Rapport de projet

Implémentation d'un IDS

Grégoire Roumache
Florian Fichet

Développement – Sécurité des systèmes
Deuxième année, groupe C-2
Année académique 2020-2021

3 Janvier 2021

1 Introduction

Pour le cours de Développement, nous avons dû implémenter un IDS (Intrusion Detection System) qui analyse le trafic qui passe par une interface réseau de la machine. Il permet de détecter des activités suspectes en fonction d'un certain nombre de règles données au préalable. En cas d'anomalie, le programme peut alerter l'utilisateur en la signalant dans les logs du système.

Nous nous sommes organisés en utilisant la plateforme github, comme recommandé, et voici le lien vers notre répertoire :

`https://github.com/groumache/Intrusion-Detection-System`

2 Organisation du travail et configuration des outils

2.1 Les issues sur github

Les issues sont utilisées pour s'organiser et garder une trace des tâches, des améliorations et des bugs dans un projet sur github.

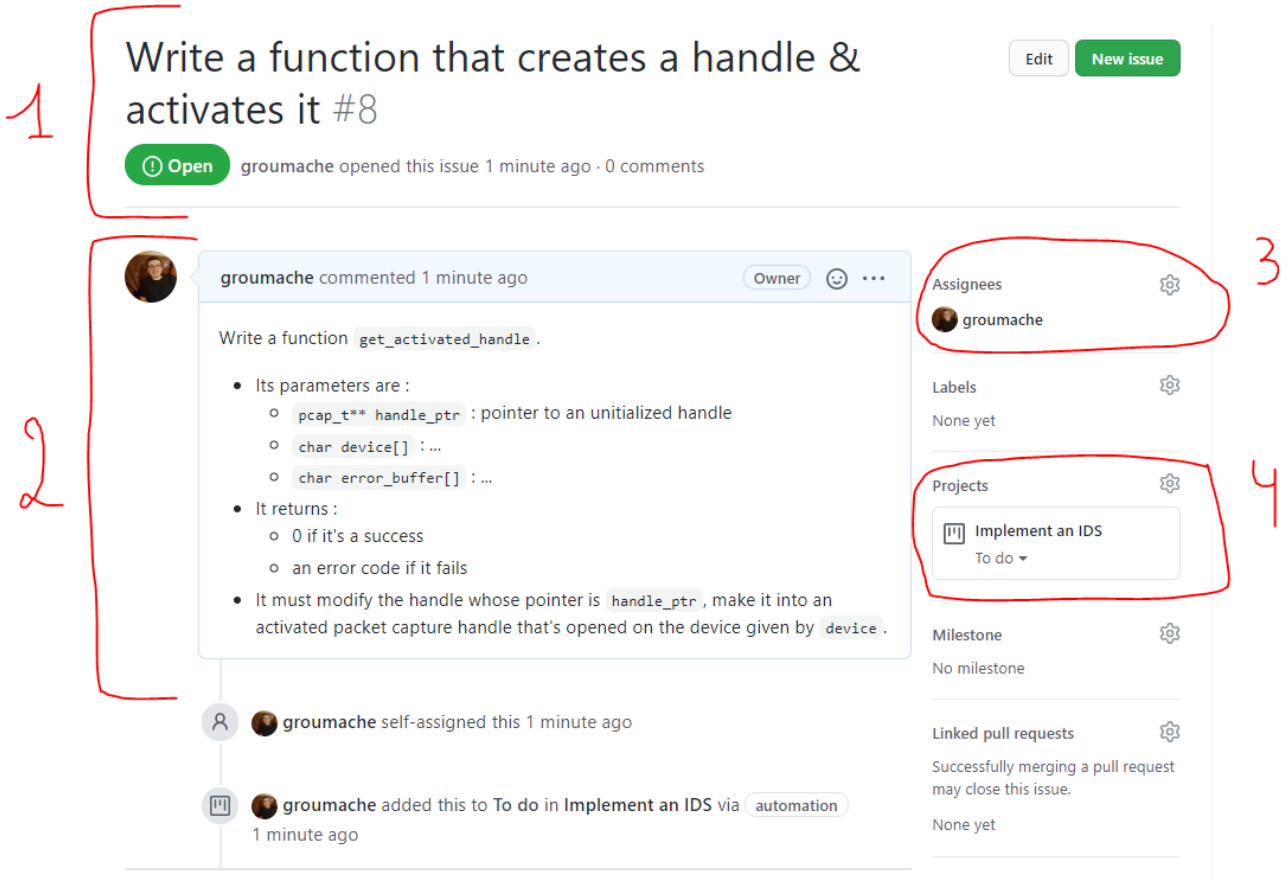

Figure 1 – Une issue sur github

Explication de la figure 1 :

1. Titre de l'issue.
2. Explication/description de l'issue.
3. On peut attribuer une issue à des personnes qui seront responsables pour résoudre l'issue.
4. Le projet dans lequel l'issue va apparaître.

2.2 Utilisation du kanban sur github

Comme nous l'avons précisé dans l'introduction, nous nous sommes organisés en utilisant github. Nous n'avons pas seulement mis le code sur github mais aussi utilisé le tableau kanban avec 4 sections :

1. **Ideas** : un endroit pour ajouter des notes/issues qui dépassent les exigences du projet.
2. **To do** : un endroit où les issues sont placées quand elles sont créées (correspond aux critères minimum de réussite).
3. **In progress** : l'endroit où les issues vont quand on travaille dessus.
4. **Done** : là où les issues vont quand elles sont fermées ou la pull request liée a été fusionnée.

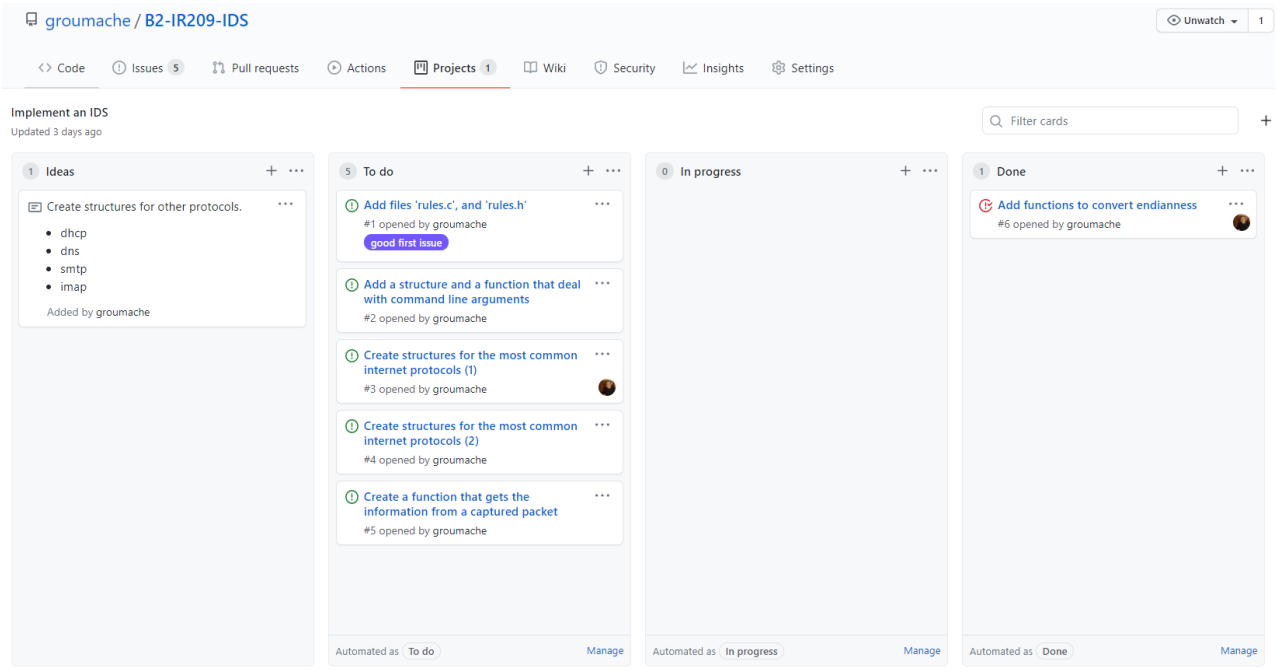

FIGURE 2 – Kanban sur github

Sur le schéma de la figure 3, on voit le network graph. Cette image illustre bien comment nous avons travaillé, c-à-d en créant des nouvelles branches à chaque fois que nous avons eu besoin de résoudre une issue. Une fois que le code a été ajouté, nous avons fait des pull requests pour fusionner les branches.

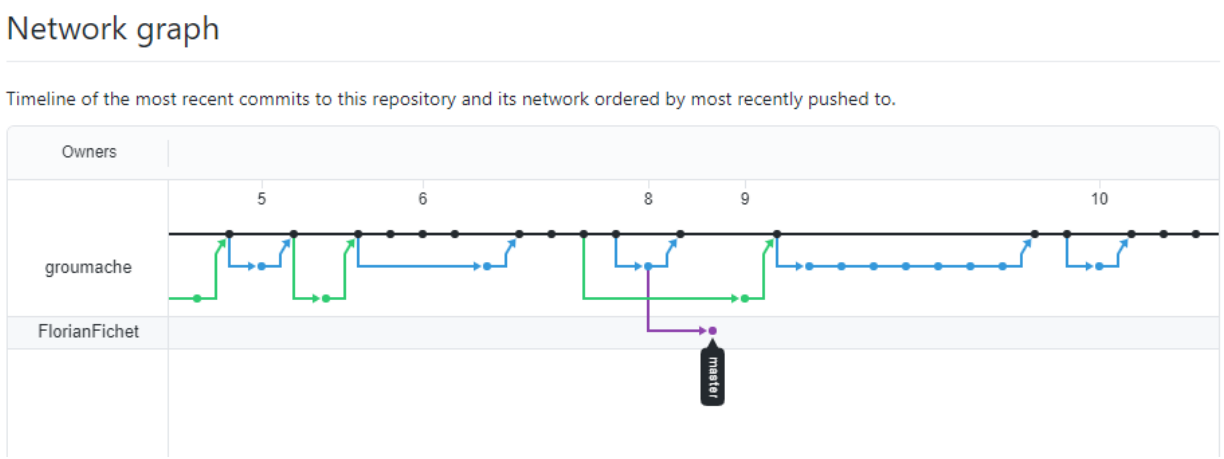

FIGURE 3 – Network graph montrant les commits et comment les branches se rejoignent

2.3 Configuration du debugger

Les fichiers de configurations sont présentés sur les figures 4 et 5.

1. Sur la figure 4, on voit le fichier *tasks.json* où on configure la compilation du projet :
   — on voit que le paramètre *command* donne bien le compilateur gcc,
   — les arguments sont précisés par le paramètre *args*, ce sont ceux recommandés dans l'énoncé,
   — si on combine le tout, c'est l'équivalent de la commande :
   ```
   gcc -Wall -o ids main.c populate.c rules.c -lpcap
   ```
2. Sur la figure 5, on voit le fichier *launch.json*, il sert à configurer le lancement du programme :
   — dans *program*, on voit qu'on a bien précisé *ids*,
   — et dans *args*, on voit les arguments du programme.
   — si on combine le tout, c'est l'équivalent de la commande :
   ```
   ids -p -d eth1 -r ids.rules -n 10
   ```

Remarquez qu'il faut lancer le programme avec des privilèges plus élevé que l'utilisateur standard. Pour réussir à faire cela, on peut lancer visual studio code avec ces mêmes privilèges.

launch.json | tasks.json

developpement-Q1 > projet > .vscode > tasks.json > ...

You, 4 days ago | 1 author (You)

```
{       You, 4 days ago • add code to close the handle and free
    "tasks": [
        {
            "type": "cppbuild",
            "label": "Build project",
            "command": "/usr/bin/gcc-10",
            "args": [
                "-Wall",
                "-g",
                "-o",
                "/home/user/Downloads/project/ids",
                "/home/user/Downloads/project/main.c",
                "/home/user/Downloads/project/populate.c",
                "/home/user/Downloads/project/rules.c",
                "-lpcap",
            ],
            "options": {
                "cwd": "${workspaceFolder}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Task generated by Debugger."
        },
```

FIGURE 4 – Fichier de configuration du debugger, pour la compilation du programme

launch.json | tasks.json

developpement-Q1 > projet > .vscode > launch.json > Launch Targets > { } gcc-10 - Build

You, a day ago | 1 author (You)

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Debug project",
            "type": "cppdbg",
            "request": "launch",
            "program": "/home/user/Downloads/project/ids",
            "args": [
                "-p",
                "-d", "eth1",
                "-r", "/home/user/Downloads/project/ids.rules",
                "-n", "10",
            ],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
```

FIGURE 5 – Fichiers de configuration du debugger, pour lancer le programme

2.4 Utilisation de valgrind

Valgrind est un outil de programmation qui sert principalement à découvrir des fuites de mémoire. Par exemple, si j'alloue un bloc de mémoire dans le tas (heap) et que je ne le libère pas, valgrind va me rappeler à l'ordre et même me dire à quelle ligne le bloc mémoire à été alloué (si le programme a été compilé avec un debug flag).

Lancement du programme avec valgrind (rappel : ids doit être lancé avec des privilèges) :

```
valgrind --leak-check=full \
    --show-leak-kinds=all \
    --track-origins=yes \
    --verbose \
    ./ids
```

L'objectif est donc d'obtenir un résultat comme celui à la figure 6.

```
==2110==
==2110== HEAP SUMMARY:
==2110==     in use at exit: 0 bytes in 0 blocks
==2110==   total heap usage: 165 allocs, 165 frees, 283,249 bytes allocated
==2110==
==2110== All heap blocks were freed -- no leaks are possible
==2110==
==2110== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

FIGURE 6 – Résultat de valgrind

2.5 Configuration du formatage automatique

L'extension C/C++ de visual studio code nous permet de faire du formatage automatique de notre code comme illustré par la figure 7. La configuration se fait au en suivant la syntaxe clang-format, voici celle que nous avons utilisé :

```
{
    BasedOnStyle: Google,
    IndentWidth: 4,
    MaxEmptyLinesToKeep: 2,
    IndentPPDirectives: BeforeHash
}
```

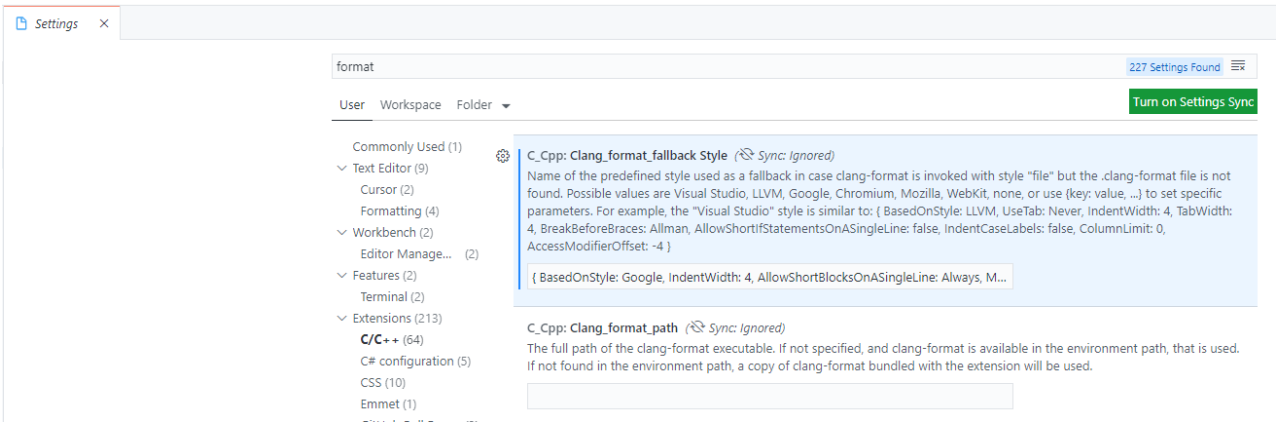

FIGURE 7 – Paramètre de formatage dans visual studio code

3 Architecture du projet

3.1 Architecture du système

Comme on peut voir sur la figure 8, l'ids est divisé en quatre parties logiques.

1. *read_rules* : cette partie est responsable d'extraire les règles du fichier pour les mettre dans les structures appropriées
2. *populate_packet* : cette partie doit extraire le paquet pour le placer dans les structures appropriées
3. *main* : cette partie a 4 responsabilités :
   — analyse les arguments du programme
   — faire appel aux parties *read_rules*, et *populate_packet*
   — vérifier si les packets correspondent aux règles
   — si ils correspondent aux règles, elle doit écrire un message dans syslog

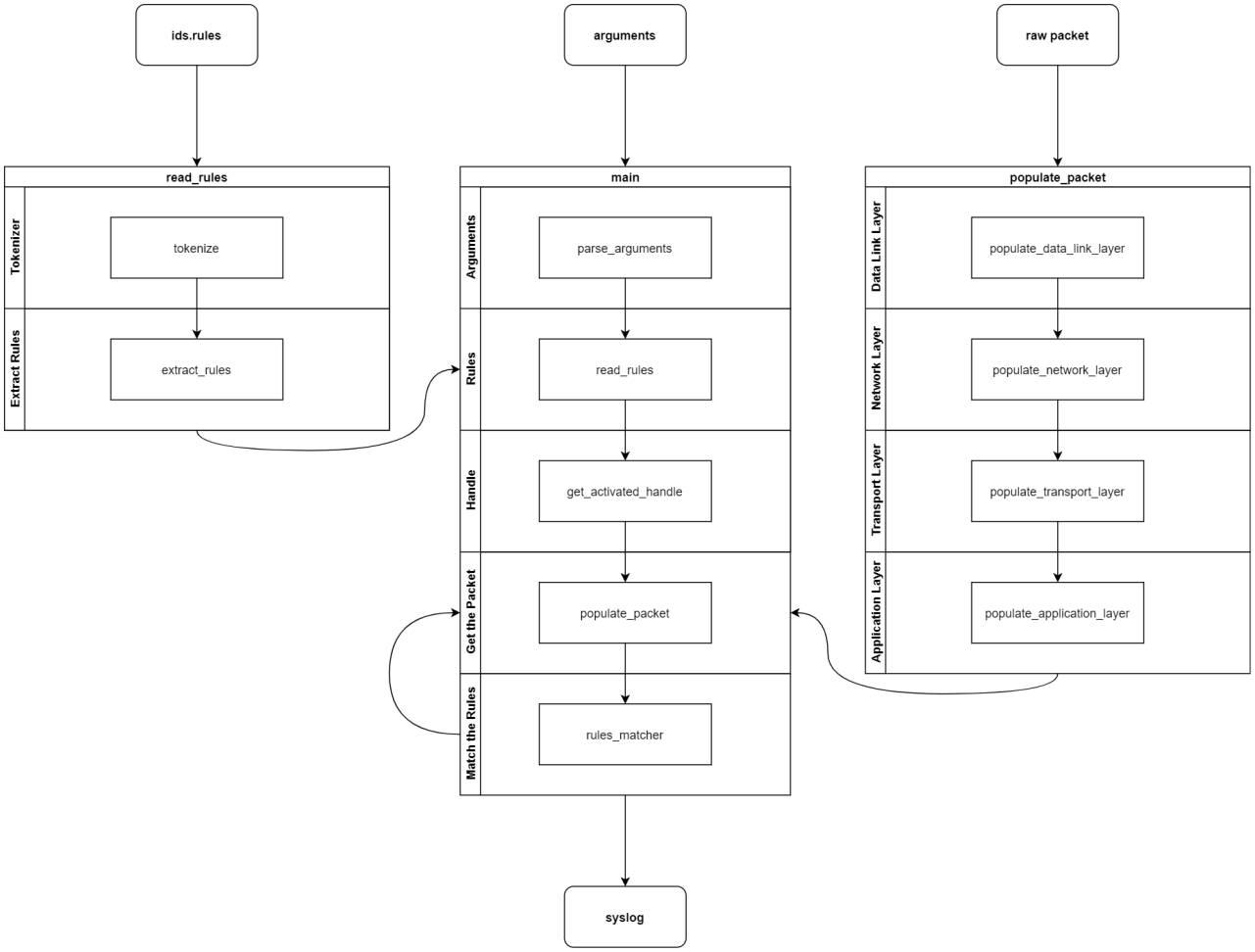

FIGURE 8 – Architecture du système

3.2 Structure du projet

Pour mieux comprendre l'organisation du projet, on peut regarder la figure 9 sert à montrer quels fichiers sont inclus dans quels autres.

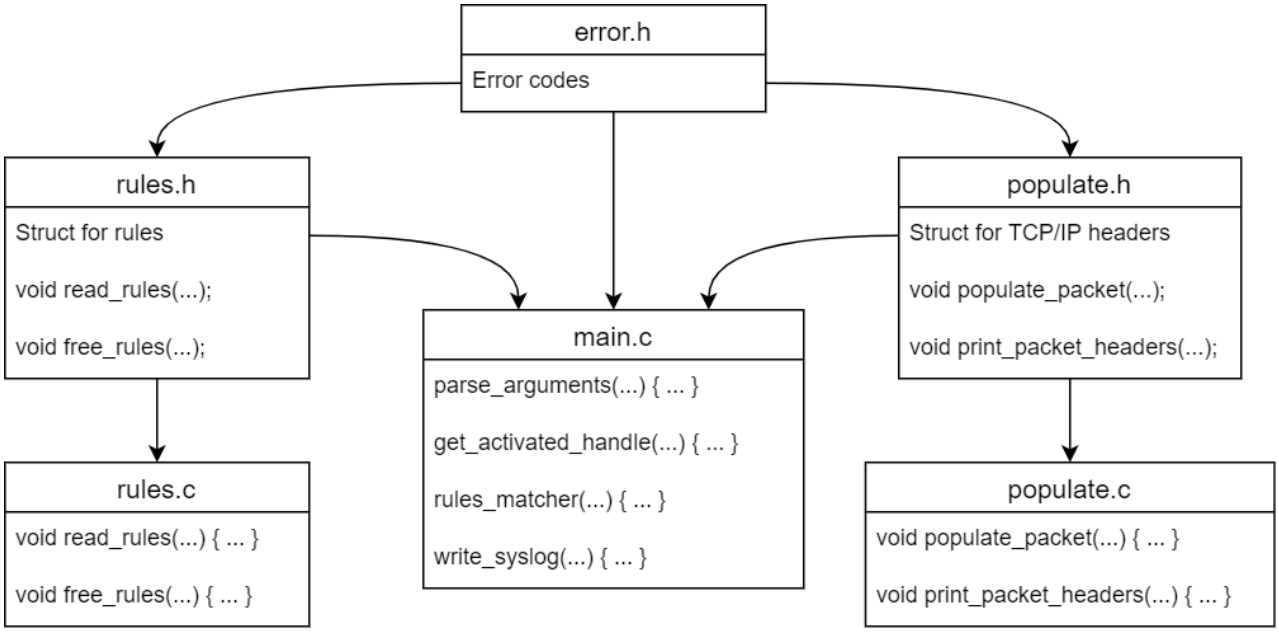

FIGURE 9 – Structure du projet

4 Explication générale sur le fonctionnement du code

5 Choix d'implémentation spécifiques et problèmes rencontrés

Voici hello world :

```c
#include <stdio.h>

// this is hello world
int main(void)
{
    printf("Hello World!");
}
```

6 Améliorations possibles

7 Conclusion

Table des matières

Table des figures

Références

[1] `https://github.com/groumache/Intrusion-Detection-System`