Louis Dufour
Darius Bertrand
Eloan Andre
Paul Squizzato

2022-2023

SAE 4.01
DÉTECTION D'INTRUSION

IUT de Clermont-Ferrand
Département informatique

Remerciements

Nous tenons à témoigner notre reconnaissance à toute personne qui a contribué de près ou de loin à la réalisation de ce projet.

Nous remercions vivement dans un premier temps notre tuteur Maxime PUYS pour sa patience, sa disponibilité et surtout pour les précieux conseils qu'il nous a prodigués tout au long de la réalisation de ce projet.

Nous tenons aussi à remercier notre tutrice de gestion de projet, Mme Chatti pour son aide, ses retours constructifs et ses recommandations pour la partie gestion de projet.

Nous remercions également, pour finir, l'ensemble des membres du jury qui ont accepté d'évaluer notre travail avec attention et professionnalisme.

Sommaire

I) Introduction

1. Contexte

Le but de notre SAE était de développer une solution de détection d'intrusion pour les systèmes industriels. Les systèmes industriels sont la cible de plus en plus d'attaques informatiques, due à leur récente connexion via des réseaux vulnérables tels qu'Internet. Cette faiblesse peut avoir des conséquences au niveau de la santé, de la sécurité et de l'environnement, ainsi que des conséquences financières et/ou des conséquences au niveau de la réputation de l'entreprise.

Dans le cadre de travaux internes, le CEA-Leti a développé une méthode d'analyse de risques spécifiques à ce type de systèmes.

Cette méthode est capable de générer des scénarios d'attaques (suites d'actions réalisées par un attaquant) permettant de causer des dommages aux systèmes.

L'attaquant se situe sur le réseau et pourra donc envoyer, recevoir, ou bloquer des messages entre les dispositifs du système.

Dans le cadre de ce projet, nous cherchions à transformer cette méthode d'analyse de risques en sonde de détection d'intrusion, afin de détecter en temps réel si ces scénarios d'attaques se produisent dans le réseau. Pour ce faire, nous devions programmer des automates d'état fini afin de pouvoir suivre l'état et l'évolution en temps réel de ces systèmes industriels. Cela nous permettra de prévoir les attaques et de bloquer les tentatives d'intrusion.

2. Genèse

L'idée à l'origine du projet provient principalement de la cause d'attaques cyber qui sont des actions malveillantes menées par un individu ou une organisation pour accéder sans autorisation à des systèmes informatiques, des réseaux et des données, en causant potentiellement des dommages ou des interruptions de service volontaires. Il ne manque pas d'exemples d'attaques.

Un exemple classique d'attaque est Stuxnet, un ver informatique utilisé pour nuire aux centrales d'enrichissement d'uranium Iraniennes en 2010. L'attaque a visé les centrifugeuses, ce qui les a ralenties, causant des explosions.

M. Puys, un de nos tuteurs chercheur à l'organisme CEA-Leti nous a donné une mission dans un contexte professionnel qui consiste à développer une solution de détection d'intrusion dans des systèmes industriels, leur sécurité étant une priorité.

3. Analyse du besoin

Notre projet a pour but de détecter et d'analyser toutes commandes exécutées sur un système industriel, il doit permettre de bloquer les commandes qui ne sont pas autorisées avec l'état du système à l'instant de l'exécution. Nous devions donc développer une sonde dans le réseau qui permet d'analyser les requêtes faites sur le système, on devait aussi enregistrer l'état courant de notre système dans une base de données en temps réel, nous comparons ensuite l'état courant avec l'état du système si il y a modification, si l'utilisateur peut exécuter sa commande alors la requête n'est pas bloquée et l'état est mis à jour dans la base de données, si l'utilisateur exécute une commande qui est interdite par rapport aux règles fixées sur le système, la requête est bloquée.

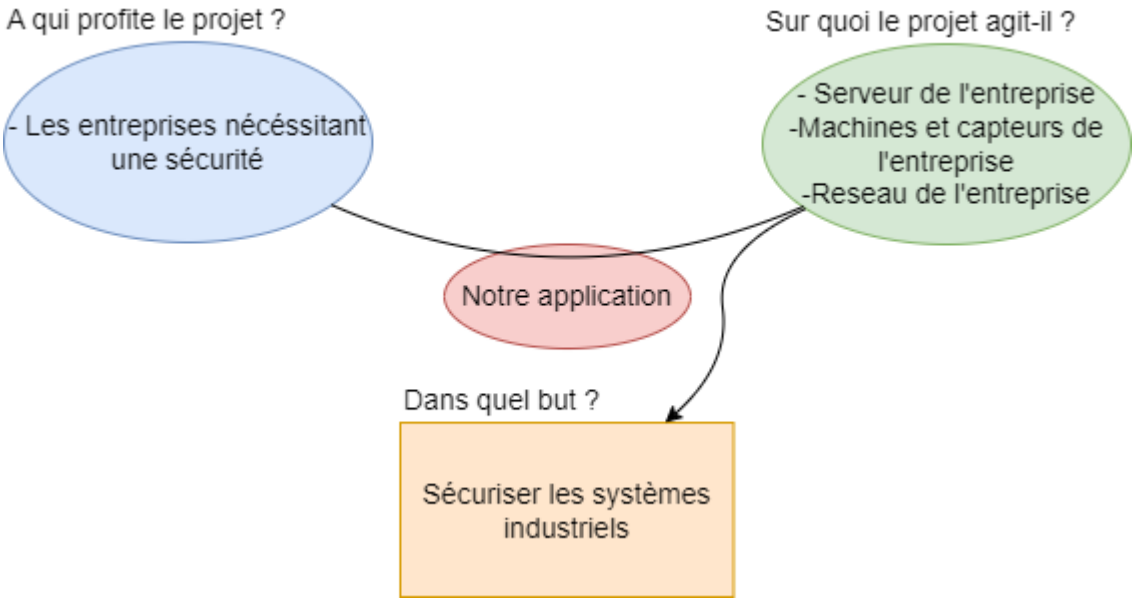

Figure 1 : Bête à corne

4. Objectifs

La solution proposée a été testée et validée pour évaluer sa fonctionnalité et ses limites. Les résultats ont montré que la solution était capable de détecter avec succès les scénarios d'attaque, en utilisant la base de données et les scripts comme sonde de détection d'intrusion. La détection en temps réel des messages dangereux a été efficace et il ne nous manque que la remontée des alertes vers un applicatif tel que Wazuh.

5. Les Types d'utilisateurs:

Les types d'utilisateurs seront dans un premier temps:

- Les organismes de systèmes d'automatisation et de commande industrielles (IACS).
- Les personnes passionnées ayant envie d'installer notre solution sur leur réseau.

Ensuite, dans de nouvelles versions, nous pourrons ouvrir notre solution a un plus grand public.

6. Annonce du plan et problématique

Ce rapport se constitue en plusieurs parties, dans un premier temps, le développement, où le sujet et sa gestion de projet seront présentés. Dans cette même partie, une analyse technique du projet et les compétences mobilisées seront abordées. Nous finirons sur une conclusion et un résumé en anglais. Suivi en annexe des hors-textes et remarques générales.

La question que l'on a pu se poser lors du début du projet est devenue notre problématique:

- Comment sécuriser un système industriel connecté au réseau de toute action malveillante ?

II) La gestion de projet

Merci de vous référer au dossier de gestion de projet rendu aux enseignantes de gestion.

III) L'analyse et le développement

1. Outils & plateformes

Voici les outils qui nous ont été utiles pour la réalisation de notre SAE :

Figure 2 : Outils utilisés

2. Fonctionnement

Notre projet fonctionne encore avec les scripts en brut, l'utilisateur devra d'abord remplir le fichier de règles qui définit quelles requêtes doivent être bloquées selon l'état actuel du système. Le système est quand même modifié, mais la base de données non, et un message d'alerte est remonté à l'utilisateur.
Nous prévoyons dans de futures versions, de créer un script permettant à l'utilisateur de créer facilement des règles, mais aussi, nous prévoyons la mise en place de notre solution en image docker, cela permettra à n'importe quel utilisateur sur n'importe quel système d'utiliser notre solution.

Pour organiser notre temps lors du projet, nous avions décidé d'utiliser un PERT temps, et un GANTT. Le PERT temps nous a permis de définir des tâches et trouver lesquelles il nous fallait prioriser pour rendre le projet dans le temps imparti. Le GANTT lui nous a permis tout d'abord de répartir nos heures de travail au cours des différentes semaines mais aussi de pouvoir suivre l'avancement de notre projet en temps réel, nous pouvions donc adapter notre travail et respecter la méthode agile. Voici ce à quoi ressemblait notre GANTT prévisionnel :

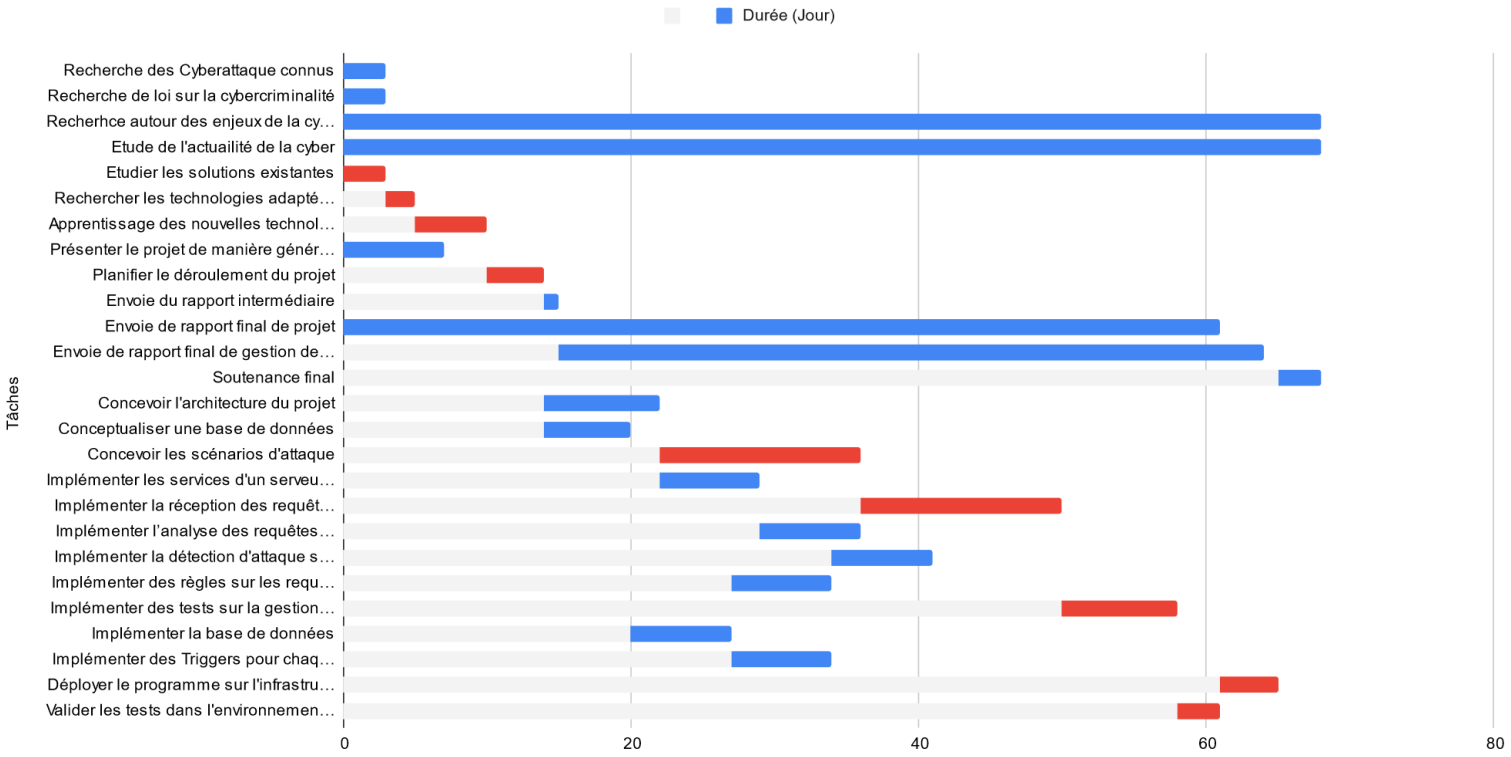

Figure 3 : Gantt Prévisionnel

Pour gérer notre travail nous avons aussi utilisé un Kanban présent sur code first, celui-ci nous permettait de répartir les tâches entre les différents membres de l'équipe en temps réel et ainsi de pouvoir lorsque l'on observe que l'on est en avance sur une d'entre elles de pouvoir continuer ou de pouvoir aider nos coéquipiers.

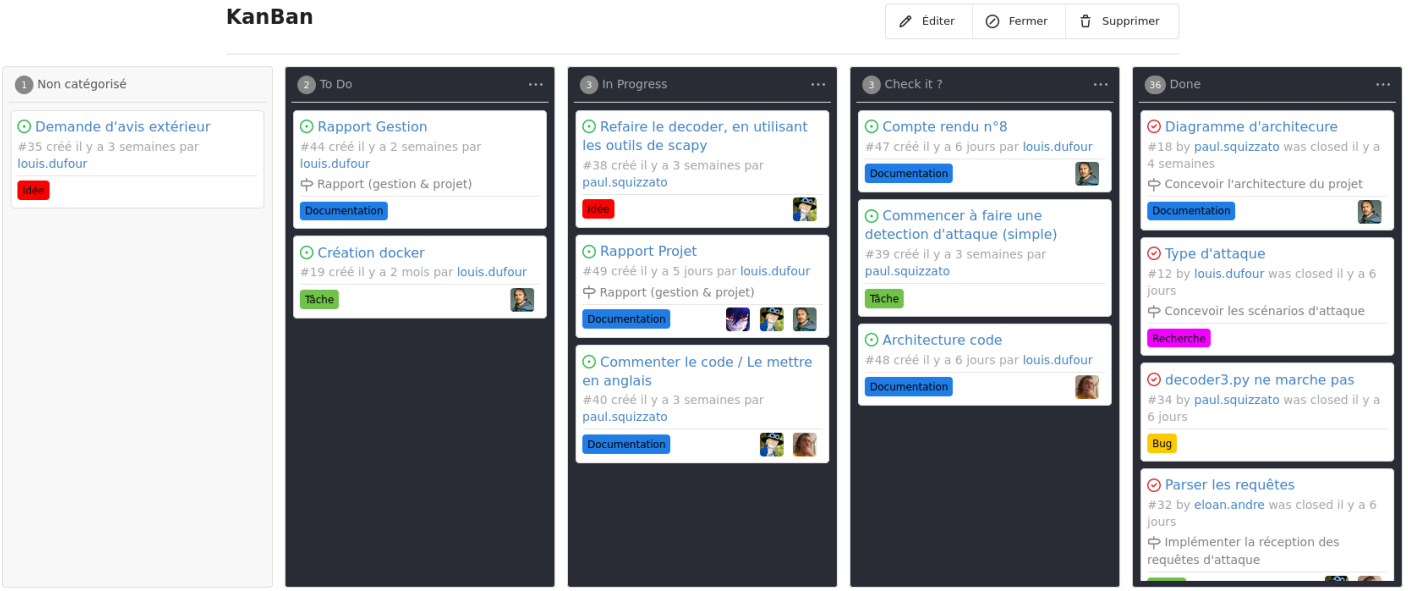

Figure 4 : Capture d'écran du KanBan de code first.

Le Kanban s'organise en 5 colonnes :

- Non catégorisé = pour les tâches qui sont destinées à la recherche principalement
- To Do = Ceux qui doit être fait
- In Progress = Ceux qui est entrain de se réaliser
- Check it ? = Ceux qu'il faudra vérifier
- Done = Ce qui est fini

La première étape de ce projet a consisté à implémenter une base de données pour stocker l'état du système industriel en temps réel. Cette base de données a été conçue pour stocker des associations de variables et de valeurs telles que le niveau de la cuve, le statut de remplissage, etc. La mise à jour en temps réel de cette base de données a été assurée en configurant le réseau de manière à ce que les paquets passent par une sonde, configurée en mode tcp_forward c'est-à-dire en gérant les paquets sur le chemin entre le client et le serveur. Un lien entre le réseau et la base de données a également été mis en place à l'aide d'une surcouche de SCAPY.

La deuxième étape a consisté à mettre en place un moyen de détecter en temps réel si un message envoyé correspondait à un scénario d'attaque. Cette détection a été réalisée en interne, à chaque requête la sonde récupère les données, les parses et appelle plusieurs fonctions, une qui nous permet de trier nos données, une autre qui vérifie si la requête est conforme aux règles et ensuite une qui enregistre dans la base de données.

Ensuite nous avons créé le script qui contient les règles qui doivent être respectées par les utilisateurs.

Nous pouvons aussi noter qu'un effort de documentation de code en anglais à été fourni afin de permettre à chaque utilisateur de l'application de bien comprendre son fonctionnement et pour faciliter sa maintenance. En effet, comme ce code est prévu pour être délivré à un client par le CEA-Leti, les entreprises qui peuvent être françaises ou internationales auront besoin de vérifier son code source avant de l'intégrer au système de sécurité de leur entreprise.

1. Diagrammes et Schémas :

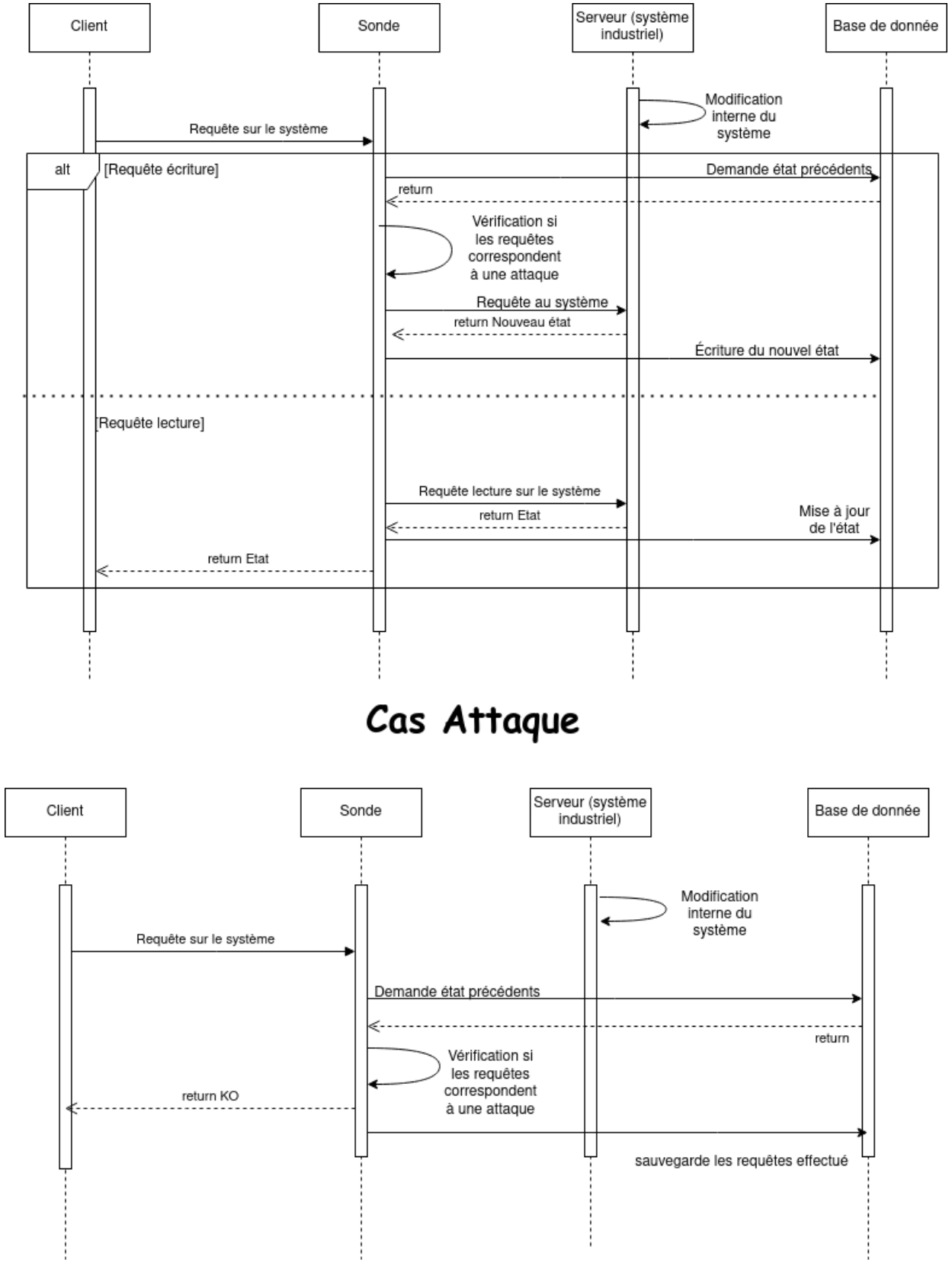

Figure 5 : Diagrammes de séquence

Le premier cas représente le fonctionnement normal du système pour une écriture et une lecture. Un client envoie une requête à la sonde l'intercepte, demande l'état précédent à la base de données, elle vérifie ensuite si la requête est conforme avec les règles attribuées au système, ici oui donc elle est exécutée sur le système industriel puis la base de données est mise à jour.
Pour une requête de lecture la elle est interceptée, elle met la BDD à jour avec l'état courant du système
Le deuxième représente la cas d'une attaque cette fois au moment de la comparaison entre les règles la requête et l'état de la base de données la requête est bloquée.

Notre Architecture Technologique

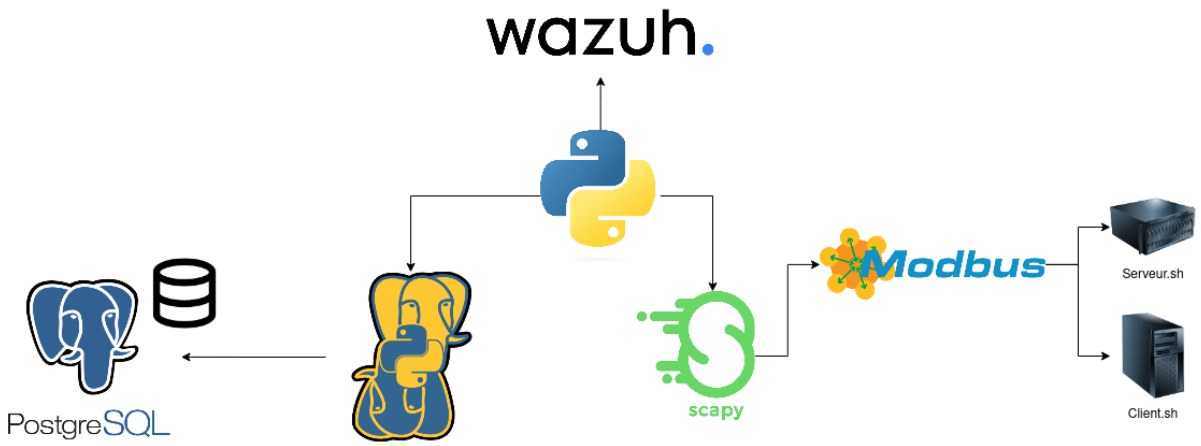

Figure 6: Diagramme d'architecture de notre solution

Les technologies que nous avons utilisé lors du projet :

- Python : offre énormément de librairies afin de faire de la sécurité réseau, mais aussi de mettre en place une communication avec la base de données.
- Scapy : offre la possibilité de voir/capturer le trafic d'une communication entre notre serveur et notre client. *(il est l'équivalent de Wireshark, mais adapté à python)*
- Postgresql : base de données énormément utilisée lors de notre apprentissage scolaire avec des serveurs fournis par notre établissement afin de réaliser nos tests.
- Psycopg2 : librairie python qui nous permet de réaliser la liaison entre notre base de données et notre code.
- Pymodbus : librairie python qui nous permet la création d'un client et d'un serveur via des scripts python.
- Wazuh : Pour intégrer Wazuh avec Python, vous pouvez utiliser l'API REST fournie par Wazuh pour accéder aux fonctionnalités de Wazuh.

Pourquoi pas Snort ou Suricata ?
La surveillance en temps réel des valeurs de variables qui prennent en compte l'état précédent n'est pas une fonctionnalité courante des systèmes de détection d'intrusion et peut nécessiter des outils de débogage spécifiques pour être réalisée.

La meilleure solution pour surveiller en temps réel les modifications de variables est d'utiliser des outils de débogage avancés qui permettent de capturer des informations sur les processus en cours d'exécution.

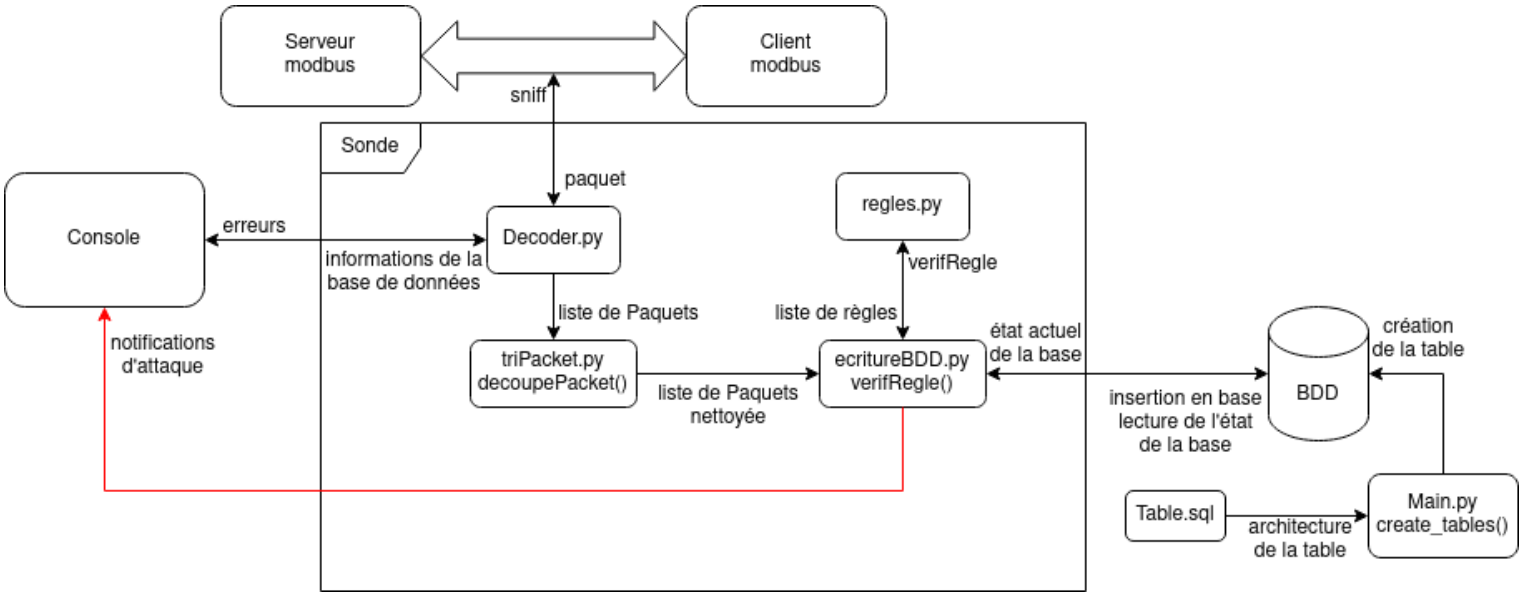

Figure 7: Schéma d'implémentation des différentes parties de notre projet

Ce schéma indique quelles sont les différentes parties que nous avons réalisées, et comment elles interagissent entre elles.
L'objectif principal du projet était la réalisation de la Sonde, et son fonctionnement est détaillé ici, avec la lecture des paquets entre les parties modbus, leur nettoyage et l'écriture en base de données.

2. Client & Serveur

Lors de notre projet, nous avons dû utiliser un client et un serveur afin de simuler une détection d'intrusion sur un réseau.

La notion de client-serveur est un modèle d'architecture logicielle très courant dans le domaine de l'informatique. Elle repose sur une séparation des responsabilités et des tâches entre deux types de programmes distincts: les clients et les serveurs.

Un client est un programme qui demande un service ou une ressource à un serveur. Le client est généralement installé sur un ordinateur ou un appareil mobile et communique avec le serveur via un réseau, tel qu'Internet ou un réseau local. Les clients peuvent prendre différentes formes, telles que des navigateurs web, des applications mobiles ou des logiciels de bureau.

Un serveur, quant à lui, est un programme qui fournit des services ou des ressources aux clients qui en font la demande. Le serveur est généralement installé sur un ordinateur puissant et connecté en permanence au réseau. Les serveurs peuvent prendre différentes formes, telles que des serveurs web, des serveurs de messagerie ou des serveurs de bases de données.

Le modèle client-serveur repose sur une communication entre ces deux types de programmes. Lorsqu'un client demande un service ou une ressource, il envoie une requête

au serveur correspondant. Le serveur traite ensuite la requête et renvoie une réponse au client. Cette réponse peut être une page web, un fichier, une information stockée dans une base de données, etc.

Le modèle client-serveur est très répandu car il permet de gérer efficacement les interactions entre plusieurs ordinateurs et utilisateurs. Il offre également une grande flexibilité et peut être utilisé dans de nombreuses applications différentes, telles que les systèmes de messagerie, les réseaux sociaux, les jeux en ligne, les applications bancaires, etc.

Pour nous, le système industriel est représenté par le serveur de la figure 7, tandis que le client joue le rôle d'une application utilisée par l'entreprise qui gère le système, ou bien un attaquant. Les requêtes du client reçoivent des réponses du serveur, et c'est à partir de cela que nous récupérons les informations nécessaires à la sauvegarde en temps réel.

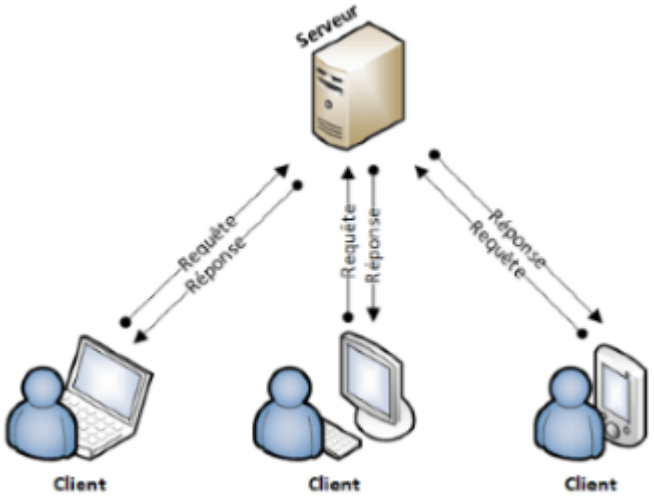

Figure 8 : Exemple de liaison Serveur/Clients

3. Base de données

Avant d'utiliser une base de données, nous devons nous poser la question de ce qu'elle va contenir.
En effet, de nombreux types de données existent et il est impératif de se demander :

- Qu'est ce qu'une base de données va apporter ?
- Qu'est ce que l'on stocke dans cette base ?

Dans notre contexte, la base de données est interne au réseau de l'entreprise qui utilise notre application, elle doit donc pouvoir être adaptable à toutes situations dans lesquelles une entreprise à besoin de stocker des informations. Comme les données stockées proviennent de dispositifs industriels, les données de ceux-ci peuvent être représentées sous 2 types de données : les booléens et les valeurs entières.

Les booléens représentent l’état d’un système, c'est-à-dire par exemple s' il est allumé ou éteint ou bien s' il est actif ou inactif . Dans notre application, cet état est représenté par les “coils” lors du paquet d’envoi.
Les valeurs entières peuvent représenter une valeur qui donne des informations sur l’état du système, par exemple, il peut indiquer si une cuve est remplie d’un certain pourcentage ou bien peut donner la valeur exacte de 5 mètres cubes d’eau. Nous ne faisons pas la différence entre les pourcentages et les autres valeurs étant donné que chaque machine gère elle-même une ligne de la table. Ces valeurs sont représentées par des “registres”.

Notre application contiendra donc une table unique avec 3 types de données :
La clé primaire sera composée de l’adresse ainsi que du type de requête. l’adresse peut être considérée comme une clé étrangère pour la donnée stockée, un capteur dans l’entreprise donnera ces informations sur une seule adresse pour ce qu’elle a capté et cette adresse ne sera pas utilisée par les autres capteurs. Le type de requête fait aussi partie de la clé primaire car elle désigne s’il s’agit d’un “coil” ou d’un “registre” et chacuns d’entre eux possèdent des adresses sur le réseau différentes. Nous avons enfin la troisième colonne qui donne l’état du système, il est partagé à la fois par le registre et par le coil car ceux-ci peuvent être représentés par un Integer avec 0 ou 1 pour le booléen.

Cette base de données est mise à jour lors de la lecture de l’état d’un système par l'extérieur et permet de connaître l’état du système lors d’une requête d’écriture afin d'empêcher les attaques.

4. Langages et Technologie

Comparatif des différentes technos pour la détection d'intrusion

Quelle technos ?

\	Avantages	Inconvénients	Objectif
Snort	- Surveille l'état des serveurs en temps réel en surveillant les fichiers de configuration, les journaux systèmes et d'autres... - Gratuit et open-source - Flexibilité *(peut-être configuré pour détecter un grand nombre de types de menaces)* - Grande communauté - Performant *(Analyse de grandes quantités de données en temps réel)*	- Configuration Complexe - False positifs *(Peut générer de nombreux faux positifs, ce qui peut entraîner un gaspillage de temps et de ressources pour les administrateurs)* - Nécessite une maintenance régulière	Ce logiciel a pour principale fonction IDS[1] et IPS[2].
Suricata	- Architecture plus moderne et des fonctionnalités de détection avancées telles que la détection de menaces SSL/TLS - Détection en temps réel - Support multi-plateforme *(macOS, Windows et Linux)* - Intégration avec plusieurs systèmes *(SIEM[3])* - Mise à jour automatique des signatures de menace	- Complexité d'installation et de configuration - Exigences de ressources élevées *(peut rendre les systèmes lent)* - Faible performance dans les environnements de grande envergure	Ce logiciel a pour principale fonction IDS, IPS et NSM[4].
Zeek (Bro)	- Surveiller l'état des serveurs en temps réel en surveillant les fichiers de configuration, les journaux système et autres... - Peut être configuré pour surveiller le trafic en temps réel - Gratuit et open-source - Grande communauté - Flexibilité (analyse poussée) - Puissance (grande)	- Complexité de programmation *(Nécessite une formation)* - Utilisation de ressources élevées *(peut rendre les systèmes lents)* - Limites dans la détection des menaces - Maintenance lourde	Ce logiciel a pour principale fonction NIDS[5] et NSM.

[1] IDS : Intrusion Detection System
[2] IPS : Intrusion Prevention System
[3] SIEM : Security Information and Event Management
[4] NSM : Network and System Management
[5] NIDS : Network Intrusion Detection System

Il existe d'autres logiciels qui peuvent être considérés comme des alternatives de qualité. Voici quelques exemples :

Security Onion : C'est une distribution Linux qui regroupe plusieurs outils de sécurité, dont Snort, Suricata et Zeek. Elle est conçue pour la surveillance et l'analyse du trafic réseau.

(=Moloch) : C'est un outil d'analyse de trafic réseau open source qui permet de stocker et d'indexer de grandes quantités de données de trafic réseau. Il peut être utilisé pour la recherche de menaces et l'investigation d'incidents.

Sagan : c'est un outil de détection d'intrusion qui utilise les règles de Snort pour identifier les menaces. Il est conçu pour être plus rapide que Snort et pour consommer moins de ressources système.

Suricata-IDS : c'est une alternative à Suricata qui propose des fonctionnalités similaires, mais avec une architecture plus moderne.

Bro-IDS : C'est une alternative à Zeek qui propose des fonctionnalités similaires, mais avec une architecture plus moderne.

OSSEC : C'est un système de détection d'intrusion open source qui surveille l'intégrité des fichiers, les journaux système, le trafic réseau et d'autres aspects de la sécurité du système en temps réel.

WAZUH : C'est une alternative à OSSEC qui offre des fonctionnalités similaires, mais avec une interface utilisateur améliorée et des fonctionnalités supplémentaires telles que la détection d'attaques de type "fileless". Il inclut également des outils pour la gestion des alertes et la gestion des politiques de sécurité.

Conclusion

En tant qu'étudiant en informatique, il est recommandé de se familiariser avec des outils de sécurité populaires et utilisés dans l'industrie. Les outils de système de détection d'attaques tels que Snort, Suricata, Zeek(Bro) et OSSEC sont des exemples de ces outils.

Cela étant dit, il n'y a pas de réponse unique à la question de savoir quel est le meilleur outil de système de détection d'attaque pour un étudiant en informatique. Le choix dépendra de l'objectif de l'étudiant et du domaine d'intérêt. Par exemple, si l'étudiant s'intéresse à la détection d'attaques réseau, Snort, Suricata et Zeek peuvent être de bons choix. Si l'étudiant s'intéresse à la détection d'attaques de système, OSSEC peut être un meilleur choix.

En général, il est recommandé de commencer par un outil de sécurité qui est bien documenté, facile à installer et à configurer, et qui a une communauté active. Cela permettra à l'étudiant de se concentrer sur l'apprentissage des concepts de sécurité, plutôt que de passer du temps à essayer de comprendre un outil complexe.

5. Problèmes techniques rencontrés

Lors du projet nous avons rencontré 3 problèmes majeur:

Le premier était avec notre fichier Pipfile vu que certains membres du groupe utilisaient leur ordinateur personnel il n'avait pas forcément la même version python et cela change les fichier des uns des autres. Nous avons donc décidé de résoudre le problème en mettant en place un gitignore afin que chacun ait son propre Pipfile.

Le deuxième était le plus compliqué, car cela venait d'un problème de version de Pymodbus, en effet lorsque nous faisions la commande `pipenv update` cela nous mettait sur une version trop élevée de Pymodbus qui nous empêchait de lancer le client et le serveur sur nos machines à l'IUT. Nous avons donc dû réduire la version de Pymodbus afin d'assurer la bonne fonctionnalité de notre projet.

Pour finir, le troisième problème était notre précipitation. Nous nous sommes précipités dans le code sans regarder profondément les possibilités que nous offraient les technologies que nous avons utilisées. Ainsi, nous avons commencé par parser le texte de résumé des paquets reçu par Scappy avant que notre tuteur nous apprene que les paquets étaient représentés par des classes avait beaucoup d'attributs qui pouvaient correspondre à nos besoin avec moins de risque de perte d'avoir à mettre à jour notre fonction si une mise à jour de Scapy venait à le demander
Pour résoudre le problème nous nous sommes répartis les tâches afin de gagner en temps. Mais cela nous a coûté tout de même un coup de développement en plus.

IV) Conclusion

Dans l'ensemble, ce projet a abouti au développement d'une solution de détection d'intrusion pour les systèmes industriels, basée sur une méthode d'analyse de risques spécifique et sur l'utilisation d'une base de données en tant que sonde de détection. La solution a été testée et validée avec succès, démontrant son efficacité à détecter les scénarios d'attaque en temps réel. Cette solution pourrait être utilisée pour renforcer la sécurité des systèmes industriels et réduire les risques liés aux cyberattaques.

V) English summary

Our team of four members was tasked with the realisation of a project under 50 hours. We weren't under the guidance of teachers only, unlike other students from our year, as we had the opportunity to have one of our tutors be an employee of CEA-Leti.

We were tasked with a project that correlates with his job, and we were feeling a certain seriousness from the start, having to satisfy needs that could correspond to a real professional experience.

Nowadays, industrial systems are targeted by more and more cyber attacks, as they represent critical assets for the people.

We had to work with two modbus profiles, a client and a server, and a function that probed packets they were exchanging.

From this, we needed to write some Python code to extract data from the packets we received thanks to the probe, and we had to store them in a database.

This was in order to watch for threats to the system, as we were given rules that should prevent some parts of the system from being activated when they shouldn't be. All of this was to detect and notify the system's users when it might be under a cyber attack

In the end, we are happy to deliver our work. This was quite an experience, as it was really far from what we did in practical courses with our studies at the IUT, but we weren't totally unprepared. The short time period was not unbearable, and we are quite satisfied with what we made through this project.

VI) Hors-texte

1) Bibliographie

https://suricata.io/
https://www.snort.org/
https://www.leti-cea.fr/cea-tech/leti
https://wazuh.com/
https://scapy.net/
https://www.python.org/
https://codefirst.iut.uca.fr/
https://docs.zeek.org/en/master/

Norme sur la criticité de la sécurité des systèmes industriels :

Norme IEC 62443-3-3
Norme IEC 62443-4-2

2) Annexes

Install

// Install d'outil
python3 -m pip install pipenv

// Install module
pip install typer
pip install scapy
pip install pymodbus

// Mise à jour
pipenv update

// Lancement du shell pipenv
pipenv shell

ATTENTION : lorsque vous effectuez un pipenv update, cela vous donne une version trop avancée de pymodub (3.2.0), qui est incompatible avec notre décodeur.
(pour voir votre version, faites pip freeze)

Démarche à suivre:

1. Désinstallez la bibliothèque pymodbus à l'aide de la commande suivante:

pipenv uninstall pymodbus

2. Installez une version antérieure de la bibliothèque pymodbus en utilisant la commande suivante:

pipenv install pymodbus==3.1.3

Lancement

Il faut lancer 2 terminaux avec pipenv shell pour utiliser notre projet :
./start_server.sh

./start_client.sh
Et enfin, avec root (pour pouvoir utiliser scapy) :
sudo python3 ./decoder.py

Côté BDD

Il faudra que vous vous connectiez à votre BDD PostgreSQL. Exécuter le script Table.sql qui se trouve dans src avec la commande ci-dessous.

```
\i /YOUR_PATH/Detection_d_intrusion/src/Table.sql
```

Si jamais vous vous retrouvez à devoir partager votre BDD il faudra exécuter la commande ci-dessous.

```
GRANT ALL ON <nom_table> TO <nom_utilisateur_à_ajouter>;
GRANT CONNECT ON DATABASE dblodufour1 TO <nom_utilisateur_à_ajouter>;
```

Pour se connecter à la BDD d'une autre personne, pensez bien à mettre le nom de sa database. Ici, c'est dblodufour1

```
psql -h londres -d <nom_DataBase> -U <votre_nom_utilisateur> -W
```

ATTENTION londres est un serveur hébergé dans l'infrastructure de notre établissement universitaire.

Notre configuration:

- Python (3.9)
- PostgreSQL
- pip (22.0.2)
- pymodbus (3.1.3)

Notre pipfile:

```
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
redis = "*"
click = "*"
prompt-toolkit = "*"
pymodbus = {extras = ["repl"], version = "*"}
sqlalchemy = "*"
scapy = "*"
ipython = "*"

[dev-packages]

[requires]
python_version = "3.9"
```