



I.U.T CLERMONT-FERRAND

PROJET N°14
OPENTHERMOSTAT
RAPPORT

Analyse de conception et de sécurité

Élèves :

Yannis MAHIOU
Gabin SALABERT
Benoît LOUVEAU
Elouan RAFFRAY
Adrien LENOIR

Tuteur :

Sébastien SALVA

Décembre 2017

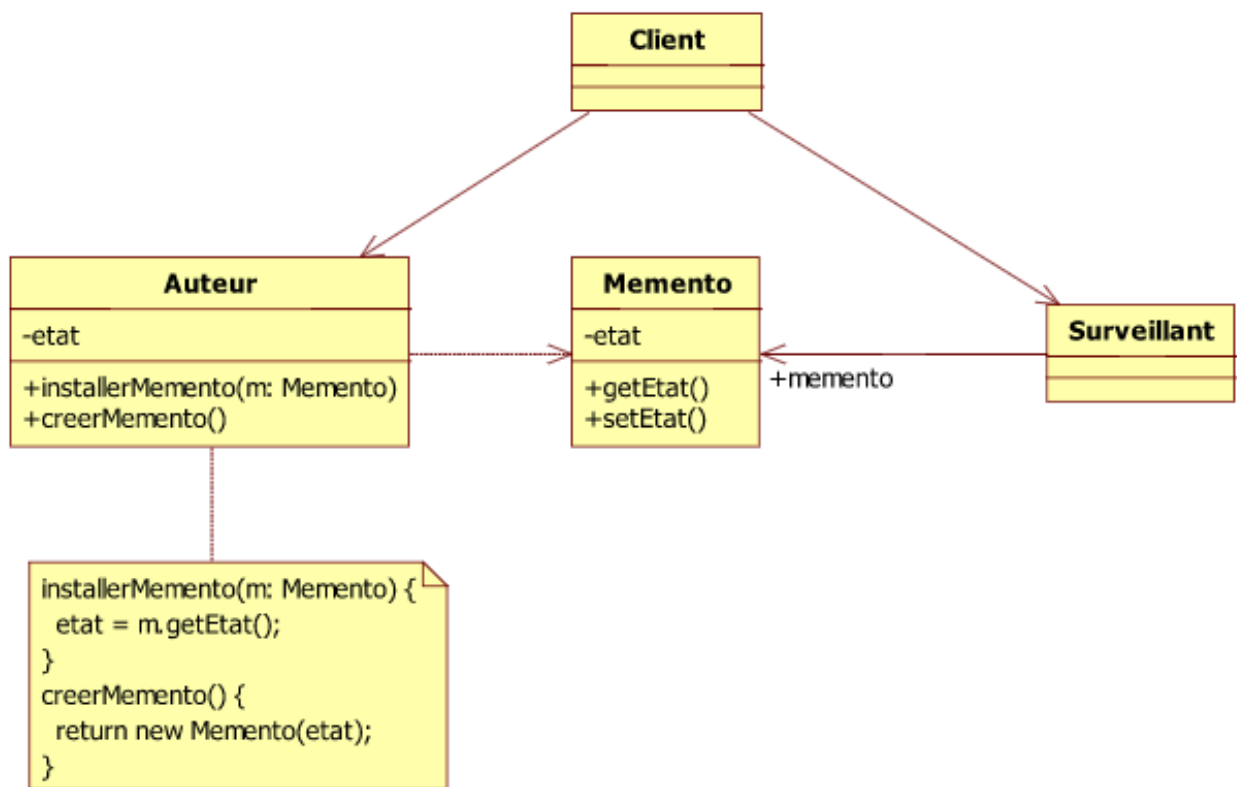
Table des matières

1	Patrons de conception	2
1.1	Patron "Memento"	2
1.2	Patron "Singleton"	3
1.3	Patron "Fabrique simple"	4
1.4	Patron "Facade"	5
2	Exécution des tests de pénétration	6
2.1	Utilisation de (Airmon/Airodump/Aircrack)-ng	6
2.2	Utilisation de ZaProxy	7
2.2.1	Précisions	8
2.3	Présentation de Krack-Attack	11
2.4	Burpsuite	12
3	Patrons de sécurité	13
3.1	Patron "Input Guard"	13
3.2	Patron "Secure Logger"	15
3.3	Patron "Single Access Point"	16
3.4	Patron "Application firewall"	18
3.5	Patron "Security context"	20

1 Patrons de conception

1.1 Patron "Memento"

Sans violer l'encapsulation, le Memento permet de saisir et de transmettre à l'extérieur d'un objet l'état interne de celui-ci, dans le but de pouvoir ultérieurement le restaurer dans cet état. Ici, il nous est presque indispensable dans le sens où un instantané de la totalité ou de parties de l'objet telles que les valeurs des capteurs ou encore les variables de configuration par défaut, doit être sauvegardé de façon à pouvoir restaurer ultérieurement celui-ci, dans cet état. En effet, nous aurions alors une sauvegarde de l'état par défaut ou à un moment précis de l'objet à disposition en permanence dans le cas où un problème surviendrait.



Memento : Mémoire de l'état interne de l'objet original. Il peut stocker autant d'informations d'état interne de l'original que nécessaire, à la discrétion de ce dernier. Protège des incursions d'objets autres que l'auteur. Les mementos ont effectivement deux interfaces. Le Surveillant voit l'interface étroite du Memento - il ne peut que faire passer le memento aux autres objets. Au contraire, l'Auteur voit l'interface large, celui qui lui permet d'accéder à toutes les données nécessaires à sa restauration dans l'état antérieur. Théoriquement, seul l'Auteur, producteur du memento, devrait avoir le droit d'accéder à l'état interne du memento.

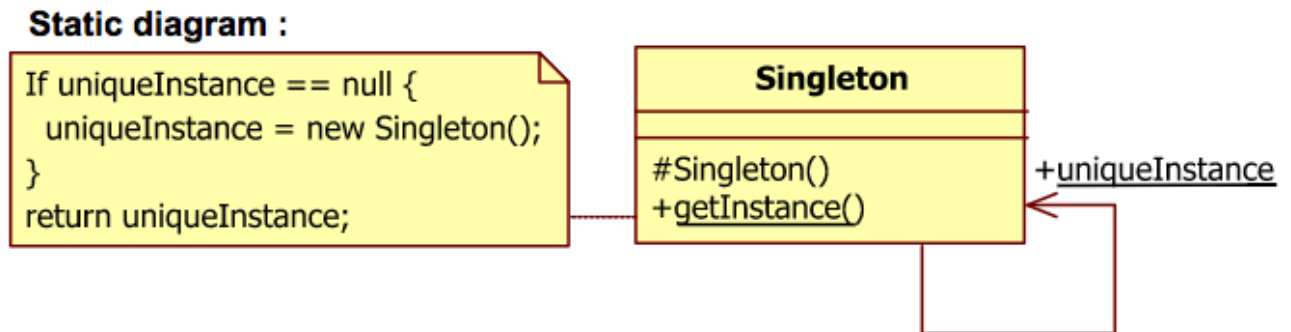
Auteur : Crée un objet Memento contenant un instantané de son état interne courant et utilise Memento pour restaurer son état interne.

Surveillant : Est responsable de la sauvegarde de Memento. N'agit jamais sur Memento,

ni ne l'examine.

1.2 Patron "Singleton"

Le principe de ce patron est de garantir qu'une classe n'a qu'une seule instance et fournir un point d'accès de type global à cette classe.



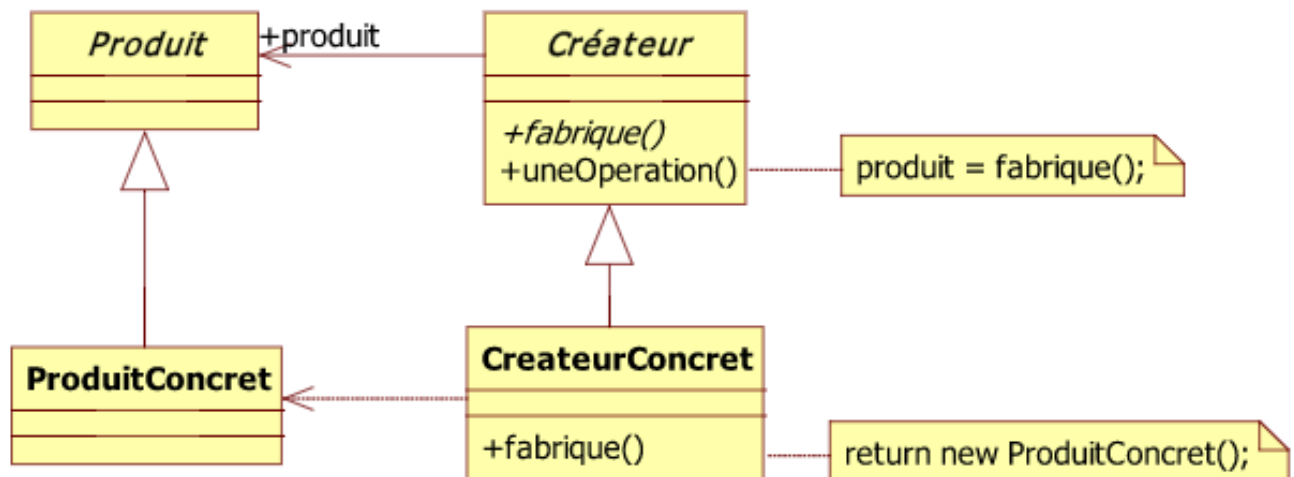
Dans le cadre de notre objet, il doit n'y avoir exactement qu'une instance d'une classe, le cas échéant étant bien trop lourd pour être supporté.

De plus, l'instance unique doit être extensible par dérivation aux autres classes qui l'utiliseront afin d'accéder aux variables.

Singleton : Définit une opération Instance qui donne au client l'accès à son unique instance. Instance est une opération de classe (c'est-à-dire, une méthode de classe en Smalltalk, et une fonction membre statique en C++). Il peut avoir la charge de créer sa propre instance unique.

1.3 Patron "Fabrique simple"

Ce patron définit une interface pour la création d'un objet, mais en laissant à des sous-classes le choix des classes à instancier. La Fabrique simple permet à une classe de déléguer l'instanciation à des sous-classes.



L'implémentation de ce patrons permettrait de gérer l'instanciation de différents objets dans des cas bien spécifiques. La plupart du temps, il est en effet utilisé dans les cas où l'instanciation d'un objet x ou y n'est pas prévisible. Or cette situation ne se présente pas au sein de notre projet étant donné que chaque objet est instancié et joue un rôle.

Produit : Définit l'interface des objets créés par la fabrique.

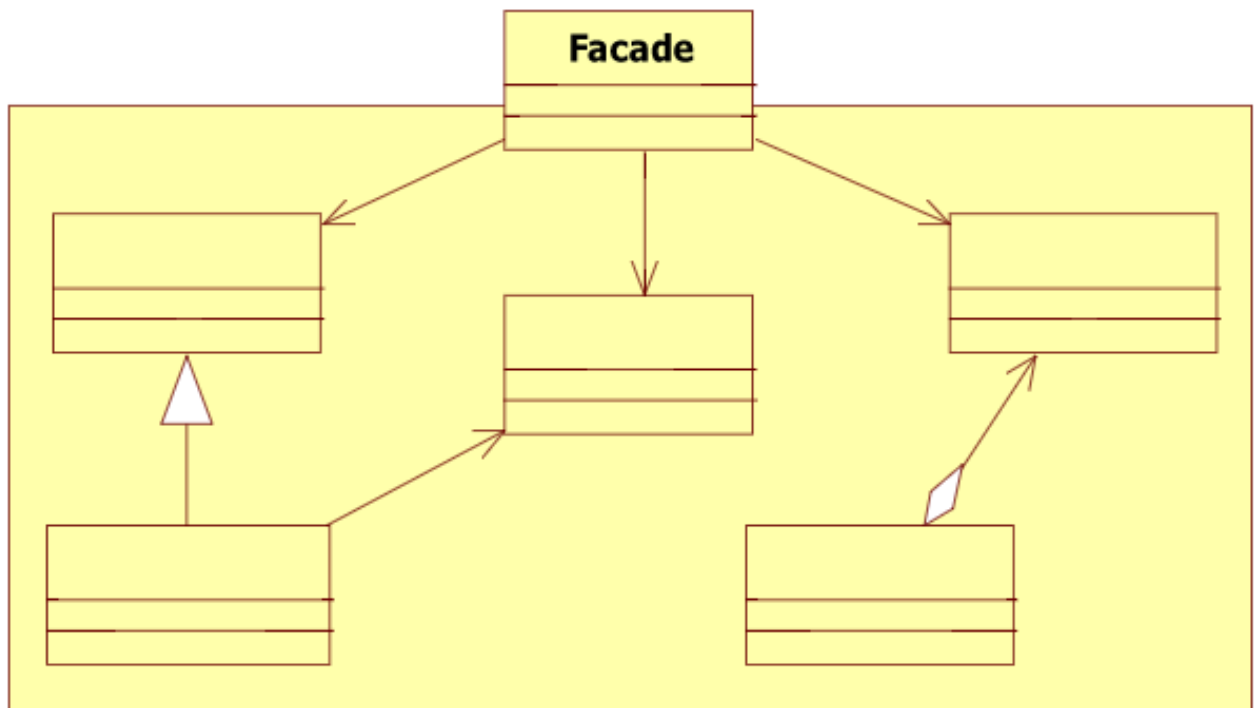
Produit Concret : Implémente l'interface Produit.

Créateur : Déclare la fabrique; celle-ci renvoie un objet de type Produit. Le Créateur peut également définir une implémentation par défaut de la fabrique, qui renvoie un objet ProduitConcret par défaut. Peut appeler la fabrique pour créer un objet Produit.

Créateur Concret : Surcharge la fabrique pour renvoyer une instance d'un Produit Concret.

1.4 Patron "Facade"

Elle fournit une couche d'encapsulation supplémentaire, à l'ensemble des interfaces d'un sous-système. La Façade fournit donc une interface de plus haut niveau, qui rend le sous-système plus facile à utiliser.



L'implémentation d'une façade faciliterait donc le passage au sous-système complexe car il n'y aurait qu'un seul point d'accès. En effet, le nombre d'associations étant très grand, il nous faudrait pouvoir réunir tous les besoins de chaque instance du projet en un seul et même point afin d'améliorer la lisibilité du code et du modèle.

Façade : Connaît les classes du sous-système compétentes pour une requête. Délègue le traitement des requêtes clients aux objets appropriés du sous-système.

Classes du sous-système : Implémentent les fonctionnalités du sous-système. Gèrent les travaux assignés par l'objet Façade. Ne connaissent pas la façade ; c'est-à-dire qu'elles n'ont pas de références à celle-ci.

2 Exécution des tests de pénétration

2.1 Utilisation de (Airmon/Airodump/Aircrack)-ng

La suite Air*, fait partie de la distribution Kali Linux, spécialisée dans le PenTesting. Elle est notamment connue pour ses capacités à “casser” les clés WPA/WEP.

Déroulement :

- 1 : Préparation de l'interface WIFI avec Airmon-ng (Mode monitor).
- 2 : Repérage des AP alentours puis récupération de paquets (sniffing) avec Airodump.
- 3 : Envoi de requêtes de désauthentification vers l'AP, pour récupérer le handshake lors de la reconnexion des équipements connectés sur le WIFI. (Il faut connaître l'adresse MAC d'un équipement connecté à l'AP)
- 4 : Attaque par dictionnaires sur le handshake sniffé précédemment.
- 5 : Résultats :

```

skyx@SkyxDebian: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide

Aircrack-ng 1.2 rc4

[00:00:00] 4/466 keys tested (453.98 k/s)

Time left: 1 second                                0.86%

KEY FOUND! [ password ]

Master Key      : F7 5E 3E 97 0D 11 3E 57 5A 7F C2 DF EB 6B 5B F2
                  7E E7 02 79 7B 6D E0 30 EE C2 6E FF 72 92 A4 56

Transient Key   : 0B 4D 42 83 88 79 8A FD 80 6F 2A 17 CF 4C 7E 19
                  6A DE 58 74 63 6D B9 3C 2A 48 DA 8A 1D CE 69 EB
                  63 34 53 4E 00 77 95 E1 6D 93 B5 00 74 1D 37 86
                  F4 17 43 9E 2E 67 CF 1B 19 78 DD B2 97 26 45 96

EAPOL HMAC     : AC 41 0F DE DB 4A A0 BE 96 7C C2 15 FC 1A A3 99
root@SkyxDebian:/home/skyx#

```

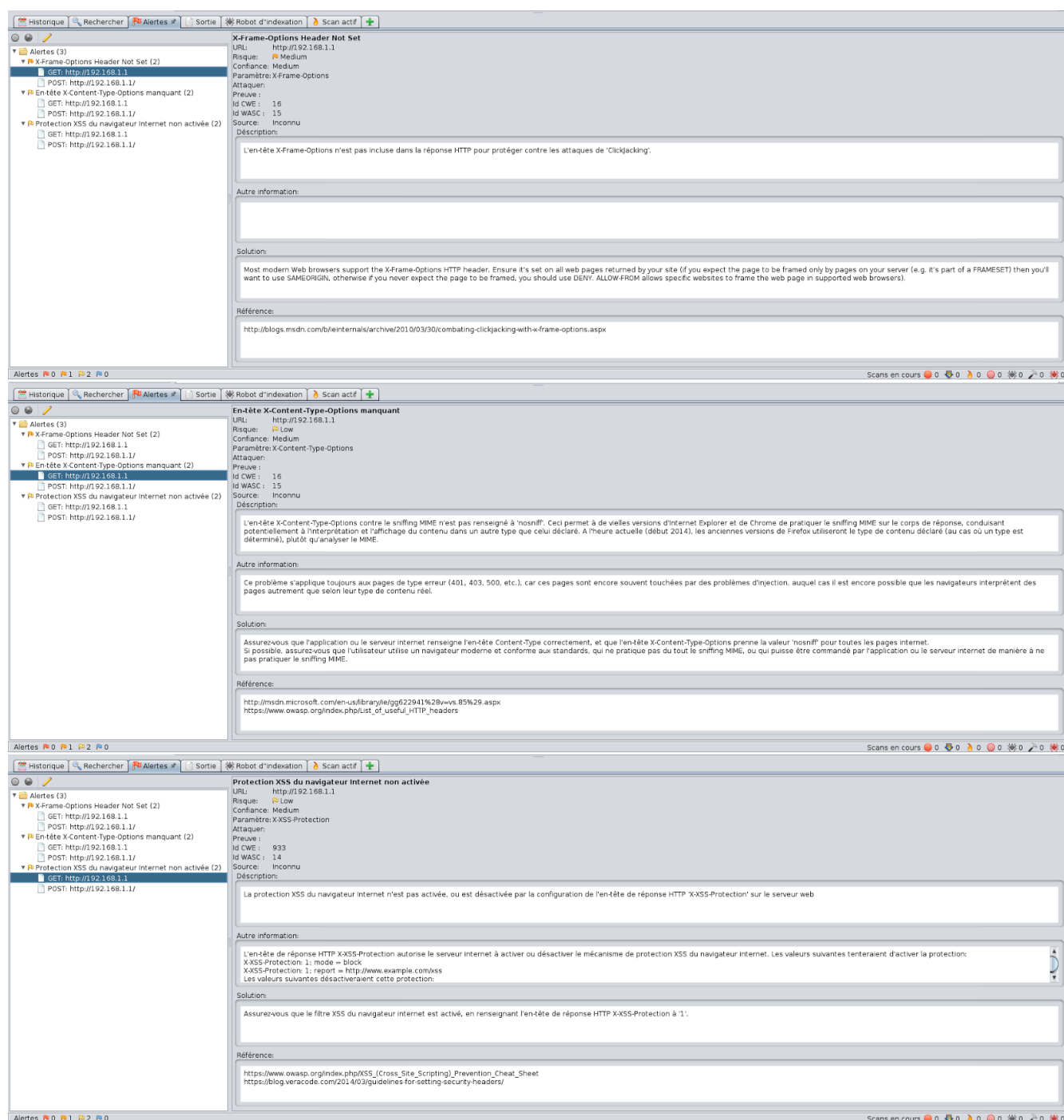
Il est possible de scanner les réseaux alentours avec Angry IP Scanner pour obtenir l'adresse IP du point d'accès puis faire un arping vers l'adresse obtenue pour récupérer une adresse MAC nécessaire à l'attaque.

2.2 Utilisation de Zaproxy

Zed Attack Proxy est un outil de sécurité Open-Source qui permet de trouver automatiquement des failles de sécurité dans des applications Web.

Déroulement :

- 1 : On donne l'adresse web (192.168.1.1) de la carte pour lancer l'attaque.
- 2 : Zaproxy enchaîne les requêtes HTTP pour découvrir des failles.
- 3 : Résultat -> 3 failles trouvées :



The image displays three screenshots of the Zaproxy alert details window, showing the results of a security scan on the target URL `http://192.168.1.1`.

Alert 1: X-Frame-Options Header Not Set

- URL:** `http://192.168.1.1`
- Risque:** Medium
- Confiance:** Medium
- Paramètre:** X-Frame-Options
- Attaquer:**
 - Preuve: 16
 - Id CWE: 15
 - Source: Inconnu
- Description:** L'en-tête X-Frame-Options n'est pas incluse dans la réponse HTTP pour protéger contre les attaques de 'Clickjacking'.
- Autre information:**
- Solution:** Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
- Référence:** <http://blogs.msdn.com/b/enternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx>

Alert 2: En-tête X-Content-Type-Options manquant

- URL:** `http://192.168.1.1`
- Risque:** Low
- Confiance:** Medium
- Paramètre:** X-Content-Type-Options
- Attaquer:**
 - Preuve: 16
 - Id CWE: 15
 - Source: Inconnu
- Description:** L'en-tête X-Content-Type-Options contre le sniffing MIME n'est pas renseigné à 'nosniff'. Ceci permet à de vieilles versions d'Internet Explorer et de Chrome de pratiquer le sniffing MIME sur le corps de réponse, conduisant potentiellement à l'interprétation et l'affichage du contenu dans un autre type que celui déclaré. A l'heure actuelle (début 2014), les anciennes versions de Firefox utiliseront le type de contenu déclaré (au cas où un type est déterminé), plutôt qu'analyser le MIME.
- Autre information:** Ce problème s'applique toujours aux pages de type erreur (401, 403, 500, etc.), car ces pages sont encore souvent touchées par des problèmes d'injection, auquel cas il est encore possible que les navigateurs interprètent des pages autrement que selon leur type de contenu réel.
- Solution:** Assurez-vous que l'application ou le serveur Internet renseigne l'en-tête Content-Type correctement, et que l'en-tête X-Content-Type-Options prenne la valeur 'nosniff' pour toutes les pages Internet. Si possible, assurez-vous que l'utilisateur utilise un navigateur moderne et conforme aux standards, qui ne pratique pas du tout le sniffing MIME, ou qui puisse être commandé par l'application ou le serveur Internet de manière à ne pas pratiquer le sniffing MIME.
- Référence:** <http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx>
https://www.owasp.org/index.php/List_of_useful_HTTP_headers

Alert 3: Protection XSS du navigateur Internet non activée

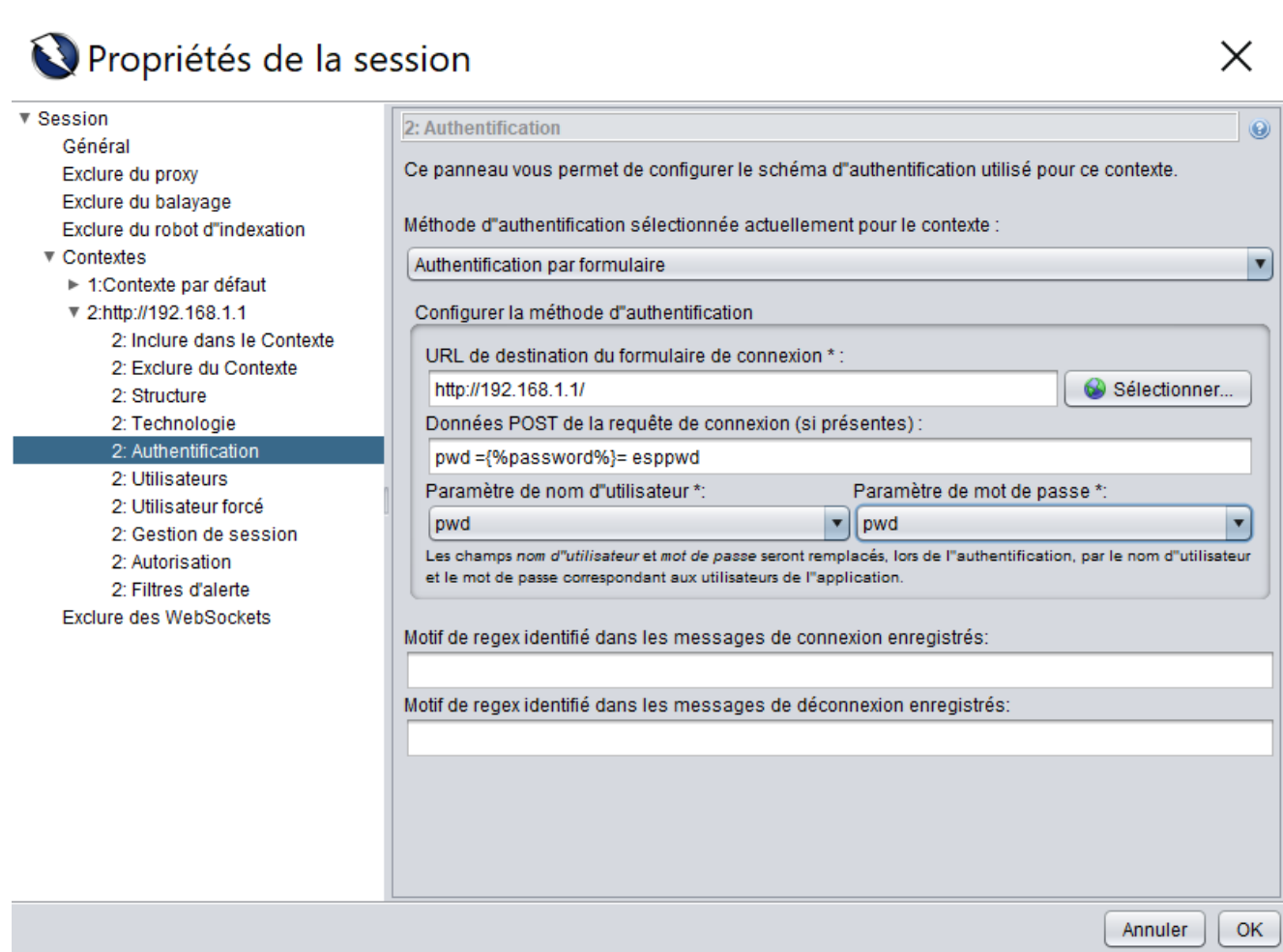
- URL:** `http://192.168.1.1`
- Risque:** Low
- Confiance:** Medium
- Paramètre:** X-XSS-Protection
- Attaquer:**
 - Preuve: 933
 - Id CWE: 14
 - Source: Inconnu
- Description:** La protection XSS du navigateur Internet n'est pas activée, ou est désactivée par la configuration de l'en-tête de réponse HTTP 'X-XSS-Protection' sur le serveur web.
- Autre information:** L'en-tête de réponse HTTP X-XSS-Protection autorise le serveur Internet à activer ou désactiver le mécanisme de protection XSS du navigateur Internet. Les valeurs suivantes tenteraient d'activer la protection: X-XSS-Protection: 1; mode = block
X-XSS-Protection: 1; report = <http://www.example.com/xss>
Les valeurs suivantes désactiveraient cette protection:
- Solution:** Assurez-vous que le filtre XSS du navigateur Internet est activé, en renseignant l'en-tête de réponse HTTP X-XSS-Protection à '1'.
- Référence:** [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
<https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/>

2.2.1 Précisions

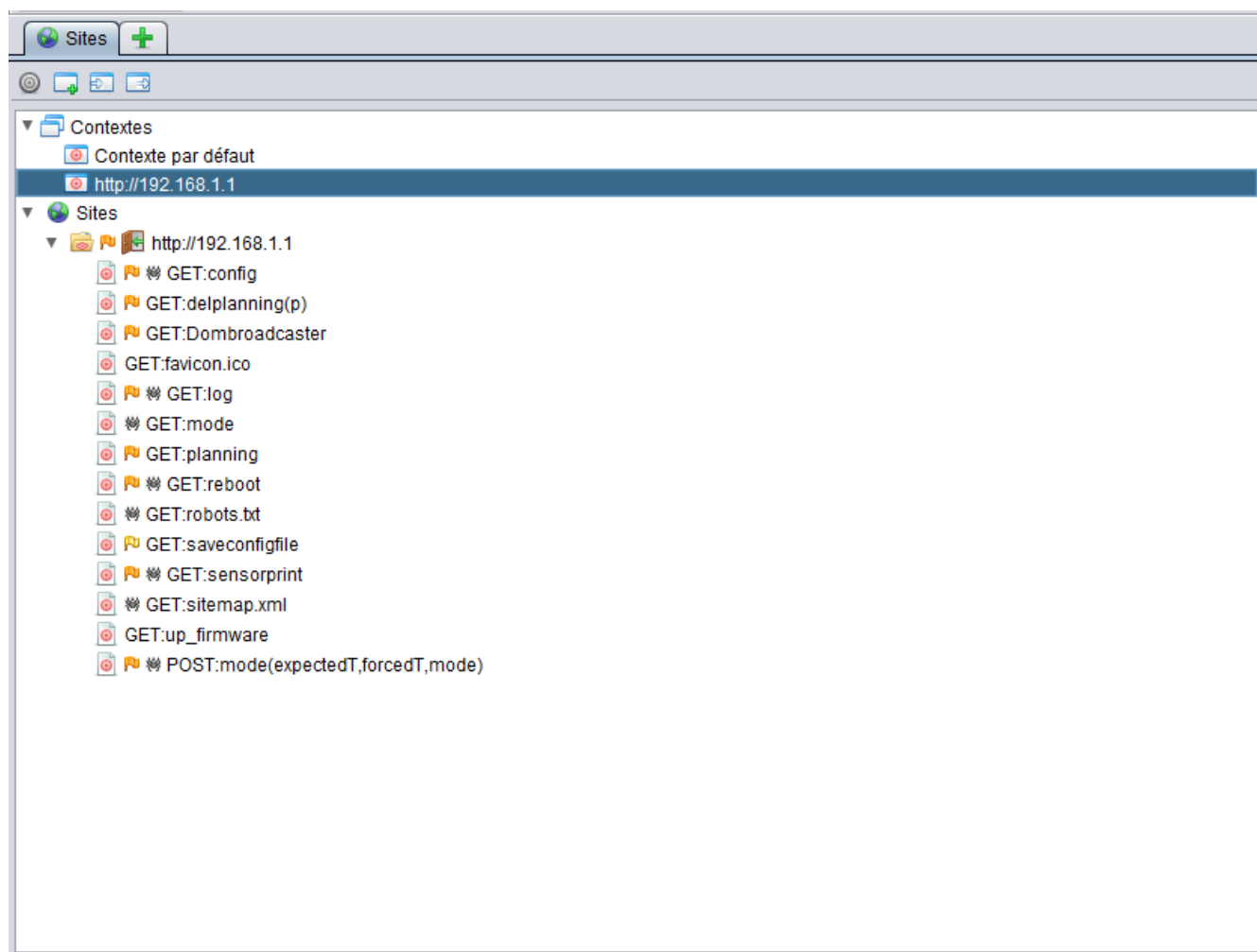
Quelques informations supplémentaires sur l'utilisation de ZaProxy.

Approfondissements :

1 : On peut donner un contexte à ZaProxy, c'est-à-dire des identifiants qu'il peut tester sur le site.



2 : De cette manière, AJAX Spider peut parcourir le site de façon plus ample. Voilà un exemple du parcours plus ample :



3 : Résultat : Une requête reboot réinitialise le système, détruisant les fichiers de config et permettant donc de se connecter à la carte sans mot de passe.

Open Thermostat Main Config Sensor State Log&Info Time 0:0

No password set!

Wifi Info

Wifi SSID:

Thermostat

Mode

Off ▼

Current Temperature

-1.00

Expected Temperature

0.00

Forced Temperature

0.00

Submit

Misc

Reboot

2.3 Présentation de Krack-Attack

Krack-Attack (Key Reinstallation Attack) se base sur une faille de sécurité du protocole WIFI WPA2. Lors de l'échange de clés (handshake), il est possible de forcer une réinstallation de la clé, alors composée uniquement de 0. On peut par conséquent écraser la clé WIFI actuelle.

Déroulement :

- 1 : Utilisation de krackattacks-scripts (<https://github.com/vanhoefm/krackattacks-scripts>) qui est un ensemble de scripts permettant la détection de vulnérabilités relatives à la faille WPA2 et leur exploitation.
- 2 : Il suffit de désactiver le cryptage matériel, le WIFI et de lancer une commande permettant au script d'utiliser le WIFI.
- 3 : Par la suite, on crée un fichier de conf wpa-supplisant en donnant les données de notre borne à tester pour faciliter la connexion.
- 4 : L'étape suivante consiste à exécuter le script principal en passant en paramètre une commande wpa-supplisant pour la connexion. Celui-ci va se connecter à la borne et effectuer les tests.
- 5 : Malheureusement, à l'heure actuelle, lors des tests, le script bloque et n'effectue pas les tests pour une raison encore inconnue.
- 6 : Cependant, il est fort probable que le protocole n'est pas été encore patché sur l'IOT et l'expose donc à cette vulnérabilité encore assez peu corrigé.

2.4 Burpsuite

Burp Suite est une application Java qui peut être utilisée pour la sécurisation ou effectuer des tests de pénétration sur les applications web. La suite est composée de différents outils comme un serveur proxy (Burp Proxy), robot d'indexation (Burp Spider), un outil d'intrusion (Burp Intruder), un scanner de vulnérabilités (Burp Scanner) et un répéteur HTTP (Burp Repeater).

Objectif : Cracker le mot de passe de l'interface Web.

Démarche :

BurpSuite crée un proxy, que l'on fait utiliser par le navigateur. Ce proxy lui permet d'intercepter les requêtes envoyées et reçues par le navigateur. En interceptant une requête de connexion, on peut ensuite marquer le champ contenant le mot de passe, et lancer une attaque, qui va renvoyer cette requête avec des mots de passe différent. Dans notre cas, une attaque en bruteforce sur les caractères alphanumériques a été lancée, uniquement sur des mots a 4 caractères. A l'ordinaire, BurpSuite trie les résultats en fonction de leur code de retour, mais notre application renvoie 200, que le mot de passe soit bon ou mauvais. J'ai du rajouter un filtre qui recherchait la chaîne "Wrong password" dans la réponse de l'objet, afin de sélectionner les bonnes réponses.

Résultats :

BurpSuite a donc trouvé le mot de passe "sasa", mais a apparemment réussi a accéder a l'interface avec de nombreux autres mot de passe (11067 pour être exact), mais en en testant une dizaine a la main, aucun d'eux ne fut une réussite, je pense donc a un bug ou une mauvaise configuration. J'ai d'abord pensé au fait qu'une fois le mot de passe trouvé, les résultat suivants furent eux aussi concluant, mais il y a un mot de passe qui fut testé avant "sasa" qui a marché (d'après BurpSuite), et certains ont été testé après sans pour autant marcher. Une partie de la liste des mot de passe testés est visible ici. (j'ai arrêté le bruteforce en voyant qu'il avait trouvé le mot de passe) Dans tout les cas, le mot de passe a été trouvé au bout de quelques heures seulement. Certes le bruteforce a été effectué sur un ordinateur portable milieu/bas de gamme, mais de toute façon la puissance de la carte limite l'utilisation du bruteforce, qui a été effectué sur 1 thread seulement.

3 Patrons de sécurité

3.1 Patron "Input Guard"

C'est un patron de sécurité que l'on place à chaque point d'accès d'un composant dans le but de vérifier la validité de l'entrée.

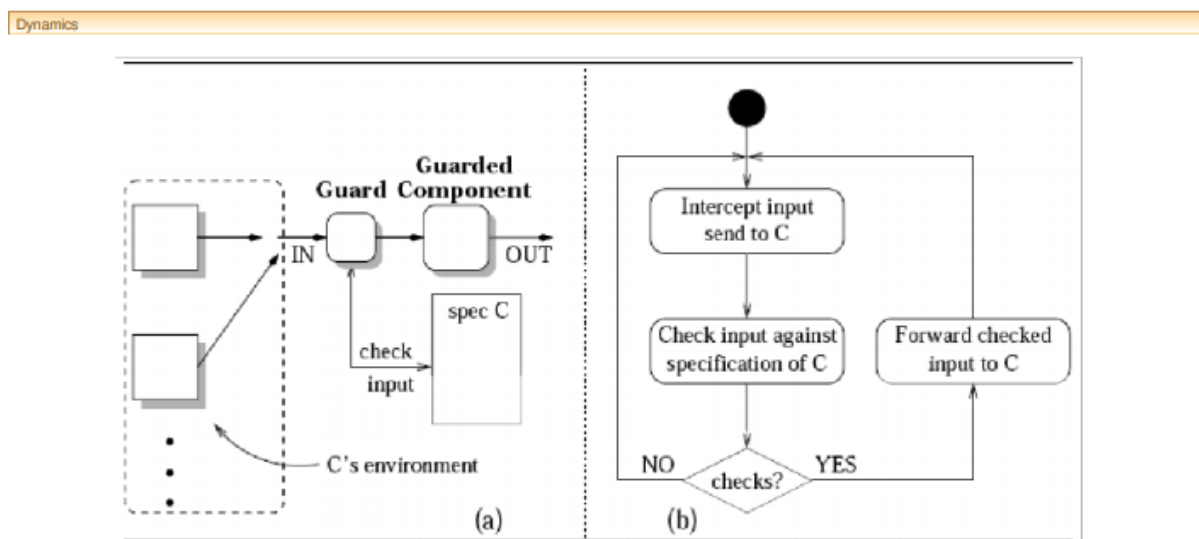
Fonction : Protéger le système contre des inputs non conforme aux specifications demandées.

Analyse : D'après la documentation fournie, le patron Input Guard s'appliquent aux systèmes qui possèdent les caractéristiques suivantes :

1 : Le système est composé de composants distincts, qui peuvent chacun être fautif et qui interagissent entre eux en input/output.

2 : Les erreurs qui peuvent être propagées dans un composant système ont la forme d'inputs erronés. Les erreurs internes (causées par des perturbations électromagnétiques) ne sont pas prises en charge par ce patron car pas sous la forme d'un input erroné.

Notre application correspond à cette description, car il y a de nombreux endroits où l'utilisateur peut entrer du texte, pouvant aboutir à de l'injection de code, etc... La récupération automatique de l'heure (ntp) et les services REST font aussi partie de l'input récupéré. Par exemple, l'utilisateur peut configurer la connexion wifi de l'appareil, et rentrant des valeurs comme "><script>alert("hello")</script>", il peut injecter du code malicieux dans la page, qui sera exécuté lors de l'affichage de la page. Il est donc nécessaire de protéger les input/output, afin d'éviter ce genre de problèmes. Le patron Input Guard pourrait donc être utile à notre application.



3 Implementations de ce patrons sont possibles :

1 : Le garde est implemente comme un nouveau composant du systeme, et son nombre est proportionnel aux nombres de points d'accès a proteger. Cet maniere rajoute du temps d'execution et augmente l'utilisation memoire car l'invocation de garde pour chaque point d'accès est couteux

2 : Le garde est implemente dans le composant a garder. Cet methode est couteuse en temps d'execution car pour rendre le systeme robuste, on check l'input a chaque appel du composant a garder.

3 : Le garde est implemente dans le composant qui envoie l'input. Le nombre d'appels du garde est proportionnel aux nombres de composant qui envoient des donnees au composant a garder. Le probleme est que les garde ne connaissent pas forcement les specifications du composant a garder afin de traiter l'input correctement. Les erreurs de communication ne sont pas geres non plus, car le check de l'input se fait avant l'envoi de celui-ci.

Dans le cadre de notre projet, les inputs sont recuperes/utilises dans les classes : Web et NTP. Pour faciliter l'utilisation de ce patron avec le Single Access Point, il est preferable d'utiliser la 1ere methode. En creant une classe ne contenant que des methodes statiques, on reduit ainsi l'utilisation memoire, et cette methode d'implementation est preferable a la 2eme, car la classe Web est suffisamment lourde et sa responsabilite n'est pas de nettoyer/valider l'input qu'elle recoit.

Notre implementation : La creation d'une classe InputGuard est necessaire, son role sera de traiter l'input avant de l'envoyer au composant a garder. Elle devra utiliser les specifications du composant qui recevra l'input traite. Ces specifications seront contenues dans une classe a part. Deux classes utiliseront l'inputguard, mais leurs specifications sont les memes, car elle sont basees sur le protocole HTTP, ce qui permet d'eviter de creer une classe de specifications supplementaire.

3.2 Patron "Secure Logger"

C'est un patron de conception permettant de centraliser et de sécuriser les logs.

Fonctions : Rassembler les log dans un seul endroit et rendre ses logs impossible a alterer.

Analyse :

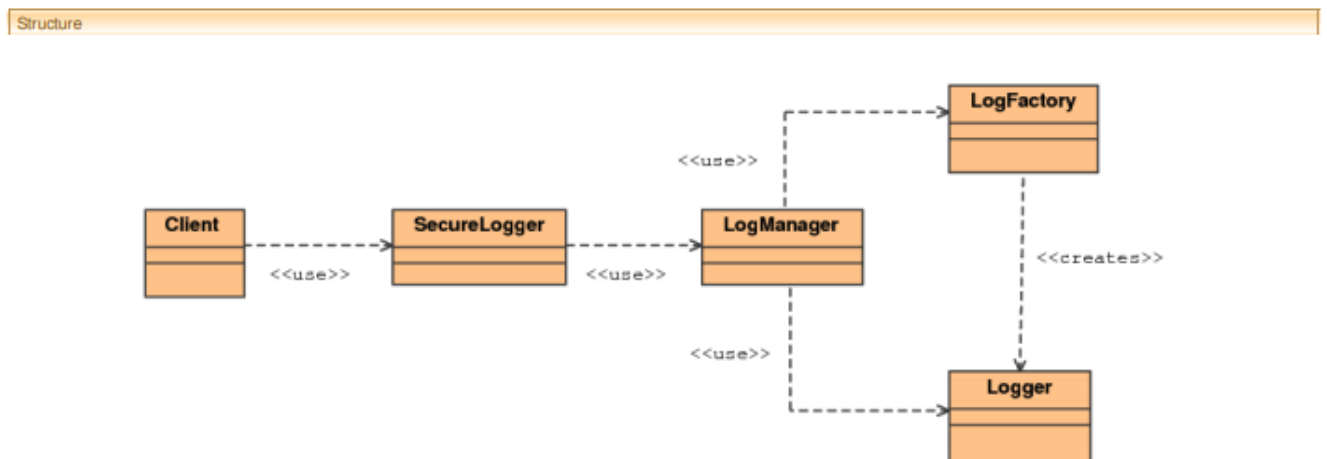
-> Ce patron n'est utile que si :

1 : On a besoin de log des informations sensibles qui ne doivent pas être accessibles à tous.

2 : On doit pouvoir être sûr que le sdonnées loggées sont entières et n'ont pas pû être altérées par un utilisateur malicieux.

3 : On peut centraliser le système de log pour améliorer la gestion de l'application. **4 :** On veut crypter les données du log pour en assurer l'intégrité et la sécurité.

Dans notre application, les donnees sont logees pour des fins de debuggage, et ne contiennent pas d'informations sensible. Le systeme de log est deja bien centralise (passage par la classe Application). Surtout, l'implementation de ce patron impliquerait la creation d'au minimum 4 classes, ce qui serait trop lourd pour notre carte, avec un apport minime sur le fonctionnement.



3.3 Patron "Single Access Point"

C'est un patron de conception permettant de protéger le système et certaines de ses transactions.

But : Apporter un point d'accès unique à l'ensemble du modèle pour l'utilisateur. Cela permet de protéger le système et ses sous-systèmes entièrement de l'extérieur. Il permet de sécuriser la connexion de l'utilisateur au système et gère les autorisations d'accès au modèle.

Utilité :

Ce système, facilement implémentable, est très utile dans une application dont le lancement ne se fait que d'une seule manière. Ce point d'accès à l'application peut permettre de contrôler l'initialisation de toutes les variables (e.g configs, etc ..). C'est ce point d'accès unique, qui contrôle si un utilisateur peut se connecter, peut se coupler avec un pattern tel que Input Guard pour vérifier les credentials d'un utilisateur. En revanche, il rend l'accès à l'application moins flexible. Dans notre cas, il pourrait permettre l'encapsulation complète du modèle pour l'extérieur en gérant les requêtes vers le modèle. En pratique, toutes les classes auraient leur constructeur privé, et le SAP serait friend de toutes les classes. Ainsi, personne à part le SAP ne pourrait instancier les classes, il s'assurerait ainsi de leur bonne initialisation et renverrait la référence de l'objet. Une requête faite pour accéder à la classe Web par exemple serait nettoyé puis validé si elle contient des inputs puis instancierait la classe et la renverrait de telle manière qu'on puisse la manipuler par la suite via ses méthodes publiques. On ne pourrait donc pas instancier la classes à l'extérieur du modèle sans passer par le SAP en ayant son autorisation.

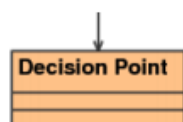
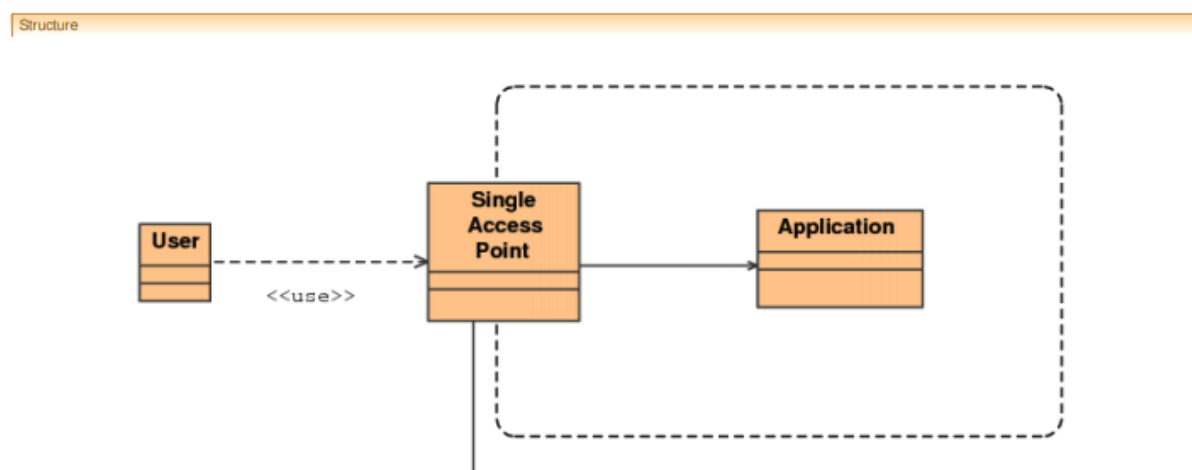
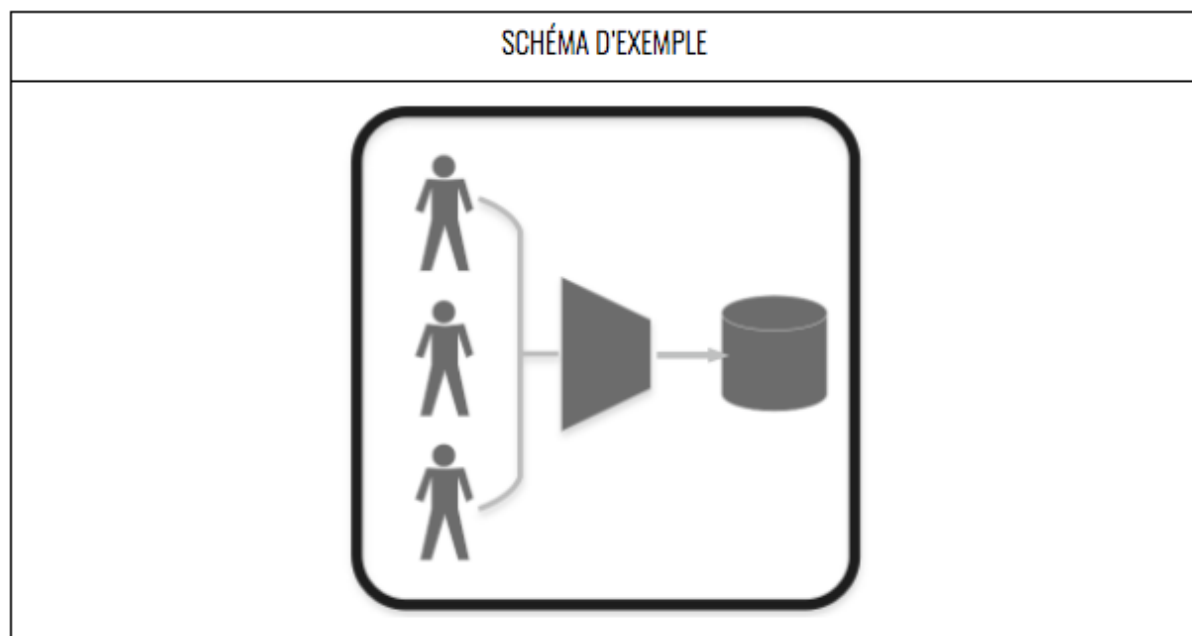


Figure 1: Single Access Point structure.

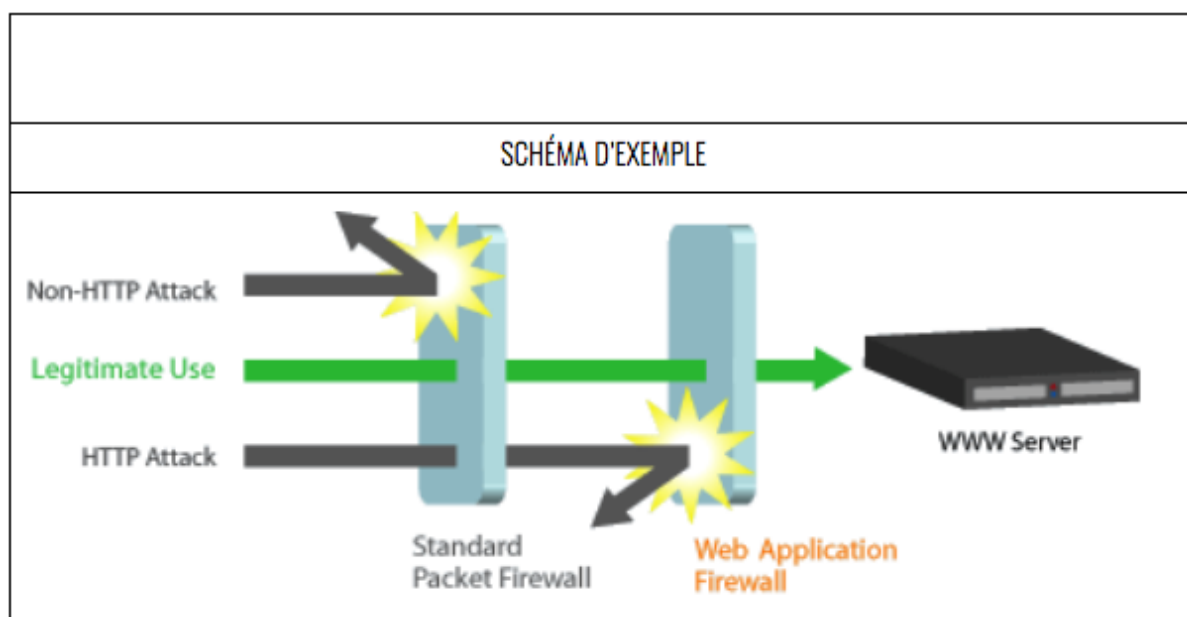


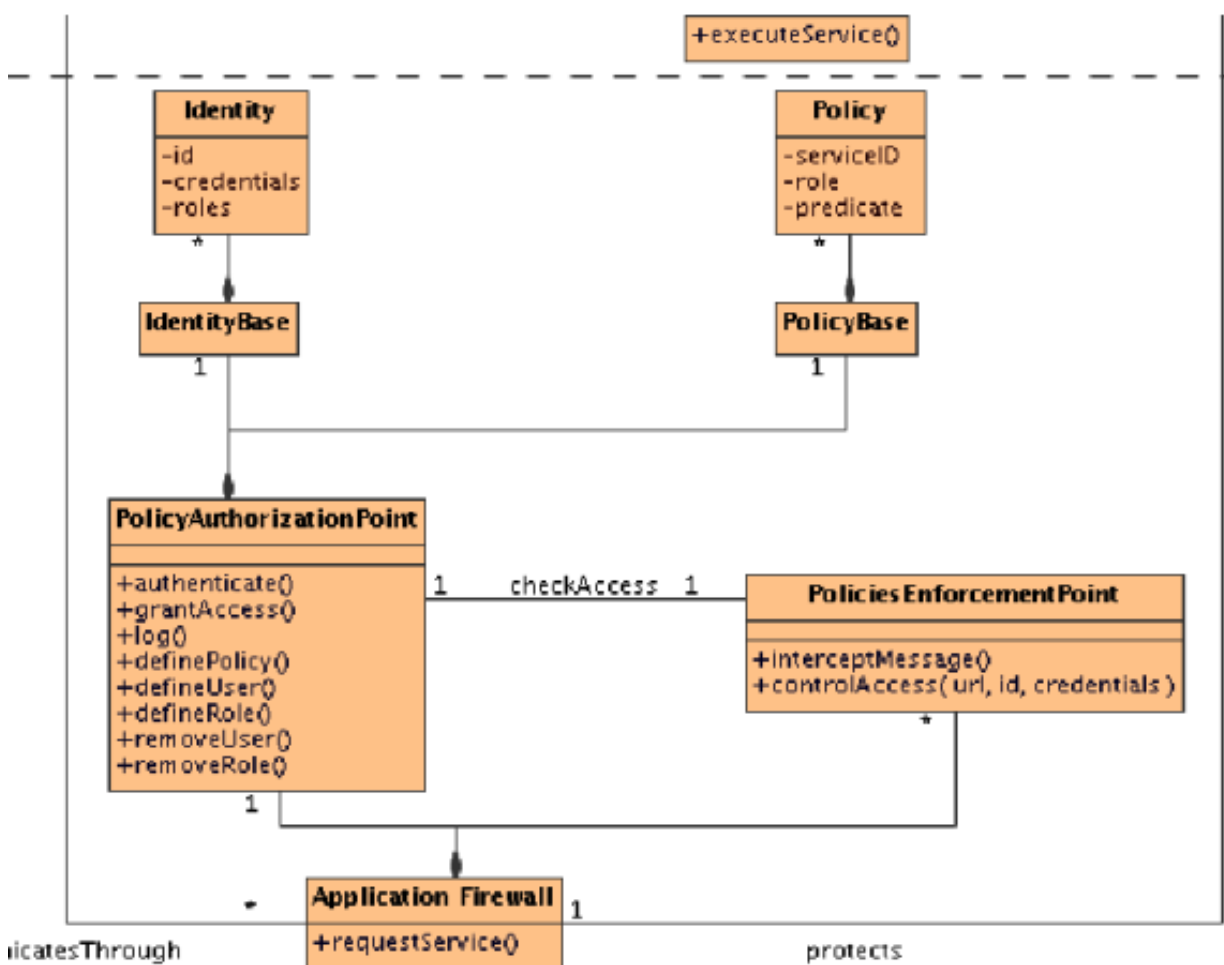
3.4 Patron "Application firewall"

C'est un patron de conception permettant de contrôler la circulation des données en fonction de normes prédéfinies.

But : Pare-feu contrôlant les entrées/sorties de l'application selon des règles établies (politiques) et les utilisateurs.

Utilité : Un tel système est intéressant pour notre IOT, car il permet de filtrer les entrées/sorties d'informations et de différencier plusieurs rôles d'utilisateurs. De plus, il protège bien des diverses attaques envers l'IOT rendant le système plus protégé des pirates. En revanche, son implémentation est lourde et irréalisable sur un si petit modèle de carte (nombre d'instances trop grand).





3.5 Patron "Security context"

C'est un patron de conception permettant d'encapsuler les attributs de sécurité ainsi que les données relatant d'exécutions particulières.

But : Ajoute une couche d'abstraction sur un objet dont on souhaite obtenir un accès régulier mais protégé. Le pattern prend tout son sens dans une situation où un objet interagit avec plusieurs autres pour former le contexte de l'application. La couche d'abstraction décrite plus haut, permet de stocker des données relatives à l'identification de l'utilisateur telles des timers d'expiration pour la session ou des clés d'identification.

Utilité : Permet l'encapsulation de l'objet dans un autre qui contient les clés cryptographique, les timers et tous autres attributs relatifs à la sécurité de l'objet. Il permet l'instauration d'un point d'accès qui contrôle les autorisations et protège les données sensibles. Ce security pattern est un dérivé du Proxy pattern en conception. Il permet à une classe de se faire passer pour une autre en ajoutant une protection à celui-ci. On appelle cette encapsulation le "contexte de sécurité". Ce pattern assez léger, peut permettre une authentification par token d'accès aux objets dont WebServer.ino a la responsabilité et permettre d'ajouter une couche d'abstraction au modèle et obtenir une API OpenThermostat propre et réutilisable. Il pourrait donc être facilement implémenter en rendant les constructeurs et méthodes de chaque classe privés et la couche d'abstraction friend de chacune d'elles.

