Prevention." Easychair Preprint 4410 (2020).
(2) XProlog. Android IDE for Logic Programming. Playstore. 2020.
(3) Github. Online Repository. Web access at Github.com. 2012.
(4) Appiah, Frank. "Open source project called Securityboard". Hosted at Github.com. Web *access, https://github.com/fanhubgt/Securityboard/blob/main/README.md* 2020.