UQAC
Université du Québec
à Chicoutimi

Abstraction de l'autorisation/authentification du code applicatif d'un système web

par Jean-Luc Cyr

Mémoire présenté à l'Université du Québec à Chicoutimi en vue de l'obtention du grade de Maître ès sciences appliquées (M.Sc.A.)

Québec, Canada

RÉSUMÉ

L'implémentation de mesures de sécurité lors du développement d'une application Web comporte des lacunes. En effet, ces fonctions, lorsqu'elles existent, sont souvent développées à chaque nouveau projet avec une approche manquant de structure et de rigueur. Cela induit une forte probabilité d'erreurs. Le point de départ du présent travail a ainsi été de trouver une réponse à la question suivante : « Est-il possible d'extraire les mécanismes d'autorisation et d'authentification du code applicatif d'un système Web? ». Si oui, est-il possible de développer un composant gérant la sécurité qui serait réutilisable et dont le fonctionnement pourrait être validé par des experts dans le domaine ?

La prépondérance de l'utilisation du serveur Apache pour livrer le contenu des applications Web sur le marché a orienté la mise en œuvre d'une solution à ce problème. Il a été établi que développer et rendre disponible un module complémentaire pouvant gérer l'authentification d'un utilisateur et ses autorisations pour accéder à différentes ressources applicatives était réalisable. Une documentation pertinente pour sa mise en place à l'intérieur d'un projet de développement applicatif a été rédigée.

Dans un premier temps, comme présenté dans tous les gabarits de gestion de la sécurité informationnelle, il a fallu identifier et classifier les différentes ressources contenues dans les logiciels. Dans un second temps, il a fallu définir les utilisateurs qui interagissent avec ces ressources et les permissions associées aux relations entre les utilisateurs et les ressources. Plusieurs méthodologies sont proposées pour définir ces relations.

Sur ces bases, un modèle a été défini et appliqué pour conserver l'identification des ressources, des utilisateurs et les relations entre les deux. Ces informations ont servi de base pour développer un module gérant l'accès aux systèmes et la protection des ressources. La gestion de ces informations a été documentée et déléguée aux concepteurs d'application ce qui permettra d'avoir un couplage fort entre l'application, son domaine et la gestion des accès. L'application du modèle, assurée par le module développé, doit être réalisée par des spécialistes en sécurité.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES FRAGMENTS DE CODES SOURCES

LISTE DES ACRONYMES ET SIGLES

AB : Apache Benchmark

ACEI : Autorité Canadienne des Enregistrement Internet

ACL : Access Control List

ACM : Association for Computing Machinery

AD : Active Directory

AJAJ : Asynchronous Javascript And JSON

AJAX : Asynchronous Javascript And Xml

APR : Apache Portable Runtime

ASP : Active Server Page

BD : Base de Données

BNC : Banque Nationale du Canada

BSD : Berkeley Software Distribution

CERN : Centre Européen de Recherche Nucléaire

CERT : Computer Emergency Response Team

CMS : Content Management System

CRUD : Create Read Update Delete

DB : DataBase

FTP: File Transfert Protocol

GDPR : General Data Protection Regulation

GID : Group Identifier

GUI : Graphical User Interface

HTML : HyperText Markup Language

HTTP(S) : Hyper Text Transfert Protocol (Secured)

IE : Internet Explorer (navigateur Web de Microsoft)

IEEE : Institute of Electrical and Electronics Engineers

IETF : Internet Engineering Task Force

IIS : Internet Information Server (serveur Web de Microsoft)

IISP : Institute of Information Security Professionnals

ISO : International Standard Organisation

ITIL : Information Technology Infrastructure Library

JSON : JavaScript Object Notation

KVM : Kernel-based Virtual Machine

LAMP : Linux Apache MySQL PHP

LDAP : Lightweight Directory Access Protocol

MIT: Massachusetts Institute of Technology

NIS(+) : Network Information Service

NIST : National Institute of Standards and Technology

NTLM : NT Lan Manager (Microsoft)

OCTAVE : Operationnaly Critical Threat Asset and Vulnerability Evaluation

OS : Operating System

OWASP : Open Web Application Security Project

PAM : Pluggable Authentification Module

PCI-DSS : Payment Card Industry – Data Security Standard

PHP: Langage PHP (Hypertext Preprocessor)

PIN : Personnal Identification Number

POSIX: Portable Operating System Interface [for Unix] / a family of IEEE standards

REST: REpresentational State Transfert

RFC: Request For Comment

SEO : Seach Engine Optimisation

SGBD: Système de Gestion de Base de Données

SOAP : Simple Object Access Protocol

SPNEGO: Simple and Protected GSS-API Negotiation

SSH : Secured Shell

SSL: Secure Sockets Layer

SSO : Single Sign-On

TLS: Transport Layer Security

UID : User IDentifier

UQAC : Université du Québec à Chicoutimi

URI : Uniform Ressource Identifier

URL : Uniform Ressource Locatorchmod

WAF : Web Application Firewall

WSDL : Web Service Description Language

WWW : World Wide Web

XML : eXtensible Markup Language

XML-RPC : eXtensible Markup Language - Remote Procedure Call protocol

REMERCIEMENTS

Mes remerciements vont à tous ceux qui m'ont supporté de près ou de loin durant ces deux années où j'ai tenté de faire cohabiter le travail, les études, la vie de famille et sociale. Une mention particulière à Daniel et Claude qui m'ont initié à l'informatique en jeune âge et aidé à développer cette passion.

Je désire souligner l'apport particulier de : Filou qui m'a aidé à maintenir l'équilibre entre travail et activité physique; Daniel, mon directeur de maîtrise, qui m'a aidé à garder le cap; ma conjointe, mes enfants et ma famille pour leur patience et leur support sur plusieurs plans; mes clients et partenaires pour leur compréhension de la réorganisation de mon horaire.

Sans oublier l'Université du Québec à Chicoutimi pour le soutien financier au travers du programme de formation long des chargé(e)s de cours et ADN Informatique pour la fourniture des ressources matérielles et logiciel.

AVANT-PROPOS

Avec la combinaison de mes formations; en enseignement, en théologie/éthique, en ingénierie, en recherche; et 20 années d'expériences professionnelles, j'ai développé une préoccupation grandissante à propos de la sécurité de l'information. Cela me pousse à vouloir éduquer la population sur les bonnes pratiques dans ce domaine en les aidant à mieux comprendre les principes de base de la sécurité et développer leur esprit critique sur les services et produits qui leur sont offerts. Ma participation au domaine des technologies de l'information m'amène également à vouloir guider les intervenants vers les meilleures pratiques du domaine et les aider à incorporer une réflexion éthique dans leurs activités professionnelles.

CHAPITRE 1

Introduction

Tel que nous l'entendons dans les médias régulièrement, les systèmes reliés à l'Internet sont régulièrement victimes d'attaques. L'OWASP [1] (Open Web Application Security Project) a rédigé un palmarès [2] (top 10) des risques de sécurité les plus critiques retrouvés dans les applications Web. Mon expérience professionnelle dans le développement de système Web confirme également plusieurs lacunes. Les demandes en gestion de la sécurité de certains clients de l'entreprise ADN Informatique corroborent également ces faits.

Plusieurs solutions ont été proposées et ont été mises en place pour essayer de sécuriser les systèmes, mais chacune apporte ses problématiques particulières. Afin de tenter de trouver une solution plus générale et plus robuste, une analyse des différentes approches existante a permis de constater que bien des problèmes étaient liés au fait que la sécurité était gérée à l'intérieur du code applicatif. Ainsi, s'il était possible d'abstraire la sécurité du code applicatif pour la rendre meilleure tout en conservant un fort niveau de cohésion entre l'application et la sécurité, cette dernière s'en trouverait grandement améliorée. Le présent projet s'attaque donc à ce problème en proposant une solution concrète dont la performance peut être facilement établie.

Suite à une étude des technologies les plus répandues, il a été décidé de développer une solution utilisant les « plate-formes » suivantes : Linux, Apache, MySQL et PHP. Le code de sécurité a été développé en langage C sous forme de module Apache. Le système de gestion utilisera les outils système et pourra utiliser une interface Web en PHP ou autre.

Ce travail démontre que cette approche est réalisable au niveau de la performance et intégrable à des systèmes existants avec certains développements supplémentaires.

1.1 Présentation de la problématique

La problématique sera abordée sous différents aspects. En premier sous l'angle du visible, soit des failles de sécurité rapportées, par la suite avec une vision des applications et de leurs développements, des constations de l'OWASP et pour conclure un sommaire des solutions existantes. La littérature scientifique propose plusieurs méthodes pour la sécurisation de systèmes informatiques, cependant aucun article n'a été répertorié à propos de l'extraction des mécanismes de sécurité des applications Web pour en faire la mise en oeuvre au niveau du serveur.

1.1.1 Failles de sécurité dans les médias

Les médias nous rapportent régulièrement des attaques de pirates contre des propriétés Internet (sites Web, bases de données, services, sites FTP …). Ces attaques touchent de multiples sites, de plus ou moins grande envergure, appartenant à toutes sortes d'entreprises. En voici quelques exemples (quelques articles sont présentés en Annexe I).

- Mai 2018 – La Banque de Montréal et « Simplii » auraient été piratées (Le Devoir)
- Mars 2018 – Une filiale de « Under Armour » victime d'une cyberattaque (Journal de Québec)
- Février 2018 – Une agence [ontarienne] de transport victime d'une cyberattaque (Radio-Canada)
- Janvier 2018 – Renseignements personnels piratés chez Bell (Le Devoir)
- Novembre 2017 – « VerticalScope » confirme que six sites ont été piratés (La presse canadienne)

Naturellement, les cas rapportés ne sont que la pointe de l'iceberg. En effet, avant d'apparaître dans les médias, ces attaques doivent avoir été détectées et le premier réflexe par la suite n'est généralement pas de les révéler au grand public. C'est pourquoi de plus en plus les organismes réglementaires tentent de légiférer pour forcer la divulgation de ces évènements afin d'en minimiser les conséquences et de permettre aux victimes d'en être informées afin qu'elles puissent agir de manière éclairée.

Un rapport de sondage [3] sur la cybersécurité de l'ACEI [4] (Autorité Canadienne des Enregistrements Internet) et commandité par Akamai datant de l'automne 2018 révèle que 40% des répondants ont fait face à une cyberattaque dans les 12 derniers mois. De plus le rapport mentionne que 45% des répondants ont pris comme mesure suite à une attaque l'installation de nouveaux logiciels pour augmenter la sécurité.

1.1.2 Applications Web

Tout d'abord, il est important de définir ce qu'est une application Web. Il s'agit d'un logiciel applicatif, développé en mode client-serveur, dans lequel l'interface entre le client et le serveur est le protocole HTTP(S) (Hyper Text Tranfert Protocol). L'interface utilisateur (GUI – Graphical User Interface) est généralement mise en oeuvre sur le poste client grâce à un navigateur (Internet Explorer, Edge, Safari, Chrome, Firefox, …) qui communique par l'intermédiaire du protocole HTTP(S) avec un serveur (Apache, Nginx, GlashFish, JBoss, …). Ce dernier interface et exécute le code applicatif.

Dans un nombre restreint de cas, le code applicatif implémente lui-même le protocole HTTP(S) directement ou par l'intermédiaire de bibliothèques de code (flask/werkzeug, django, etc.). Ces exceptions ne seront pas prises en considération dans le présent travail.

1.1.3 Développement de produits

Un grand nombre de sites Web sont développés à l'aide de logiciels tels que Wordpress [5], Joomlà [6], Drupal [7]; alors que d'autres sont réalisés sous forme de projets autonomes en divers langages tels que PHP [8], Ruby [9], Python [10], Java ou autre. La plupart des systèmes offerts tant commerciaux que provenant du domaine des logiciels libres « Open Source » ont tous une implémentation distincte pour la gestion de la sécurité.

Un point commun entre la plupart de ces systèmes est qu'ils sont, pour la majorité d'entre eux, mis en place derrière un serveur Web. Un faible nombre utilise plutôt des bibliothèques de code tel que Flask [11] et Django [12] en Python, à même l'application, pour implémenter le protocole HTTP(S) plutôt que d'utiliser un serveur Web tel qu'Apache HTTP.

La gestion de la sécurité hors du code applicatif pourrait donc être réalisée à l'intérieur du serveur Web de manière à pouvoir être utilisée par la majorité de ces systèmes.

1.1.4 Systèmes utilisés

Parmi les serveurs Web les plus répandus, l'on retrouve

- Apache [13]
- Nginx [14]
- IIS [15]

Ces serveurs offrent plusieurs modules pour la gestion de la sécurité tels que *mod_auth_basic* pour le serveur Apache. Cependant, mon expérience sur le marché du travail m'a permis de constater que peu de ces fonctions sont couramment utilisées dans des projets. Seule l'utilisation de *mod_auth_basic* conjointement avec des fichiers locaux *« .htaccess »* est parfois utilisée en cours de développement.

1.1.5 Constatations de l'OWASP

L'OWASP [1] est une organisation internationale sans but lucratif qui a pour but de faire la promotion de la sécurité dans les applications Web. Elle a diffusé à trois reprises un palmarès (top 10) des éléments de sécurité les plus critiques dans ces applications.

Ces trois documents diffusés respectivement en 2010, 2013 et 2017 identifient tous des lacunes importantes au niveau de la gestion des autorisations et authentifications pour accéder aux ressources des systèmes.

Tableau 1.1 - Palamares des failles de sécurité de l'OWASP

La version 2013 présente les points suivants :	La version 2017 quant à elle présente
A1 Injection	A1 Injection
A2 Cross-Site Scripting (XSS)	**A2 Broken Authentication**
A3 Broken Authentication and Session Management	A3 Sensitive Data Exposure
A4 Insecure Direct Object References	A4 XML External Entities (XXE)
A5 Cross-Site Request Forgery (CSRF)	**A5 Broken Access Control**
A6 Security Misconfiguration	**A6 Security Misconfiguration**
A7 Insecure Cryptographic Storage	A7 Cross-Site Scripting (XSS)
A8 Failure to Restrict URL Access	A8 Insecure Deserialization
A9 Insufficient Transport Layer Protection	A9 Using Components with Known Vulnerabilities
A10 Unvalidated Redirects and Forwards	A10 Insufficient Logging & Monitoring

Parmi les éléments présentés au Tableau 1.1, trois points (A3, A4 et A8 2013 et A2, A5 et A6 2017) sont directement reliés à la gestion des authentifications et l'autorisation. Le détail de chacun pour 2017 se retrouve en Annexe II.

1.1.6 Systèmes existants

Plusieurs systèmes existants tentent d'élever le niveau de sécurité des serveurs Web. Plusieurs d'entre eux ajoutent une couche supplémentaire indépendante devant les systèmes.

- Les appareils WAF (Web Application Firewall) (Imperva [16], BigIP F5 [17], Cisco [18])
- Les logiciels WAF (ModSecurity [19])
- Les passerelles de procuration « proxy » (Squid [20], Varnish [21])
- Les modules serveur (Apache : mod_auth_xxxx, nginx : naxsi, modsecurity)
- Les bibliothèques de code « library » (Flask avec Basic Auth [22])

Dans certains cas, notamment les WAF, le but premier est de détecter les attaques (Owasp 2013 A1, A2, A5, 2017 A1, A7) contre les systèmes plutôt que d'en gérer l'accès. Les passerelles de procuration « proxys » peuvent servir à la gestion des permissions, cependant ce n'est pas leur fonction première.

De plus, ces solutions sont souvent complètement découplées du code applicatif donc fonctionnent de manière totalement indépendante. La plupart engendrent des coûts supplémentaires en matériel ou logiciel de même qu'une équipe spécialisée pour leur mise en place et leur gestion.

La solution proposée devra limiter les coûts engendrés, par l'utilisation de logiciels à code ouvert, être opérable par les programmeurs ou les opérateurs déjà en place et offrir un couplage fort avec le code applicatif.

1.2 Objectifs du projet

L'objectif de ce projet est de développer et diffuser des outils pour permettre une meilleure gestion de l'autorisation et l'authentification dans les projets de développement Web. Ces outils pouvant être mis en œuvre par l'entreprise ADN Informatique pour les besoins de ses clients.

La base de ces outils doit reposer sur l'utilisation des techniques courantes dans les domaines traditionnels (non-Web), l'abstraction de la couche de sécurité du code applicatif tout en conservant un couplage fort entre l'application pour la gestion et les outils pour l'exécution des règles de sécurité.

L'approche proposée consiste à développer un module Apache incorporant plusieurs fonctionnalités des modules existant (autorisation, authentification, journalisation) en utilisant les fonctions du système d'exploitation lorsque possible, conjointement avec une base de

données permettant aux applications d'en faire la gestion (interface entre les fonctions de sécurité du serveur Apache et la souplesse de gestion par l'application).

Afin de déterminer les pratiques les meilleures et les plus répandues pour la gestion de la sécurité, de même que les habitudes des programmeurs, un parallèle sera établi avec le développement des applications de bureau et des applications mobiles.

1.3 Méthodologie

Le projet a été réalisé en suivant les étapes suivantes :

1. Revue de l'état de l'art
2. Choix d'un système cible
3. Mise en place d'un environnement de développement et de test
4. Développement et documentation du système (pistes de solutions et analyse)
5. Réalisation du module et outils (conception et développement/programmation)
6. Statistiques de performance et diagnostic (validation)
7. Application à un environnement de production (validation)
8. Livraison publique

La méthodologie de développement utilisée aux étapes 4 et 5 suivra les processus définis par la norme ISO29110 [23, 24, 25]. Cette norme vise à standardiser les pratiques de développement logiciel dans les petites entreprises. Elle encadre les processus de gestion de projet et de développement. Les extraits pertinents de la documentation normalisée sont inclus tout au long de ce travail. La liste des documents rédigés est disponible en Annexe III.

Le développement du module est fait sur la base des modèles et modules existants, avec l'utilisation d'une base de données MySQL pour permettre le couplage avec le code applicatif. La documentation rédigée s'aligne avec un environnement de production similaire « LAMP » (

Linux Apache Mysql Php) basée sur une distribution standard de Linux, incluant Apache, Mysql et PHP tel qu'indique son nom.

1.4 Échéancier

Le Tableau 1.2 présente la répartition de la charge de travail du projet réparti sur 2 ans.

Tableau 1.2 - Échéancier du projet

Éléments	**Nb Heures**	**Session de travail**
1. État de l'art	300h	2017-2 à 2017-3
2. Élaboration pistes de solutions	300h	2017-3 à 2018-2
3. Développement	200h	2018-2 à 2018-3
4. Validation	150h	2018-3 à 2019-1
5. Documentation (utilisation)	150h	2018-3 à 2019-1
6. Rédaction et présentation	200h	2018-2 à 2019-1
Total	**1200h**	

1.5 Contribution

Les concepts exposés dans ce projet pour la sécurisation des sites Web en découplant la gestion des authentifications et autorisations du code applicatif tout en réutilisant des composants existants apportent une vision différente de la structure des applications Web.

Afin de faciliter la mise en œuvre de cette vision, les documents de ce projet ainsi que les codes sources seront distribués gratuitement afin de permettre leur réutilisation comme base pour de nouveaux projets.

CHAPITRE 2

État de l'art

Le premier point à valider est la prédominance de certains systèmes sur le marché pour le développement et l'exploitation de site Web. Sans connaître ces tendances, il serait impossible de cibler correctement l'impact que pourrait avoir la disponibilité d'outils pour la sécurisation de ces sites. (section 2.1)

Par la suite, la revue des standards (section 2.2), l'analyse des technologies existantes et le ciblage des éléments pertinents au projet ont été effectués. Pour ce faire, une revue des projets « Open Source » sur le sujet ainsi que de certains codes sources a été réalisée. En complément d'information, les pages de manuels électroniques d'Unix (« Unix Man Page ») ont été utilisées pour obtenir de l'information à jour sur l'utilisation des bibliothèques, fichiers et programmes du système d'exploitation.

Pour terminer, une expérimentation avec différents logiciels sous différents systèmes d'exploitation et différents appareils a été conduite afin d'observer les mécanismes utilisés pour l'authentification et les autorisations. (sections : 2.3, 2.4 et 2.5)

Systèmes d'exploitation : Unix (Linux), macOS (Unix/BSD), Windows 10, Android, iOS.

Plateformes : Ordinateurs (PC Dell, iMac, MacBookPro), Tablettes (Dell Android/Dell Windows, Apple iPadPro), Téléphones (Apple iPhone).

Logiciels : Microsoft Office, Google Docs, DropBox, Logiciels bancaires (BNC, Tangerine, Desjardins), Paiement ApplePay, ainsi que diverses applications selon les plateformes et système d'exploitation.

2.1 Systèmes utilisés sur le marché

Afin de bien cibler la pertinence du projet et d'établir le marché cible, une étude du taux d'utilisation des principaux produits a été effectuée. L'étude a porté sur les systèmes d'exploitation, les serveurs Web, les langages de développement ainsi que les principaux gestionnaires de contenu Web.

2.1.1 Les serveurs HTTP

Comme l'on peut le constater sur le rapport statistique de BuiltWith [26] (Figure 2.2), le serveur Web le plus rependu en juin 2018 est Apache avec 31 % suivi de Nginx avec 27 % et IIS avec 15%. Les données de w3techs [27] (Figure 2.1) corroborent ces informations attribuant 46% du marché à Apache suivi de Nginx avec 39% en août 2018.

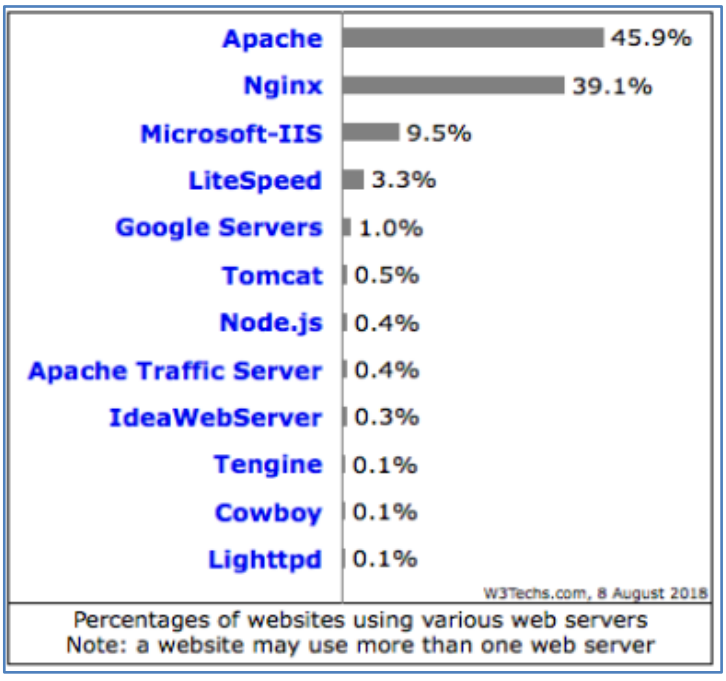


Figure 2.1 - Statistiques d'utilisation des serveurs Web (w3techs)

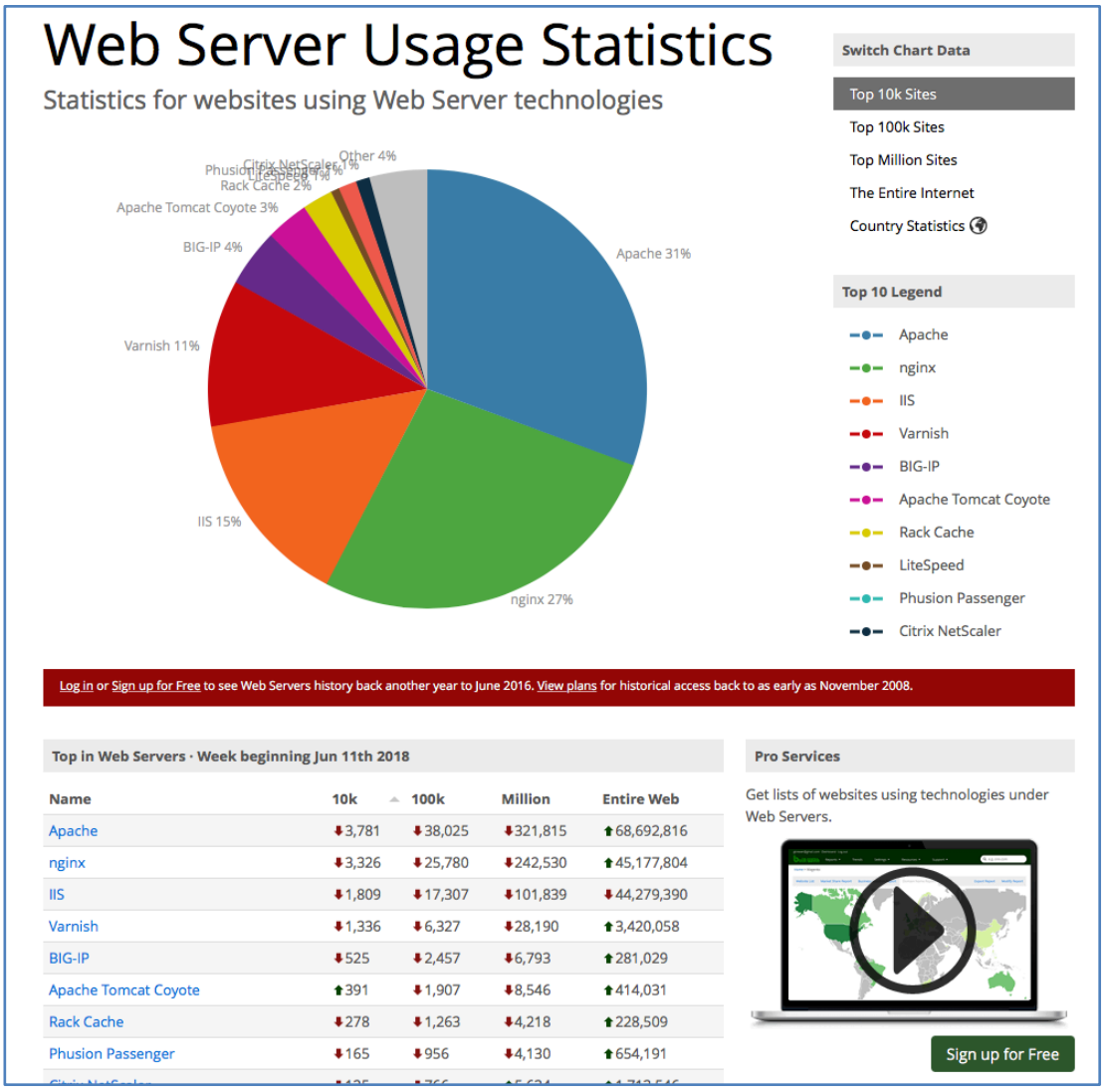


Figure 2.2 - Statistiques d'utilisation des serveurs Web (BuiltWith)

On peut donc envisager le développement d'un module accompagné d'un guide visant la sécurisation des sites Web, utilisant Apache comme serveur, ayant un marché potentiel intéressant.

Advenant son adoption, la seconde plateforme cible pourrait alors être Nginx avec la part de marché suivante. IIS (Internet Information Server - Microsoft) pourrait être également intéressant, mais notons que les deux premiers ont l'avantage d'être des projets « Open Source ».

2.1.2 Les langages de programmation

De même, le langage de programmation le plus répandu selon BuiltWith [28] (Figure 2.4) est PHP avec 54%, suivi par les technologies ASP.NET avec 37%. Les données de w3techs [29] (Figure 2.3) présentent un grand écart, attribuant 84% à PHP, mais la prédominance de ce langage dans les deux cas est notable. Le langage PHP étant fortement lié à l'utilisation d'un interpréteur et un serveur HTTP, particulièrement mod_php sous Apache, tend à confirmer l'exactitude des statistiques d'utilisation des serveurs.

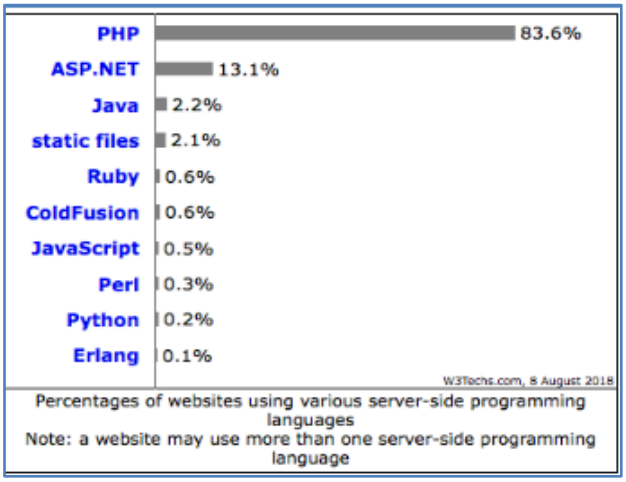


Figure 2.3 - Statistiques d'utilisation des langages (w3techs)

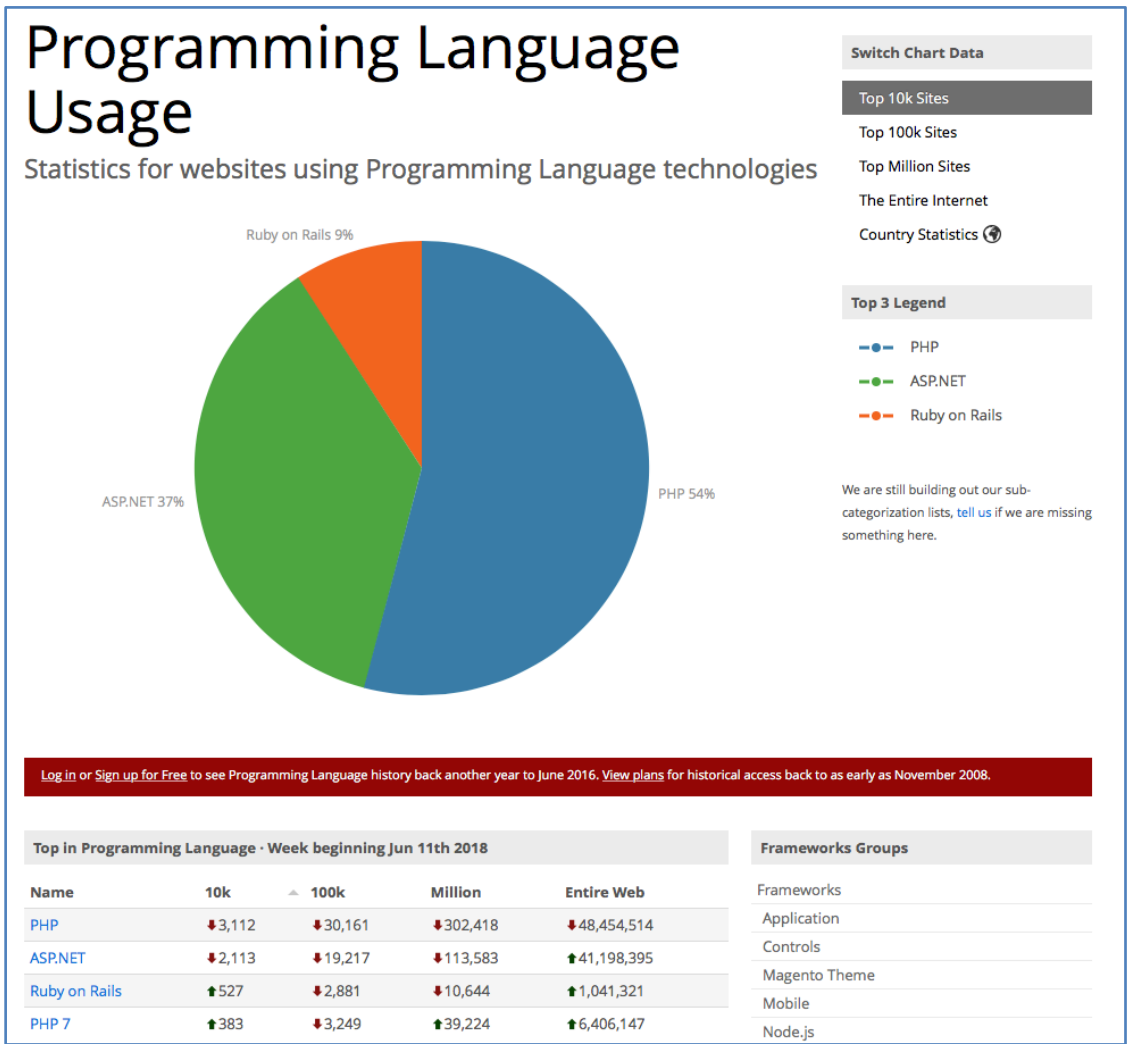


Figure 2.4 - Statistiques d'utilisation des langages (BuiltWith)

2.1.3 Les gestionnaires de contenu

Toujours dans la même lignée, si l'on regarde l'utilisation d'applications Web disponibles pour la mise en ligne de sites, le joueur majeur est Wordpress avec 36% du marché selon BuiltWith [30] (Figure 2.5) et 32% selon w3techs [31] (Figure 2.6). Le fait que celui-ci est en langage PHP et généralement utilisé sous Apache appuie les statistiques précédentes.

La combinaison de ces trois éléments nous donne donc une indication sur la pertinence d'avoir comme première cible la sécurisation de système derrière le serveur Apache.

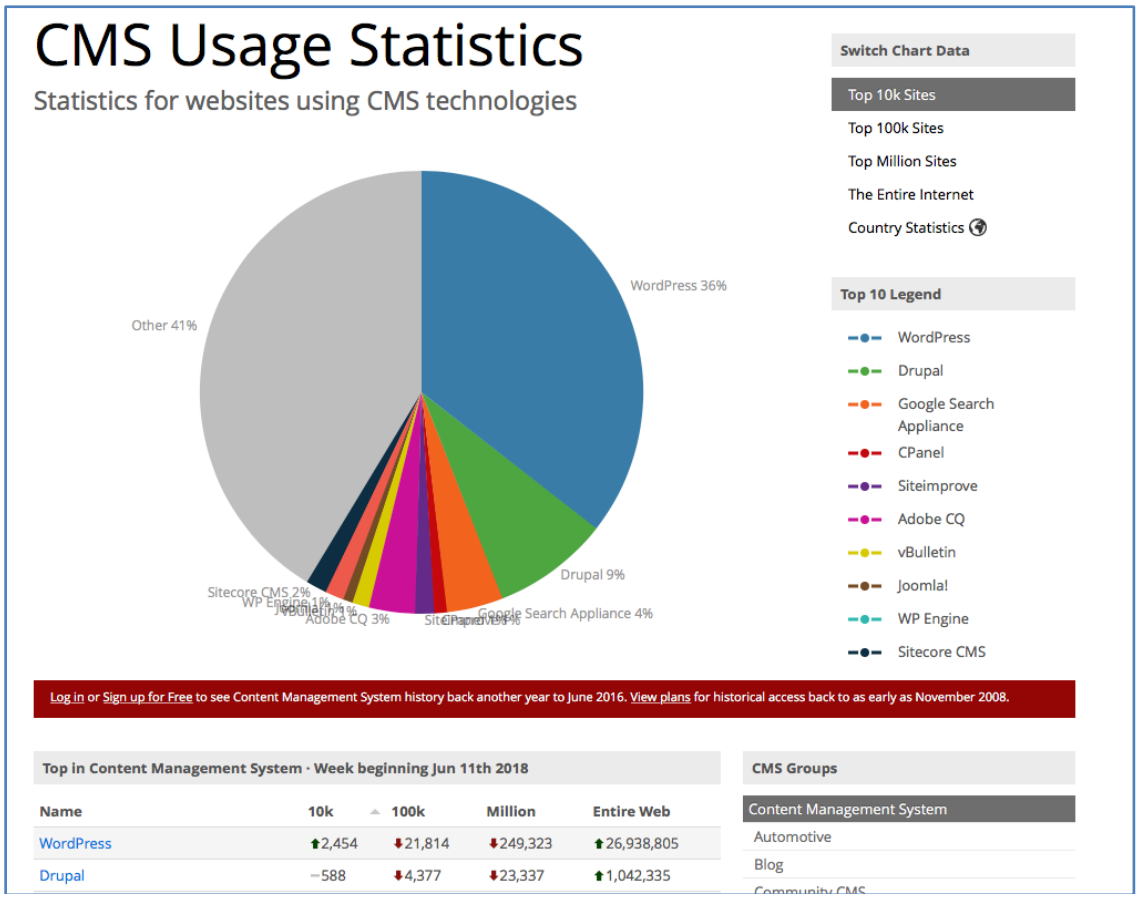


Figure 2.5 - Statistiques d'utilisation des CMS (BuiltWith)

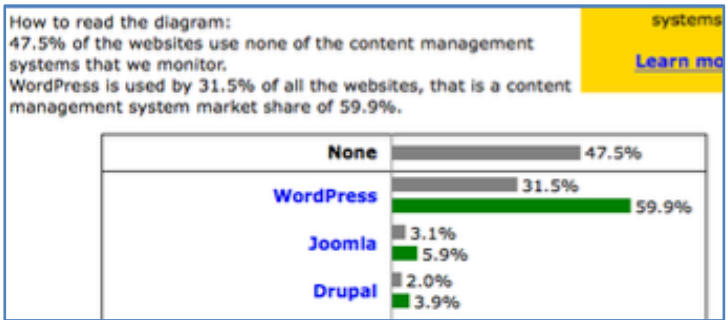


Figure 2.6 - Statistiques d'utilisation des CMS (w3techs)

2.1.4 Les systèmes d'exploitation

Selon w3Techs [32] (Figure 2.7), le système d'exploitation le plus utilisé pour opérer des sites Web est « Unix ». Il serait donc pertinent de s'attarder à ce système d'exploitation en premier.

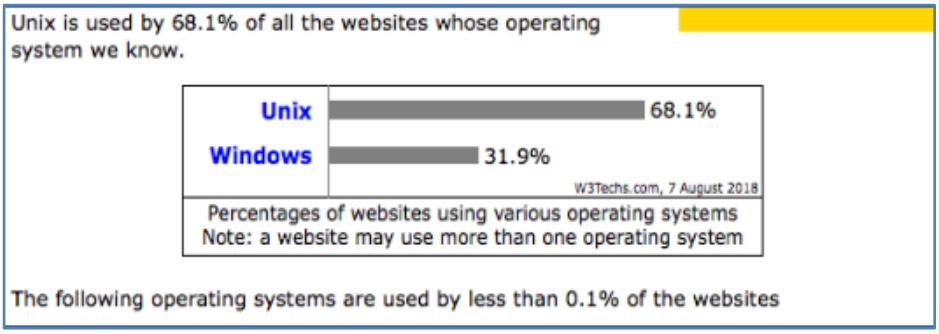


Figure 2.7 - Statistiques d'utilisation des systèmes exploitation (w3techs)

2.2 Les standards en sécurité informatique

Plusieurs standards existent concernant la sécurité des systèmes informatiques en général. Différents organismes travaillent à mettre en place des outils pour la sécurisation des données, particulièrement en relation avec la vie privée des utilisateurs. Une des dernières grandes initiatives mises en place est la GDPR (General Data Protection Regulation) européenne [33] qui tente de responsabiliser les propriétaires de systèmes où transigent, sont traitées ou sont entreposées des informations personnelles.

Cependant, l'adoption des normes est souvent laissée à la bonne volonté des parties. Dans quelques cas, la législation donne des directives ou impose des obligations, mais même dans ces cas, les moyens pour la mise en œuvre diffèrent souvent d'un système à l'autre.

Un autre élément s'ajoutant à l'équation est le fait que le domaine des technologies de l'information n'appartient à aucun groupe de professionnels. On retrouve donc des gens de tout acabit participant aux développements des systèmes avec des connaissances qui leur

sont propres et ne suivant pas nécessairement les bonnes pratiques de sécurité établies par la majorité de l'industrie.

Parmi les acteurs de l'industrie élaborant diverses normes l'on retrouve notamment:

- PCI-DSS (payment card industry – data security standard) V3.2.1 Mai 2018 [34]
  Norme de l'industrie des cartes de paiement pour la sécurité données relatives aux propriétaires de cartes de paiement.

- ISO 27000 Information security management systems [35]
  Gestion de manière responsable et sécuritaire des informations (financières, propriété intellectuelle, informations personnelles des employés ou informations sensibles fournies par des tiers) en possession de l'entreprise.

- ITIL (Information Technology Infrastructure Library) Security management [36]
  Un ensemble de pratiques pour la gestion des services d'information en alignement avec les besoins d'affaire des entreprises.
  Particulièrement, la section sur le design des services et la sous-section sur la gestion de la sécurité.
  2- ITIL Service Design: *turns the service strategy into a plan for delivering the business objectives.*
  2.7- Security management
  Cette section est basée sur la norme ISO27001 mettant l'emphase sur la confidentialité, l'intégrité et la disponibilité des informations reliées aux qualités ou au but d'authenticité, de responsabilité, de non-répudiation et de fiabilité des données.

- NIST (National Institute of Standards and Technology / U.S. Department of Commerce) [37]
  Organisme qui a plusieurs publications sur la sécurité informatique. La série de publication « NIST Special Publication 800 / 1800 series » qui traite de sécurité informatique et cyber sécurité.

- IETF (Internet Engineering Task Force) [38]
  Organisme pour le développement des standards de l'Internet dans un processus ouvert. Il est composé de nombreux groupes de travail sur différents sujets, dont l'IAB (Internet Architecture Board) traitant de l'architecture de l'Internet.

- IISP (Institute of Information Security Professionals) [39]
  Organisation indépendante à but non lucratif pour l'avancement du professionnalisme dans le domaine de la sécurité de l'information.

- CERT (Computer Emergency Response Team) [40]
  Division de l'institut d'ingénierie logiciel de l'université Carnegie Mellon. Ils maintiennent notamment une base de données des vulnérabilités des systèmes informatiques [41]. Ils ont également développé la méthode OCTAVE (Operationally Critical Threat Asset and Vulnerability Evaluation) pour le classement des actifs et la détermination des risques associés.

De même, chaque institution de grande envergure, proposent généralement des guides d'opérations, des règlements ou autres documents traitants des responsabilités relativement

à l'obtention, au traitement et à l'entreposage de données critiques. Par exemple, on retrouve ces directives pour le gouvernement du Québec sur le site su secrétariat du Conseil du trésor à l'adresse [42]: https://www.tresor.gouv.qc.ca/ .

Toutes ces normes présentent des éléments essentiels à prendre en compte lors de transaction, entreposage ou manipulations de données critiques. Cependant, aucune ne présente formellement un modèle ou une méthode à mettre en place afin d'obtenir le résultat escompté.

À la lecture des différents documents standards, on remarque que la plupart donnent des lignes directrices sur les comportements ou les méthodologies à adopter. Cependant, aucune implémentation ou guide de mise en œuvre n'est proposée, à l'exception de la norme ISO29110 pour la gestion qualité du développement logiciel qui propose des gabarits prêts à l'utilisation. Cependant, cette norme s'applique plus au processus de développement qu'à la sécurisation des applications.

2.3 Les systèmes Web

Les systèmes Web sont composés de différents éléments échangeant de l'information. Les principaux sont le navigateur et le serveur. Leurs interactions sont régies par différents protocoles. Voici une description de ces principaux éléments.

2.3.1 Fonctionnement global

Le fonctionnement du Web est une interaction entre un client (navigateur) et un serveur HTTP (Figure 2.8). L'environnement du client est cloisonné (« sandboxed ») à l'intérieur du navigateur (Safari, Chrome, Firefox, Edge, Internet Explorer …) et n'a pas accès facilement aux fonctionnalités du système d'exploitation. Il est cependant possible que des exceptions de sécurité puissent être acceptées par l'utilisateur pour l'utilisation des ressources locales (par exemple par des scripts Java, JavaScript ou ActiveX contenus dans un document).

Le serveur, quant à lui, peut accéder à certaines fonctionnalités du système d'exploitation sur lequel il est installé tout en respectant les restrictions de privilèges. C'est-à-dire qu'il est limité aux droits accordés à l'utilisateur qui l'exécute.

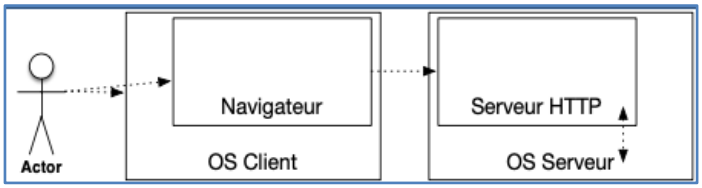


Figure 2.8 - Composants de l'environnement Web client-serveur

Les connexions entre le navigateur et le serveur sont sans état « stateless », c'est-à-dire qu'il n'y a pas de continuité entre les échanges. Ainsi, par défaut, il n'y a pas de session de travail. Le protocole HTTP n'utilise pas non plus, par défaut, de connexions persistantes « keep-alive » pour la réalisation de plusieurs transferts d'objets [43]. Tous les échanges sont initiés par le client. Cependant, cela tend à changer avec l'adoption de HTTP/2 (Figure 2.9) et l'utilisation du Web pour des connexions bidirectionnelles (push, websockets server side events).

Il est à noter que la composition et le rendu des données sont faits par le logiciel client et nécessitent plusieurs connexions ou échanges avec le serveur afin d'obtenir tous les éléments nécessaires.

Lors de l'exécution d'une requête typique, l'utilisateur saisit l'adresse désirée dans la barre de son navigateur. Le logiciel client transmet alors une requête au serveur en utilisant le protocole HTTP(S). Cette requête est composée d'un entête comprenant plusieurs éléments dont la méthode à utiliser (exemple : GET) et la ressource désirée (exemple : fichier.txt). Le serveur qui reçoit cette requête valide l'existence de la ressource, effectue l'action appropriée selon la méthode et en retourne le contenu résultant accompagné des entêtes nécessaires au respect du protocole.

La documentation sur les systèmes Web, l'utilisation de serveurs HTTP(S) et le développement d'applications autant serveur que client abonde sous toutes sortes de formes; livres papiers ou numériques, vidéos, sites Internet, formations en classe ou en ligne.

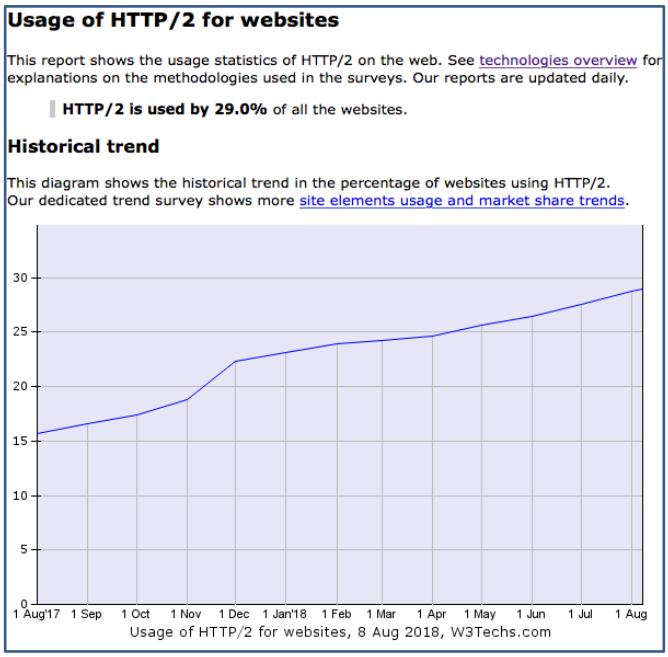


Figure 2.9 - Statistiques du taux d'adoption de http/2 (w3techs)

2.3.2 Le serveur Apache HTTP

Le serveur Apache HTTP [13] de « Apache Foundation [44] » est un logiciel à code ouvert. La version la plus récente est de la série 2.4 (en date de ce document, 2.4.34). L'architecture de ce serveur a grandement évolué entre la version 1 et 2. La version 2 propose une architecture modulaire plus complète, facilitant ainsi l'intégration de nouvelles fonctionnalités. Ce serveur est disponible pour plusieurs systèmes d'exploitation, ceux des familles « Unix », « Windows » et « Mac » (OS X maintenant un Unix basé sur BSD). Le livre « The Apache Modules Book – Application Development with Apache » [45] présente bien le logiciel et la manière de développer des modules complémentaires.

Le logiciel est conçu de manière modulaire et utilise des bibliothèques de composantes portables d'un système d'exploitation à un autre (Apache Portable Runtime ou APR). De plus, la chaine de traitement de l'information permet aisément d'introduire des traitements supplémentaires. Les Figure 2.10 à 2.12 extraites du livre cité ci-dessus illustrent ces faits. La Figure 2.10 illustre la structure de développement basée sur un cœur accompagné de différents modules tout basés sur l'APR. La Figure 2.11 montre le traitement linéaire des requêtes dans Apache 1.x de la réception jusqu'à la journalisation et la transmission. La Figure 2.12 décrit le même fonctionnement pour Apache 2.x avec l'ajout d'un second axe pour la composition du document à retourner.

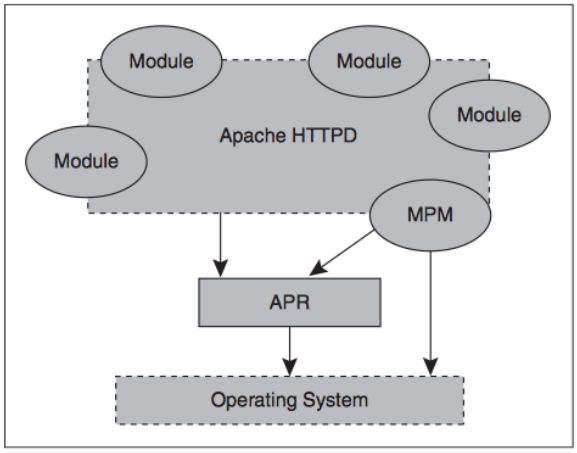


Figure 2.10 - Architecture globale d'Apache - P.22

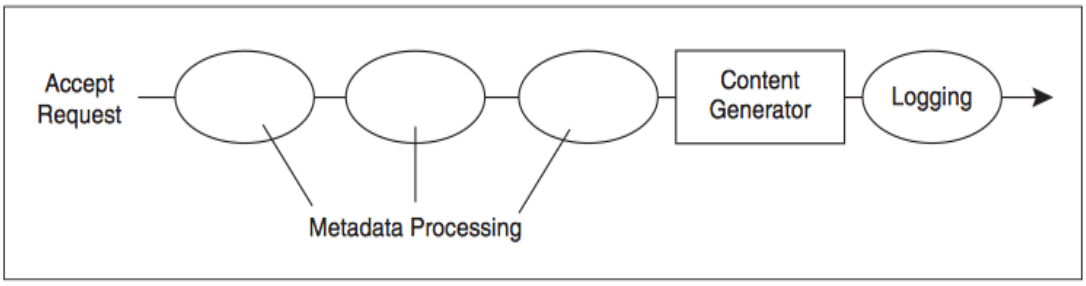


Figure 2.11 - Processus de traitement des requêtes d'Apache (P.44)

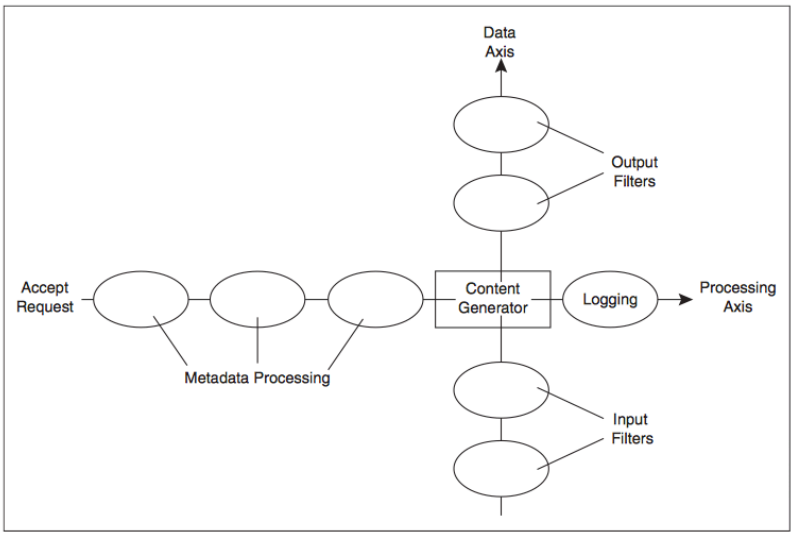


Figure 2.12 - Chaîne de filtre de contenu d'Apache 2.x (P.47)

Ces informations confirment donc qu'Apache HTTP, particulièrement en version 2.x, est un bon candidat pour la mise en place d'un module gérant la sécurité, autant en amont du traitement « Metadata Processing » qu'en aval de la génération de contenu « Output Filters ». De plus, plusieurs modules sont déjà disponibles et pourraient servir de base au projet [46].

La documentation sur le serveur Apache HTTP est relativement commune sous toutes sortes de formes; livres papiers ou numériques, vidéos, sites Internet, formations en classe ou en ligne. Cependant si l'on restreint le champ de recherche au développement proprement dit du serveur, on retrouve principalement deux ouvrages « **Writing Apache Modules with Perl and C: The Apache API and mod_perl »** pour les versions antérieures et « **The Apache Modules Book: Application Development with Apache »** pour la version 2.x. Les autres ressources se limitent à de la documentation en ligne.

2.3.3 Le protocole HTTP(S)

Le protocole de communication entre les logiciels clients et les serveurs est HTTP(S). Présentement, deux versions sont très répandues soit 1.1 et 2.0. Ce protocole est défini par des spécifications de l'IETF sous forme de RFC (Request For Comment) [47]. Ces

spécifications sont en constante révision sous la supervision du IETF HTTP Working Group [48]. Plus précisément, la version 1.1 est définie dans le RFC 2616 et la version 2 dans le RFC 7540.

Ces deux versions du protocole HTTP définissent des éléments pour la sécurisation des applications Web. La plus couramment utilisée est « basic authentication » basée sur la transmission par le serveur d'une réponse avec le statut « 401 » et retour du client avec un entête d'autorisation « Authorization ». Plusieurs autres éléments tels que des entêtes supplémentaires, des statuts de réponse, l'utilisation de témoins et la transmission de paramètres peuvent également servir à implémenter une couche d'autorisation et authentification.

Il faut également mentionner que le protocole HTTP, en version 1 ou 2 peut être utilisé de manière sécurisée en version HTTP**S**. Le protocole est alors utilisé au travers de TLS/SSL plutôt que directement en texte clair. En effet, TLS (Transport Layer Security) et SSL (Secure Sockets Layer) sont deux protocoles cryptographiques qui fournissent à la fois l'authentification et le cryptage des données [49]. Le standard TLS créé par l'IETF est le successeur de SSL créé par Netscape [50]. Cette information est importante, car elle influence la sécurité lors de l'utilisation de « basic authentication » du protocole. Déjà en 2015 [51], des discussions avaient lieu pour déclarer la désuétude de HTTP au profit de HTTPS pour des communications plus sécuritaires à la grandeur de la toile.

Les RFC de l'IETF étant la documentation officielle de ces protocoles, la recherche documentaire sur ce sujet n'a pas été plus exhaustive.

Il est également important de mentionner l'ouvrage « Architectural Styles and the Design of Network-based Software Architectures » [52] par Roy Fielding dont l'extrait suivant démontre la pertinence pour l'utilisation de ces normes.

« This dissertation defines a framework for understanding software architecture via architectural styles and demonstrates how styles can be used to guide the architectural design of network-based application software. A survey of architectural styles for network-based applications is used to classify styles according to the architectural properties they induce on an architecture for distributed hypermedia. I then introduce the Representational State Transfer (REST) architectural style and describe **how REST has been used to guide the design and development of the architecture for the modern Web**.»

En effet, M. Fielding est un membre actif du développement des normes définissant le protocole HTTP et cet ouvrage permet de bien saisir les principes sous-jacents à son utilisation dans le développement d'application Web.

La définition du protocole quant à elle décrit deux éléments importants que l'on voit expliqués dans la documentation sur sa sémantique.

Le RFC7231 « **Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content »**

HTTP/1.1 request and response semantics in terms of the architecture defined in [RFC7230]. »

À la section 2 qui définit ce qu'est une ressource, l'explication nous donne deux informations qui seront des hypothèses de bases pour l'élaboration de notre solution. La première, que chaque ressource est identifiée par un URI et la seconde mentionne la préséance des méthodes du protocole sur le contenu de ces dernières.

« The target of an HTTP request is called a "resource". HTTP does not limit the nature of a resource; it merely defines an interface that might be used to interact with resources. Each resource is identified by a Uniform Resource Identifier (URI), as described in Section 2.7 of [RFC7230]. »

« One design goal of HTTP is to separate resource identification from request semantics, which is made possible by vesting the request semantics in the request method (Section 4) and a few request-modifying header fields (Section 5). If there is a conflict between the method semantics and any semantic implied by the URI

itself, as described in Section 4.2.1, the method semantics take precedence. »

De plus, la section 4.1 (voir Tableau 2.1) présente la sémantique des méthodes du protocole dont voici un extrait :

Tableau 2.1 - Sémantique des méthodes du protocole HTTP

Method	Description	Sec.
GET	Transfer a current representation of the target and header section.	4.3.1
HEAD	Same as GET, but only transfer the status line and header section.	4.3.2
POST	Perform resource-specific processing on the request payload.	4.3.3
PUT	Replace all current representations of the target resource with the request payload.	4.3.4
DELETE	Remove all current representations of the target resource.	4.3.5
CONNECT	Establish a tunnel to the server identified by the target resource.	4.3.6
OPTIONS	Describe the communication options for the target resource.	4.3.7
TRACE	Perform a message loop-back test along the path to the target resource.	4.3.8

2.3.4 Les documents

Les objets transigés entre le client et le serveur peuvent être de plusieurs natures (texte, image, son …). Lors d'une navigation traditionnelle, le premier objet transféré est normalement une page Web (document HTML- HyperText Markup Language). Cet objet contient les informations du document ainsi que instructions pour le rendu tel que l'inclusion de feuilles de style, d'image ou autres. Chacun des objets peut être déjà existant sur le serveur ou généré à la demande et est identifié par une adresse unique nommée URL (Uniform Ressource Location). Plus de détails sont présentés à la section 2.5.5.

2.4 Gestion de l'authentification

L'identification d'un utilisateur peut être faite de diverses manières. La plus conventionnelle et répandue est l'utilisation d'un identifiant conjointement avec un mot de passe. Des systèmes de clé publique/privée et de certificats sont également répandus. De plus en plus de nouvelles technologies émergent telles que l'utilisation de données biométriques. Chaque type d'application implémente des mécanismes qui lui sont propres tel que décrit dans les sous-

sections suivantes. Les méthodes les plus répandues et déjà en place dans les environnements ciblés seront privilégies dans ce travail.

2.4.1 Système d'exploitation Unix

L'ouverture d'une session de travail dans l'environnement Unix, incluant l'environnement Mac, se fait traditionnellement à partir d'une liste d'utilisateurs et de mots de passe contenus dans les fichiers /etc./passwd et /etc./shadow. Ces fichiers appartiennent généralement au super-utilisateur (« root »). Dans un environnement réseau, il est fréquent d'utiliser une base d'utilisateur centralisée avec un autre mécanisme tel que NIS (Network Information Service), NIS+, LDAP (Lightweight Directory Access Protocol) ou autre. Dans ce cas, les logiciels de base utilisés pour l'authentification (connexion par exemple) détectent, dans la configuration du système, la stratégie à utiliser pour obtenir les informations nécessaires souvent par le contenu du fichier /etc./nsswitch. Un autre mécanisme utilisé sous Linux permettant la configuration du système d'exploitation et de plusieurs logiciels au bon système de permission est le système PAM (Pluggable Autentification Module) [53].

Éléments du fichier /etc./passwd – Informations sur les comptes utilisateurs

Chaque ligne de ce fichier comprend :

- L'identifiant texte de l'utilisateur (login name)
- Un mot de passe optionnel crypté
- L'identifiant numérique de l'utilisateur (UID – User IDentifier)
- L'identifiant numérique du groupe de base de l'utilisateur (GID – Group Identifier)
- Le nom complet ou un commentaire
- Le répertoire de base de l'utilisateur
- Le nom (optionnel) de l'interpréteur de commande lancé lors d'une connexion

Éléments du fichier /etc./shadow – Fichier optionnel de mot de passe crypté

- L'identifiant texte de l'utilisateur (doit exister dans /etc./passwd)
- Un mot de passe crypté ou un identifiant spécial tel que ! ou *

- La date du dernier changement de mot de passe
- L'âge minimal du mot de passe (nb de jour avant la possibilité de le)
- L'âge maximal du mot de passe (nb de jour avant la nécessité de le changer)
- Le nombre de jours avant l'âge maximal l'utilisateur reçoit un avertissement
- Le nombre de jours après l'âge maximal avant l'expiration du compte
- La date d'expiration du compte
- Un champ réservé pour usage futur

Éléments du fichier /etc./group – Fichier des groupes supplémentaires

- Le nom du groupe
- Un mot de passe de groupe
- L'identifiant numérique du groupe (GID)
- La liste des identifiants texte d'utilisateurs membre du groupe

Le contenu de ces fichiers peut être manipulé par différents utilitaires en ligne de commande tels que : useradd, usermod, passwd, groupadd, groupmod. Des fonctions et structures de données sont également disponibles pour leur manipulation en langage C. Un mécanisme de clé publique/privé est aussi utilisé dans certains cas, notamment avec les communications via SSH, par la mise en place d'une clé publique dans le répertoire racine de l'utilisateur dans le sous-répertoire .ssh dans un fichier nommé authorized_keys.

2.4.2 Système d'exploitation Windows

Le mécanisme de gestion des sessions de travail Windows n'a pas été exploré, car il est hors de la portée du présent projet. Cependant, les technologies telles que l'Active Directory (AD) pourraient peut-être apporter des éléments intéressants.

2.4.3 Applications natives

Les applications natives (applications de bureau Windows, OS X ou Unix) utilisent soit les informations de la session de travail du système d'exploitation, soit leur propre système d'identification propriétaire local ou avec un serveur de licence local ou distant.

2.4.4 Applications mobiles

Les applications mobiles (iOS, Android …) se comparent aux applications natives de bureau. Il faut cependant distinguer ici deux catégories d'application, celles qui fonctionnent principalement en mode local et celles qui sont des clientes de services Internet. Plusieurs applications délèguent au système d'exploitation (iOS, Android, Windows) l'identification de l'utilisateur. Ces systèmes utilisent des PINs (Personnal Identification Number), une association d'usager (adresse courriel) et mot de passe, empreinte digitale ou faciale. En complément, certaines applications demandent une identification supplémentaire gérée localement ou, souvent dans la seconde catégorie, sur un serveur distant (identification sur un service Web, un service nuagique ou autre). Cela s'explique par le fait que les applications qui sont en fait des « clients Web » personnalisées se basent sur les technologies des applications Web.

2.4.5 Applications Web

L'authentification de l'utilisateur peut se faire de plusieurs manières. Le protocole HTTP inclut des fonctionnalités de base supportées par la plupart des navigateurs qui reposent sur la transmission d'un code de statut 401 et de l'entête WWW-Authenticate tel qu'illustré sur « http gallery » [54]. Les mécanismes les plus utilisés sont: Basic, Digest et NTLM (NT Lan Manager). D'autres mécanismes plus complets et complexes peuvent également être utilisés tels que: utilisation de certificats et utilisation d'un jeton (« token ») (en témoins ou en paramètre) conjointement avec une authentification directe sur le serveur ou délégation à un tiers (SSO ou Single Sign-on [55]).

Presque toutes ces méthodes nécessitent une interaction avec l'utilisateur pour fournir une ou des informations (utilisateur, mot de passe, certificat …) pour confirmer son identité, due aux restrictions de l'environnement du navigateur, qui rend inaccessible l'identité de l'utilisateur sur le système pour une transmission directe sauf dans des cas comme l'utilisation de NTLM/Kerberos sur des postes Windows avec des sessions reliées à un domaine. Kerberos [56] est un protocole pour l'échange de manière sécurisée des authentifications. Une implémentation gratuite est fournie par le MIT (Massachusetts Institute of Technology). Il est utilisé pour l'authentification dans plusieurs systèmes. Le serveur Apache fournit d'ailleurs un module pour en faire l'utilisation pour la validation des informations reçues via la Basic Authentification avec un serveur Kerberos [57]. L'utilisation de Kerberos directement entre le client (Navigateur) et le serveur implique une modification des paramètres du navigateur [58] et l'ajout de composants et/ou la modification de configuration du serveur pour supporter le mécanisme SPNEGO (Simple and Protected GSS-API Negotiation) [59, 60].

Outre quelques applications Web basées sur les technologies Microsoft (Windows, Internet Explorer, Internet Information Server) utilisant NTLM et une authentification basée sur la session de travail sur le poste client, la majorité des systèmes utilisent une identification basée sur la saisie d'informations (identifiant/mot de passe) ou l'utilisation de certificats d'identité.

Les mécanismes prévus dans le protocole HTTP sont mis à contribution pour réaliser ces opérations. La validation des informations par le serveur varie grandement d'un système à un autre. Plusieurs systèmes ont une base de données interne, d'autres utilisent des informations système locales ou en réseau. D'autres délèguent même ces fonctionnalités en partie à des tiers (utilisation de SSO par exemple).

2.5 La gestion des autorisations

La gestion des autorisations d'actions sur une ressource accordées à un utilisateur peut être faite de diverses manières. Chaque type d'application implémente des mécanismes qui lui sont propres. Les méthodes utilisés par les systèmes de fichiers et déjà en place dans les

environnements ciblés, tels que le système de permission POSIX ou les ACL (Access Control List), seront privilégies dans ce travail. D'autres méthodes de gestion telles que RBAC (Role-Based Access Control) [61], Bell-LaPadula [62], Biba Model [63] apportent un niveau de granularité et des principes d'assignations différents et intéressants. Cependant ces modèles nécessitent l'utilisation de composants supplémentaires que nous éviterons.

2.5.1 Système d'exploitation Unix

Les ressources protégées dans les systèmes Unix sont de plusieurs types, mémoire, fichiers, processus, ports réseau ... Les ressources se rapprochant le plus de celles des systèmes Web sont les fichiers emmagasinés sur le disque. Nous allons donc nous attarder au mécanisme de permission du système de fichiers.

Le système de base de permission d'Unix, souvent nommé le système de permission POSIX (Portable Operating System Interface), date des premiers jours de la création de ce système d'exploitation. Bien que des systèmes plus complets d'ACL soient maintenant disponibles sous Linux [64], le système de base est toujours le plus utilisé. Les permissions permettent le contrôle sur la possibilité de lire, modifier, naviguer et exécuter le contenu du système de fichiers. Une fois l'utilisateur identifié lors de l'ouverture de session, les identifiants assignés comprennent un identifiant numérique (UID) pour le compte, un identifiant numérique (GID) pour le groupe de base. L'utilisateur peut également être membre de groupes supplémentaires (Liste de groupes dans /etc./groups).

Ces différents identifiants numériques sont utilisés pour déterminer les droits associés avec les fichiers. En effet, chaque fichier possède également deux identifiants numériques (Owner et Group) qui correspondent à l'utilisateur et au groupe propriétaire. Adjoint à ces informations se retrouve un masque (« mask ») de permission pour le propriétaire, le groupe et le public. Les attributs possibles sont la lecture, l'écriture et l'exécution (R,W,X). L'on retrouve donc trois

triplets d'information RWX RWX RWX représentant respectivement les permissions du propriétaire, du groupe et du public. (voir Tableau 2.2 et Tableau 2.3) [65]

Ces permissions s'appliquent à tous les objets du système de fichiers tel que les fichiers réguliers, les liens symboliques ou lourds, les répertoires, etc. De légères variations sur leur interprétation apparaissent selon le type d'objet.

Tableau 2.2 - Permissions de base système de fichiers Unix

Permissions textuelles	**Valeurs numériques (octal)**	**Description**
r	1	Permission en lecture
w	2	Permission en écriture
x	4	Permission d'exécution

Ces valeurs sont utilisées en combinaison pour définir les permissions pour trois catégories d'acteurs soit : le propriétaire, le groupe propriétaire et le public. Quelques exemples de permissions possibles sont présentés au Tableau 2.3.

Tableau 2.3 - Exemple d'attribution de permission de fichiers Unix

Permissions textuelles	Permissions numériques (octal)	Description
- r w x r w x r w x	0777	Lecture, écriture et exécution permises pour le propriétaire, le groupe et le public
- r w x r - - r - -	0744	Lecture, écriture et exécution pour le propriétaire, Lecture seulement pour le groupe et le public

Les permissions associées aux objets sur le système de fichiers peuvent être manipulées par différents utilitaires en ligne de commande telles que : ls, chown, chmod. Dans le cas des systèmes implémentant les ACL, des fonctions telles que getfacl, setfacl peuvent être utilisées. Des fonctions et structures de données sont également disponibles pour leur manipulation en langage C.

2.5.2 Système d'exploitation Windows

La protection des ressources dans les systèmes Windows, comme pour les systèmes Unix, est de plusieurs types : mémoire, fichiers, processus, ports réseau ... Nous allons également ici regarder le mécanisme de permission du système de fichiers.

Le système de permission de fichiers sous Windows est plus élaboré que celui de base sous Unix. Il est basé sur une liste de contrôle (ACL) développée plus récemment que le système de base de permission d'Unix. Il est principalement utilisé sous Windows ou avec les systèmes qui supportent les systèmes de fichiers NTFS ou ReFS. Ce système permet une granularité plus grande que le système de permission traditionnel d'Unix.

2.5.3 Applications natives

Les applications natives (applications de bureau Windows, OS X ou Unix) utilisent généralement les informations de la session de travail en cours ou une identification demandée au lancement de l'application pour restreindre l'accès aux ressources de l'application par l'utilisateur. Cette gestion des ressources, dans la majorité des cas, est interne à l'application et ne peut être analysée sans l'accès au code source. Dans certains cas, les applications développées dans un modèle n-tiers, délèguent la gestion des données à un moteur de données (SGBD) ou au système de permission du système d'exploitation.

2.5.4 Applications mobiles

Les applications mobiles (iOS, Android …) se comparent aux applications natives de bureau. Plusieurs applications délèguent au système d'exploitation (iOS, Android, Windows) la gestion de la session de travail. Basées sur cette session du système ou de l'application, les restrictions d'accès aux ressources sont appliquées. Comme pour l'authentification, le comportement des applications mobiles diffère selon qu'ils soient complètement autonomes ou qu'ils accèdent à des ressources en ligne. Dans le second cas, la gestion des autorisations est souvent déléguée sur serveur en ligne fournissant les données basées sur l'identifiant fourni, soit un comportement similaire aux applications Web.

2.5.5 Applications Web

Avant de définir plus en détail la méthode pour accorder des autorisations (ou privilèges d'accès) il faut d'abord déterminer clairement les ressources (« assets ») et les actions possibles sur ces dernières. La composition des ressources Web associées à des URL était de prime abord une relation 1 :1, dans la version 0.9 de HTTP qui n'avait que la méthode GET et retournait le contenu de fichiers présents sur le système. C'est ce qui est toujours utilisé pour les ressources statiques dans les systèmes modernes. Cependant, déjà à ce point, par la nature multimédia des documents de base transmis qui sont généralement en HTML (hyper text markup language), chacun des documents nécessite une étape de composition par le client pour être affiché et cette étape comprend la récupération d'autres éléments (images par exemple) associés à d'autres URL qui sont des ressources distinctes (voir exemple de composition d'une page Web à la Figure 2.14).

Une partie du contenu des sites Web est conservé sous forme de ressources statiques (fichiers) et délivré directement par le serveur Web. Pour ces ressources, la correspondance entre l'URI et le contenu est généralement de 1 à 1. Dans ce cas, la gestion des autorisations se résume souvent à l'application des règles d'attribution des permissions du système de fichiers du système d'exploitation ou des règles de bases dans la configuration du serveur Web à l'authentification préalablement établie. Les principales ressources auxquelles cette stratégie s'applique sont les images, feuilles de style et code JavaScript.

Les autorisations appliquées à la partie dynamique du contenu, souvent généré par du code exécutable, reposent également sur la session de travail établie en phase d'autorisation. Cependant, le même code étant utilisé pour délivrer le contenu de plusieurs ressources, la relation entre l'URI et le code est quant à elle de N à 1. L'utilisateur est encore une fois identifié sur la base de l'étape de l'authentification, mais la ressource ne correspondant pas à un fichier propre sur le système d'exploitation, les droits ne peuvent être gérés par le système de fichiers.

La configuration du serveur Web prévoit certains mécanismes à cette fin, mais ils sont peu utilisés sur de gros systèmes en production.

Tel que présenté, avec la génération dynamique du contenu (par programmation PHP ou autre) la relation 1 : 1 entre l'URL et le document a changé. Un ensemble d'URL est maintenant associé à un « programme » qui génère dynamiquement le contenu, souvent à partir d'une base de données. Ce contenu étant généralement du HTML (Hyper Text Markup Language) et incluant la nécessité d'accéder à d'autres ressources statiques ou également dynamiques.

Pour complexifier davantage, en plus de la composition à partir du HTML (récupération des images et autres), l'avènement du « Web 2.0 » avec une grande utilisation de JavaScript pour faire cette composition (allant même jusqu'à avoir des applications clients s'exécutant dans le navigateur) rend le document de base « obsolète ». L'utilisation d'AJAX (Asychronous JavaScript And XML)[1] pour la modification dynamique de ce document de base, tel qu'illustré à la Figure 2.13, rend donc inutilisables des permissions de base accordées au document HTML.

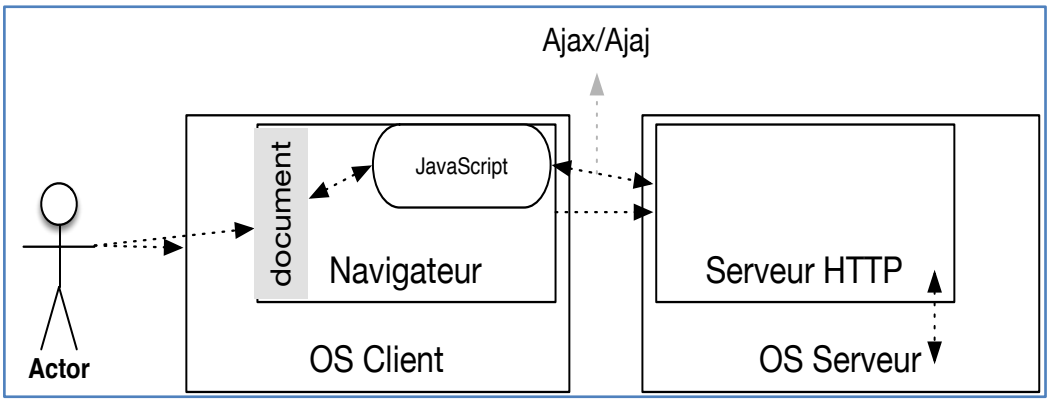


Figure 2.13 – Flux d'information avec une requête Ajax/Ajaj

[1] Aussi quelques fois remplacé par AJAX (Asynchronous JavaScript And Json)

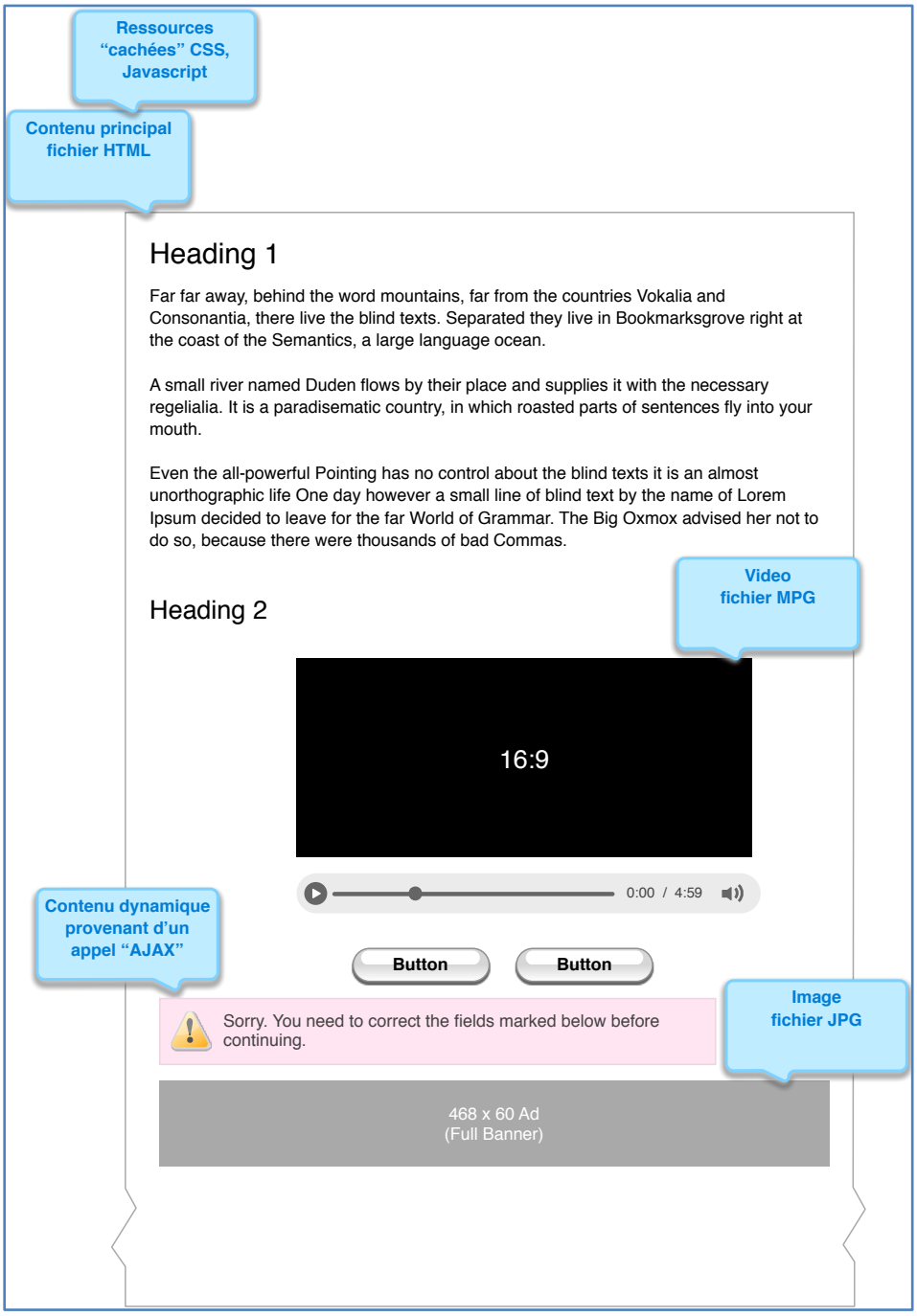


Figure 2.14 - Composition pour le rendu d'une page Web

Un point demeure, chaque ressource accédée (page de base, objets inclus tels que les images, scripts inclus pour exécution, objets récupérés via Ajax) est récupérée sur le serveur,

a un identifiant URI qui lui est propre et peut être soumise indépendamment à une gestion des privilèges.

Les pages Web étant des objets « composites », la gestion des autorisations sur la page ne permet pas un contrôle strict des informations qui s'y trouvent. En effet, l'inclusion d'autres objets pour son rendu peut mener à la diffusion ou au blocage d'information de manière inapproprié. Le suivi de la relation entre une donnée (image, texte ou autre) et son emplacement dans une ou plusieurs pages est difficile. Une méthode pour y arriver sera présentée dans les développements futurs au CHAPITRE 6.

L'utilisation de REST (REpresentational State Transfert), présenté en 2.3.3, peut être un point de départ pour définir l'identification des ressources et éventuellement les accès. Cependant, nous ne pouvons pas établir avec certitude que les systèmes applicatifs ou les services Web adhèrent aux principes de REST pour délivrer les objets.

Tableau 2.4 – Méthodes HTTP pour représenter les actions CRUD et SQL

HTTP Verb	CRUD Action	Équivalent SQL
POST	Create	Insert
GET	Read	Select
PUT	Update	Update
DELETE	Delete	Delete

L'utilisation de REST attribue l'utilisation des verbes GET, POST, PUT et DELETE du protocole HTTP aux actions possibles sur les objets, soit CRUD ou CREATE, READ, UPDATE, DELETE tel qu'illustré dans le Tableau 2.4, cela conjointement avec l'affirmation que les URLs identifient les ressources de manière précise et unique. Cette attribution unique des URLs combinés avec des définitions des actions possibles nous donne une piste intéressante pour définir les privilèges qui peuvent être accordés aux utilisateurs pour interagir avec les différents objets. La gestion des permissions par des systèmes Web, dans la plupart des cas, est gérée de façon rudimentaire par le serveur Web. Afin d'ajouter cette fonctionnalité, une expression régulière est souvent utilisée dans la configuration pour passer l'accès à un ou plusieurs URLs

(dans un programme faisant partie de l'application qui gère l'accès aux ressources) sans la participation du serveur HTTP. L'application gère donc, à l'interne, une liste de contrôle (ACL) pour les objets/URL qu'il délivre.

Pour définir les autorisations, des protocoles tels que OAuth 2.0 [66] peuvent être utilisés. Ce dernier spécifiquement gère des autorisations d'accès, mais sa granularité n'est pas à la base suffisante pour atteindre l'objectif de ce projet.

CHAPITRE 3

Pistes de solution

Afin de déterminer la première piste à explorer afin de solutionner la problématique identifiée, certaines hypothèses ont été posées pour en déterminer les paramètres principaux.

3.1 Hypothèses

Les pistes de solutions proposées se basent sur les hypothèses suivantes :

Hypothèse 1: Les statistiques du marché (section 2.1) demeureront stables pendant un certain temps, donc l'utilisation d'Apache est un choix judicieux

Hypothèse 2: L'implémentation du module sera indépendante de la version du protocole HTTP 1.1/2.0.

Hypothèse 3: Les logiciels et services Web à protéger adhèrent aux principes de « REST » (section 2.3.3) et ne sont pas des implémentations de XML-RPC (eXtensible Markup Language - Remote Procedure Call protocol) ou SOAP (Simple Object Access Protocol).

i. C'est-à-dire qu'ils utilisent les méthodes HTTP pour définir l'action.
ii. Et des URI uniques pour définir les ressources.

Hypothèse 4: Les logiciels peuvent interagir avec le système de gestion des authentifications du système d'exploitation (directement ou via une autre application Web telle que Webmin)

Hypothèse 5: Les administrateurs des applications ou services Web ont accès à l'administration du système.

Hypothèse 6: Le système d'exploitation utilise le système « Shadow » pour la gestion des mots de passe.

Hypothèse 7: Le système a une base de données (SGBD) disponible pour entreposer les informations d'autorisation.

Hypothèse 8: Les logiciels clients (navigateur) supportent adéquatement le protocole HTTP(S) et réagissent de manière adéquate au message avec un statut « 401 ».

Hypothèse 9: Le système PAM (Pluggable Authentification Module) est disponible.

3.2 Véhicule pour l'implémentation

Basé sur l'état de l'art, section 1.1.4, les systèmes cibles identifiés sont les applications développées principalement en PHP, utilisées derrière un serveur HTTP(S) apache et exécutées sur un système UNIX. Une particularité de ces systèmes est que la plupart implémentent leur propre système d'authentification et d'autorisation et inversement ils utilisent presque tous un logiciel serveur pour l'implémentation du protocole HTTP(S). Le serveur Apache HTTP [13] de « Apache Foundation [44] » tel que présenté en 2.3.2 présente une structure intéressante pour le développement d'un tel projet. Il dispose en effet déjà de mécanisme pour gérer la sécurité des applications sous forme de modules bien que ces derniers sont sous-utilisés.

Une piste de solution qui ressort est donc de déléguer la gestion de la sécurité (authentification et autorisation) à ce même serveur puisqu'il est déjà une partie prenante de ces systèmes. L'adoption de la solution en sera favorisée, ne nécessitant pas de composants supplémentaires.

3.3 Possibilités d'authentification

Parmi les méthodes d'authentifications présentées en 2.4, l'utilisation de composants existants pour la gestion des autorisations pourrait se faire relativement facilement au niveau du système d'exploitation. En effet, ce dernier a déjà en place plusieurs mécanismes pour ce faire. La mise en place de cette approche doit être validée avec la possibilité d'intégration du contrôle de ce mécanisme par les applications. On doit également tenir compte des méthodes

possibles pour collecter les informations d'authentification auprès de l'utilisateur. Ces méthodes doivent être supportées par l'application client, c'est-à-dire le navigateur Web.

3.3.1 Basées sur le système d'exploitation

L'utilisation des composants du système d'exploitation Unix nous offre plusieurs choix dont :

1. L'utilisation du mécanisme standard basé sur les fichiers de permission (/etc./passwd, /etc./shadow, /etc./group)
2. L'utilisation de Linux Pam
3. L'utilisation de répertoire centralisé tel que LDAP

Le premier étant le plus utilisé et applicable à tous les systèmes UNIX doit être exploré, le second étant répandu sous Linux et offrant une plus grande flexibilité également. Le dernier sera mis de côté, car il nécessite un système complémentaire au système d'exploitation.

3.3.2 Basées sur les applications existantes

Dû à l'intégration avec des applications existantes et sur le doute inhérent à l'intégration des fonctions du système d'exploitation aux applications, on se doit d'également considérer les méthodes d'authentification des applications existantes soit :

1. L'utilisation d'une base de données (tel que MySQL) avec une structure propre à l'application.
2. L'utilisation d'un répertoire centralisé tel que LDAP dans un contexte applicatif.

La première approche sera conservée comme une alternative possible à l'utilisation des fonctions du système d'exploitation. La seconde, quant à elle, sera mise de côté pour les raisons citées en 4.3.1 puisqu'elle nécessite un composant supplémentaire.

3.3.3 Interactions client-serveur

L'utilisation du navigateur comme client nécessite également la mise en place de l'interface utilisateur pour déclencher l'authentification sous forme de saisie d'un identifiant et d'un mot

de passe. Nous assumerons l'hypothèse 8 valide considérant que les navigateurs réagissent aux messages avec un statut « 401 » par l'affichage d'une boite de dialogue de saisie de l'identifiant et du mot de passe.

De plus, il serait intéressant de pouvoir supporter des échanges personnalisés entre l'application et l'utilisateur.

3.4 Possibilités d'autorisation

Parmi les méthodes de gestion des autorisations présentées en 3.4, la réutilisation de composants existants au niveau du système d'exploitation est possible. En effet, ce dernier a déjà en place le mécanisme de gestion pour les fichiers qui peut s'appliquer. Cependant, l'identification des ressources à l'aide des URI couplés à un système de génération dynamique du contenu peut devenir un problème avec les mécanismes de gestion des fichiers qui est utilisé pour emmagasiner du contenu statique (Hypothèse 3-ii). La mise en place de cette approche doit être validée conjointement avec la possibilité d'intégration du contrôle de ce mécanisme par les applications.

3.4.1 Basées sur le système d'exploitation

L'utilisation du système de fichiers est possible, compte tenu des réserves mentionnées en 2.3.3, que si l'attribution d'URI est en relation 1 :1 avec des objets existants.

1. Système de fichiers (POSIX)
2. ACL

Le premier étant le standard sur les systèmes Unix doit être analysé. Le second, offert dans plusieurs environnements, est possible, il augmenterait les possibilités, mais sera gardé en alternative.

3.4.2 Basées sur les applications existantes

1. BD
2. LDAP

La majorité des applications existantes utilisent leur propre modèle de base de données pour la gestion des utilisateurs ou un répertoire tel que LDAP. Dans le premier cas, le modèle étant propriétaire à l'application, son utilisation est moins intéressante du fait qu'elle limiterait son application à une application en particulier. Le second cas quant à lui est mis de côté pour être compatible avec les raisons citées aux sections 4.3.1 et 4.3.2.

3.4.3 Interactions client-serveur

Les autorisations étant basées sur l'utilisateur préalablement identifié en phase d'authentification, le résultat de cette dernière doit être conservé et retransmis. L'utilisation du navigateur comme client nécessite donc que ce dernier conserve et retransmette les informations pertinentes. Nous assumerons l'hypothèse 8 valide considérant que les navigateurs retransmettent les informations dans les entêtes Basic Autentication à chaque requête.

Il serait intéressant d'explorer la possibilité de supporter un « cookie » de session permettant ainsi d'exécuter la phase d'authentification une fois pour plusieurs autorisations successives durant une période prédéterminée.

3.5 Solution proposée

Au regard des arguments présentés (hypothèse 1 et 2), la solution retenue pour l'expérimentation est d'utiliser le serveur Web Apache HTTP de l'Apache Foundation sous le système d'exploitation est Linux (ubuntu 16.04).

Le système d'authentification retenu est celui du système d'exploitation en premier lieu, sous deux formes soit l'utilisation directe (fonction pour accéder à /etc./passwd, /etc./shadow et /etc./group) ou au travers (à tout le moins sous Linux) du composant système PAM plutôt que l'accès direct aux fichiers au travers des fonctions de l'OS (Operating System) (hypothèse 4,5,6). La gestion des autorisations, quant à elle, sera effectuée avec les fonctions du système de fichiers (ugo+/-rwx) dans une première itération, puis avec l'utilisation une base de données (Hypothèse 7).

Le tout en assumant que les applications à protéger telles que WordPress respectent les principes d'utilisation du protocole HTTP (hypothèse 3) et que les logiciels client, les navigateurs Web, l'implémentent également de manière adéquate (hypothèse 8).

CHAPITRE 4

Réalisation

Pour les besoins de l'entreprise commanditaire, le développement de l'application est documenté tel que défini par la norme ISO29110 [24]. Les principaux éléments sont incorporés dans le présent document. Les documents complets, en date de ce document, sont disponibles à l'adresse : http://absec.adninformatique.com/. Les codes sources complets de l'application et des tests ont été déposés dans le dépôt git sur un système Kalithea [67] à l'adresse : https://source.mescours.ca/ADN/Maitrise/projet.

4.1 Création d'un environnement de développement

Deux environnements ont été créés pour le développement du module. Le premier similaire à un environnement de production pour la réalisation des divers tests et la validation du module, le second destiné au développement. Cet environnement de travail comprendra quant à lui tous les outils pour le développement (les fichiers d'entête Apache, APR, le compilateur …). Les deux environnements ont été construits à partir de la version Linux Ubuntu Server 16.04.3 LTS x86_64 (64 Bits).

À l'intérieur des deux environnements, le serveur Web HTTPD Apache de l'Apache Foundation a été installé ainsi que le module complémentaire permettant l'exécution de code en langage PHP. Les instructions d'installation sont les suivantes (voir Fragment 4.1) :

Installation d'Apache et de PHP à partir des « paquets » standards

```
1:    apt-get install apache2
2:    apt-get install PHP libapache2-mod-php
```

Fragment 4.1 - Commandes pour l'installation d'Apache et PHP

Dans les deux cas, l'installation du serveur et de l'interpréteur PHP a été faite à partir des dépôts officiels de « package » afin d'assurer la reproductibilité et de se rapprocher le plus possible d'un environnement de production réel.

4.1.1 Serveur

L'environnement serveur se veut un environnement se rapprochant le plus possible d'un environnement de production. Sa configuration n'inclut aucun outil de développement. Sa version initiale se limite aux composants minimaux et dans la version finale, une application de test (tel que WordPress) y sera déployée pour permettre de conclure les tests.

Les paramètres de création de cette machine virtuelle ProxMox [68] KVM (Kernel-based Virtual Machine) sur le serveur de l'entreprise commanditaire prox01.adninformatique.com sont les suivants :

- 1 CPU virtuel
- 2G Mémoire vive
- 32G Disque dur

Les spécifications du serveur physique sont les suivantes :

- 8 x Intel core i7-2600 @3.40 GHZ (1 socket)
- 16G Mémoire vive
- 500G Disque dur SATA

4.1.2 Poste de travail

L'environnement de travail, dans le but d'être autonome et fonctionnel, a été réalisé sur une Machine virtuelle « Parallel Desktop » [69] sous macOS [70] sur un iMac. Cela permet de partager le système de fichiers pour une édition plus aisée des codes sources et de permettre une indépendance du réseau durant la période de développement. Les paramètres de création de cette machine sont les suivants :

- 2 CPUs Virtuels
- 1G Mémoire vive
- 64G Disque dur

Les spécifications de la machine physique sont les suivantes :

- Intel Core i5 @3.2 GHZ
- 24G de mémoire vive
- 500G Disque dur (SATA over FireWire)

L'environnement de travail comprend un disque partagé entre la machine hôte et la machine virtuelle pour simplifier l'édition des fichiers. Les outils de développement sur la machine hôte comprennent notamment

- Visual Code Studio [71] de Microsoft pour l'édition
- GIT pour la gestion des codes sources avec l'application SourceTree [72] de Atlassian.
- Le navigateur Safari [73] sous macOS a été le principal client Web utilisé, des tests supplémentaires avec Firefox [74] et Chrome [75] également sous macOS ont été effectués.

Les outils de développement sur la machine virtuelle comprennent notamment

- GCC pour la compilation du module
- Les outils de développement d'Apache notamment APSX
- Les fichiers d'entête d'Apache, d'APR, du système d'exploitation
- GIT pour la gestion des codes sources

Les instructions pour l'installation des outils de développement sont détaillées au Fragment 4.2.

```
3:    #Outils de developpements
4:    sudo apt-get install build-essential
5:    sudo apt-get install autoconf automake
6:
7:    #Dependances de compilation de module pour apache
8:    sudo apt-get install apache2-dev
9:    sudo apt-get install libpcre3-dev
10:   sudo apt-get install libapr1-dev libaprutil1-dev
```

Fragment 4.2 - Commandes pour l'Installation des outils de développement

Note : Pour plus de souplesse comme utilisateur, il est intéressant d'utiliser un autre service pour écouter sur le port 80 (qui nécessite des droits de super utilisateur) et permettre d'exécuter Apache sur un autre port. Cela a comme avantage, lors du redémarrage d'Apache qui est sur un numéro de port supérieur à 1024, il ne sera pas nécessaire d'utiliser des droits de super utilisateur. Le logiciel opérant sur le port 80 peut être une autre copie d'Apache (en mode proxy), un mandataire HTTP quelconque (NGINX, Squid, Varnish, ...) ou un mandataire TCP comme rinetd, redir .... ou même faire une configuration dans la pile TCP avec ipchains.

Pour les besoins de la présente, le déverminage du code n'a pas été nécessaire. Cependant, étant donné que le serveur Apache HTTPD est multiprocessus/multi-thread, la

familiarisation avec les techniques de déverminage dans ces environnements est recommandée pour aller plus loin dans le développement du module. Des informations de complémentaires à propos de l'utilisation de l'outil GDB pour le déverminage d'Apache HTTP Server peuvent être trouvées sur les sites suivants :

https://httpd.apache.org/dev/debugging.html

https://sourceware.org/gdb/onlinedocs/gdb/index.html.

4.2 Création d'un module de base

La première étape a consisté à réaliser un module de base en suivant les informations fournies dans la documentation en ligne d'Apache. Ce module intégré à l'application sous forme de « hook » est la base pour les développements ultérieurs. Cette version est incorporée à Apache sous forme de règle de traitement pour un regroupement d'URL. Elle est responsable du processus de livraison des données. La réalisation de cette étape a également permis une familiarisation avec l'environnement de développement d'Apache, notamment l'Apache Portable Runtime (APR). L'utilisation de l'APR sera privilégiée tout au long du développement afin d'assurer la portabilité du code et la possibilité de son utilisation sur de multiples plateformes telles que le serveur Apache HTTPD lui-même. Nous basons notre développement sur plusieurs ressources en ligne. [76]

La seconde étape a consisté à récupérer les informations d'authentification de l'utilisateur sous forme d'identifiants (identifiant et mot de passe) afin de pouvoir en faire la validation auprès du système d'exploitation, d'abord avec l'utilisation directe des fichiers /etc./passwd, /etc./shadow puis au travers de PAM. La déclaration des fonctions /etc. et PAM a été fait de manière uniforme afin de permettre l'interchangeabilité.

La troisième étape sera d'obtenir les informations sur la ressource accédée ainsi que l'action désirée sur cette dernière afin de valider les autorisations de l'usager authentifié pour l'accès à cette ressource en utilisant comme premier mécanisme les fonctions de gestion du

système de fichiers de l'OS, puis à l'aide d'une base de données (SGBD). La déclaration des fonctions basées sur le système de fichiers et sur une base de données (SGBD) a été faite de manière uniforme afin de permettre l'interchangeabilité.

4.2.1 Gabarit d'un module Apache

La création d'un module de base pour le serveur HTTPD Apache est documentée dans plusieurs sources. La méthode la plus simple et reproductible consiste à utiliser l'outil « apxs ». Comme expliqué sur le site Web d'Apache [77] et diverses sources [78], les instructions dans le Fragment 4.3 ont été utilisées. Cette commande génère un gabarit de base pour le développement d'un module pour Apache présenté au même endroit. (les commentaires ont été supprimés)

```
Génération du gabarit
11:   /usr/local/apache2/bin/apxs -g -n absec

Fichier résultant
12:   #include "httpd.h"
13:   #include "http_config.h"
14:   #include "http_protocol.h"
15:   #include "ap_config.h"

16:   /* The sample content handler */
17:   static int absec_handler(request_rec *r)
18:   {
19:   if (strcmp(r->handler, "absec")) {
20:   return DECLINED;
21:   }
22:   r->content_type = "text/HTML";

23:   if (!r->header_only)
24:   ap_rputs("The sample page from mod_absec.c\n", r);
25:   return OK;
26:   }

27:   static void absec_register_hooks(apr_pool_t *p)
28:   {
29:   ap_hook_handler(absec_handler, NULL, NULL, APR_HOOK_MIDDLE);
30:   }

31:   /* Dispatch list for API hooks */
32:   module AP_MODULE_DECLARE_DATA absec_module = {
33:   STANDARD20_MODULE_STUFF,
34:   NULL,                  /* create per-dir    config structures */
35:   NULL,                  /* merge  per-dir    config structures */
36:   NULL,                  /* create per-server config structures */
37:   NULL,                  /* merge  per-server config structures */
38:   NULL,                  /* table of config file commands       */
39:       absec_register_hooks  /* register hooks                      */
40:   };
```

Fragment 4.3 - Gabarit de module autogénéré

La compilation de ce module peut être effectuée à l'aide de la commande présentée dans le Fragment 4.4 qui s'assure de l'inclusion des bibliothèques nécessaires, de l'installation du module dans l'environnement du serveur et du redémarrage de ce dernier pour s'assurer du chargement de la nouvelle version du code.

```
41:   /usr/local/apache2/bin/apxs -i -a -c mod_absec.c
```

Fragment 4.4 – Commande pour la compilation du module

Afin que le serveur charge le module et attribue un groupe d'URI à l'appel de ce dernier, la configuration doit comprendre quelques modifications comme indiqué dans le Fragment 4.5 Il faut prêter une attention particulière lors de la mise en place de la configuration à la déclaration du nom de module « absec_module », au nom du fichier « mod_absec.so » et du nom du « handler » nommé « absec ».

```
Fichier /etc./apache2/mods-available/absec.load et lien symbolique dans
/etc./apache2/mods-enable/absec.load

42:   LoadModule absec_module modules/mod_absec.so

Fichier /etc./apache2/mods-available/absec.conf et lien symbolique dans
/etc./apache2/mods-enable/absec.conf

43:   <IfModule absec_module>
44:       <Location /absec>
45:           SetHandler absec
46:       </Location>
47:   </IfModule>
```

Fragment 4.5 - Extrait de la Configuration d'Apache HTTP

L'accès au serveur à l'aide du client pour l'adresse de base « / » nous retourne bien la page d'accueil par défaut du serveur et l'accès à la sous-page « /absec/ » nous retourne bien le texte transmis par le module tel que présenté aux Figure 4.1 et Figure 4.2. (mod_absec_521.c)

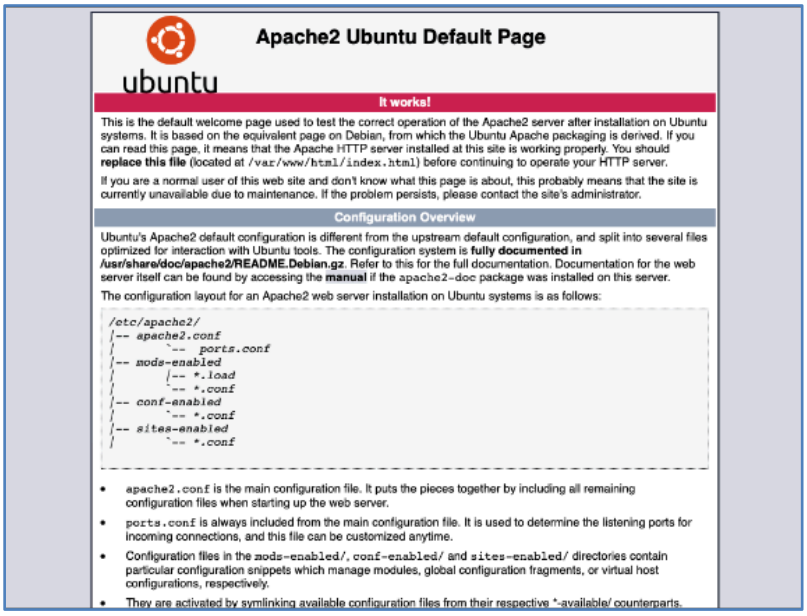


Figure 4.1 - Page d'accueil d'Apache

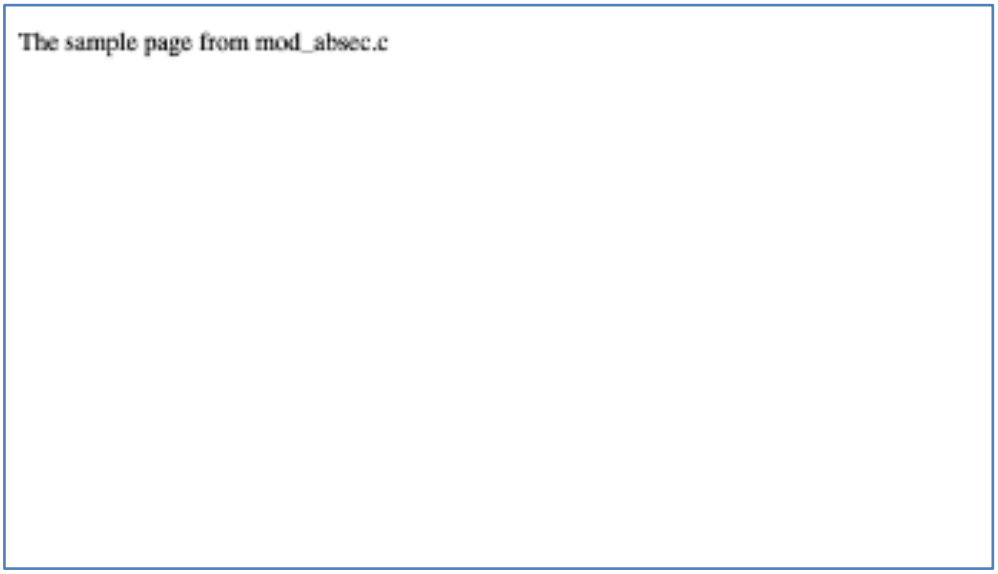


Figure 4.2 - Page d'accueil du module

4.2.2 Récupération des informations d'authentification

L'étape suivante a consisté en la modification de ce module de base pour la récupération des informations de l'utilisateur à partir du navigateur. Pour ce faire l'utilisation du mécanisme

(« Basic authentification ») prévu dans le protocole a été utilisée en vérifiant le retour du code « 401 », tel que présenté dans le Fragment 4.6 L'exécution du module avec l'ajout de cette fonctionnalité nous retourne les informations désirées soit une demande d'authentification ou, si l'information est déjà présente, le nom d'utilisateur et le mot de passe fournis comme le montrent les Figure 4.3 et Figure 4.4. (mod_absec_522.c)

```
48:   // Verification si on a un entête d'authentification
49:   auth64p = apr_table_get(r->headers_in,"Authorization");
50:   if (auth64p==NULL) {
51:       apr_table_setn(r->err_headers_out,
52:          (PROXYREQ_PROXY == r->proxyreq) ? "Proxy-Authenticate"
53:          : "WWW-Authenticate",
54:          apr_pstrcat(r->pool, "Basic realm=\"", ap_auth_name(r),
55:          "\"", NULL));
56:       return HTTP_UNAUTHORIZED;
57:   }
58:
59:   // Recuperation du user/pass à partir du contenu de l'entête
60:   // qui est une chaine base64
61:   // début au caractère 6 après le mot 'Basic '
62:   char *auth64;
63:   auth64 = apr_pstrdup(r->pool, auth64p+6);
64:   // décode la chaine base64
65:   char *auth;
66:   auth = apr_pcalloc(r->pool, 64);
67:   apr_base64_decode(auth, auth64);
68:   // separe le user/pass
69:   char *user;
70:   char *pass;
71:   user = apr_strtok(auth, ":", &pass);
```

Fragment 4.6 - Code de récupération de l'identifiant et du mot de passe

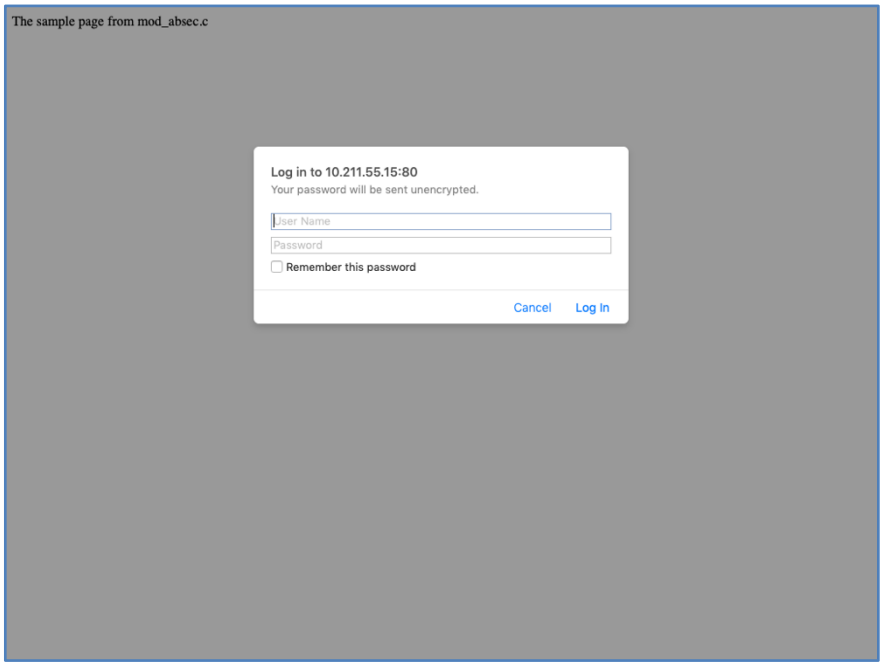


Figure 4.3 - Boite de dialogue de demande d'authentification

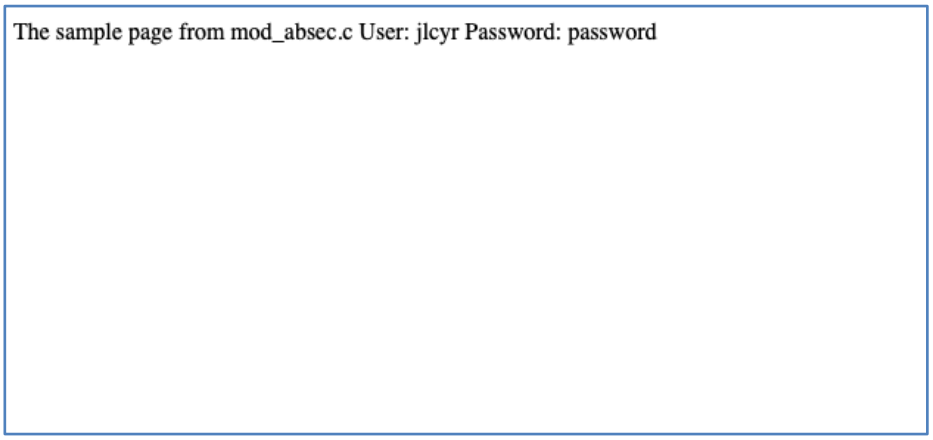


Figure 4.4 - Page du module affichant l'authentification reçu

4.2.3 Récupération des informations sur la ressource

Par la suite, des fonctionnalités pour la récupération des informations sur la page (URI) ont été ajoutées (voir Fragment 4.7). Les fonctions de bases d'Apache HTTPD permettent de récupérer la majorité des informations sur les requêtes par l'intermédiaire de la variable « r » qui est une structure « request_rec » telle que définie dans le fichier « httpd.h ». Nous y retrouvons notamment l'URL demandée, le nom du fichier correspondant à cette URL et la méthode invoquée qui nous sera nécessaire pour valider les autorisations. La Figure 4.5 montre un exemple d'exécution du module dans lequel on a obtenu une correspondance pour la méthode GET. (mod_absec_523.c)

```
72:   // Récupération des permissions base sur le nom de fichier vs URL/URI
73:   status = stat(r->filename, &fperm);
74:
75:   // Récupération et vérification de la méthode vs l'action
76:       int permmask = 0;
77:       if (strcmp(r->method,"GET")==0) permmask=0444;     // r
78:       if (strcmp(r->method,"PUT")==0) permmask=0222;     // w
79:       if (strcmp(r->method,"POST")==0) permmask=0222;    // w
80:       if (strcmp(r->method,"DELETE")==0) permmask=0111;  // x
```

Fragment 4.7 - Récupération des informations de la requête

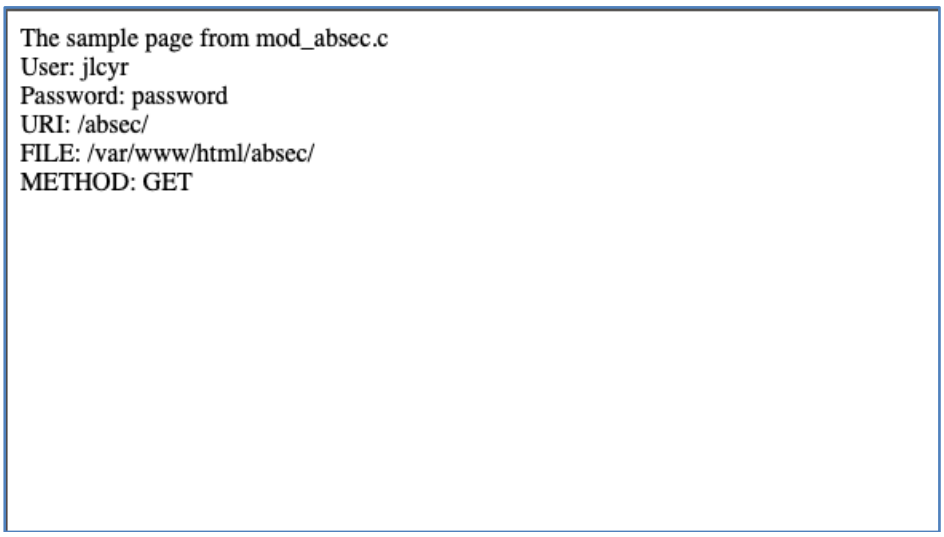


Figure 4.5 - Page du module affichant les informations de la requête

4.3 Composants d'authentification

Un entête de fonction standard a été établi pour l'authentification afin de pouvoir interchanger les composants utilisant les fichiers /etc./passwd et celui utilisant PAM. Les structures existantes telles que « request_rec » et « stat » ont été utilisées comme structures de base. L'entête établi est présenté dans le Fragment 4.8.

```
81:   int check_user(char* user, char* pass);
```

Fragment 4.8 - Entête de la fonction d'authentification

4.3.1 Utilisation de /etc./passwd pour l'authentification

L'accès des fichiers de gestion des authentifications du système Unix (/etc./passwd) pourrait être fait directement. Cependant dans un souci de portabilité et de respect des standards, les fonctions de plus haut niveau disponible ont été considérées en premier lieu. À ce chapitre, l'APR d'Apache comprend quelques fonctions à cette fin documentées à la section « user and group id services » [79]. Cependant, pour les besoins du serveur, seules la récolte d'information sur l'utilisateur exécutant le processus ou des informations sur un utilisateur propriétaire de ressource sont nécessaires. Les fonctions permettent donc de récupérer à

partir de son identifiant (login) ou son UID, le nom, le UID, le nom de son groupe, le GID et le répertoire de base, mais ne permettent pas la validation de son mot de passe. L'accès aux fonctions de base du système d'exploitation [80] est donc nécessaire.

Utilisation d'apr

Les prototypes des fonctions disponibles via l'APR, retournant toutes un « apr_status_t », sont présentés Fragment 4.9. Elles permettent de récupérer plusieurs informations à propos d'un compte utilisateur telles que le nom, le uid, le répertoire de base et le groupe. Cependant, aucune fonction n'est disponible pour récupérer ou valider le mot de passe de l'utilisateur ce qui est prérequis pour authentifier ce dernier.

```
Entête des fonctions de l'APR

82:   apr_uid_current (apr_uid_t *userid, apr_gid_t *groupid, apr_pool_t *p)
83:   apr_uid_name_get (char **username, apr_uid_t userid, apr_pool_t *p)
84:   apr_uid_get (apr_uid_t *userid, apr_gid_t *groupid, const char *username,
apr_pool_t *p)
85:   apr_uid_homepath_get (char **dirname, const char *username, apr_pool_t *p)
86:   apr_gid_name_get (char **groupname, apr_gid_t groupid, apr_pool_t *p)
87:   apr_gid_get (apr_gid_t *groupid, const char *groupname, apr_pool_t *p)

Exemple de récupération des informations
88:   /* Recuperation du UID avec les fonctions de l'APR à partir du username reçu */
89:   apr_status_t ret;
90:   apr_uid_t i;
91:   apr_gid_t g;
92:   ret = apr_uid_get(&i, &g, user, r->pool);
```

Fragment 4.9 - Fonctions de l'APR pour manipuler les informations d'un utilisateur

Ces fonctions ne permettent pas d'obtenir les informations nécessaires à l'authentification. Ces dernières doivent donc être récupérées par les fonctions du système d'exploitation décrites dans le Fragment 4.11. À la lecture des informations sur les fonctions de l'APR, on peut noter deux points qui devront être pris en considération lors de l'utilisation de ces fonctions du système soit : la « portabilité » et l'utilisation sécuritaire en mode processus multiples (« multi thread »). En effet, ces deux critères sont à la base du développement de l'APR. Certaines fonctions du système d'exploitation sont conçues pour une utilisation en mode processus multiple comme, par exemple, « getpwnam_r » qui peut être utilisée selon certaines

conditions de compilation. Ces détails sont illustrés dans le code de la fonction « getpwnam_safe » de l'APR présenté dans le Fragment 4.10 (voir les lignes débutantes par « #if »).

```
93:
94:   ////////////////////////////////////////////////////////////////
95:   static apr_status_t getpwnam_safe(const char *username,
96:                                     struct passwd *pw,
97:                                     char pwbuf[PWBUF_SIZE])
98:   {
99:       struct passwd *pwptr;
100:  #if APR_HAS_THREADS && defined(_POSIX_THREAD_SAFE_FUNCTIONS) &&
defined(HAVE_GETPWNAM_R)
101:      apr_status_t rv;
102:
103:      /* POSIX defines getpwnam_r() et al to return the error number
104:       * rather than set errno, and requires pwptr to be set to NULL if
105:       * the entry is not found, imply that "not found" is not an error
106:       * condition; some implementations do return 0 with pwptr set to
107:       * NULL. */
108:      rv = getpwnam_r(username, pw, pwbuf, PWBUF_SIZE, &pwptr);
109:      if (rv) {
110:          return rv;
111:      }
112:      if (pwptr == NULL) {
113:          return APR_ENOENT;
114:      }
115:  #else
116:      /* Some platforms (e.g. FreeBSD 4.x) do not set errno on NULL "not
117:       * found" return values for the non-threadsafe function either. */
118:      errno = 0;
119:      if ((pwptr = getpwnam(username)) != NULL) {
120:          memcpy(pw, pwptr, sizeof *pw);
121:      }
122:      else {
123:          return errno ? errno : APR_ENOENT;
124:      }
125:  #endif
126:      return APR_SUCCESS;
127:  }
128:
```

Fragment 4.10 - Définition de la fonction getpwnam_safe

Utilisation des fonctions du système d'exploitation

Les prototypes des fonctions disponibles via le système d'exploitation sont présentés dans le Fragment 4.11 de même qu'un exemple de leur utilisation et du résultat obtenu.

```
129:  struct passwd *getpwnam(const char *name); // getpwnam_r
130:  struct passwd *getpwuid(uid_t uid);
131:
132:  int getgrouplist(const char *user, gid_t group, gid_t *groups, int *ngroups);

Exemple de code
133:  #include <stddef.h>
134:  #include <pwd.h>
135:  #include <stdio.h>
136:
```

```
137:  int uid = 1000;
138:
139:  int main(int argc, char** argv) {
140:      struct passwd *pwd;
141:
142:      if ((pwd = getpwuid(uid)) == NULL) {
143:          printf("Erreur");
144:          /* erreur */
145:      }
146:
147:      printf("name: %s\r\n", pwd->pw_name);
148:      printf("uid: %d\r\n", pwd->pw_uid);
149:      printf("gid: %d\r\n", pwd->pw_gid);
150:      printf("pw: %s\r\n",  pwd->pw_passwd);
151:
152:      return 0;
153:  }

Exemple du résultat
154:  gcc exemple_5312.c ; ./a.out
155:  name: jlcyr
156:  uid: 1000
157:  gid: 1000
158:  pw: x
```

Fragment 4.11 - Utilisation de la fonction getpw* de l'OS

Dans cet exemple, on constate que le mot de passe (pw) obtenu du système d'exploitation est invalide (valeur « x »). En effet, dans la majorité des systèmes UNIX récents, le fichier /etc./passwd, qui doit être accessible à tous les utilisateurs, et le fichier /etc./shadow, utilisé pour la conservation des mots de passe, n'est accessible que par le super-utilisateur (« root ») et les services qui en nécessitent l'accès. Nous devons donc modifier le fonctionnement de ce code pour utiliser le contenu de ce dernier. Nous établissons donc l'hypothèse 6 : Le système d'exploitation utilise le système « Shadow » pour la gestion des mots de passe.

L'accès à la liste des utilisateurs était la première étape. La seconde, qui consistait à accéder à la validation de mot de passe, est triviale dans certains cas (utilisation seulement de passwd). Mais, dans la majorité des systèmes, les mots de passe sont maintenant découplés de la liste d'utilisateurs (par l'utilisation de shadow).

4.3.2 Utilisation /etc./shadow

Pour la récupération du mot de passe et des autres informations contenues dans le fichier /etc./shadow, des fonctions du système d'exploitation sont disponibles. Le Fragment 4.12 illustre leur utilisation et la Figure 4.6 montre le résultat de l'intégration à notre module [81].

```
Exemple de code
159:  #include <shadow.h>
160:  #include <stddef.h>
161:  #include <stdio.h>
162:
163:  char* user="jlcyr";
164:
165:  int main(int argc, char** argv) {
166:      struct spwd *spw;
167:
168:      if ((spw = getspnam(user)) == NULL) {
169:          printf("Erreur\r\n");
170:          return -1;
171:      }
172:
173:      /* Traitement des cas spéciaux de mot de passe */
174:       if (spw->sp_pwdp[0] == 'x' || spw->sp_pwdp[0] == '*' || spw->sp_pwdp[0] ==
'!') {
175:          printf("Password non autorisé");
176:          /* non authorise */
177:      }
178:
179:      printf("pw: %s\r\n", spw->sp_pwdp);
180:
181:      return 0;
182:  }

Exemple de compilation et exécution
183:  gcc exemple_532.c -o a.out;
184:  ./a.out
185:  Erreur
186:  sudo ./a.out
187:  pw:
$6$j4YBlwAh$CkongzsmbERrZlS0KT8azUYZ8R21XIz335BDpi5ocmMkFtZBS4lBon9b73crITPdxSN4yUb..P
V5ul5.9Dc1c.
```

Fragment 4.12 - Exemple d'utilisation des informations du fichier "shadow"

Figure 4.6 - Page du module affichant les informations du fichier shadow

Nous pouvons donc avec succès récupérer le mot de passe crypté pour fin de comparaison. Il est à noter que, pour exécuter avec succès le programme, l'utilisateur « root » doit être employé (utilisation de la commande « sudo »). Nous poursuivons en ajoutant la comparaison du mot de passe reçu avec celui obtenu du fichier /etc./shadow. Pour ce faire, nous utilisons

la fonction « crypt » du système d'exploitation. Le système d'exploitation permet l'existence de comptes pour lesquels aucun mot de passe n'est défini dans le but de bloquer le compte. Il sera important de tenir compte de cette fonctionnalité. De plus, les algorithmes sont non réversibles. La comparaison des informations cryptées doit donc être réalisée [82] [83] telle qu'illustrée dans le Fragment 4.13.

```
188:  #include <crypt.h>
189:
190:  // pass : mot de passe reçu dans la requête
191:  // spw->sp_pwdp : mot de passe récupéré du fichier /etc./shadow
192:
193:  char *encrypted;
194:  const char *correct;
195:  int diff_value;
196:  encrypted = crypt(pass, spw->sp_pwdp);
197:  diff_value = strcmp(encrypted, spw->sp_pwdp);
198:
199:  if (diff_value!=0) {
200:      return HTTP_UNAUTHORIZED;
201:  }
```

Fragment 4.13 - Code de comparaison mot de passe crypté

Tenant compte de l'hypothèse 6, nous intégrons donc ce code à notre module tel qu'illustré dans le Fragment 4.14.

```
202:  /* Retrieve PW from /etc/shadow */
203:  /* Should #include <shadow.h> */
204:      struct spwd *spw;
205:      errno = 0;
206:      if((spw = getspnam(user)) == NULL)
207:      {
208:          ap_rprintf(r, "NULL %d\n<br/>", errno);
209:          apr_table_setn(r->err_headers_out,
210:              (PROXYREQ_PROXY == r->proxyreq) ? "Proxy-Authenticate"
211:                                              : "WWW-Authenticate",
212:              apr_pstrcat(r->pool, "Basic realm=\"", ap_auth_name(r), "\"",
NULL));
213:          return HTTP_UNAUTHORIZED;
214:      }
215:      else
216:      {
217:          ap_rprintf(r, "Shadow PW : %s \n<br/>", spw->sp_pwdp);
218:      }
219:
220:      if (spw->sp_pwdp[0] == 'x' || spw->sp_pwdp[0] == '*' || spw->sp_pwdp[0] ==
'!') {
221:          apr_table_setn(r->err_headers_out,
222:              (PROXYREQ_PROXY == r->proxyreq) ? "Proxy-Authenticate"
223:                                              : "WWW-Authenticate",
224:              apr_pstrcat(r->pool, "Basic realm=\"", ap_auth_name(r), "\"",
NULL));
225:          return HTTP_UNAUTHORIZED;
226:      }
227:
228:  ////////
229:  /* Encrypt and compare shadow password */
230:
231:  // TODO : Valider qu'on a un user
232:  // TODO : Valider qu'il y a un password (pas * ! rien)
233:      char *encrypted;
234:      const char *correct;
235:      int rrr;
236:      encrypted = crypt(pass, spw->sp_pwdp);
237:      rrr = strcmp(encrypted, spw->sp_pwdp);
238:      ap_rprintf(r, "compare pw : %s \n<br/>", encrypted);;
239:      ap_rprintf(r, "compare : %d \n<br/>", rrr);
240:      if (rrr!=0) {
241:          apr_table_setn(r->err_headers_out,
242:                  (PROXYREQ_PROXY == r->proxyreq) ? "Proxy-Authenticate"
243:                                                  : "WWW-Authenticate",
244:                  apr_pstrcat(r->pool, "Basic realm=\"", ap_auth_name(r),
245:                              "\"", NULL));
246:          return HTTP_UNAUTHORIZED;
247:      }
```

Fragment 4.14 - Integrationde du code pour l'utilisation du fichier shadow au module

Lors de l'exécution de cette version du module, nous constatons que le serveur n'a pas accès au fichier /etc./shadow et retourne donc sans fin une demande d'authentification puisque la fonction retourne sans cesse une erreur. (getspnam(user) == NULL) Ce problème est causé par le fait que les fichiers de stockage de mots de passe « shadow » ne sont pas accessibles par les utilisateurs réguliers du système. Le serveur Web HTTPD n'a donc pas accès par

défaut au contenu de ces fichiers, directement ou au travers des fonctions du système d'exploitation.

La première alternative envisagée pour accéder à ces éléments a été l'exécution du serveur avec les privilèges administrateur « root ». Dans ce mode, nous obtenons un message signifiant que la version compilée d'Apache HTTPD ne permet pas cette configuration. Pour contourner ce problème, une réinstallation complète à partir des codes sources est nécessaire en utilisant l'option de compilation « DBIG_SECURITY_HOLE ». La documentation indique cependant que cette méthode peut causer des problèmes de sécurité. Suite à ces tests, cette approche a été écartée puisqu'elle contrevient au but premier de ce projet, soit d'augmenter la sécurité des systèmes. Pour plus de détails concernant la compilation utilisant cette option, voir l'ANNEXE VI [84].

Une seconde approche a donc été développée. Celle-ci consiste à permettre à l'utilisateur du serveur Apache (www-data) d'accéder au contenu de ce fichier par l'utilisation d'un groupe supplémentaire. Les droits assignés à ce fichier sur un système « propre » sont les suivants :

-rw-r----- 1 root shadow 970 Jul 11 2018 /etc/shadow.

Les autres fichiers accessibles avec les mêmes permissions peuvent être déterminés avec la commande présentée dans le Fragment 4.15. Le résultat est présenté à la Figure 4.7. On peut y constater que le fait d'ajouter le serveur Web au groupe shadow a un impact mineur sur la sécurité du système.

```
248:  find /* -group shadow -name "*" | xargs ls -l
```

Fragment 4.15 - Recherche des fichiers "shadow"

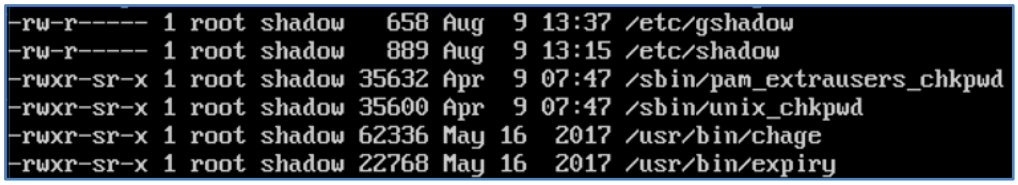


Figure 4.7 - Liste des fichiers appartenant au groupe "shadow"

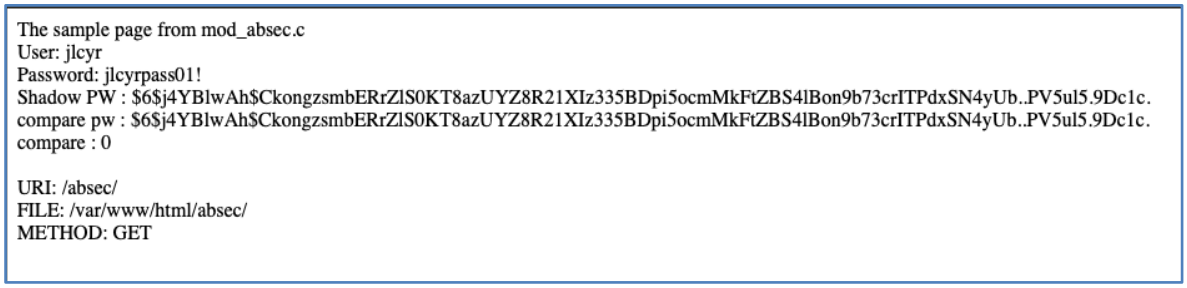


Figure 4.8 - Page du module affichant la comparaison du mot de passe reçu et celui valide

Après cette modification aux permissions de l'utilisateur www-data, on peut exécuter la version de notre module affichant l'utilisateur, le mot de passe reçu reçu et sa validation. On constate que la comparaison des deux chaines de caractères nous retourne la valeur zéro (0) donc elles sont semblables(Figure 4.8). (mod_absec_532.c)

4.3.3 Utilisation de PAM pour l'authentification

L'utilisation de PAM élève le niveau d'abstraction au-dessus des fichiers de /etc. Il permet d'utiliser des instructions de haut niveau et d'ajouter des fonctionnalités de sécurité supplémentaires telles que : l'expiration des mots de passe, la vérification de leur complexité et permettre l'utilisation d'autres systèmes tels que LDAP. La documentation complète est disponible en ligne [85].

Compilation et l'exécution doivent comprendre la liaison avec la bibliothèque dynamique « PAM » d'où l'introduction de l'argument « -lpam » tel qu'illustré dans le Fragment 4.16. Le fonctionnement de PAM nécessite l'enregistrement de l'application dans la configuration du système dans le répertoire /etc./pam.d/ [86]. Un exemple est présenté dans le Fragment 4.17.

Son utilisation dans le code demande l'implémentation d'une fonction de rappel (« CallBack ») ici définie sous le nom « converse ». En faisant l'hypothèse que le système possède le système PAM (Hypothèse #9), les fonctions suivantes sont alors disponibles : initialisation de l'appel, appel de validation du mot de passe, fermeture du système. Leur utilisation est illustrée dans le Fragment 4.16 et son incorporation au module dans le Fragment 4.18.

```
Exemple d'utilisation de LDAP
249:  #include <security/pam_appl.h>
250:  #include <security/pam_misc.h>
251:  #include <stddef.h>
252:  #include <stdio.h>
253:
254:  struct pam_response *reply;
255:
256:  /* function de rappel (callback) de PAM */
257:  int converse(int n, const struct pam_message **msg,
258:      struct pam_response **resp, void *data)
259:  {
260:    *resp = reply;
261:    return PAM_SUCCESS;
262:  }
263:
264:  /* initialisation pour les rappels */
265:  struct pam_conv conv = { converse, 0 };
266:
267:  /*************************************/
268:  /* Extrait du code pour l'appel à PAM */
269:  int main(int argc, char** argv) {
270:
271:      reply = (struct pam_response *)malloc(sizeof(struct pam_response));
272:
273:      char* user="jlcyr";
274:      char* pass="jlcyrpass01!";
275:
276:      pam_handle_t * pamh = NULL;
277:      int rret;
278:
279:      /* Initialisation de PAM */
280:      if((rret = pam_start("httpd", user/*pw->pw_name*/, &conv, &pamh)) !=
PAM_SUCCESS) {
281:              printf("Pam start failed\n");
282:              exit(-1);
283:      }
284:
285:      /* Passage du mot de passe reçu vers PAM */
286:      reply[0].resp = strdup(pass);
287:      reply[0].resp_retcode = 0;
288:
289:      /* Appel de la function d'authentification */
290:      if((rret = pam_authenticate(pamh, 0)) != PAM_SUCCESS) {
291:          printf("Pam authenticate failed\n");
292:          exit(-1);
293:      }
294:
295:      /* Deconnexion de PAM */
296:      if(pam_end(pamh, rret) != PAM_SUCCESS) {
297:          printf("Pam end failed\n");
298:          exit(-1);
299:      }
300:
301:      printf("Pam success\n");
302:
303:  }

Instruction de compilation et d'exécution
```

```
304:  gcc exemple_533.c -lpam -o a.out

305:  ./a.out
306:  Pam success
```

Fragment 4.16 - Exemple de code pour l'utilisation de PAM

```
Contenu du fichier /etc./pam.d/httpd
307:  # fichier base sur /etc./pam.d/common-auth - authentication settings common to
all services
308:  #
309:  # here are the per-package modules (the "Primary" block)
310:  auth  [success=1 default=ignore]    pam_unix.so nullok_secure,try_first_pass
311:  # here's the fallback if no module succeeds
312:  auth  requisite              pam_deny.so
313:  # prime the stack with a positive return value if there isn't one already;
314:  # this avoids us returning an error just because nothing sets a success code
315:  # since the modules above will each just jump around
316:  auth  required               pam_permit.so
317:  # and here are more per-package modules (the "Additional" block)
318:  # end of pam-auth-update config
```

Fragment 4.17 - Fichier de configuration de PAM

L'arrimage au module nous retourne effectivement le statut de l'authentification par l'intermédiaire de PAM tel qu'illustré à la Figure 4.9 (mod_absec_533.c). Pour compléter la validation de cette approche, différents cas d'erreurs ont été testés (mauvais utilisateur, mauvais mot de passe …). Il a été constaté qu'en cas d'erreur, la réception de la réponse subissait un ralentissement significatif.

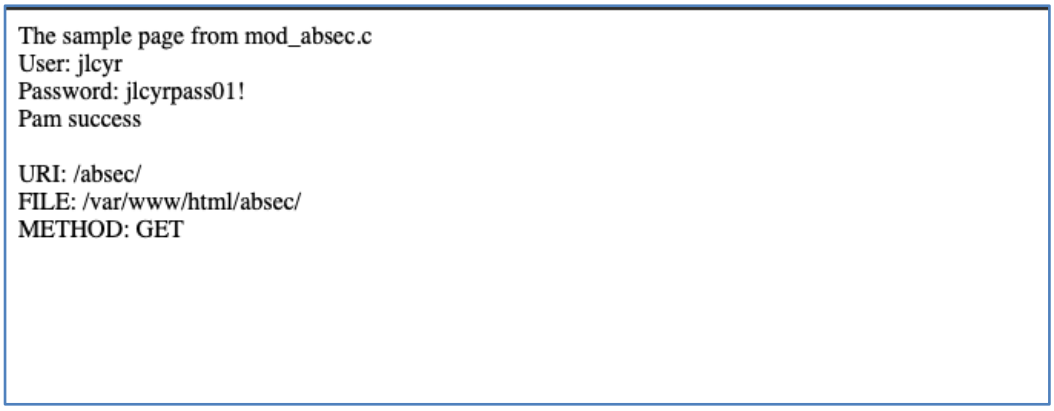


Figure 4.9 - Page du module affichant les informations de validation de PAM

```
Incorporation de PAM au module
319:  // Global var for passing fake response to PAM callback
320:  struct pam_response *reply;
321:
322:  ////////////////////////////////////////////////////////////////
323:  // PAM response callback function
324:  int converse(int n, const struct pam_message **msg,
325:      struct pam_response **resp, void *data)
326:  {
327:      // Return globally set response
328:      *resp = reply;
329:      return PAM_SUCCESS;
330:  }
331:
332:  ////////////////////////////////////////////////////////////////
333:  // define PAM callback function
334:  struct pam_conv conv = { converse, 0 };
335:
336:  /* The sample content handler */
337:  static int absec_handler(request_rec *r)
338:  {
339:      if (strcmp(r->handler, "absec")) {
340:          return DECLINED;
341:      }
342:      r->content_type = "text/HTML";
343:
344:      if (!r->header_only) {

… Code illustré précédemment …

345:          pam_handle_t * pamh = NULL;
346:          int rret;
347:
348:          /* Initialisation de PAM */
349:          if((rret = pam_start("httpd", user, &conv, &pamh)) != PAM_SUCCESS) {
350:              ap_rputs("Pam start failed<br/>", r);
351:              return HTTP_INTERNAL_SERVER_ERROR;
352:          }
353:
354:          /* Passage du mot de passe reçu vers PAM */
355:          reply = (struct pam_response *)malloc(sizeof(struct pam_response));
356:          reply[0].resp = strdup(pass);
357:          reply[0].resp_retcode = 0;
358:
359:          /* Appel de la function d'authentification */
360:          if((rret = pam_authenticate(pamh, 0)) != PAM_SUCCESS) {
361:              ap_rprintf(r, "NULL %d\n<br/>", errno);
362:              apr_table_setn(r->err_headers_out,
363:                  (PROXYREQ_PROXY == r->proxyreq) ? "Proxy-Authenticate"
364:                                                  : "WWW-Authenticate",
365:                  apr_pstrcat(r->pool, "Basic realm=\"", ap_auth_name(r), "\"",
NULL));
366:              return HTTP_UNAUTHORIZED;
367:          }
368:
369:          /* Deconnexion de PAM */
370:          if(pam_end(pamh, rret) != PAM_SUCCESS) {
371:              ap_rputs("Pam end failed<br/>", r);
372:              return HTTP_INTERNAL_SERVER_ERROR;
373:          }
374:          ap_rputs("Pam success<br/>", r);
375:      }
376:
377:      return OK;
378:  }
379:

Compilation du module avec PAM
380:  /usr/local/apache2/bin/apxs -lpam -lpam_misc -c -i mod_absec.c ; sudo make
install ; sudo /etc/init.d/apache2 restart
```

Fragment 4.18 - Code de l'incorporation de PAM au module

4.3.4 Récupération des groupes supplémentaires

Il est important que la récupération des groupes supplémentaires auxquels appartient l'utilisateur soit effectuée. En effet, pour chacun des groupes trouvés, une comparaison pourra être faite avec le groupe propriétaire de la ressource pour valider les autorisations. Le code nécessaire pour ce faire est illustré dans le Fragment 4.19, le résultat de son application dans le module est présenté à Figure 4.10. (mod_absec_534.c)

```
Exemple récupération des groupes supplémentaires
381:  #include <grp.h>
382:  #include <stdio.h>
383:
384:  char* user="jlcyr";// nom d'utilisateur reçu dans la requête
385:  int g=1000;// group
386:
387:  int main(int argc, char** argv) {
388:
389:      gid_t grouplist[16];
390:      int grouplistsize = 16;
391:      int groupreturn;
392:      groupreturn = getgrouplist(user, g, grouplist, &grouplistsize);
393:      if (groupreturn != -1) {
394:          for (int i=0; i<grouplistsize; i++) {
395:              /* Iteration sur chacun des groups */
396:              printf("%d\r\n", grouplist[i]);
397:          }
398:      } else {
399:          printf("Aucun groupe\r\n");
400:      }
401:  }

Compilation et exécution
402:  gcc exemple_534.c -o a.out; ./a.out
403:  1000
404:  100
```

Fragment 4.19 - Code pour la vérification des groupes supplémentaires

Figure 4.10 - Page du module affichant les informations des groupes supplémentaires

4.3.5 Journalisation des authentifications

La journalisation des tentatives d'authentification a été mise de côté pour le moment. Divers mécanismes peuvent être utilisés tels que l'insertion d'informations dans les journaux du serveur Web, dans les journaux du système ou au travers de la journalisation du module PAM.

4.4 Composants d'autorisation

Comme pour l'authentification, un entête de fonction standard a été établi afin de pouvoir interchanger le composant utilisant le système de fichiers et celui utilisant une base de données. L'entête établi est basé sur les structures de données disponibles, soit « request_rec » (voir « httpd.h ») pour les éléments de la requête et stat (voir « stat.h ») pour les permissions associées à l'objet. Le Fragment 4.20 démontre leur utilisation.

```
405:  int perms_lookup(request_rec *r, struct stat *fperm);
```

Fragment 4.20 - Entête de la fonction d'autorisation

Le but principal ici est de récupérer les permissions associées avec une ressource afin de faire la validation des accès conjointement avec les informations d'authentification préalablement récupérées, les permissions associées à la ressource et l'action demandée au travers du « verbe » utilisé dans le protocole HTTP.

Le cheminement de la validation devra être optimisé afin de favoriser le plus court trajet avant le retour d'une réponse. Le diagramme à la Figure 4.11 illustre ce processus. Ce dernier consiste à répondre aux questions suivantes :

- Est-ce que l'objet est privé (aucune permission)? – retour accès refusé
- Est-ce que l'objet est public (permis à tout/tous)? – retour accepté
- Est-ce que l'utilisateur authentifié correspond à l'utilisateur propriétaire de l'objet et aux droits assignés?
- Est-ce que le groupe de l'utilisateur authentifié correspond au groupe propriétaire de l'objet et aux droits assignés?
- Est-ce qu'un groupe supplémentaire de l'utilisateur authentifié correspond au groupe propriétaire de l'objet aux droits assignés?

Ces validations sont réalisées à partir des informations sur l'utilisateur récupérés en phase d'authentification (login/pass, uid, gid, suppgid), des informations contenues dans la requête (URI, Methode, entête …) et des informations qui seront fournies par le composant d'autorisation sur l'objet accédé (owner, group, owner_mask, group_mask, public_mask).

Tel que présenté au Tableau 2.4, les méthodes HTTP sont validées avec les permissions correspondantes en lecture (r) /écriture (w) et l'attribut d'exécution (x) sera utilisé pour la suppression (GET=r, POST/PUT=w, DELETE=x).

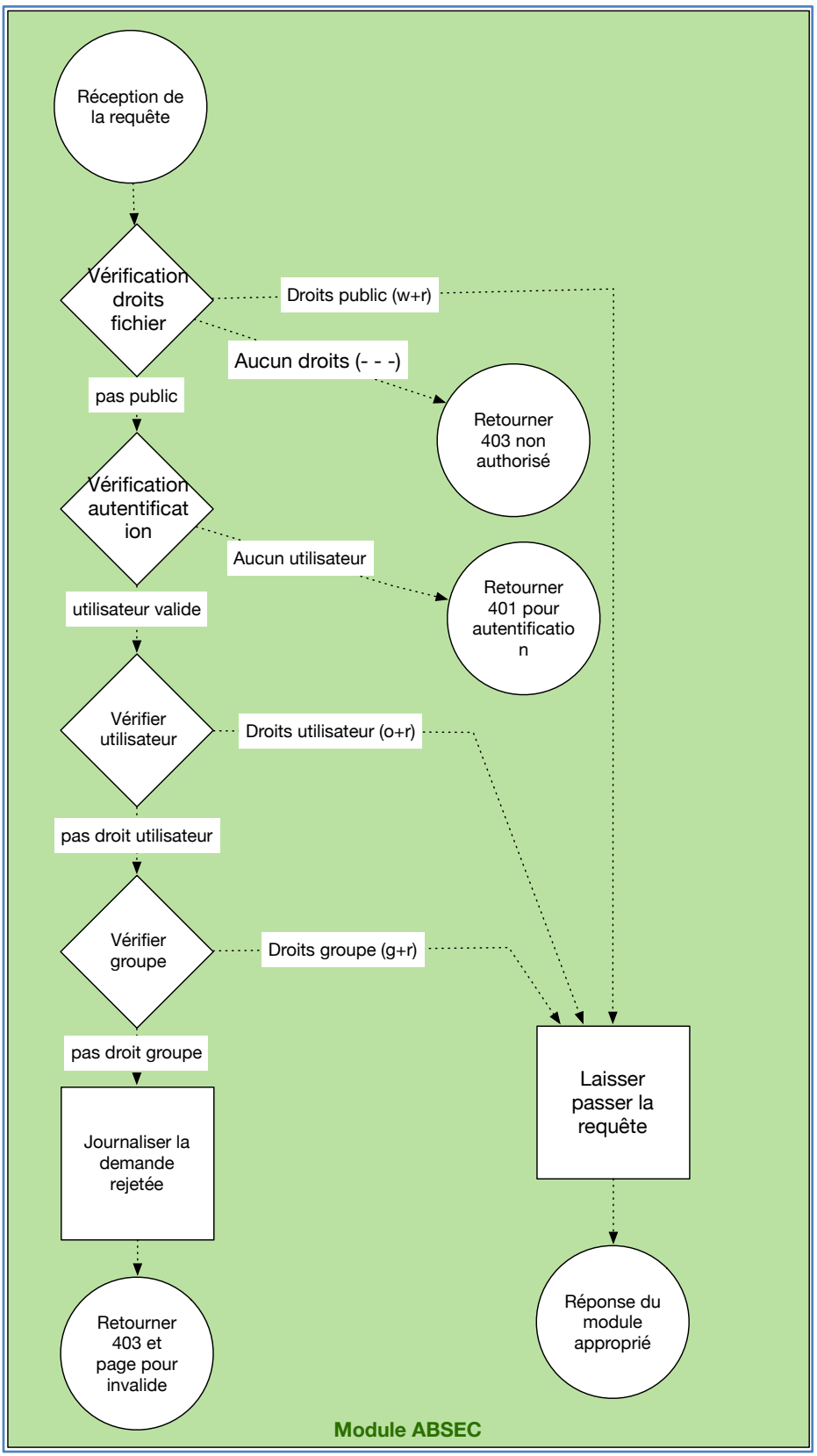


Figure 4.11 - Processus de validation autorisation

4.4.1 Utilisation du système de fichiers pour l'autorisation

L'utilisation du système de fichiers pour la validation des autorisations est triviale dans le cas de ressources statiques (documents existants sur le système de fichiers), cependant dans le cas de la génération dynamique des documents, un défi supplémentaire est présent. En effet, les permissions sont associées aux fichiers et non aux URLs. La relation entre eux doit donc être de type 1 à 1. La structure de base sur les systèmes POSIX pour représenter les autorisations, struct stat [87], comprend plusieurs éléments (Fragment 4.21). Les éléments st_uid, st_gid et st_mode sont suffisants pour conserver les informations minimales nécessaires à notre projet. Notons que cette structure ne peut être modifiée. Deux approches sont possibles pour traiter ces informations soit l'utilisation des fonctions de l'APR (0) ou du système d'exploitation (0).

```
406:  dev_t     st_dev     Device ID of device containing file.
407:  ino_t     st_ino     File serial number.
408:  mode_t    st_mode    Mode of file (see below).
409:  nlink_t   st_nlink   Number of hard links to the file.
410:  uid_t     st_uid     User ID of file.
411:  gid_t     st_gid     Group ID of file.
412:  dev_t     st_rdev    Device ID (if file is character or block special).
413:  off_t     st_size    For regular files, the file size in bytes.
414:                       For symbolic links, the length in bytes of the
415:                       pathname contained in the symbolic link.
416:                       For a shared memory object, the length in bytes.
417:                       For a typed memory object, the length in bytes.
418:                       For other file types, the use of this field is
419:                       unspecified.
420:  time_t    st_atime   Time of last access.
421:  time_t    st_mtime   Time of last data modification.
422:  time_t    st_ctime   Time of last status change.
423:  blksize_t st_blksize A file system-specific preferred I/O block size for
424:                       this object. In some file system types, this may
425:                       vary from file to file.
426:  blkcnt_t  st_blocks  Number of blocks allocated for this object.
```

Fragment 4.21 - Structure de donnée "stat" contenant les métadonnées des fichiers

Utilisation de l'APR

Les fonctions de l'APR sont triviales à utiliser. Le nom du fichier est passé en paramètre et une structure similaire à « stat » propre à l'APR est retournée. Les informations complètes sont présentées dans le Fragment 4.22 et l'incorporation à notre module donne le résultat montré à la Figure 4.12. (mod_absec_5411.c)

```
Définition de la structure
427:  typedef struct apr_finfo_t    apr_finfo_t

428:  apr_pool_t *      pool
429:  apr_int32_t       valid
430:  apr_fileperms_t   protection // correspond à st_mode
431:  apr_filetype_e    filetype
432:  apr_uid_t   user // correspond à st_uid
433:  apr_gid_t   group // correspond à st_gid
434:  apr_ino_t   inode
435:  apr_dev_t   device
436:  apr_int32_t       nlink
437:  apr_off_t   size
438:  apr_off_t   csize
439:  apr_time_t  atime
440:  apr_time_t  mtime
441:  apr_time_t  ctime
442:  const char *      fname
443:  const char *      name
444:  struct apr_file_t *     filehand

Définition de la fonction
445:  apr_status_t      apr_stat (apr_finfo_t *finfo, const char *fname, apr_int32_t
wanted, apr_pool_t *pool)

Exemple d'utilisation
446:       struct apr_finfo_t finfo;
447:       int apr_status_t;
448:       apr_status_t = apr_stat (&finfo, r->filename, APR_FINFO_NORM, r->pool);
```

Fragment 4.22 - Code pour l'utilisation de "fstat" au travers de l'APR

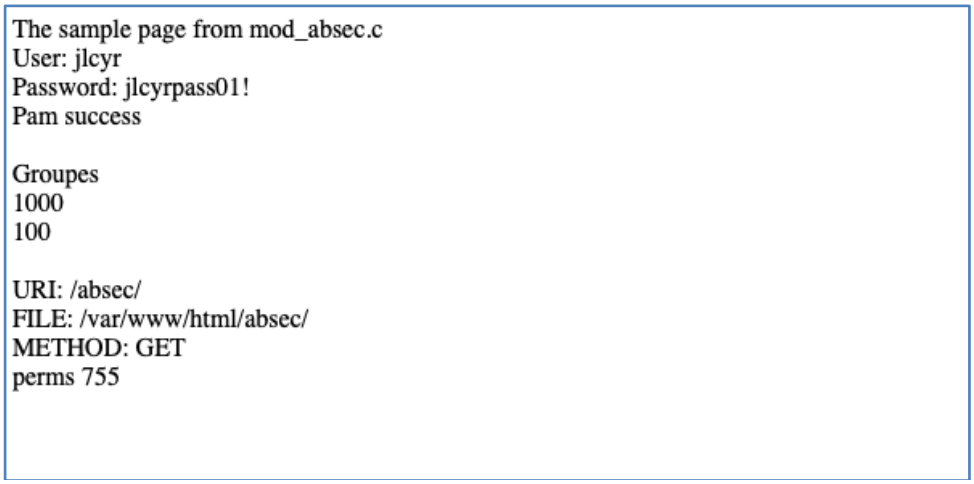


Figure 4.12 - Page du module affichant les informations de permissions des fichiers

Ces fonctions donnent le résultat escompté dans le cadre de l'utilisation du module. Cependant, pour le développement d'outils de configuration, l'utilisation des commandes

standards du système est probablement plus adéquate, ne nécessitant pas l'incorporation de l'APR pour de simples petits outils en ligne de commande.

Utilisation des fonctions du système d'exploitation

Les fonctions du système d'exploitation sont également triviales à utiliser. Le nom du fichier est passé en paramètre de même qu'un pointeur vers la structure « stat » (Fragment 4.21). Cette dernière est remplie avec les informations au retour de l'appel. L'exemple d'utilisation est présenté dans le Fragment 4.23 et l'incorporation à notre module donne le résultat montré à la Figure 4.13. (mod_absec_5412.c)

```
449:  ////////
450:  /* check file permission on filesystem */
451:  /* should #include <sys/stat.h> */
452:      struct stat fperm;
453:      int status;
454:      status = stat(r->filename, &fperm);
455:      ap_rprintf(r, "File perms %o, owner %d, group %d (status %d)",
fperm.st_mode, fperm.st_uid, fperm.st_gid, status);
456:
457:      return OK;
458:  }
```

Fragment 4.23 - Exemple d'utilisation de la fonction stat

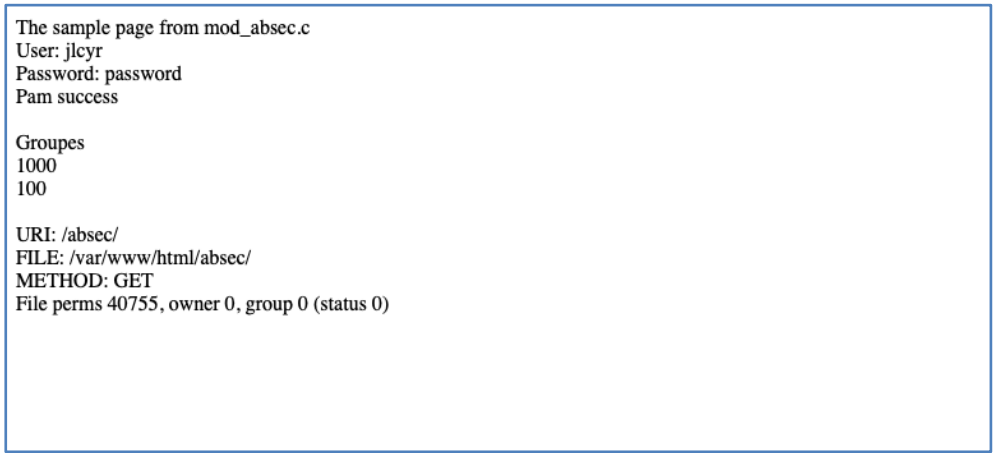


Figure 4.13 - Page du module affichant les informations le résultat de l'appel à stat

Dans les deux cas, il subsiste un problème de consultation des permissions si l'utilisateur n'a pas de droits sur le fichier. Nous nous retrouvons donc avec la même problématique que pour l'authentification et l'accès à /etc./shadow. L'utilisation d'Apache en mode super-

utilisateur (« root ») a été mise de côté à la section 4.3.2, car elle ne solutionnait pas notre problème. De plus, l'ajout du groupe Shadow ne peut pas être non plus la solution. L'inclusion de l'utilisateur à tous les groupes utilisés par les ressources pourrait être une solution, mais elle serait plutôt fastidieuse. Plutôt que d'élaborer des solutions plus complexes, il est préférable de considérer une alternative plus simple, soit l'utilisation d'une base de données. En effet le problème principal associé à cette approche provient du fait qu'il faut créer une arborescence complète de sous-répertoires et de fichiers pour représenter les URL devant exister. De plus, les entrées « stat » liées à chacun des éléments comportent les informations nécessaires pour la gestion des autorisations (uid, gid, perms), mais ne nous laissent pas beaucoup de latitude pour inclure des éléments de journalisation autre que la dernière date d'accès. Les problèmes de permission combinés avec les limitations du système nous dirigent donc vers la seconde alternative identifiée soit l'utilisation d'une base de données.

4.4.2 Utilisation d'une base de données pour l'autorisation

Contrairement à la méthode présentée à la section 4.4.1, l'utilisation d'une base de données pour maintenir à jour les permissions associées à des ressources n'oblige pas l'existence de cesdites ressources de manière statique dans le système. De plus, l'arborescence complète n'a pas besoin d'exister et peut être calculée. Son utilisation avec des générateurs de contenu dynamique est donc simplifiée. En contrepartie, des outils propres doivent être développés pour la gestion de cette base de données. Elle sous-tend également l'hypothèse #7.

Installation de MySQL

L'Installation du moteur de base de données, du client en ligne de commande et de l'environnement de développement forme la première étape illustrée dans le Fragment 4.24.

```
459:  sudo apt-get install mysql-server mysql-client
460:  sudo apt-get install libmysqlclient-dev
```

Fragment 4.24 - Commandes pour l'installation de MySQL

Format de la base de données

Par la suite, nous créons une base de données d'autorisation (voir Fragment 4.25) basée sur les informations utilisées de la structure du système de fichiers « stat » présenté dans le Fragment 4.21. Il sera possible d'ajouter des éléments de journalisation dans une phase ultérieure.

```
461:  CREATE TABLE `urls` (
462:    `url` varchar(64) NOT NULL DEFAULT '/',
463:    `uid` int(11) DEFAULT '0',
464:    `gid` int(11) DEFAULT '0',
465:    `perms` int(11) DEFAULT '0',
466:    PRIMARY KEY (`url`),
467:    UNIQUE KEY `url_UNIQUE` (`url`)
468:  )
```

Fragment 4.25 - Modèle de la base de données

```
Exemple d'utilisation
469:  /* Simple C program that connects to MySQL Database server*/
470:  #include <mysql.h>
471:  #include <stdio.h>
472:
473:  int main() {
474:     MYSQL *conn;
475:     MYSQL_RES *res;
476:     MYSQL_ROW row;
477:
478:     char *server = "localhost";
479:     char *user = "jlcyr";
480:     char *password = "password"; /* set me first */
481:     char *database = "mysql";
482:
483:     conn = mysql_init(NULL);
484:
485:     /* Connect to database */
486:     if (!mysql_real_connect(conn, server,
487:           user, password, database, 0, NULL, 0)) {
488:        fprintf(stderr, "%s\n", mysql_error(conn));
489:        exit(1);
490:     }
491:
492:     /* send SQL query */
493:     if (mysql_query(conn, "show tables")) {
494:        fprintf(stderr, "%s\n", mysql_error(conn));
495:        exit(1);
496:     }
497:
498:     res = mysql_use_result(conn);
499:
500:     /* output table name */
501:     printf("MySQL Tables in mysql database:\n");
502:     while ((row = mysql_fetch_row(res)) != NULL)
503:        printf("%s \n", row[0]);
504:
505:     /* close connection */
506:     mysql_free_result(res);
507:     mysql_close(conn);
508:  }
```

```
509:  gcc mysql_test.c $(mysql_config --libs) -I/usr/include/mysql/ -o mysql_test

La compilation du module avec MySQL deviendra

510:  apxs -lpam -lpam_misc $(mysql_config --libs) -c -I/usr/include/mysql/ -i
mod_absec.c ; sudo make install ; sudo /etc/init.d/apache2 restart
```

Fragment 4.26 - Code d'exemple pour l'utilisation de MySQL

Connexion et manipulation

Avec ces éléments en place, il est possible de créer un test de connexion à MySQL et de tester l'exécution d'une requête (voir Fragment 4.26) [88]. La compilation et la liaison doivent incorporer des paramètres supplémentaires pour accéder aux entêtes de développement mysql et aux bibliothèques dynamiques d'où l'introduction des arguments « $(mysql_config --libs) » pour les bibliothèques et « -I/usr/include/mysql/ » pour les entêtes. À noter que la sous-commande $(mysql_config --lib) permet d'obtenir les options de compilation correspondantes à l'installation de MySQL. Nous pouvons appliquer cette option à la vérification des permissions associées à une URL dans notre module (voir Fragment 4.27) (mod_absec_5423.c) et on peut en vérifier l'exécution à la Figure 4.14.

```
511:
512:  MYSQL *conn;
513:  MYSQL_RES *res;
514:  MYSQL_ROW row;
515:
516:  char *server = "localhost";
517:  char *user = "jlcyr";
518:  char *password = "password"; /* set me first */
519:  char *database = "absec";
520:
521:  conn = mysql_init(NULL);
522:
523:  /* Connect to database */
524:  if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
525:     printf("%s\n", mysql_error(conn));
526:     return(0);
527:  }
528:
529:  char query[256];
530:  sprintf(query, "select * from urls where url='%s'", r->uri);
531:  /* send SQL query */
532:  //if (mysql_query(conn, "show tables")) {
533:  if (mysql_query(conn, query)) {
534:     ap_rprintf(r, "%s\n", mysql_error(conn));
535:     return(-1);
536:  }
537:
538:  res = mysql_use_result(conn);
539:
540:  /* output table name */
541:  ap_rprintf(r, "MySQL data:\n<br/>");
542:  int cnt = 0;
543:  while ((row = mysql_fetch_row(res)) != NULL) {
544:     ap_rprintf(r, "Perms: %s %d %d %o \n<br/>", row[0], atoi(row[1]),
atoi(row[2]), atoi(row[3]));
```

```
545:     cnt = cnt + 1;
546:  }
547:
548:  /* close connection */
549:  mysql_free_result(res);
550:  mysql_close(conn);
```

Fragment 4.27 - Incorporation du code de MySQL au module

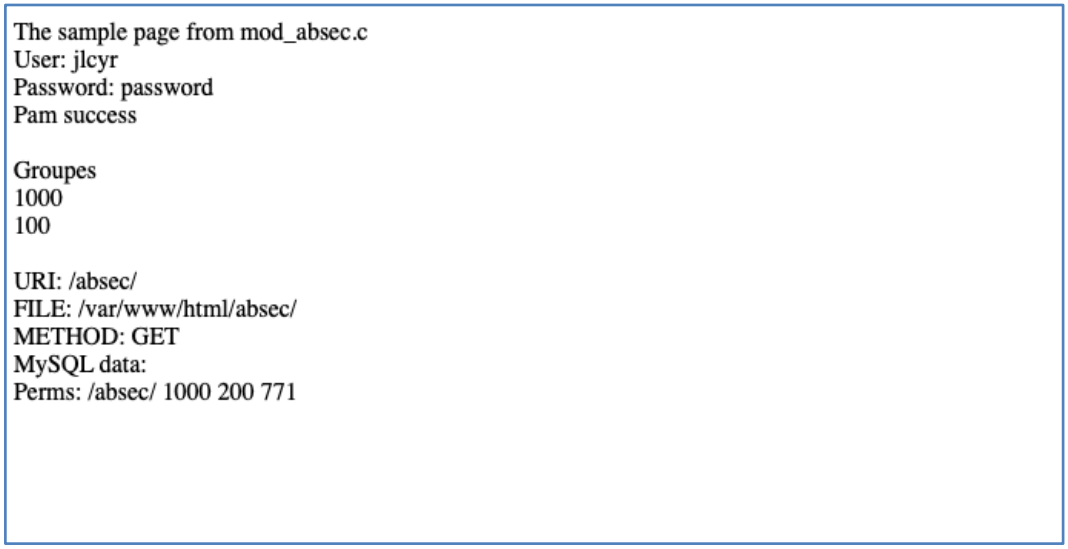


Figure 4.14 - Page du module affichant les informations reçues de MySQL

4.4.5 Journalisation des autorisations

La journalisation des autorisations a été mise de côté pour le moment. Divers mécanismes peuvent être utilisés tels que l'insertion d'informations dans les journaux du serveur Web, dans les journaux du système. Il est envisagé, pour plus de souplesse, d'inclure ultérieurement une journalisation complète dans la base de données. Dans une première phase, seules les requêtes pour des URLs non définis seront journalisées afin d'identifier les lacunes dans la définition des autorisations.

4.5 Création d'un module complet

La création d'un module complet peut se décliner en quatre versions combinant successivement les composants d'authentification et d'autorisation développées précédemment. Cependant, comme notée à la section 5.4.1, l'utilisation du système de fichiers

pour conserver les autorisations est limitative. Cela nous force à éliminer les combinaisons utilisant cette option.

Deux versions du module ont donc été développées, soit celle utilisant /etc./passwd et une base de données, et celle utilisant PAM et une base de données (voir Figure 4.15). Le module d'authentification présenté à la section 4.3 est relativement complet. Cependant, le module d'autorisation décrit à la section 4.4 doit être amélioré pour associer la validation de la requête (GET/POST/PUT/DELETE) et les divers éléments d'authentification (UID, GROUP, GROUP SUPP). Les éléments de la structure « stat » (perms, uid, gid) sont utilisés tels que décrits dans le Fragment 4.28 pour valider la requête. De plus, dans la version complète du module, pour permettre le traitement des requêtes par d'autres « handlers » tel que « mod_php », on doit utiliser un mode d'intégration au serveur différent de « handler ». Les premières informations recueillies suggèrent l'utilisation du mode « filter ».

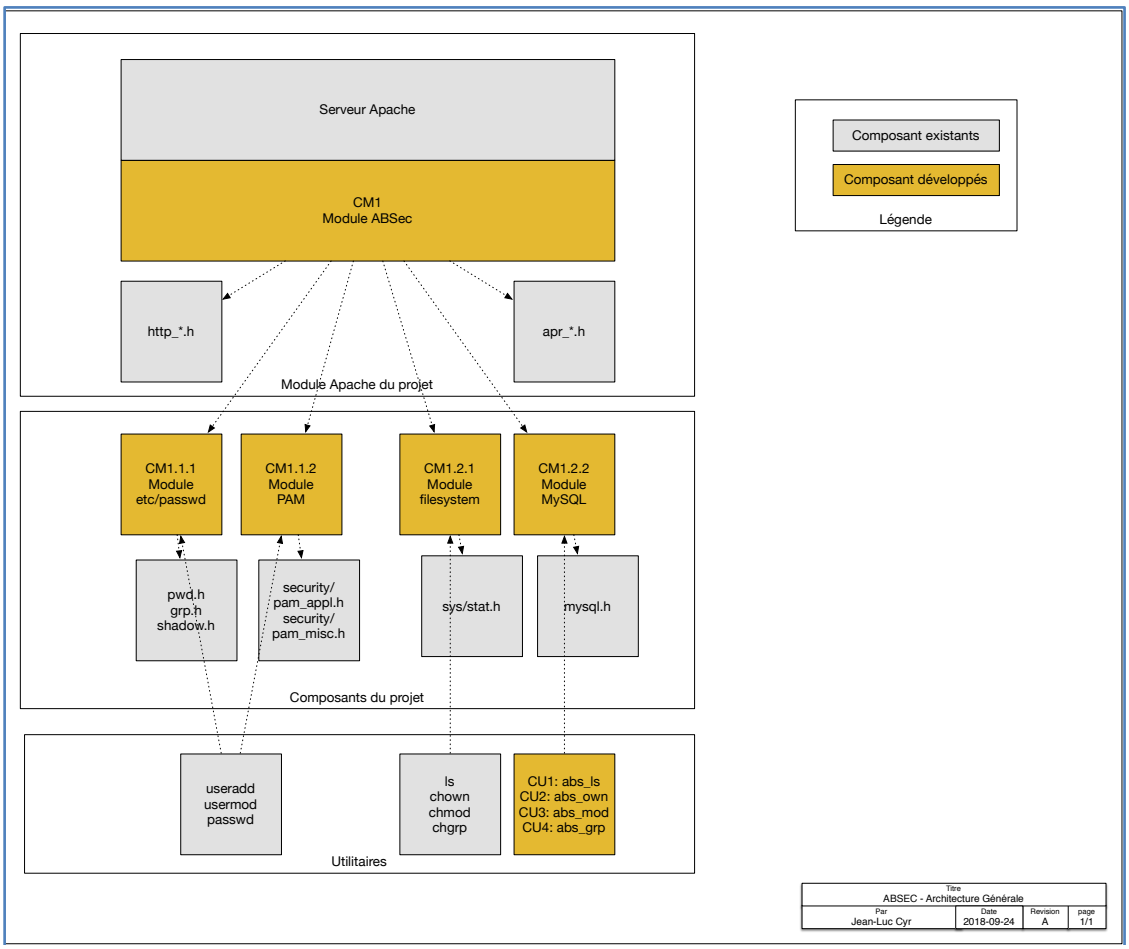


Figure 4.15 - Sous modules du projet

```
551:  // Creation d'un masque de validation à partir de la méthode HTTP
552:  int permmask = 0;
553:  if (strcmp(r->method,"GET")==0) permmask=0444;     // r
554:  if (strcmp(r->method,"PUT")==0) permmask=0222;     // w
555:  if (strcmp(r->method,"POST")==0) permmask=0222;    // w
556:  if (strcmp(r->method,"DELETE")==0) permmask=0111;  // x
. . .

557:  /* check if any permission (ogw) match method (get r, put/post w, delete x) */
558:  if ((fperm.st_mode & permmask)==0) {
559:      /* no permission match, return don't even have to check user perms */
560:      ap_rprintf(r, "Aucune permission pour la méthode %s (%o, %o)", r->method,
fperm.st_mode, permmask);
561:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Aucune permission pour la méthode %s (%o, %o)", r->method,
fperm.st_mode, permmask);
562:      return AUTHZ_DENIED;
563:  }
564:
565:  // If file is world accessible for asked method return content
566:  if (fperm.st_mode & 0x7 & permmask) {
567:      /* if so, return, no need to check user perms */
568:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier accessible a tous");
569:      return AUTHZ_GRANTED;
570:  }
571:
. . .

572:
573:  // If file is user readable and user match return content
574:  if ((fperm.st_uid==pw->pw_uid) && (fperm.st_mode & 0700 & permmask)) {
575:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier propriétaire<br/>\r\n");
576:      return AUTHZ_GRANTED;
577:  } else {
578:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : No user access %d", pw->pw_uid);
579:  }
580:
581:  // If file is group readable and primary group match return content
582:  if ((fperm.st_gid==pw->pw_gid) && (fperm.st_mode & 0070 & permmask)) {
583:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier groupe<br/>\r\n");
584:      return AUTHZ_GRANTED;
585:  } else {
586:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : No group access %d", pw->pw_gid);
587:  }
588:
589:  // check supplemental groups
590:  gid_t grouplist[16];
591:  int grouplistsize = 16;
592:  int groupreturn;
593:  groupreturn = getgrouplist(user, pw->pw_gid, grouplist, &grouplistsize);
594:  if (groupreturn >= 0) {
595:      ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : liste groups (%d)", grouplistsize);
596:      for (int i=0; i<grouplistsize; i++) {
597:          ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : group %d", grouplist[i]);
598:          // If file is group readable and match a supplemental group return
content
599:          if ((fperm.st_gid==grouplist[i]) && (fperm.st_mode & 0070 & permmask)) {
600:              ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier group supp");
601:              return AUTHZ_GRANTED;
602:          }
603:      }
604:  }
```

Fragment 4.28 - Code de validation de la requête en fonction des permissions

Afin de pouvoir cohabiter avec d'autres modules Apache de génération de contenu tel que mod_php, notre module doit être modifié pour être invoqué en premier et permettre de poursuivre au prochain « hook » si les permissions sont valides. En effet l'invocation de notre module comme « hook », utilisé jusqu'à maintenant, fait que peu importe l'URL invoquée nous obtenons en retour la page générée par notre module. Par exemple, les appels à /absec/, /absec/test.html et /absec/test.php nous retournent tous les trois la même page avec pour seule variante l'URI et le fichier demandé qui est affiché (Figure 4.16). C'est la démonstration que ni le module standard pour retourner un fichier (ex. : HTML) ni celui pour exécuter le code PHP n'est invoqué.

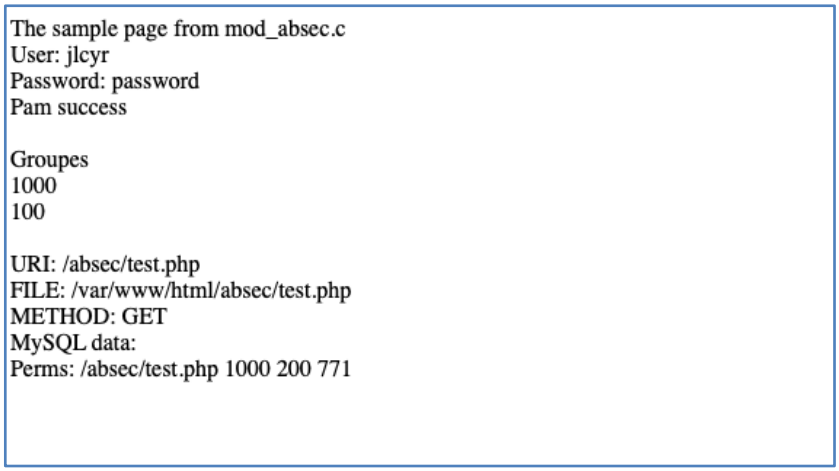


Figure 4.16 - Page du module affichant un exemple de non-cohabitation des modules

Pour obtenir le fichier HTML ou forcer l'exécution du code PHP, il faut effectuer quelques modifications. La première consiste à redéfinir l'ordre d'appel de notre module pour le mettre en premier (changement de APR_HOOK_MIDDLE pour APR_HOOK_FIRST) puis de modifier le code de retour pour poursuivre la chaine d'exécution (changement du retour OK pour DECLINED) tel qu'illustré au Fragment 4.29.

De plus, la configuration doit être changée comme indiqué au Fragment 4.30 pour utiliser les filtres plutôt que les « handlers ». Nous pouvons ensuite constater que le fichier « PHP » s'exécute bien après le code de notre module tel qu'illustré à la Figure 4.17.

```
Changer le code

605:  static void absec_register_hooks(apr_pool_t *p)
606:  {
607:      ap_hook_handler(absec_handler, NULL, NULL, APR_HOOK_MIDDLE);
608:  }

par

609:  static void absec_register_hooks(apr_pool_t *p)
610:  {
611:      ap_hook_handler(absec_handler, NULL, NULL, APR_HOOK_FIRST);
612:  }

Et changer les retours

613:  // Retourner le contenu
614:  return (OK);

par

615:  // Passer le contrôle au prochain “hook”
616:  return (DECLINED);
--- Et laisser notre module s’exécuter même s’il n’est pas le hook car il ne le sera
jamais

617:  //if (strcmp(r->handler, "absec")) {
618:  //    return DECLINED;
619:  //}
```

Fragment 4.29 - Code nécessaire pour le changement de "hook" à "filter"

```
620:          <Location /absec>
621:           #SetHandler absec
622:           SetInputFilter absec
623:           #SetOutputFilter absec
624:           </Location>
```

Fragment 4.30 - Configuration d'Apache pour l'utilisation du module filter

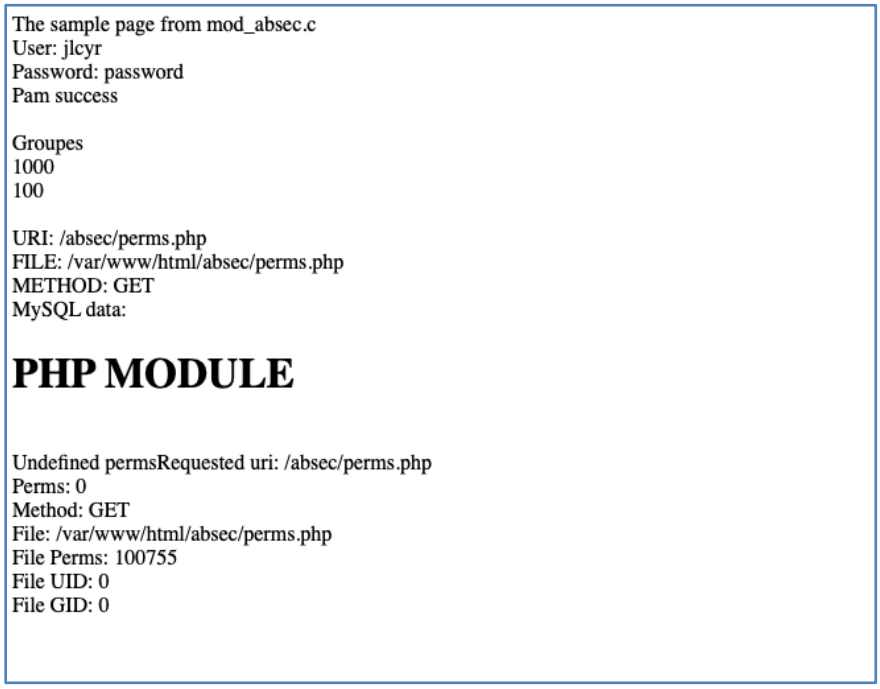


Figure 4.17 - Page du module démontrant la cohabitation du module et mod_php

À cette étape, nous ajoutons également la possibilité d'inclure certains paramètres dans la configuration d'Apache, dont les permissions par défaut à utiliser en cas d'absence de permission pour une URL.

Lors du développement de la version du module en mode « filter », les informations trouvées nous ont amenés à considérer une autre méthode d'intégration intéressante pour notre module. En effet, les informations contenues dans « The Apache Modules Book » au chapitre 7 [45] nous orientent vers l'utilisation de « authn/authz » plutôt que le mode « filter ». Cependant le livre ayant été écrit pour la version 2.2, il faut adapter les directives pour la version 2.4 en utilisant des informations disponibles en ligne. Basé sur le code des modules existants d'Apache, particulièrement le code en lien avec « authn/authz » [89], nous obtenons la structure de code présentée au Fragment 4.31. On peut constater que le code a été découpé en deux fonctions distinctes pour l'authentification et l'autorisation. Chacun a été lié à la

fonction « authn/authz » correspondante. De plus, les valeurs de retour ont dû être adaptées aux valeurs appropriées d'authentification/autorisation plutôt qu'aux retours standards.

```
Déclaration et initialisation du module
625:  //////////////////////////////////////////////////////////////////
626:  static const authn_provider authn_absec_provider =
627:  {
628:      &authn_check_absec,
629:      NULL
630:  };
631:
632:  //////////////////////////////////////////////////////////////////
633:  static const authz_provider authz_absec_provider =
634:  {
635:      &authz_check_absec,
636:      NULL
637:  };
638:
639:  //////////////////////////////////////////////////////////////////
640:  static void absec_register_hooks(apr_pool_t *p)
641:  {
642:      ap_register_auth_provider(p, AUTHN_PROVIDER_GROUP, "absec", "0",
&authn_absec_provider, AP_AUTH_INTERNAL_PER_CONF);
643:      ap_register_auth_provider(p, AUTHZ_PROVIDER_GROUP, "absec", "0",
&authz_absec_provider, AP_AUTH_INTERNAL_PER_CONF);
644:  }
645:
646:  //////////////////////////////////////////////////////////////////
647:  /* Dispatch list for API hooks */
648:  AP_DECLARE_MODULE(absec) = {
649:      STANDARD20_MODULE_STUFF,
650:      authnz_absec_config,   /* create per-dir    config structures */
651:      NULL,                  /* merge  per-dir    config structures */
652:      NULL,                  /* create per-server config structures */
653:      NULL,                  /* merge  per-server config structures */
654:      absec_auth_basic_cmds, /* table of config file commands       */
655:      absec_register_hooks   /* register hooks                      */
656:  };

Valeurs de retour authn
AUTH_DENIED
AUTH_GRANTED

Valeurs de retour authz
AUTHZ_DENIED
AUTHZ_GRANTED
AUTHZ_DENIED_NO_USER
```

Fragment 4.31 - Modification du code pour l'utilisation de Authn et Authz

```
Fichier /etc./apache2/sites-enables/000-default.conf

657:  <Directory "/var/www/HTML">
658:        <IfModule absec_module>
659:              AuthName "Absec"
660:              AuthType Basic
661:              AuthBasicProvider absec
662:              Require absec
663:              ABSECDefaultPerms 700
664:              ABSECDefaultUID 100
665:              ABSECDefaultGID   1000
666:        </IfModule>
667:        Options FollowSymLinks Indexes
668:        AllowOverride None
669:        Order allow,deny
670:        allow from all
671:  </Directory>
```

Fragment 4.32 - Configuration d'Apache pour utiliser le module sous la forme d'Authn et Authz

La configuration du serveur doit également être modifiée pour faire appel à cette version du module. Les paramètres doivent être ajoutés dans la configuration d'un répertoire sous forme de gestion des permissions, plutôt que d'associer un « handler » ou un « filter » tel que montré au Fragment 4.32.

La structure de notre code permet, en utilisant les options de compilation, de générer les deux versions qui seront utilisées pour les tests. En effet, la sélection peut se faire au moment de la liaison, en choisissant parmi les modules d'authentification (absec_pam et absec_etc) et d'autorisation (absec_fs et absec_mysql).

4.6 Journalisation des accès

Comme présenté aux sections 4.3.5 et 0, l'implémentation de la journalisation des accès a été considérée comme une extension au projet et n'a pas été implémentée sauf pour conserver une trace des appels à des ressources pour lesquelles aucune autorisation n'est définie.

4.7 Création d'outils de gestion

Le développement en sous-composants du module a été réalisé afin de pouvoir réutiliser ces derniers pour développer des outils de gestion complémentaires au cas où les outils système ne seraient pas disponibles comme, par exemple, lorsque la base de données pour la gestion des autorisations est employée. Les outils à utiliser pour la gestion des authentifications et des autorisations selon l'arrimage des composants sont présentés dans ce qui suit.

4.7.1 Module /etc./passwd

Dans le cas du composant requérant l'accès aux fichiers /etc./passwd et /etc./shadow pour l'authentification, l'utilisation des outils « système » est possible pour la gestion. Les commandes disponibles sont :

Useradd
Usermod
Groupadd

4.7.2 Module PAM

Dans le cas de l'utilisation du composant PAM, les mêmes outils que pour l'utilisation directe des fichiers /etc./passwd et /etc./shadow sont disponibles.

4.7.3 Module filesystem

Pour la gestion des autorisations lors de l'utilisation du système de fichiers du système d'exploitation, plusieurs outils système sont disponibles tels que les commandes :

Ls
Chown
Chgrp
Chmod
Mkdir
Chdir
Touch

4.7.4 Module database

Dans le cas de la gestion des autorisations à l'aide d'une base de données, aucun outil système n'existe. Nous avons donc développé des outils complémentaires.

Le gabarit de base de ces fonctions a été calqué sur les fonctions du système présentées en 4.7.3. Le préfixe « ab » a été ajouté aux noms des commandes. Les utilitaires en ligne de commande acceptent, en paramètre, le nom de la ressource sous forme d'URI et sensiblement

les mêmes paramètres pour changer les autorisations. Des pages d'informations (« unix man page ») [90] [91] ont également été créées pour décrire ces outils.

Pour la version expérimentale, la connexion à la base de données a été incluse directement dans le code. Dans une version finale, elle devrait être paramétrisable pour l'ensemble des outils grâce à un fichier de configuration commun (ex : /etc./absec/autorization.conf).

Les commandes créées sont les suivantes :

abls : liste des ressources

Exemple de la « man page » correspondant

```
672:  man(1)                            abls man page
man(1)
673:
674:  NAME
675:         abls - list defined URI permission in database
676:
677:  SYNOPSIS
678:         abls [URI Pattern]
679:
680:  DESCRIPTION
681:         abls is high level shell program for listing URI permission defined in
the
682:         database.  This database configure ABSEC Apache HTTPD Module.
683:
684:         Exit status:
685:
686:         0   if OK
687:
688:         -1  if ERROR
689:
690:  OPTIONS
691:         The abls does not take any options. However, you can supply URI Pattern
to
692:         list.
693:
694:  SEE ALSO
695:         abchmod(1), abchgrp(1), abchown(1)
696:
697:  BUGS
698:         No known bugs.
699:
700:  COPYRIGHT
701:         Copyright © 2019 ADN Informatique, Enr.  License  GPLv3+:  GNU  GPL
version  3
702:         or  later <http://gnu.org/licenses/gpl.html>.   This  is  free  software:
you
703:         are free to change and redistribute it.  There is NO WARRANTY, to the
extent
704:         permitted by law.
705:
706:  AUTHOR
707:         Jean-Luc Cyr (jlcyr@adninformatique.com)
708:
709:  1.0                                 05 March 2019
man(1)
```

abchown : changement du propriétaire d'une ressource

abchgrp : changement du groupe propriétaire d'une ressource

abchmod : changement des droits associés au propriétaire, groupe et public pour une ressource

4.8 Développement d'un site Web de diffusion

À la rédaction de ce document, les informations relatives au projet ont été placées sur un site Web identifié par l'adresse http://absec.adninformatique.com/ pour la diffusion au public. L'ensemble de la documentation et du code source y sont disponibles. Le développement de ce site a fait partie du présent projet.

CHAPITRE 5

Tests et Analyse

Le module développé a été mis en place et validé dans différents environnements. Le premier est un test indépendant, le second dans un environnement expérimental et le dernier devant un environnement similaire à de la production.

5.1 Tests de performance sous différentes conditions

Les tests de performance varient grandement en fonction du matériel et des logiciels utilisés. Une variabilité supplémentaire est ajoutée par l'utilisation d'environnements virtuels. Les mesures de performance obtenues lors des tests doivent donc être comparées de manière relative et non en termes absolus. De plus, l'utilisation de MatLab pour l'automatisation des tests induit un décalage des résultats en comparaison avec AB (Apache Benchmark). La comparaison des résultats laisse donc part à interprétation sans le calcul exact de ce décalage.

Les premiers tests de performance ont été réalisés avec l'outil AB de l'Apache Foundation. Cet outil se limite à l'exécution de requêtes auprès du serveur et ne permet pas d'automatiser le changement de configuration entre les requêtes. Ces tests confirment les observations notées lors du développement du module avec l'utilisation de PAM (Pluggable Authentication Module). En effet, on a noté un temps de réponse relativement important lors de la vérification de l'authentification.

Avant le début des tests, le chargement de la page par défaut d'Apache (/index.html) a été validé (Figure 5.1), de même que le chargement d'un page simple en langage PHP (/test.php) (Figure 5.2). Les fichiers associés à ces deux pages ont été copiés dans le sous-répertoire /absec/ où le module a pu être activé.

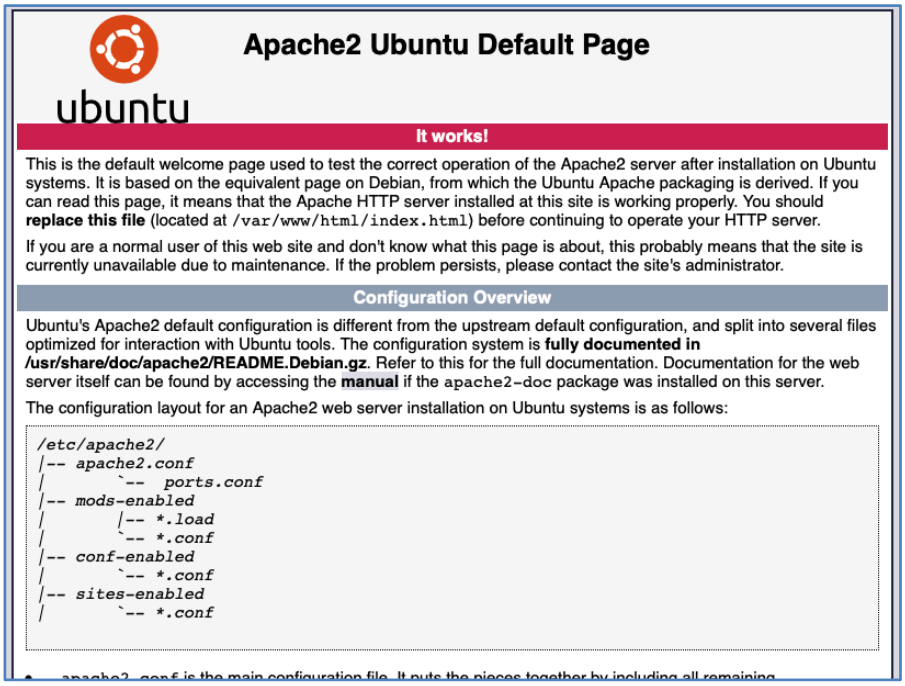


Figure 5.1 - Page de test HTML d'Apache

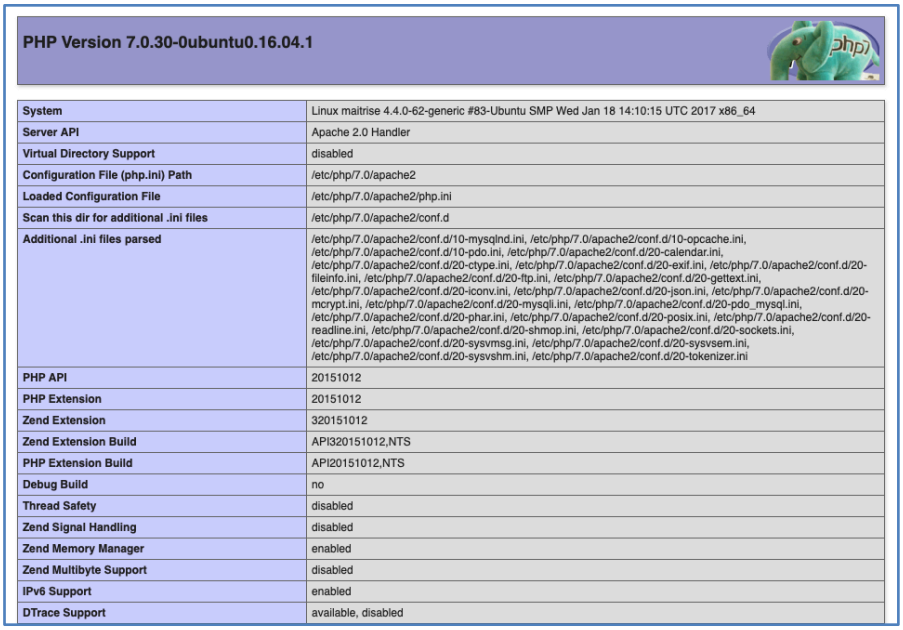


Figure 5.2 - Page de test de PHP

Après l'activation du module dans la configuration du serveur Apache à l'aide des directives présentées au Fragment 5.1 la demande d'authentification présentée à la Figure 5.3 a été obtenue (pour les pages /absec/index.html et /absec/test.php).

```
710:        <Directory "/var/www/HTML/absec">
711:              <IfModule absec_module>
712:                    AuthName "Absec"
713:                    AuthType Basic
714:                    AuthBasicProvider absec
715:                    Require absec
716:                    ABSECDefaultPerms 700
717:                    ABSECDefaultUID 100
718:                    ABSECDefaultGID   1000
719:              </IfModule>
720:              Options FollowSymLinks Indexes
721:              AllowOverride None
722:              Order allow,deny
723:              allow from all
724:        </Directory>
```

Fragment 5.1 - Configuration d'Apache pour le module

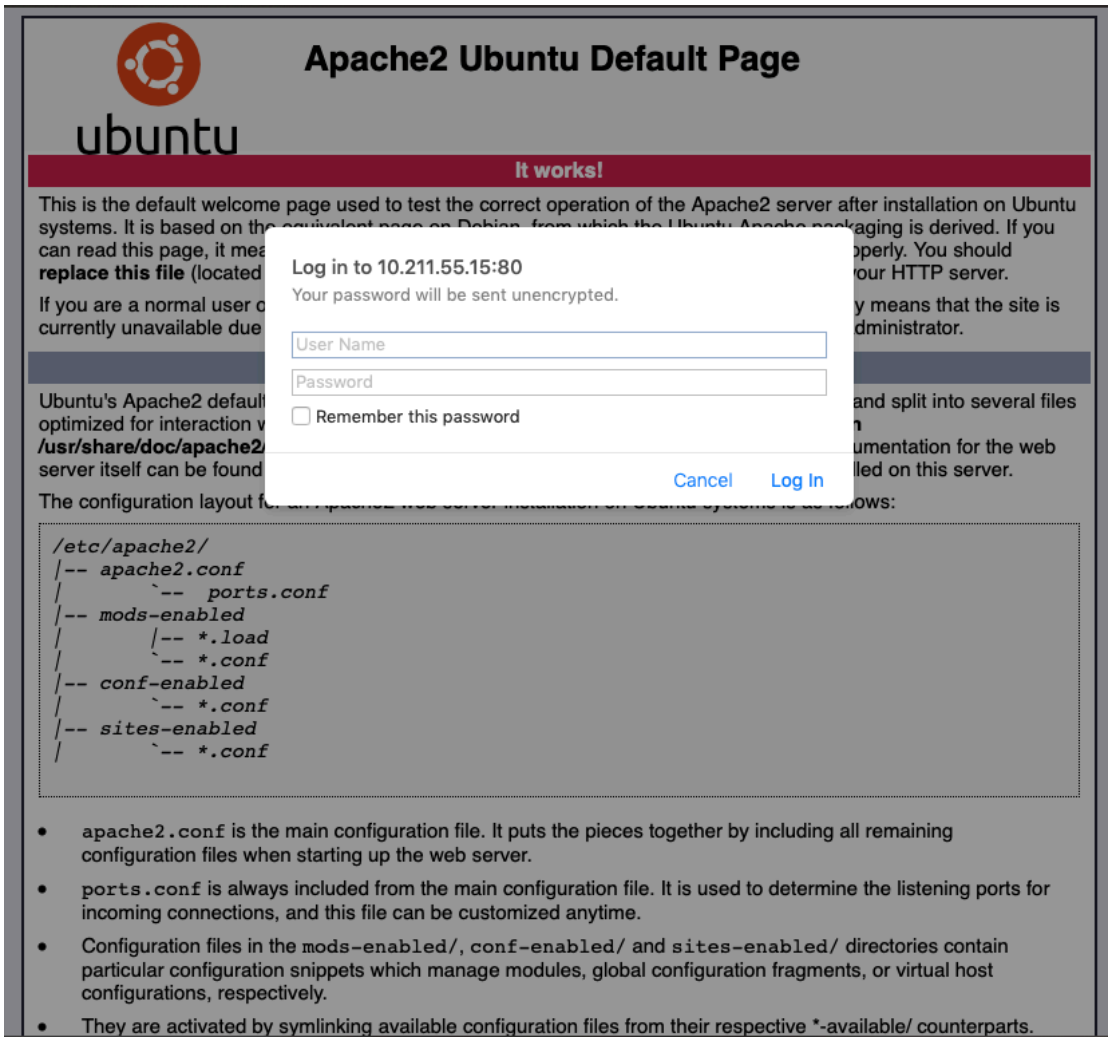


Figure 5.3 - Boite de dialogue affichée par la demande d'authentification

Des mesures de temps de réponse ont été réalisées pour les cas suivants :

1. page sans le module (donc accessible à tous)
2. page accessible à tous 777 (avec module pam et /etc./passwd)
3. page non accessible 000 (avec module pam et /etc./passwd)
4. page accessible à un utilisateur 700 – connexion valide (avec module pam et /etc./passwd)
5. page accessible à un utilisateur 700 – connexion invalide (avec module pam et /etc./passwd)

Figure 5.4 - Liste des cas testés

Comme montré à la Figure 5.4, une première mesure a été réalisée sur une page non protégée n'utilisant pas le module (cas 1). Les autres cas (2 à 5) ont permis d'évaluer le temps d'accès à la page sous différents privilèges d'accès, et ce, en utilisant les deux versions du module.

5.1.1 Tests avec « ab »

Les tests menés avec le client « ab » (Apache Benchmark) furent lancés avec la commande présentée dans le Fragment 5.2. Cette commande permet d'effectuer cent (100) requêtes successives sur le serveur avec les paramètres fournis et affiche les temps de réponse en millisecondes. Un exemple de rapport complet du test est présenté à la Figure 5.5. On y retrouve des informations sur la requête ainsi que des temps de réponse. Pour faciliter l'analyse, le sommaire des temps de réponse obtenus pour l'ensemble des cas est présenté dans le Tableau 5.1.

```
725:
726:  ab [-A jlcyr:password] -n 100 http://10.211.55.15/
727:
```

Fragment 5.2 - Exemple d'utilisation de la commande "AB"

```
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 10.211.55.15 (be patient).....done


Server Software:        Apache/2.4.18
Server Hostname:        10.211.55.15
Server Port:            80

Document Path:          /index.html
Document Length:        11321 bytes

Concurrency Level:      1
Time taken for tests:   0.097 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      1159500 bytes
HTML transferred:       1132100 bytes
Requests per second:    1030.27 [#/sec] (mean)
Time per request:       0.971 [ms] (mean)
Time per request:       0.971 [ms] (mean, across all concurrent requests)
Transfer rate:          11665.99 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     0    1   0.2      1       2
Waiting:        0    1   0.2      1       2
Total:          0    1   0.3      1       4

Percentage of the requests served within a certain time (ms)
  50%      1
  66%      1
  75%      1
  80%      1
  90%      1
  95%      1
  98%      1
  99%      4
 100%      4 (longest request)
```

Figure 5.5 - Résultats complets d'une requête "AB"

On peut constater dans le Tableau 5.1 que l'utilisation du module avec PAM est plus lente que l'utilisation avec les fonctions basées sur les fichiers /etc./passwd et /etc./shadow. Nous pouvons aussi identifier que l'appel à la validation de l'authentification (permission 700 où l'accès est limité à un utilisateur) est plus lent que la simple validation de l'autorisation (permission 000 où aucun accès n'est autorisé; permission 777 où l'accès est accordé à tous les usagers) sans validation de l'authentification. Les comparaisons avec le système sans l'utilisation du module n'ont pas été élaborées ici.

Tableau 5.1 - Résultats des tests effectués avec "ab"

Temps moyen obtenu pour les différentes requêtes (ms)	
Sans le module ABSEC - Temps moyen 0.971 ms C'est à dire sans aucune validation de permission.	
Avec le module (pam), permission 777 **Temps moyen 1.654 ms**	Avec le module (fichiers), permission 777 **Temps moyen 1.591 ms**
Avec le module (pam), permission 700, avec une autorisation valide **Temps moyen 8.386 ms**	Avec le module (fichiers), permission 700, avec une autorisation valide **Temps moyen 4.915 ms**
Avec le module (pam), permission 700, avec autorisation invalide **Temps moyen 2045.918 ms**	Avec le module (fichiers), permission 700, avec autorisation invalide **Temps moyen 4.798 ms**
Avec le module (pam), permission 000 **Temps moyen 0.844 ms**	Avec le module (fichiers), permission 000 **Temps moyen 0.804 ms**

L'exécution à répétition de ces tests nous a donné des résultats légèrement différents d'un cas à l'autre. Une autre série de tests a donc été réalisée en lançant plusieurs exécutions et en calculant la moyenne des résultats. Les modifications nécessaires à effectuer entre chaque banc de test étant sujettes à l'erreur, leur automatisation était de mise. La deuxième série des tests a été réalisée à l'aide de MatLab. Cet outil a permis l'ajout de mécanismes (code supplémentaire) pour permettre de modifier la configuration du système entre les tests, et ce, de manière automatisée en plus de réaliser les exécutions multiples et la compilation des résultats.

5.1.2 Tests avec MatLab

Des tests ont donc été refaits avec MatLab [92] en incluant des mécanismes permettant de modifier les privilèges d'accès (permissions) du fichier demandé. La fonction « WebRead » [93] de MatLab a permis d'effectuer les requêtes de la page. Deux méthodes ont été employées pour obtenir le temps d'exécution de la requête. La fonction « timeit » [94] a été

employée en premier. On doit mentionner que la documentation stipule que : « timeit calls the specified function multiple times, and computes the median of the measurements. » [94]. Après quelques essais, on a pu constater que le nombre de répétitions semblait limité à 10. Pour plus de certitude au niveau de la reproductibilité des résultats, les fonctions « tic » et « toc » [95] ont été utilisées (voir Fragment 5.3).

Un autre problème a également été rencontré lors de l'exécution de la fonction « WebRead ». En effet, celle-ci ne se termine pas avec succès lorsque le résultat de la requête Web n'a pas un statut « 200 ». L'appel à cette fonction a été inséré dans une structure « try / catch » (voir Fragment 5.4) bien que cela introduise un délai supplémentaire lors de l'exécution de la requête. Ce délai sera présent dans toutes les requêtes similaires donc n'empêchera pas leur comparaison.

Pour la configuration automatique des permissions, l'accès à la base de données est nécessaire. Le module « database » de MatLab n'étant pas disponible sur la version utilisée, l'extension Python de MatLab a été utilisée pour réaliser ces opérations comme le montre le Fragment 5.5. Le code complet du test se retrouve à l'ANNEXE VII.

Afin de réaliser de multiples exécutions dans un temps raisonnable, les requêtes sont effectuées par groupe de cent, comme avec « ab », avec un délai d'attente d'une seconde entre chaque groupe. La valeur moyenne du temps de réponse a été calculée sur dix exécutions.

Il a été observé que l'utilisation de MatLab (langage interprété) combinée à l'emploi de la structure TRY/catch a un effet important sur le temps de réponse observé comparativement à l'utilisation de l'outil « ab » qui, lui, est en langage compilé. Par exemple, les résultats obtenus pour une requête à une page non protégée (permission777) avec le module PAM montrent

que le temps de réponse moyen avec « ab » était de 1.654 ms (Tableau 5.1) et celui avec MatLab est de 18.55 ms (Figure 5.7).

```
% Version de la function avec TimeIt
function results = run_test(url, hits, options)
    % create function handle to use with timeit accepting params
    f = @() tcwebread(url,options);

    % Clear previous results
    clear y;
    % Make the requests and time them
    disp('Running queries');
    for x = 1:hits
        y(x) = timeit(f);
        pause(1);
    end
    % Convert time in second to miliseconds
    results = y * 1000;
end

% Version de la function avec Tic et Toc
function results = run_test(url, hits, options)
    % Clear previous results
    clear y;
    % Make the requests and time them
    disp('Running queries');
    for x = 1:hits
        display('itération: '+string(x))
        tic;
        for c = 1:100
            tcwebread(url,options);
        end
        y(x) = toc / 100.0;
        pause(1);
    end
    % Convert time in second to miliseconds
    results = y * 1000;
end
```

Fragment 5.3 - Fonction MatLab de mesure des exécutions

```
function tcwebread(URL, options)
    try
        webread(url,options);
    catch
        disp('.');
    end
end
```

Fragment 5.4 - Fonction MatLab d'exécution de la requête avec un Try/Catch

```
function perm = get_perms(url)
    py.importlib.import_module('pymysql');
    conn = py.pymysql.connect('10.211.55.15','jlcyr','password','absec');
    cur = conn.cursor;
    cur.execute('select * from urls where url="'+url+'"');
    res = cur.fetchone();
    perm = string(dec2base(res{4},8));
end

function set_perms(URL, perm)
    perm = floor(perm/100)*8^2+floor(mod(perm,100)/10)*8+floor(mod(perm,10));
    py.importlib.import_module('pymysql');
    conn = py.pymysql.connect('10.211.55.15','jlcyr','password','absec');
    cur = conn.cursor;
    cur.execute('update urls set perms='+string(perm)+' where url="'+url+'"');
    conn.commit();
end
```

Fragment 5.5 - Fonction MatLab de lecture/écriture des permissions dans la base de données

Le temps moyen de traitement des requêtes pour une page statique (HTML) d'Apache donne une valeur de 16.7 ms (Figure 5.6) pour accéder au contenu qui n'est pas protégé (permissions 777). La requête équivalente avec la version du module PAM nous donne une valeur moyenne de 18.55 ms (Figure 5.7) et la version utilisant /etc./passwd une valeur moyenne de 18.47 ms (Figure 5.8). On constate donc une légère augmentation du temps de réponse dû à l'utilisation du module. Les deux versions nous donnent sensiblement le même résultat pour une page non protégée puisque la validation ne fait pas appel à l'authentification du fait que l'accès est autorisé pour tous et que cette étape serait superflue dans le cheminement optimal (Figure 4.11).

L'on peut constater sur plusieurs figures (dont Figure 5.6, Figure 5.12, Figure 5.13 et Figure 5.14) que le premier point sur l'axe de gauche est plus élevé que les autres. Cela peut s'expliquer par le démarrage du système sans aucun tampon et le lancement des processus nécessaires à l'exécution des requêtes.

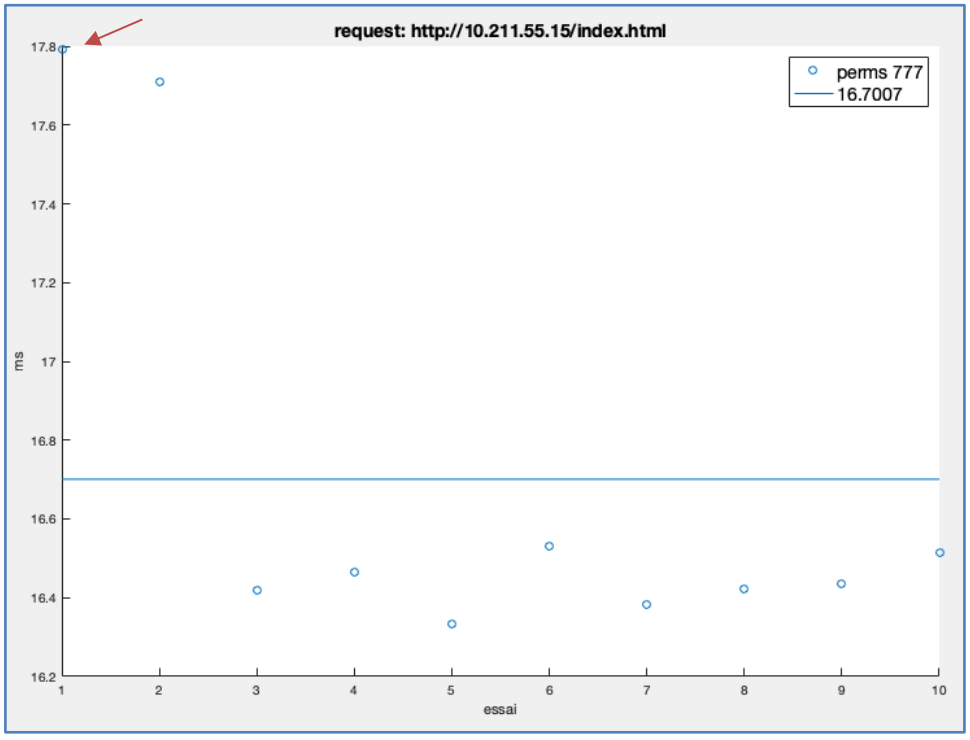


Figure 5.6 - Temps de réponse sans le module

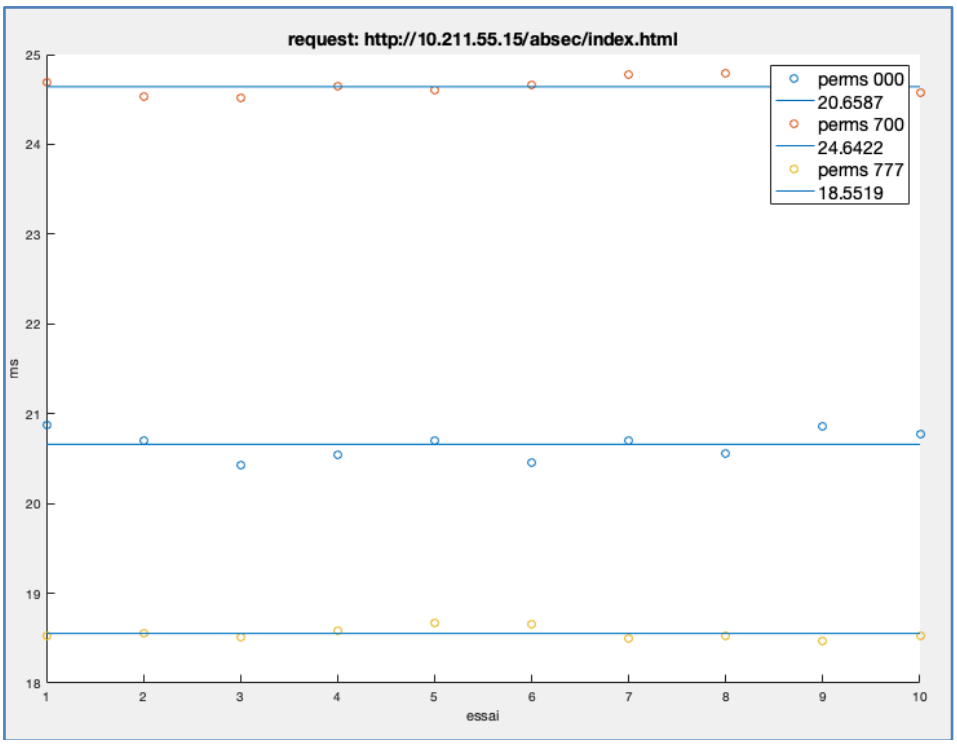


Figure 5.7 - Temps de réponse du module version PAM

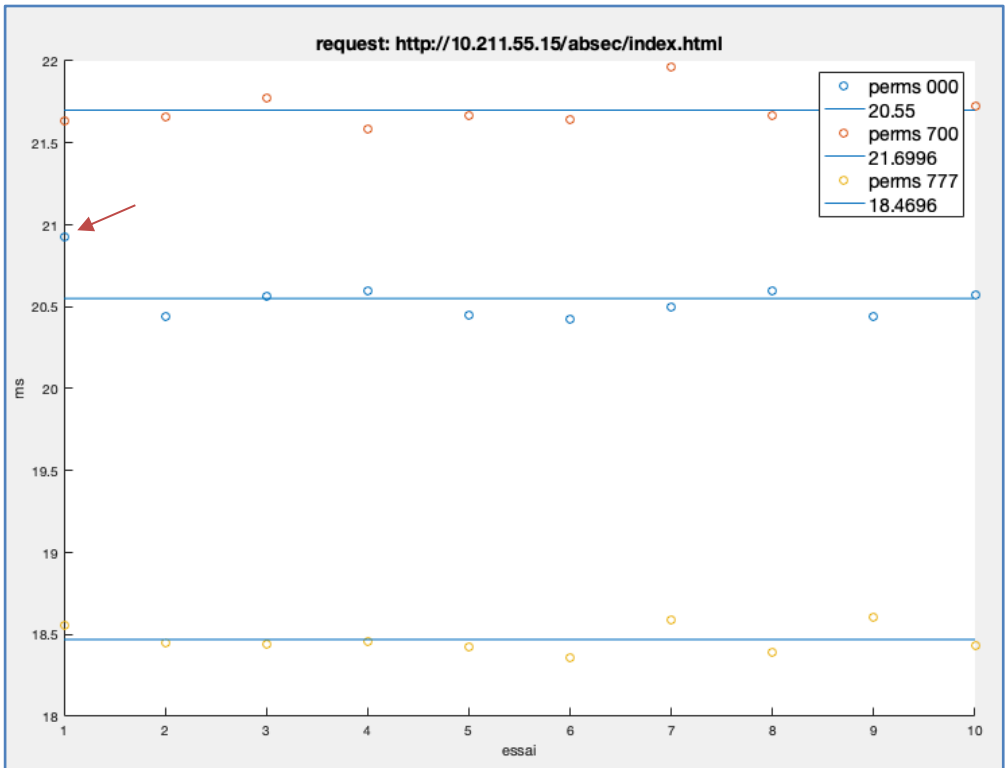


Figure 5.8 - Temps de réponse du module version /etc./passwd

La comparaison du temps de réponse entre les deux versions du module et ce, pour une requête demandant une authentification (700), nous permet de constater une meilleure performance lors de l'utilisation de /etc./passwd qui présent un temps de réponse moyen de 21.70 ms (Figure 5.8). Par comparaison, l'utilisation de PAM présente un temps de réponse de 24.64 ms (Figure 5.7). L'écart s'accroit entre les deux lorsque les informations d'authentification sont erronées (« perms 700 « invalid »). En effet, un temps de réponse de près de 4 secondes a été obtenu avec PAM et de 21 ms avec /etc./passwd (Figure 5.9 et Figure 5.10).

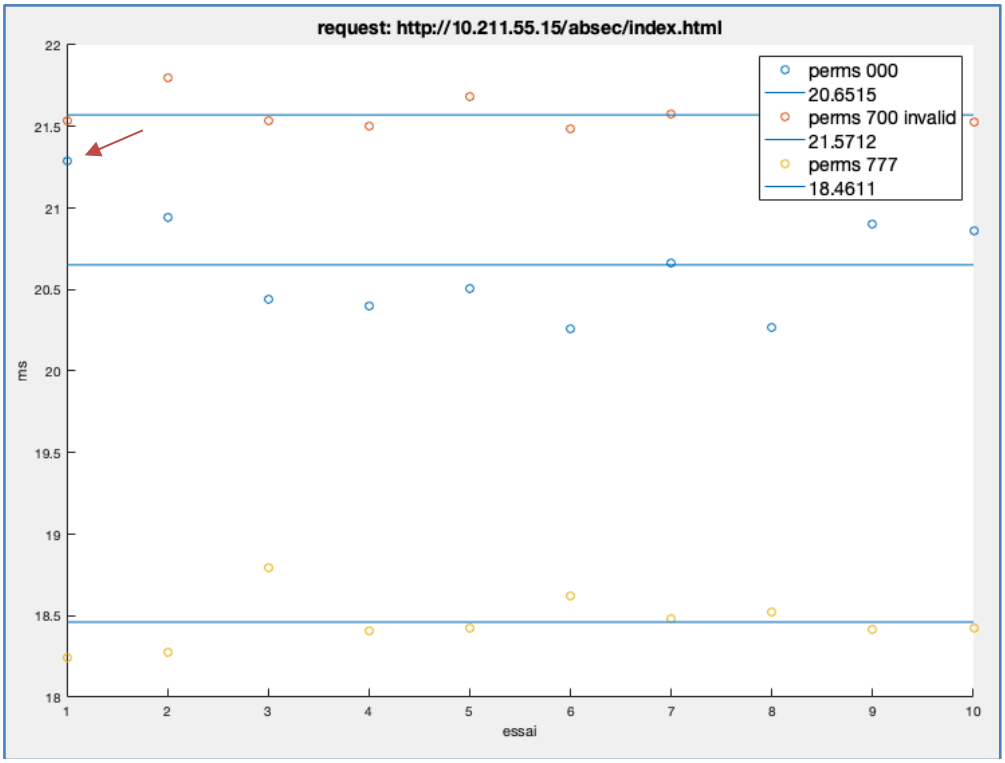


Figure 5.9 – Temps de réponse du module version /etc./passwd avec authorisation erronée

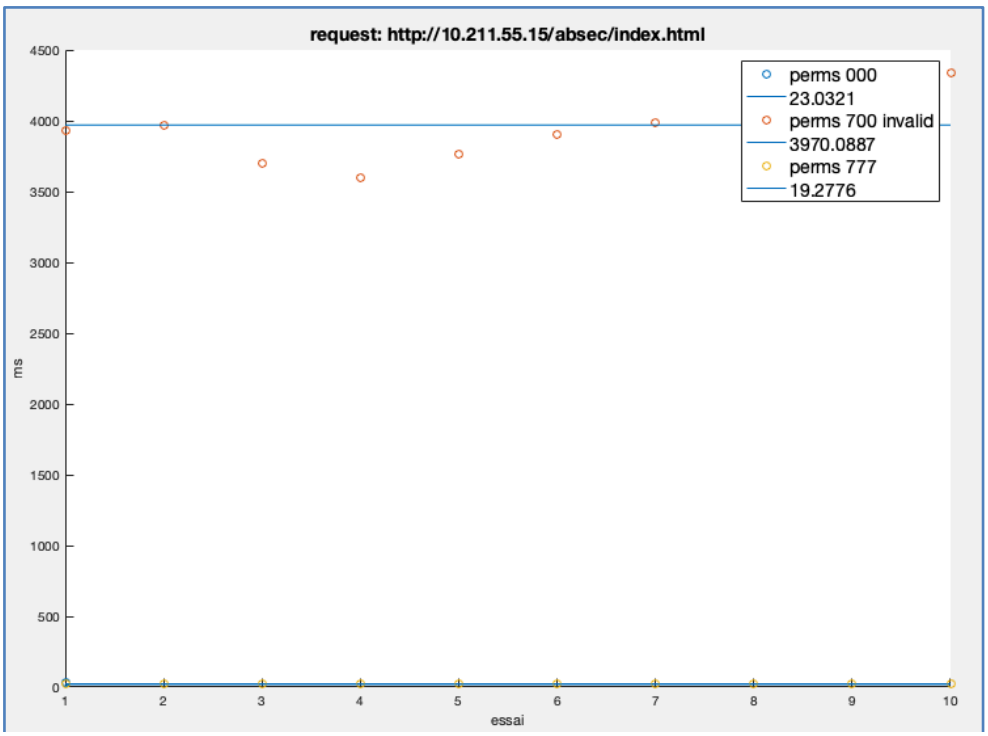


Figure 5.10 – Temps de réponse du module version PAM avec authorisation erronée

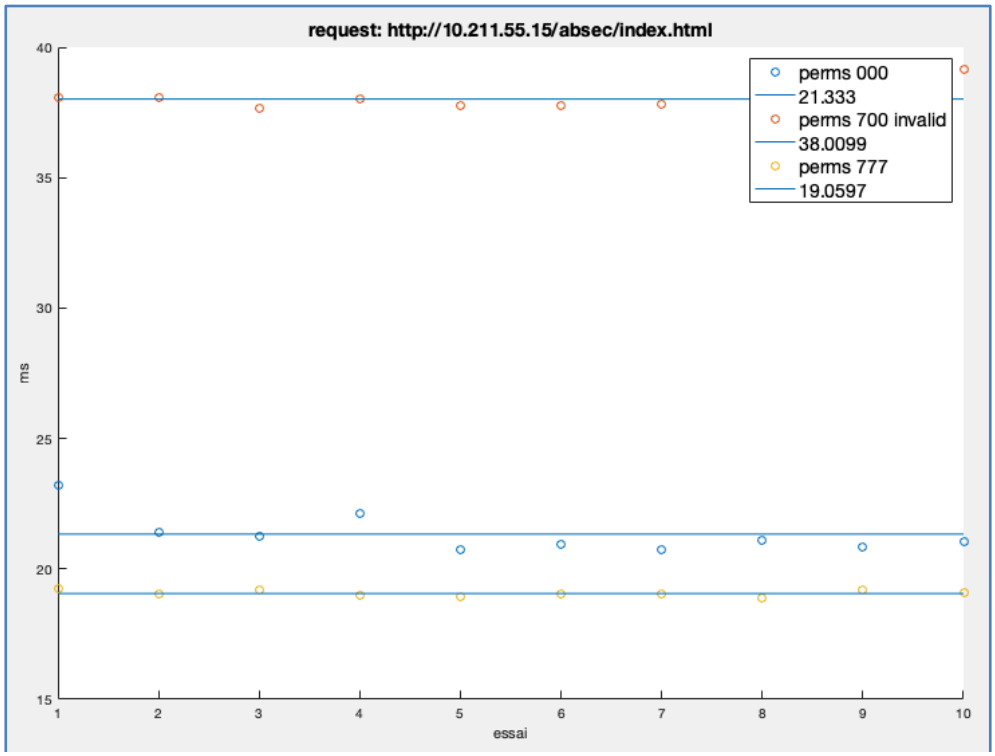


Figure 5.11 – Temps de réponse du module version PAM avec ajustement de la configuration du délai

La documentation de PAM mentionne que le délai de réponse en cas d'identification erronée peut être ajusté. En effet, après la modification de la configuration de PAM, nous obtenons un temps de réponse beaucoup plus acceptable en cas d'authentification invalide (voir Figure 5.11). Pour ce faire nous avons modifié les fichiers /etc./pam.d/httpd et /etc./pam.d/common-auth en ajoutant « nodelay » comme option dans le premier et en ajustant le délai dans le second comme indiqué dans le manuel du module pam_unix (« man pam_unix ») [96].

Pour obtenir une évaluation supplémentaire de la performance du module, nous avons effectué des essais en configurant Apache avec son module basic_auth ainsi qu'en modifiant les permissions sur le système de fichiers pour obtenir une demande d'authentification ou un refus d'accès sans l'utilisation de notre module. L'utilisation d'un fichier .htaccess a permis d'initialiser le module basic_auth et ainsi obtenir un retour 401 et l'ajustement des permissions sur le système de fichiers à 000 pour le fichier d'obtenir un retour 403. Les valeurs obtenues, étant associées à un retour différent de 200, peuvent être comparées avec celles des tests effectués sur notre module, car dans les deux cas il y a déclenchement du « try/catch » dans le code engendrant le même délai supplémentaire. La configuration utilisée est présentée au Fragment 5.6 et les différents résultats obtenus sont montrés aux Figure 5.12, Figure 5.13 et Figure 5.14.

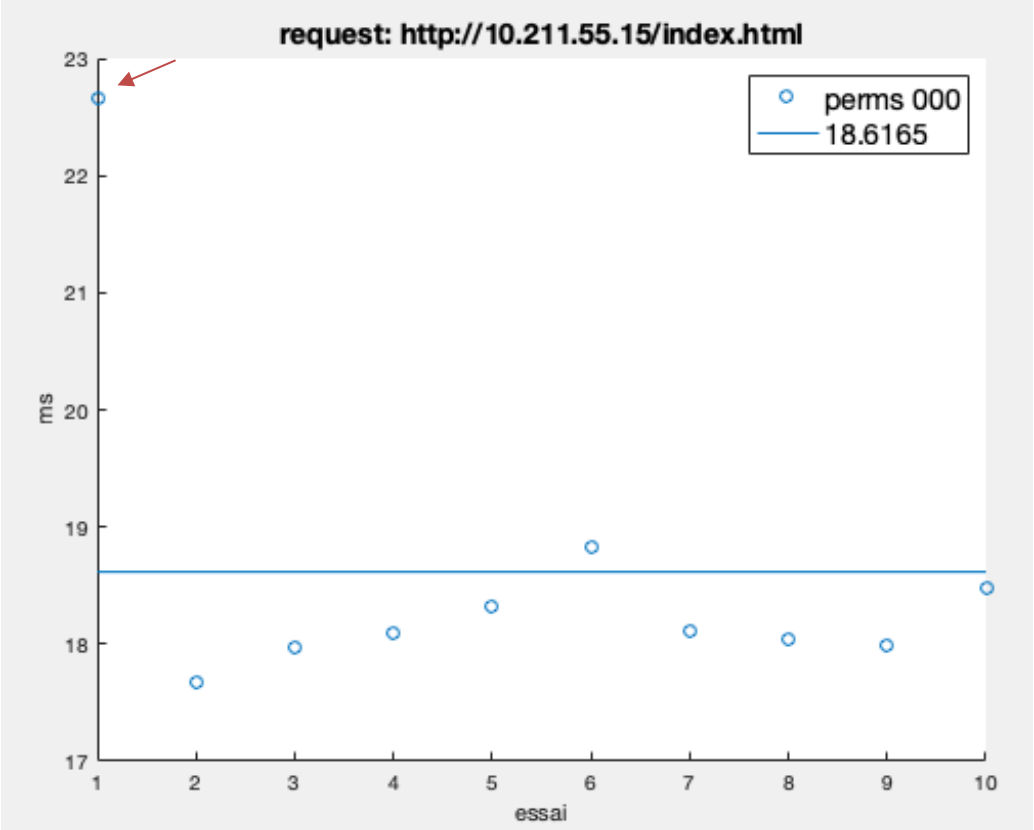


Figure 5.12 - Temps de réponse d'Apache sans le module pour un refus d'accès

```
Fichier .htaccess (retour 403) aucune authentification validée
787:  <Files index.html>
788:  Order allow,deny
789:  Deny from all
790:  </Files>

Fichier .htaccess (retour 401/403) validation d'une bonne ou mauvaise identification
791:  <Files index.html>
792:  order allow,deny
793:  allow from all
794:  require valid-user
795:  Authname "Password access required."
796:  Authtype Basic
797:  AuthUserFile /var/www/HTML/.htpasswd
798:  </Files>

Fichier .htpasswd correspondant
799:  jlcyr:$apr1$FKhyRJN7$quk1if4ze0wyBwuNz2gCH/
```

Fragment 5.6 - Configuration d'Apache pour l'utilisation du module basic_auth

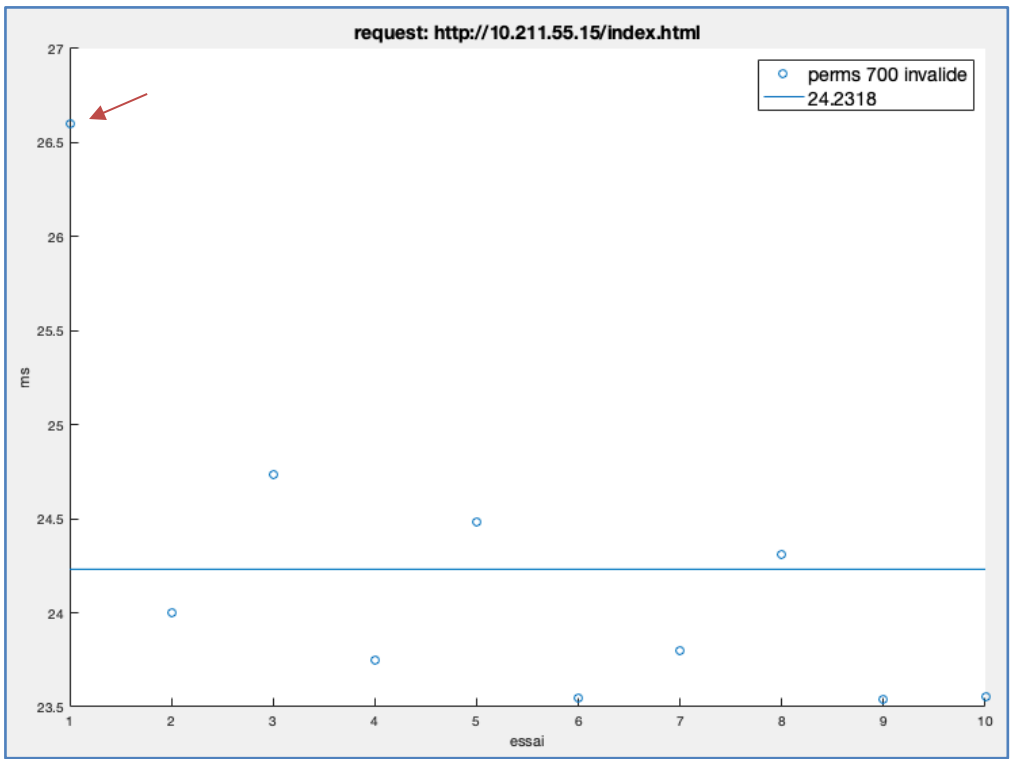


Figure 5.13 - Temps de réponse d'Apache sans le module pour une permission invalide

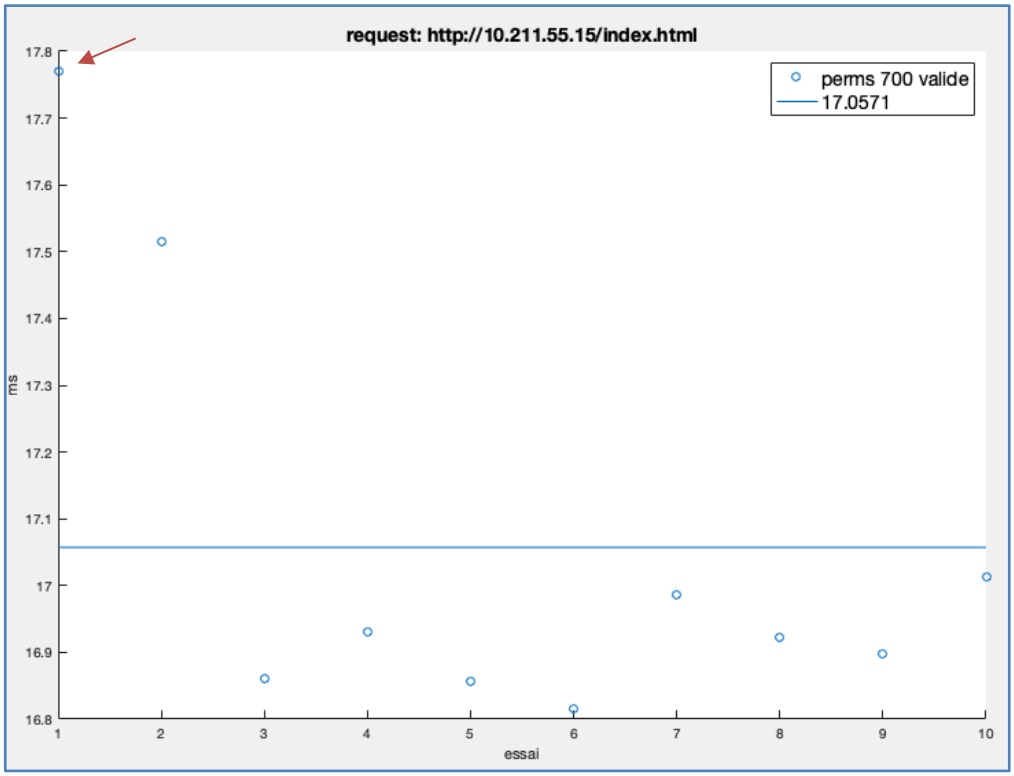


Figure 5.14 - Temps de réponse d'Apache sans le module pour une permission valide

On peut constater que les temps de réponse obtenus lors de l'utilisation du module en version /etc./passwd (Figure 5.8, Figure 5.9) se rapprochent de ceux sans l'utilisation du module (Figure 5.6, Figure 5.12, Figure 5.13, Figure 5.14). La version du module PAM, quant à elle ajoute, un délai plus important qui pourrait causer des problèmes de performances sur un système très sollicité. Le sommaire des résultats obtenus est présenté au Tableau 5.2. Nous pouvons y constater une différence d'environ 0.2 ms pour l'utilisation du module lors d'un accès valide pour tous et une différence variant entre -2.6 ms et 13.8ms pour les autorisations individuelles selon les cas valides ou invalides.

Tableau 5.2 - Sommaire des temps de réponse pour les cas testés

	Module PAM	Module /etc./passwd	Sans le module
Retour 200 (777)	18.5 ms (fig.39)	18.5 ms (fig.40)	16.7 ms (fig.38)
Retour 401 (700 Invalide)	38.0 ms (fig.43)	21.6 ms (fig.41)	24.2 ms (fig.45)
Retour 401 (700 Valide)	24.6 ms (fig.39)	21.7 ms (fig.40)	17.0 ms (fig.46)

Il est important de noter que dans un environnement de production, les appels successifs sont souvent effectués sur des ressources différentes ne permettant pas à la mise en œuvre de mécanisme de tampon (cache) qui peut accélérer le processus lors d'appels successifs à la même ressource. L'utilisation de ces mécanismes peut donc être un biais dans les résultats obtenus.

5.1.3 Comparaison avec un code d'authentification/autorisation similaire écrit en PHP

Après avoir évalué la performance du module, un code équivalent à celui du module a été réalisé en langage PHP, tel qu'il pourrait exister dans n'importe lequel des logiciels les plus populaires. Cela permet d'établir un point de comparaison supplémentaire. Le code de ce programme est présenté à l'ANNEXE VIII. Le code PHP a été mis en place derrière Apache à l'aide de la configuration au Fragment 5.7. Les temps de réponse obtenus sont présentés à la Figure 5.15.

```
800:  <IfModule mod_rewrite.c>
801:  RewriteEngine On
802:  RewriteBase /
803:  RewriteRule ^index\.PHP$ - [L]
804:  RewriteCond %{REQUEST_FILENAME} !-f
805:  RewriteCond %{REQUEST_FILENAME} !-d
806:  RewriteRule . /PHP/index.php [L]
807:  </IfModule>
```

Fragment 5.7 - Configuration d'Apache pour le test d'un système en PHP

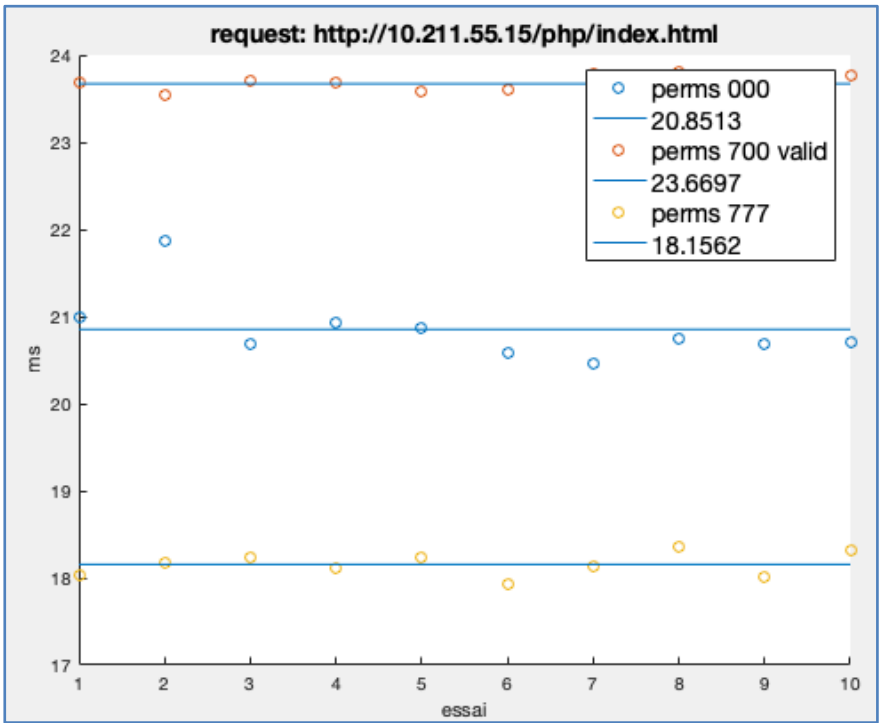


Figure 5.15 - Temps de réponse de l'exécution du code PHP

Ce test a permis de constater (voir Tableau 5.3) que la performance du module est meilleure ou similaire à celle du code PHP. L'écart le plus important est obtenu lors de la validation complète de l'authentification et de l'autorisation (permission 700). Ce phénomène s'explique fort probablement par le temps requis pour le calcul du mot de passe crypté et la comparaison avec celui contenu dans le fichier /etc./shadow. Les performances du module PAM n'ont pas été comparées, même si l'équivalent aurait pu être réalisé en PHP [97]. Cette comparaison n'a tout simplement pas été réalisée, car les résultats obtenus avec PAM, en comparaison de /etc./passwd, se sont révélés plus lents.

Tableau 5.3 - Comparaison des performances de PHP vs Module

	Module /etc./passwd	Code PHP
Aucune permission (000)	20.5 ms	20.8 ms
Permission valide (700)	21.7 ms	23.7 ms
Public (777)	18.5 ms	18.2 ms

5.2 Tests de détection

Comme la journalisation n'a pas été réalisée complètement, aucun test de détection n'a été réalisé. Le but de la journalisation était de détecter des problèmes de configuration des permissions et de détecter d'éventuelles attaques ou intrusions. Lors de la réalisation du module d'autorisation avec la base de données, une fonctionnalité rudimentaire de journalisation a été mise en place soit l'insertion d'un enregistrement de permission lorsqu'un URI non défini est appelé avec l'assignation automatique des permissions par défaut tel que défini dans la configuration du module. Cette fonctionnalité pourrait facilement être augmentée afin de détecter les accès à des pages pour lesquelles aucune permission n'est définie.

5.3 Tests d'intégration

L'intégration à un système simple n'ayant aucun système d'authentification et autorisation se compare å l'intégration faite avec le fichier test.php dans les tests effectués. En effet, les tests font appel à des pages statiques ou dynamiques présentes sur le système de fichiers et n'appliquant aucune gestion des authentifications ou permissions et le module a démontré sa capacité de les gérer.

L'intégration à un système plus complet a été réalisée afin de déterminer les difficultés d'intégration de l'approche proposée à un système existant. Le logiciel WordPress a donc été sélectionné, car, comme il a été mentionné à la section 2.1.3, ce dernier est le plus répandu chez les concepteurs de sites Web. Le but de ce test n'était pas de modifier le logiciel pour obtenir des résultats, mais de déterminer la faisabilité et le niveau de complexité de l'intégration.

À la section 3.1, il a été mentionné que l'intégration devrait être possible si le logiciel adhère aux principes de REST (hypothèse 3). De plus, pour une intégration simplifiée, il est important que le logiciel utilise les usagers du système (/etc./passwd) et que les permissions associées aux URL soient dans une base de données.

On peut donc énoncer que :

- Wordpress doit avoir accès aux usagers système ou le module avoir accès aux usagers de WP
- Wordpress doit avoir accès à la base de données de permissions ou le module avoir accès aux permissions de WP
- Wordpress doit utiliser un schéma d'URL compatible avec REST
- Wordpress doit utiliser les méthodes HTTP pour définir les actions.

Pour cette étape de validation, le logiciel WordPress a été téléchargé et installé à partir de l'adresse https://wordpress.org/download/. L'installation s'est faite selon les instructions fournies sur le site.

Notre analyse sommaire du logiciel nous indique premièrement qu'il n'utilise pas la « basic authentication » du protocole HTTP pour gérer les authentifications, mais plutôt un « cookie » de session (présenté à la Figure 5.16). Nous avions déjà proposé cette approche à la section 3.3.3 en remplacement du « basic authentication » pour permettre le développement d'une interface d'identification de l'utilisateur différente de celle du navigateur.

Name	Value	Domain	Path	Expires	Size
wordpress_2d71a1e0e527a69fc2...	jlcyr%7C1554394199...	10.211.55.15	/wordpress/wp...	Session	173 B
wordpress_logged_in_2d71a1e0e...	jlcyr%7C1554394199...	10.211.55.15	/wordpress/	Session	183 B
wordpress_test_cookie	WP+Cookie+check	10.211.55.15	/wordpress/	Session	36 B
wp-settings-time-1	1554221399	10.211.55.15	/wordpress/	2020-04-01, 12:09:59 ...	28 B

Figure 5.16 - Cookies présents dans le navigateur lors d'une connexion à WordPress

La structure de la base de données (disponible à l'ANNEXE IX) permet de constater que la gestion des utilisateurs (distribué sur 2 tables wp_users et wp_usermeta) utilise un identifiant (wp_users.user_login) et un mot de passe crypté (wp_users.user_pass). L'étude de l'interface de gestion des utilisateurs (Figure 5.17) suggère l'utilisation de rôles pour classifier le type

d'utilisateur (Subscriber, contributor, Author, Editor, Administrator, No Role). Cependant, dans la structure de données, on retrouve un niveau d'accès (wp_usermeta metakey = wp_user_level). Ces types d'information peuvent être associés aux groupes et aux modes (lecture/écriture). En étudiant la structure des publications (wp_posts, wp_postmeta), on constate que la première association visible est le propriétaire (wp_posts.post_author).

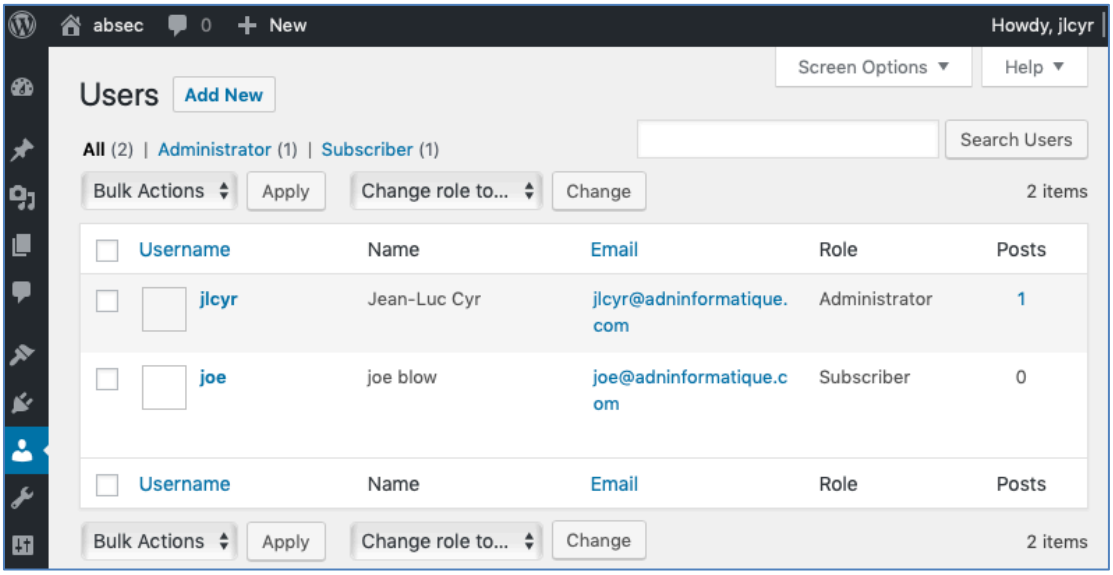


Figure 5.17 - Écran d'édition des utilisateurs de WordPress

Le gestionnaire de contenu par défaut est nommé index.php. Aucune configuration supplémentaire n'est présente. C'est pourquoi les adresses du contenu passent toutes par l'adresse /wordpress/index.php avec, à la suite, les arguments pour accéder au contenu (/index.php/2019/04/02/hello-world/). Une configuration plus élégante est possible en activant une structure d'URL différente dans la configuration du logiciel (Figure 5.18) conjointement avec l'utilisation de règles de réécriture des URL dans un fichier de configuration Apache .htaccess (Fragment 5.8) [98].

Le système d'administration du logiciel présente des schémas d'URL complètement différents. De plus, l'interface utilise « AJAX » pour plusieurs des appels. La liste ci-dessous présente quelques URLs utilisés.

http://10.211.55.11/wordpress/wp-admin/
http://10.211.55.15/wordpress/wp-admin/edit.php
http://10.211.55.15/wordpress/wp-admin/post.php?post=1&action=edit
http://10.211.55.15/wordpress/wp-admin/admin-ajax.php

À l'analyse de ces patrons (« patterns ») d'adresse, l'on constate que les « scripts » sont invoqués directement et que les actions sont en partie présentes en paramètre dans l'URL. Les hypothèses impliquant une relation 1 :1 entre l'URL et le contenu et l'utilisation des verbes de HTTP pour représenter les actions ne sont donc pas respectées. Il est donc difficile dans le temps imparti de réaliser un test d'intégration complet concluant.

Nous pouvons cependant déterminer qu'avec une modification du module pour valider les autorisations basées sur le « cookie » de WordPress, conjointement avec un accès à la base de données de WordPress nous serions en mesure d'arriver à un résultat intéressant sans apporter de modifications majeures au logiciel. Pour avoir un résultat optimal, diverses modifications devraient cependant y être apportées telles que la standardisation des URL, l'identification des permissions par URL et la suppression du code des validations qui serait remplacé par le module.

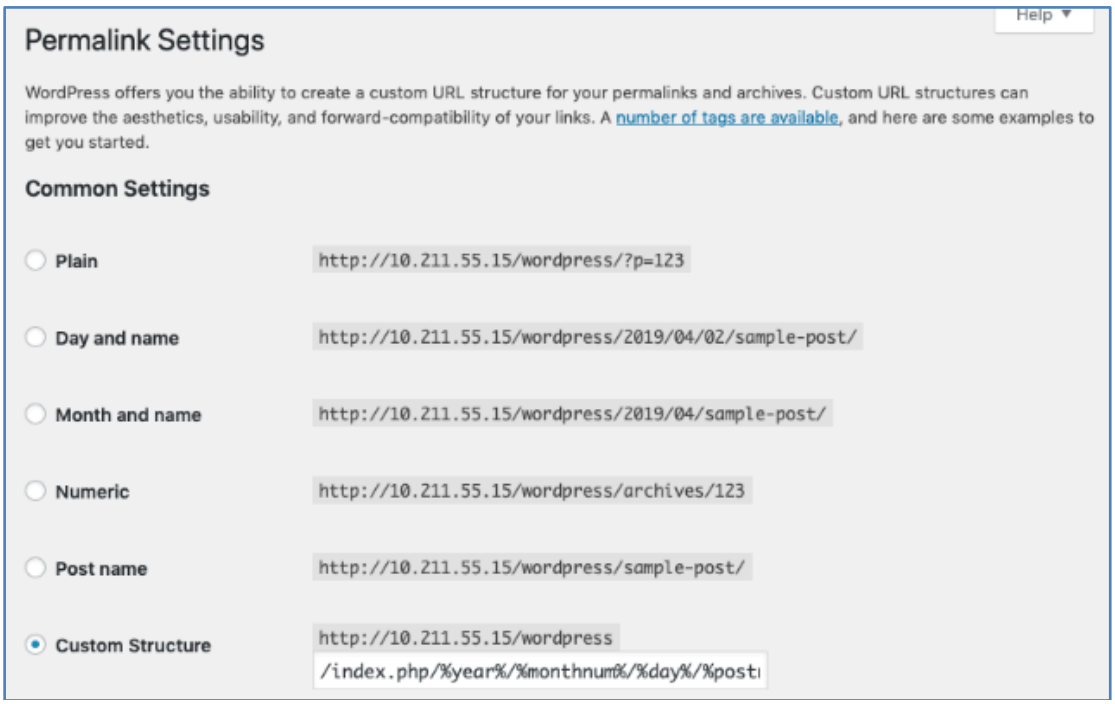


Figure 5.18 - Écran de configuration du format d'URL de WordPress

```
808:  # BEGIN WordPress
809:  <IfModule mod_rewrite.c>
810:  RewriteEngine On
811:  RewriteBase /
812:  RewriteRule ^index\.PHP$ - [L]
813:  RewriteCond %{REQUEST_FILENAME} !-f
814:  RewriteCond %{REQUEST_FILENAME} !-d
815:  RewriteRule . /wordpress/index.php [L]
816:  </IfModule>
817:  # END WordPress
```

Fragment 5.8 - Configuration .htaccess d'Apache pour WordPress avec réécriture des URLs

CHAPITRE 6

Conclusion

La réalisation de ce projet démontre qu'il est possible de mettre en œuvre un module pouvant être intégré au code source d'un serveur Apache. Il permet de gérer les authentifications et les autorisations d'une application Web de manière complète sans avoir à les intégrer au code de cette dernière. Le module utilise des composants déjà existants et validés incluant le processus d'authentification du système d'exploitation.

Cette approche, dans le cas d'applications réalisées en langage PHP, apporte un gain de performance lors de l'utilisation de technique d'authentification basée sur le cryptage tel que présenté à la section 5.1.3.

Le positionnement du module dans la chaine de traitement d'Apache assure également qu'aucune ressource associée à une URL ne peut être accédée sans une validation des permissions qui lui sont associées. Les paramètres de la configuration du module permettent également de restreindre les permissions en cas de défaut, ce qui assure la sécurité malgré les erreurs qui pourraient survenir dans le code de l'application.

L'intégration du module au développement d'une nouvelle application se fait de manière simple. Cependant, son intégration à une application existante, évaluée seulement avec le logiciel WordPress, demande de certains ajustements. Dans ces cas, deux approches sont possibles, soit la modification de l'application ou l'adaptation du module. Considérant que le module devrait être maintenu par une équipe spécialisée et utilisable universellement, les seules modifications acceptables seraient l'ajout de paramètres pour définir la base de données à utiliser pour gérer les informations de manière commune avec l'application. Des modifications apportées à l'application seraient donc la solution à privilégier.

Pour favoriser son intégration et simplifier les modifications à apporter, tel que l'intégration de diverses fonctionnalités du protocole HTTP au langage PHP derrière Apache, le module devrait offrir une API permettant son intégration aux applications de manière simple. De plus, l'utilisation des « cookies » pour gérer l'authentification plutôt que la « basic authentication » tel que défini dans le protocole HTTP offrirait plus de souplesse aux applications pour implémenter leur propre interface de saisie des identifiants.

Dans une étape intermédiaire, des outils « web » tels que « Webmin » [99] pourraient être utilisés pour permettre la gestion des utilisateurs du système au sein de l'application. Afin de faciliter cette intégration, la gestion de l'authentification pourrait être également basée sur une base de données plutôt que les composantes du système d'exploitation.

Pour favoriser l'adoption de l'approche proposée, il serait pertinent de mettre en place un standard pour la définition des URL utilisées par les applications, tel que les cartes de sites (« sitemap ») pour le SEO (Search Engine Optimisation), auquel s'ajouteraient les informations d'autorisation. Ce standard, tel que le protocole HTTP, pourrait devenir un incontournable pour le développement d'application Web. Il s'agirait d'un répertoire d'URLs existante et de permissions associées, comparables à LDAP (Lightweight Directory Access Protocol) pour l'authentification ou WSDL (Web Service Description Language) pour les services, qui pourraient à la fois être utilisés par le module pour la sécurité et les robots d'indexation pour le SEO.

Dans le but de contribuer au domaine, à la date de la présente, l'entièreté du projet sera mise en ligne et accessible publiquement.[2] Le code actuel, la documentation ainsi que les options qui seront explorées dans l'avenir font partie des éléments qui s'y retrouveront.

[2] L'adresse en date de ce document est http://absec.adninformatique.com/

6.1 Développements futurs

Cette section présente des pistes supplémentaires à explorer pour permettre une meilleure intégration du module avec des logiciels existants et pour améliorer son efficacité et sa versatilité pour sécuriser les applications qui l'utiliseront.

6.1.1 Arrimage avec les principaux CMS (Content Management System)

Pour faciliter son arrimage à différentes applications, une méthode de communication standardisée devrait être mise en place pour définir les URI correspondant aux ressources et l'assignation des permissions qui leur sont associées. L'utilisation actuelle d'une base de données dédiée au module peut être une contrainte importante. Il pourrait être intéressant de pouvoir paramétrer la base de données utilisée dans un fichier de configuration d'Apache afin de l'interfacer avec les bases de données existantes des logiciels moyennant quelques contraintes (comme, par exemple, la disponibilité de certaines valeurs).

6.1.2 Intégration à d'autres serveurs HTTP(S)

L'intégration d'un module similaire avec d'autres serveurs HTTP tel que NGINX pourrait également être une piste intéressante pour favoriser l'adoption de l'approche proposée.

6.1.3 Suivi des données ressources

L'association des droits avec des blocs de données plutôt qu'une « page Web » associée à une URL serait également utile. En effet, comme présenté à la section 2.5.5, les « pages Web » sont des objets composites et une granularité plus fine des permissions pourrait être un incitatif additionnel pour l'adoption du module.

Pour ce faire, dans un premier temps, le module pourrait traiter les autorisations originales de la page pour en valider le retour. Dans un cas positif, soit lorsque l'accès à la page est permis, cette dernière pourrait être analysée pour déterminer si des blocs exigent des

permissions supplémentaires. Ces dernières pourraient être incluses sous forme d'attributs XML supplémentaires associés à différents éléments tels que <DIV>, <IMG> ou autre tel que présenté au Fragment 6.1.

```
818:  <div uid=10 gid=100 perms=740>
819:  Contenu avec des permissions plus fines.
820:  <img src="img.jpg" uid=10 gid=100 perms=444 />
821:  </div>
```

Fragment 6.1 - Exemple de granularité des permissions incorporée au XML

6.1.4 Modèle d'authentification/autorisation différent

L'identification de l'utilisateur (authentification) pourrait être améliorée en utilisant une association avec l'adresse IP du client. Le transport de l'authentification pourrait également permettre l'utilisation de « cookies » en complément ou en remplacement de la « basic authentication » du protocole HTTP. Cette dernière modification permettrait le support d'une page de connexion personnalisée de l'application plutôt que l'utilisation de la boite de dialogue standard du navigateur.

LISTE DE RÉFÉRENCES

[1] «Open Web Application Security Project,» 2019. [En ligne]. Available: https://www.owasp.org/. [Accès le 04 2019].

[2] «OWASP Top 10,» 2019. [En ligne]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. [Accès le 01 2019].

[3] Akamai, «ACEI - Sondage sur la cybersécurité,» 2019. [En ligne]. Available: https://acei.ca/rapport-cybersecurite-2018. [Accès le 04 2019].

[4] «Autorité Canadienne des Enregistrements Internet,» 2019. [En ligne]. Available: https://acei.ca/. [Accès le 04 2019].

[5] «Logiciel : Wordpress,» 2019. [En ligne]. Available: https://wordpress.org/. [Accès le 01 2019].

[6] «Logiciel : Joomlà,» 2019. [En ligne]. Available: https://www.joomla.org/. [Accès le 01 2019].

[7] «Logiciel : Drupal,» 2019. [En ligne]. Available: https://www.drupal.org/. [Accès le 2019].

[8] «Logiciel : Langage PHP,» The PHP Group, 2019. [En ligne]. Available: http://www.php.net/. [Accès le 01 2019].

[9] «Logiciel : Langage Ruby,» Ruby community, 2019. [En ligne]. Available: https://www.ruby-lang.org/. [Accès le 01 2019].

[10] «Logiciel : Langage Python,» Python Software Foundation, 2019. [En ligne]. Available: https://www.python.org/. [Accès le 01 2019].

[11] A. Ronacher, «Logiciel : microframework Flask,» 2019. [En ligne]. Available: http://flask.pocoo.org/. [Accès le 01 2019].

[12] «Logiciel : Django,» 2019. [En ligne]. Available: https://www.djangoproject.com/. [Accès le 01 2019].

[13] «Logiciel : Apache http server,» Apache Software Fundation, 2019. [En ligne]. Available: http://httpd.apache.org/. [Accès le 01 2019].

[14] «Logiciel : Nginx,» nginx inc., 2019. [En ligne]. Available: https://www.nginx.com/. [Accès le 01 2019].

[15] «Logiciel : IIS,» Microsoft, 2019. [En ligne]. Available: https://www.iis.net/. [Accès le 01 2019].

[16] Imperva, 2019. [En ligne]. Available: https://www.imperva.com/. [Accès le 06 2018].

[17] «BIG-IP,» F5, 2019. [En ligne]. Available: https://f5.com/products/big-ip. [Accès le 06 2018].

[18] CISCO, 2019. [En ligne]. Available: https://www.cisco.com/c/en/us/products/security/index.html. [Accès le 06 2018].

[19] «Logiciel : ModSecurity,» 2019. [En ligne]. Available: http://www.modsecurity.org/. [Accès le 01 2019].

[20] «Logiciel : Squid,» 2019. [En ligne]. Available: http://www.squid-cache.org/. [Accès le 01 2019].

[21] «Logiciel : Varnish,» 2019. [En ligne]. Available: https://varnish-cache.org/. [Accès le 01 2019].

[22] «Logiciel : Flask Basic Auth,» 2019. [En ligne]. Available: http://flask.pocoo.org/snippets/8/. [Accès le 01 2019].

[23] «ISO29110,» Wikipedia, [En ligne]. Available: https://en.wikipedia.org/wiki/ISO_29110. [Accès le 06 2017].

[24] «ISO/IEC 29110-2-1:2015,» International Organization for Standardization, 2015. [En ligne]. Available: https://www.iso.org/standard/62712.html. [Accès le 06 2017].

[25] C. Y. Laporte, «Public site of the ISO WG - ISU/IEC 29110,» Ecole de technologie supérieure Montréal, [En ligne]. Available: http://profs.etsmtl.ca/claporte/English/VSE/index.html. [Accès le 06 2017].

[26] «Web Server Usage Statistics,» BuildWith Pty Ltd, 2018. [En ligne]. Available: https://trends.builtwith.com/web-server. [Accès le 06 2018].

[27] «Usage of web servers,» Q-Success Web-Based Services, 2018. [En ligne]. Available: https://w3techs.com/technologies/overview/web_server/all. [Accès le 08 2018].

[28] «Programming Language Usage,» BuiltWith Pty Ltd, 2018. [En ligne]. Available: https://trends.builtwith.com/framework/programming-language. [Accès le 06 2018].

[29] «Server-side programming languages,» Q-Success Web-Based Services, 2018. [En ligne]. Available: https://w3techs.com/technologies/overview/programming_language/all. [Accès le 06 2018].

[30] «CMS Usage Distribution,» BuiltWith Pty Ltd, 2019. [En ligne]. Available: https://trends.builtwith.com/cms. [Accès le 06 2018].

[31] «Usage of content management systems,» Q-Success Web-Based Services, 2018. [En ligne]. Available: https://w3techs.com/technologies/overview/content_management/all. [Accès le 06 2018].

[32] «Usage of operating systems,» Q-Success Web-Based Services, 2018. [En ligne]. Available: https://w3techs.com/technologies/overview/operating_system/all. [Accès le 06 2018].

[33] «EU General Data Protection Regulation (GDPR),» 2019. [En ligne]. Available: https://www.eugdpr.org/. [Accès le 01 2019].

[34] «Payment Card Industry - Data Security Standard,» 2018. [En ligne]. Available: https://www.pcisecuritystandards.org/. [Accès le 06 2018].

[35] «ISO/IEC 27000 family - Information security,» International Organization for Standardization, 2019. [En ligne]. Available: https://www.iso.org/isoiec-27001-information-security.html. [Accès le 06 2018].

[36] «Information Technology Infrastructure Library (ITIL),» Axelos, 2019. [En ligne]. Available: https://www.axelos.com/best-practice-solutions/itil. [Accès le 06 2018].

[37] «Computer Security Resource Center,» National Institute of Standards and Technology / U.S. Department of commerce, 2019. [En ligne]. Available: https://csrc.nist.gov/publications. [Accès le 06 2018].

[38] «Accueil,» The Internet Engineering Task Force (IETF), 2019. [En ligne]. Available: https://www.ietf.org. [Accès le 06 2018].

[39] «Accueil,» The institute of information security professionnals, 2019. [En ligne]. Available: https://www.iisp.org. [Accès le 06 2018].

[40] «CERT (Computer Emergency Team),» Carnegie Mellon University - Software Engineering Institute, 2019. [En ligne]. Available: https://www.sei.cmu.edu/about/divisions/cert/index.cfm. [Accès le 06 2018].

[41] «Vulnerability Notes Database,» Carnegie Mellon University - Software Engineering Institute, 2018. [En ligne]. Available: https://www.kb.cert.org/vuls/. [Accès le 06 2018].

[42] «Directive sur la sécurité de l'information gouvernementale,» Secrétariat du conseil du trésor - Québec, 2009. [En ligne]. Available: https://www.tresor.gouv.qc.ca/ressources-informationnelles/securite-de-linformation/directive-sur-la-securite-de-linformation-gouvernementale/. [Accès le 01 2019].

[43] A. A. M. S. B. T. D. G. Sailu Reddy, HTTP: The Definitive Guide, O'Reilly Media, Inc., 2002, pp. Ch4,Sec5.

[44] «Accueil,» Apache Software Foundation, 2019. [En ligne]. Available: http://apache.org. [Accès le 06 2018].

[45] N. Kew, The Apache Modules Book : Application Development with Apache, Prentice Hall, 2007.

[46] «List of Apache modules,» Wikimedia Foundation Inc., 2019. [En ligne]. Available: https://en.wikipedia.org/wiki/List_of_Apache_modules. [Accès le 06 2018].

[47] «RFCs,» The Internet Engineering Task Force (IETF), 2018. [En ligne]. Available: https://ietf.org/standards/rfcs/. [Accès le 06 2018].

[48] «HTTP Working Group,» The Internet Engineering Task Force (IETF), 2019. [En ligne]. Available: https://httpwg.org. [Accès le 06 2018].

[49] J. Olenski, «GlobaSign Blog - SSL vs TLS,» GlobaSign, 07 07 2016. [En ligne]. Available: https://www.globalsign.com/en/blog/ssl-vs-tls-difference/. [Accès le 20 04 2019].

[50] «Difference between SSL and TLS,» Indiana University, 18 02 2019. [En ligne]. Available: https://kb.iu.edu/d/anjv. [Accès le 06 2019].

[51] R. Barnes, «Deprecating Non-Secure HTTP,» 30 Avril 2015. [En ligne]. Available: https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/. [Accès le 06 2018].

[52] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» 2000.

[53] «Pluggable Authentication Module for Linux,» 2018. [En ligne]. Available: http://www.linux-pam.org. [Accès le 06 2018].

[54] «HTTP Gallery - HTTP Authentication,» Neumetrix Limited, 2019. [En ligne]. Available: https://www.httpwatch.com/httpgallery/authentication/. [Accès le 06 2018].

[55] «Single Sign-on,» Wikimedia Foundation Inc., 2018. [En ligne]. Available: https://en.wikipedia.org/wiki/Single_sign-on. [Accès le 06 2018].

[56] «Kerberos: The network Authentication Protocol,» MIT, 08 01 2019. [En ligne]. Available: https://web.mit.edu/kerberos/. [Accès le 06 2019].

[57] G. Shaw, «Configure Apache to use Kerberos,» MicroHowto, 2018. [En ligne]. Available: http://www.microhowto.info/howto/configure_apache_to_use_kerberos_authentication.html. [Accès le 06 2019].

[58] «How to configure browsers for kerberos authentification,» claudera, 22 05 2019. [En ligne]. Available: https://www.cloudera.com/documentation/enterprise/latest/topics/cdh_sg_browser_access_kerberos_protected_url.html. [Accès le 06 2019].

[59] «Configuring Kerberos authentication in different browser,» Windows OS Hub, 01 08 2018. [En ligne]. Available: http://woshub.com/enable-kerberos-authentication-in-browser/. [Accès le 06 2019].

[60] G. Shaw, «Configure Apache to use Kerberos authentication,» micro howto, 2018. [En ligne]. Available: http://www.microhowto.info/howto/configure_apache_to_use_kerberos_authentication.html. [Accès le 06 2019].

[61] «Role-basec access control,» Wikipedia, [En ligne]. Available: https://en.wikipedia.org/wiki/Role-based_access_control. [Accès le 06 2019].

[62] «Bell-LaPadula,» Wikipedia, [En ligne]. Available: https://en.wikipedia.org/wiki/Bell–LaPadula_model. [Accès le 06 2019].

[63] «Biba Model,» Wikipedia, [En ligne]. Available: https://en.wikipedia.org/wiki/Biba_Model. [Accès le 06 2019].

[64] A. Grünbacher, «POSIX Access Control Lists on Linux,» SuSE Labs, 2003. [En ligne]. Available: https://www.usenix.org/legacy/publications/library/proceedings/usenix03/tech/freenix03/full_papers/gruenbacher/gruenbacher_html/main.html. [Accès le 01 2019].

[65] «File system permissions,» Wikimedia Foundation, Inc, 2019. [En ligne]. Available: https://en.wikipedia.org/wiki/File_system_permissions. [Accès le 06 2018].

[66] A. Parecki, «OAuth 2..0,» OAuth, [En ligne]. Available: https://oauth.net. [Accès le 06 2019].

[67] «Kallithea,» Software Freedom Conservancy, 2019. [En ligne]. Available: https://kallithea-scm.org. [Accès le 06 2018].

[68] «Logiciel : Proxmox,» Proxmox Server Solutions GmbH, 2019. [En ligne]. Available: https://www.proxmox.com/en/. [Accès le 06 2018].

[69] «Parallel Desktop,» Parallels International GmbH, 2019. [En ligne]. Available: https://www.parallels.com/ca/. [Accès le 06 2018].

[70] «macOS,» Wikimedia Foundation, Inc, 2019. [En ligne]. Available: https://en.wikipedia.org/wiki/MacOS. [Accès le 06 2018].

[71] «Logiciel : Visual Code Studio,» Microsoft, 2019. [En ligne]. Available: https://code.visualstudio.com/. [Accès le 01 2019].

[72] «Logiciel : SourceTree,» Atlassian, 2019. [En ligne]. Available: https://www.sourcetreeapp.com/. [Accès le 01 2019].

[73] «Logiciel : Safari,» Apple Inc., 2019. [En ligne]. Available: https://www.apple.com/ca/safari/. [Accès le 06 2018].

[74] «Firefox,» Mozilla Foundation, 2019. [En ligne]. Available: https://www.mozilla.org/en-CA/firefox/. [Accès le 06 2018].

[75] «Chrome,» Google, 2019. [En ligne]. Available: https://www.google.com/chrome/. [Accès le 06 2018].

[76] I. Seiichiro, «libapr programming tutorial,» 2005. [En ligne]. Available: http://dev.ariel-networks.com/apr/apr-tutorial/html/apr-tutorial.html#toc3. [Accès le 01 2019].

[77] «Developing modules for the Apache HTTP Server 2.4,» The Apache Software Foundation, 2019. [En ligne]. Available: https://httpd.apache.org/docs/2.4/developer/modguide.html. [Accès le 06 2018].

[78] J. R. Ziviani, «How to create an Apache module,» 2011. [En ligne]. Available: http://www.ziviani.net/2011/how-to-create-an-apache-module. [Accès le 06 2018].

[79] «User and Group ID Services,» Apache Software Foundation, 2018. [En ligne]. Available: https://apr.apache.org/docs/apr/1.6/group__apr__user.html. [Accès le 06 2018].

[80] «The Open Group Base Specifications Issue 7, 2018 edition,» IEEE and The Open Group, 2001-2018. [En ligne]. Available: http://pubs.opengroup.org/onlinepubs/9699919799/functions/contents.html. [Accès le 01 2018].

[81] S. Frampton, «6.6. Linux Password & Shadow File Formats,» 2019. [En ligne]. Available: https://www.tldp.org/LDP/lame/LAME/linux-admin-made-easy/shadow-file-formats.html. [Accès le 01 2019].

[82] «correct_password.c,» 2013. [En ligne]. Available: https://github.com/cobyism/edimax-br-6528n/blob/master/AP/busybox-1.11.1/libbb/correct_password.c. [Accès le 01 2019].

[83] «crypt(3) - Linux man page,» 2019. [En ligne]. Available: https://linux.die.net/man/3/crypt. [Accès le 01 2019].

[84] «Compiling and Installing,» Apache Software Foundation, 2019. [En ligne]. Available: https://httpd.apache.org/docs/2.4/en/install.html. [Accès le 04 2019].

[85] T. K. Andrew G. Morgan, «The Linux-PAM Application Developers' Guide,» 08 2010. [En ligne]. Available: http://www.linux-pam.org/Linux-PAM-html/Linux-PAM_ADG.html. [Accès le 01 2019].

[86] T. K. Andrew G. Morgans, «The Linux-PAM System Administrators' Guide,» 08 2010. [En ligne]. Available: http://www.linux-pam.org/Linux-PAM-html/Linux-PAM_SAG.html. [Accès le 01 2019].

[87] «sys/stat.h,» The Open Group / IEEE, 2004. [En ligne]. Available: http://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/stat.h.html. [Accès le 01 2019].

[88] V. Gite, «Howto: Connect MySQL server using C program API under Linux or UNIX,» 05 2007. [En ligne]. Available: https://www.cyberciti.biz/tips/linux-unix-connect-mysql-c-api-program.html. [Accès le 01 2019].

[89] «Apache HTTP Server,» GitHub, 2019. [En ligne]. Available: https://github.com/apache/httpd. [Accès le 04 2019].

[90] «man - macros to format man pages,» Canonical Ltd., 2019. [En ligne]. Available: http://manpages.ubuntu.com/manpages/precise/man7/man.7.html. [Accès le 04 2019].

[91] V. Gite, «HowTo: Linux / UNIX Create a Manpage,» nixCraft, 16 11 2017. [En ligne]. Available: https://www.cyberciti.biz/faq/linux-unix-creating-a-manpage/. [Accès le 01 2019].

[92] «Logiciel: MatLab,» MatWorks, 2019. [En ligne]. Available: https://www.mathworks.com. [Accès le 01 2019].

[93] «Matlab : WebRead,» MatWorks, 2019. [En ligne]. Available: https://www.mathworks.com/help/matlab/ref/webread.html?s_tid=srchtitle. [Accès le 01 2019].

[94] «MatLab : TimeIt,» MatWorks, 2019. [En ligne]. Available: https://www.mathworks.com/help/matlab/ref/timeit.html?searchHighlight=timeit&s_tid=doc_srchtitle. [Accès le 01 2019].

[95] «MatLab : Tic,» MatWorks, 2019. [En ligne]. Available: https://www.mathworks.com/help/matlab/ref/tic.html?searchHighlight=tic%20toc&s_tid=doc_srchtitle. [Accès le 01 2019].

[96] «pam_unix(8) - Linux man page,» 2019. [En ligne]. Available: https://linux.die.net/man/8/pam_unix. [Accès le 01 2019].

[97] Neelesh, «How to do Authentication by PAM in PHP code,» 2019. [En ligne]. Available: https://neelesh-gurjar.livejournal.com/16491.html. [Accès le 01 2019].

[98] «htaccess,» wordpress.org, 2019. [En ligne]. Available: https://codex.wordpress.org/htaccess. [Accès le 01 2019].

[99] «Logiciel: WebMin,» WebMin, 2019. [En ligne]. Available: http://www.webmin.com. [Accès le 01 2019].

[100] M. Stephens, «Using Apache htaccess to password protect documents,» IDRsolutions Ltd, 2019. [En ligne]. Available: https://blog.idrsolutions.com/2014/01/using-apache-htaccess-password-protect-documents/. [Accès le 01 2019].

ANNEXE I

Articles des médias

LE DEVOIR

Le Devoir

Économie, mardi 29 mai 2018 - 562 mots, p. B1

Les données de clients de la Banque de Montréal et de Simplii auraient été piratées

François Desjardins

La Banque de Montréal et Simplii Financial tentent de comprendre toute l'étendue de ce qui s'est passé dimanche, au moment où chacune des deux institutions a appris que des fraudeurs avaient peut-être mis la main sur les renseignements personnels de certains clients.

Coup sur coup, la Banque de Montréal et Simplii, banque virtuelle de la CIBC, ont diffusé des communiqués lundi avisant leurs clients de surveiller leurs comptes. Simplii a estimé que 40 000 clients sont potentiellement touchés, tandis qu'à la Banque de Montréal, " moins de 50 000 clients ont été touchés ", a indiqué une porte-parole en fin de journée.

" Le dimanche 27 mai, des fraudeurs ont communiqué avec BMO, alléguant être en possession de certains renseignements personnels et financiers concernant un petit nombre de clients. Nous croyons que cette attaque a été menée de l'extérieur du pays ", a écrit la Banque de Montréal en matinée.

" Nous avons pris les mesures nécessaires dès que l'incident est survenu et nous sommes persuadés d'avoir réussi à écarter les risques relevés menaçant les comptes des clients visés. [...] Nous conseillons à nos clients de surveiller leurs comptes et d'aviser BMO de toute activité suspecte ", a ajouté la Banque en mentionnant un travail étroit avec " les autorités compétentes ".

Les grandes fuites de renseignements personnels dans le secteur bancaire n'ont pas été particulièrement nombreuses au cours des dernières années. De manière plus large dans le domaine des services financiers, toutefois, Equifax a été frappée l'an dernier par un piratage qui a mis en péril les données d'environ 19 000 personnes.

Le Journal de Québec (réf. site web)

29 mars 2018 - 353 mots

Une filiale du groupe américain Under Armour victime d'une cyberattaque

AFP

NEW YORK | L'équipementier sportif américain Under Armour a annoncé jeudi qu'une de ses filiales, MyFitnessPal, a été victime d'un piratage informatique portant sur le vol de données de... Voir l'article

Ce document référence un lien URL de site non hébergé par CEDROM-SNi.

Aussi paru dans 29 mars 2018 -
30 mars 2018 -

ICI Radio-Canada - Toronto (site web)
Toronto, lundi 26 février 2018 - 268 mots

Une agence de transport en commun ontarienne victime d'une cyberattaque

La présidente du Conseil du Trésor affirme qu'aucune information confidentielle n'a été compromise lors de la cyberattaque contre Metrolinx le mois dernier.

L'agence de transport régional Metrolinx, qui dessert les régions de Toronto et de Hamilton, a confirmé avoir été la cible de pirates informatiques nord-coréens. La présidente du Conseil du Trésor de l'Ontario, Eleanor McMahon Photo : Radio-Canada/Barry Smith

Le gouvernement se fait toutefois rassurant, disant qu'une équipe de pirates informations travaillent pour le gouvernement pour déceler ce genre d'attaques et les prévenir.On s'inquiète toujours quand il y a une attaque comme ça, mais cette fois-ci, je veux rassurer tout le monde....Il n'y a aucun problème.Eleanor McMahon, présidente du Conseil du Trésor

Metrolinx gère les trains de banlieue GO et le train UP Express qui relie le centre-ville de Toronto à l'aéroport Pearson, ainsi que la carte de paiement électronique Presto utilisée par les usagers.

Or, un logiciel malveillant, qui était passé par un serveur en Russie, a infecté des ordinateurs de l'agence.

Sa porte-parole, Anne Marie Aikins, assure toutefois qu'il n'y a eu aucun risque pour la sécurité et que les renseignements personnels des usagers n'ont pas été compromis.

Mme Aikins refuse toutefois de donner plus de détails au sujet de l'attaque informatique, « pour des raisons de sécurité ».

« De grandes organisations comme la nôtre font face à de possibles attaques chaque jour », dit-elle.

D'après les renseignements de CBC

À lire aussi :Près de 100 000 clients de Bell Canada victimes de piratage informatiquePlus de 19 000 Canadiens touchés par la cyberattaque contre Equifax

Cet article est paru dans ICI Radio-Canada - Toronto (site web)

LE DEVOIR
Le Devoir
Économie, mercredi 24 janvier 2018 - 459 mots, p. B1

Renseignements personnels piratés chez Bell

"Moins de 100 000 clients" seraient affectés, affirme un porte-parole de l'entreprise

Sarah R. Champagne

Bell Canada a avisé mardi certains de ses clients que leurs renseignements personnels pourraient avoir fait l'objet d'un piratage informatique.

La Gendarmerie royale du Canada (GRC) confirme qu'elle enquête présentement sur ce dossier, sans pouvoir donner plus de détails. La compagnie a également avisé le Commissariat à la protection de la vie privée du Canada que ces données " avaient été compromises ".

Ils seraient " moins de 100 000 clients " à être affectés, a indiqué à La Presse canadienne Nathan Gibson, porte-parole de l'entreprise. Rien n'indique pour le moment que le pirate ait pu accéder à des données financières ou à des mots de passe, avance-t-on.

C'est ainsi la deuxième fois en huit mois que Bell avise ses clients d'une fuite de renseignements personnels. Cette compagnie de télécommunications, parmi les principales au pays, avait ainsi révélé en mai 2017 qu'un pirate informatique avait obtenu illégalement 1,9 million d'adresses courriel actives et environ 1700 noms et numéros de téléphone actifs.

Utilisations possibles des informations

" Ils sont au courant de la brèche de sécurité, ce qui est en soi bon signe ", affirme le chercheur montréalais en cybersécurité Olivier Bilodeau.

Impossible de savoir si cette attaque particulière aura ou a déjà des conséquences. M. Bilodeau ne cache cependant pas que les pirates dérobent la plupart du temps ces informations à des fins malveillantes. " Accumuler autant d'informations, c'est peu souvent pour des utilisations légitimes, en effet ", expose-t-il, commentant les cas qui se multiplient.

M. Bilodeau, qui travaille pour la firme GoSecure, explique qu'un courriel peut par exemple devenir une sorte de point d'infection. Le pirate peut les utiliser " dans des campagnes d'attaques ciblées " pour simuler la provenance d'un courriel par exemple et hameçonner une personne pour lui soutirer encore davantage de renseignements.

LA PRESSE CANADIENNE
La Presse Canadienne
Nouvelles Générales, Affaires, lundi 13 novembre 2017 - 14:22:54 UTC -0500 - 256 mots

VerticalScope confirme que six de ses sites internet ont été piratés

La Presse canadienne

TORONTO - VerticalScope a indiqué lundi que six des sites internet communautaires qu'elle gère avaient été piratés le mois dernier. Cet incident vient s'ajouter à une série de récents piratages des informations personnelles en ligne des Canadiens.

Dans une déclaration transmise par courriel, la société a précisé qu'une « partie non autorisée » avait réussi à accéder aux données des membres des communautés de six de ses sites web du 17 au 21 octobre. Une partie de ces données a été publiée en ligne, a ajouté VerticalScope.

Les données volées ne comprenaient pas d'informations de cartes de crédit ou de comptes bancaires, mais certains noms d'utilisateurs, adresses courriel, mots de passe chiffrés et adresses IP pour les sites web DIYchatroom.com, PBnation.com, HysterSisters.com, ToyotaNation.com, JeepForum.com et WatchUSeek.com.

VerticalScope, détenue en majorité par l'éditeur Torstar (TSX:TS.B), a dit avoir fait expirer tous les mots de passe des sites internet touchés et a averti les utilisateurs touchés par le piratage. Il affirme avoir informé le commissaire fédéral à la protection de la vie privée, le service de police de Toronto et la division des cybercrimes de la Gendarmerie royale du Canada.

Environ 8000 Canadiens ont été victimes, plus tôt cette année, d'un important vol d'informations personnelles détenues par Equifax. Les données piratées comprenaient, dans certains cas, des informations de cartes de crédit. La brèche de sécurité a eu lieu en juillet, mais elle n'a été annoncée par l'agence de crédit qu'en septembre.

Aussi paru dans 13 novembre 2017 -

ANNEXE II

OWASP TOP 10 2017

A1:2017-Injection

A2:2017-Broken Authentication
The prevalence of broken authentication is widespread due to the design and implementation of most identity and access controls. Session management is the bedrock of authentication and access controls, and is present in all stateful applications.

Attackers can detect broken authentication using manual means and exploit them using automated tools with password lists and dictionary attacks.

A3:2017-Sensitive Data Exposure

A4:2017-XML External Entities (XXE)

A5:2017-Broken Access Control
Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers.

Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc.

A6:2017-Security Misconfiguration
Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Automated scanners are useful for detecting misconfigurations, use of default accounts or configurations, unnecessary services, legacy options, etc.

A7:2017-Cross-Site Scripting (XSS)

A8:2017-Insecure Deserialization

A9:2017-Using Components with Known Vulnerabilities

A10:2017-Insufficient Logging & Monitoring

ANNEXE III

Documents de projet

Les documents de projets réalisés pour ADN Informatique suivant la norme ISO29110 sont disponibles sur le site Web de l'entreprise à l'adresse http://absec.adninformatique.com/

Les documents réalisés sont :

- Plan de projet
- Spécifications
- Composantes
- Architecture/Conception
- Plan et procédure de tests

En complément, un guide d'utilisation du module est en cours de rédaction et sera disponible pour téléchargement sur le site.

ANNEXE IV

Code source du module

Code source du module de base d'apache généré avec « apxs »

```
822:  /*
823:  **  mod_absec.c -- Apache sample absec module
824:  **  [Autogenerated via ``apxs -n absec -g'']
825:  **
826:  **  To play with this sample module first compile it into a
827:  **  DSO file and install it into Apache's modules directory
828:  **  by running:
829:  **
830:  **    $ apxs -c -i mod_absec.c
831:  **
832:  **  Then activate it in Apache's apache2.conf file for instance
833:  **  for the URL /absec in as follows:
834:  **
835:  **    #   apache2.conf
836:  **    LoadModule absec_module modules/mod_absec.so
837:  **    <Location /absec>
838:  **    SetHandler absec
839:  **    </Location>
840:  **
841:  **  Then after restarting Apache via
842:  **
843:  **    $ apachectl restart
844:  **
845:  **  you immediately can request the URL /absec and watch for the
846:  **  output of this module. This can be achieved for instance via:
847:  **
848:  **    $ lynx -mime_header http://localhost/absec
849:  **
850:  **  The output should be similar to the following one:
851:  **
852:  **    HTTP/1.1 200 OK
853:  **    Date: Tue, 31 Mar 1998 14:42:22 GMT
854:  **    Server: Apache/1.3.4 (Unix)
855:  **    Connection: close
856:  **    Content-Type: text/HTML
857:  **
858:  **    The sample page from mod_absec.c
859:  */
860:
861:  #include "httpd.h"
862:  #include "http_config.h"
863:  #include "http_protocol.h"
864:  #include "ap_config.h"
865:
866:  /* The sample content handler */
867:  static int absec_handler(request_rec *r)
868:  {
869:      if (strcmp(r->handler, "absec")) {
870:          return DECLINED;
871:      }
872:      r->content_type = "text/HTML";
873:
874:      if (!r->header_only)
875:          ap_rputs("The sample page from mod_absec.c\n", r);
876:      return OK;
877:  }
878:
879:  static void absec_register_hooks(apr_pool_t *p)
880:  {
881:      ap_hook_handler(absec_handler, NULL, NULL, APR_HOOK_MIDDLE);
882:  }
883:
884:  /* Dispatch list for API hooks */
885:  module AP_MODULE_DECLARE_DATA absec_module = {
886:      STANDARD20_MODULE_STUFF,
```

```
887:     NULL,                  /* create per-dir    config structures */
888:     NULL,                  /* merge  per-dir    config structures */
889:     NULL,                  /* create per-server config structures */
890:     NULL,                  /* merge  per-server config structures */
891:     NULL,                  /* table of config file commands       */
892:     absec_register_hooks  /* register hooks                      */
893: };
```

Code source pour accéder /etc/passwd, /etc/shadow, /etc/group

```
894: ////////////////////////////////////////////////////////////////
895: // Unix file based auth
896: int check_user(char* user, char* pass) {
897: ////////
898: // Get UID, GIDs for the user
899: /* Working example, but just UID not PW */
900:     apr_status_t ret;
901:     apr_uid_t i;
902:     apr_gid_t g;
903:     apr_pool_t *pool;
904: 
905:     apr_initialize();
906:     apr_pool_create(&pool, NULL);
907:     ret = apr_uid_get(  &i, &g, user, pool);
908:     printf("Result2: G:%d, I:%d \n<br/>", g,i);
909: 
910: ////////
911: /* Retrieve PW from /etc/passwd */
912: /* Should include <pwd.h> */
913:     struct passwd *pw;
914:     if((pw = getpwnam(user)) == NULL)
915:     {
916:         printf("NULL \n<br/>");
917:         return HTTP_UNAUTHORIZED;
918:     }
919:     else
920:     {
921:         printf("Unix PW : %s \n<br/>", pw->pw_passwd);
922:     }
923: 
924: ////////
925: /* Retrieve PW from /etc/shadow */
926: /* Should include <shadow.h> */
927:     struct spwd *spw;
928:     errno = 0;
929:     if((spw = getspnam(user)) == NULL)
930:     {
931:         printf("NULL %d\n<br/>", errno);
932:         return HTTP_UNAUTHORIZED;
933:     }
934:     else
935:     {
936:         printf("Shadow PW : %s \n<br/>", spw->sp_pwdp);
937:     }
938: 
939:     if (spw->sp_pwdp[0] == 'x' || spw->sp_pwdp[0] == '*' || spw->sp_pwdp[0] ==
'!') {
940:         return HTTP_UNAUTHORIZED;
941:     }
942: 
943: ////////
944: /* Encrypt and compare shadow password */
945: 
946:     char *encrypted;
947:     const char *correct;
948:     int rrr;
949:     encrypted = crypt(pass, spw->sp_pwdp);
950:     rrr = strcmp(encrypted, spw->sp_pwdp);
951:     printf("compare pw : %s \n<br/>", encrypted);;
952:     printf("compare : %d \n<br/>", rrr);
953:     if (rrr!=0) {
954:         return HTTP_UNAUTHORIZED;
```

```
955:      }
956:
957:      // now check supplemental groups
958:       gid_t grouplist[16];
959:      int grouplistsize = 16;
960:      int groupreturn;
961:      groupreturn = getgrouplist("jlcyr", g, grouplist, &grouplistsize);
962:      if (groupreturn != -1) {
963:          printf("OK liste des groupes (%d)<br/>\r\n", grouplistsize);
964:          for (i=0; i<grouplistsize; i++) {
965:              printf("group: %d\r\n", grouplist[i]);
966:              // If file is group readable and match a supplemental group return
content
967:          }
968:      } else {
969:          printf("Erreur<br/>\r\n");
970:          return OK;
971:      }
972:  }
```

Code source pour utiliser les fonctions de « PAM »

```
973:  // Global var for passing fake response to PAM callback
974:  struct pam_response *reply;
975:
976:  ////////////////////////////////////////////////////////////////
977:  // PAM response callback function
978:  int converse(int n, const struct pam_message **msg,
979:        struct pam_response **resp, void *data)
980:  {
981:      // Return globally set response
982:      *resp = reply;
983:      return PAM_SUCCESS;
984:  }
985:
986:  ////////////////////////////////////////////////////////////////
987:  // define PAM callback function
988:  struct pam_conv conv = { converse, 0 };
989:
990:  int check_user(char* user, char* pass) {
991:      ////////
992:      // Connect to PAM to auth user
993:      pam_handle_t * pamh = NULL;
994:      int rret;
995:
996:      if((rret = pam_start("httpd", user, &conv, &pamh)) != PAM_SUCCESS) {
997:        return PAM_ERROR_START;
998:      }
999:
1000:           // Set the PAM callback function response (would call for password)
1001:           reply = (struct pam_response *)malloc(sizeof(struct pam_response));
1002:           reply[0].resp = strdup(pass); // password received in basic auth
1003:           reply[0].resp_retcode = 0;
1004:
1005:           if((rret = pam_authenticate(pamh, 0)) != PAM_SUCCESS) {
1006:             return PAM_ERROR_INVALID_CRED;
1007:           }
1008:
1009:           if(pam_end(pamh, rret) != PAM_SUCCESS) {
1010:           }
1011:       }
```

Code source pour utiliser une base de données MySQL

```
1012:       ////////////////////////////////////////////////////////////////
1013:       // mysql based permission lookup
1014:       int perms_lookup(request_rec *r, struct stat *fperm)
1015:       {
1016:          MYSQL *conn;
1017:          MYSQL_RES *res;
```

```
1018:          MYSQL_ROW row;
1019:
1020:          char *server = "localhost";
1021:          char *user = "jlcyr";
1022:          char *password = "password"; /* set me first */
1023:          char *database = "absec";
1024:
1025:          conn = mysql_init(NULL);
1026:
1027:          /* Connect to database */
1028:          if (!mysql_real_connect(conn, server,
1029:                user, password, database, 0, NULL, 0)) {
1030:             printf("%s\n", mysql_error(conn));
1031:             return(0);
1032:          }
1033:
1034:           char query[256];
1035:           sprintf(query, "select * from urls where url='%s'", r->uri);
1036:          /* send SQL query */
1037:          //if (mysql_query(conn, "show tables")) {
1038:          if (mysql_query(conn, query)) {
1039:             printf("%s\n", mysql_error(conn));
1040:             return(-1);
1041:          }
1042:
1043:          res = mysql_use_result(conn);
1044:
1045:          /* output table name */
1046:          printf("MySQL data:\n<br/>");
1047:          int cnt = 0;
1048:          while ((row = mysql_fetch_row(res)) != NULL) {
1049:             printf("%s %d %d %o \n<br/>", row[0], atoi(row[1]), atoi(row[2]),
atoi(row[3]));
1050:                fperm->st_uid = atoi(row[1]);
1051:                fperm->st_gid = atoi(row[2]);
1052:                fperm->st_mode = atoi(row[3]);
1053:                cnt = cnt + 1;
1054:          }
1055:
1056:          if (cnt==0) {
1057:              sprintf(query, "insert into urls (url, uid, gid, perms) values
('%s', %d, %d, %d)", r->uri, fperm->st_uid, fperm->st_gid, fperm->st_mode);
1058:               if (mysql_query(conn, query)) {
1059:                   printf("%s\n", mysql_error(conn));
1060:                   return(0);
1061:               }
1062:              printf("Aucune donnée\n<br/>");
1063:          }
1064:
1065:          /* close connection */
1066:          mysql_free_result(res);
1067:          mysql_close(conn);
1068:          return(0);
1069:       }
```

Code principal du module

```
1070:       ////////////////////////////////////////////////////////////////
1071:       // Macro déclaration du module
1072:       module AP_MODULE_DECLARE_DATA absec_module;
1073:
1074:       ////////////////////////////////////////////////////////////////
1075:       // structure de configuration du module
1076:       typedef struct {
1077:           int default_uid;
1078:           int default_gid;
1079:           int default_perms;
1080:       } authnz_config_rec;
1081:
1082:       static void *authnz_absec_config(apr_pool_t *pool, char *x)
1083:       {
1084:           return apr_pcalloc(pool, sizeof(authnz_config_rec));
```

```
1085:       }
1086:
1087:       static const char* set_default_perms(cmd_parms* cmd, void* cfg, const
char* val) {
1088:           int octal, decimal;
1089:           octal = atoi(val);
1090:           decimal = 0;
1091:           int i=0;
1092:           while (octal != 0)
1093:           {
1094:               decimal =  decimal +(octal % 10)* pow(8, i++);
1095:               octal = octal / 10;
1096:           }
1097:
1098:           ((authnz_config_rec*)cfg)->default_perms = decimal;
1099:           return NULL;
1100:       }
1101:
1102:       static const char* set_default_uid(cmd_parms* cmd, void* cfg, const char*
val) {
1103:           ((authnz_config_rec*)cfg)->default_uid = atoi(val);
1104:           return NULL;
1105:       }
1106:
1107:       static const char* set_default_gid(cmd_parms* cmd, void* cfg, const char*
val) {
1108:           ((authnz_config_rec*)cfg)->default_gid = atoi(val);
1109:           return NULL;
1110:       }
1111:
1112:       ////////////////////////////////////////////////////////////////
1113:       static const command_rec absec_auth_basic_cmds[] =
1114:       {
1115:       /*    AP_INIT_ITERATE("AuthBasicProvider", add_authn_provider, NULL,
OR_AUTHCFG,
1116:                           "specify the auth providers for a directory or
location"),*/
1117:           AP_INIT_TAKE1("ABSECDefaultPerms", set_default_perms, NULL,
OR_AUTHCFG,
1118:                         "Set to default octal value of perms "),
1119:           AP_INIT_TAKE1("ABSECDefaultUID", set_default_uid, NULL, OR_AUTHCFG,
1120:                         "Set to default octal value of perms "),
1121:           AP_INIT_TAKE1("ABSECDefaultGID", set_default_gid, NULL, OR_AUTHCFG,
1122:                         "Set to default octal value of perms "),
1123:       /*    AP_INIT_TAKE12("AuthBasicFake", add_basic_fake, NULL, OR_AUTHCFG,
1124:                           "Fake basic authentication using the given expressions
for "
1125:                           "username and password, 'off' to disable. Password
defaults "
1126:                           "to 'password' if missing."),
1127:           AP_INIT_TAKE1("AuthBasicUseDigestAlgorithm", set_use_digest_algorithm,
1128:                           NULL, OR_AUTHCFG,
1129:                           "Set to 'MD5' to use the auth provider's authentication
"
1130:                           "check for digest auth, using a hash of
'user:realm:pass'"),*/
1131:           {NULL}
1132:       };
1133:
1134:       ////////////////////////////////////////////////////////////////
1135:       // Validate authentification (user/pass)
1136:       static authn_status authn_check_absec(request_rec *r, const char* user,
const char* password)
1137:       {
1138:           ap_log_rerror("mod_absec.c", 269, 1, APLOG_ERR, APR_SUCCESS, r,
"authn_check_absec : user : %s, pass : %s", user, password);
1139:
1140:           int pam_result = check_user(user, password);
1141:           if ((pam_result==PAM_ERROR_START) || (pam_result==PAM_ERROR_STOP)) {
1142:               return HTTP_INTERNAL_SERVER_ERROR;
1143:           }
1144:           if (pam_result==PAM_ERROR_INVALID_CRED) {
```

```
1145:                ap_log_rerror("mod_absec.c", 269, 1, APLOG_ERR, APR_SUCCESS, r,
"authn_check_absec : DENIED");
1146:                return AUTH_DENIED;
1147:            }
1148:            ap_log_rerror("mod_absec.c", 269, 1, APLOG_ERR, APR_SUCCESS, r,
"authn_check_absec : GRANTED");
1149:            return AUTH_GRANTED;
1150:
1151:            // Example code
1152:            /*
1153:            if (strcmp(user, "joe")) {
1154:                return AUTH_USER_NOT_FOUND;
1155:            } else {
1156:                if (strcmp(password, "poi")) {
1157:                    return AUTH_DENIED;
1158:                } else {
1159:                    return AUTH_GRANTED;
1160:                }
1161:            }
1162:            */
1163:
1164:            // Possible return status
1165:            // AUTH_GENERAL_ERROR
1166:            // AUTH_USER_NOT_FOUND
1167:            // AUTH_USER_FOUND
1168:            // AUTH_DENIED
1169:            // AUTH_GRANTED
1170:        }
1171:
1172:        ////////////////////////////////////////////////////////////////
1173:        // Validate autorisation (ressource access)
1174:        // request_rec contain user/pass if basic auth
1175:        // it could also contain cookies if cookies based auth
1176:        static authz_status authz_check_absec(request_rec *r, const char
*require_args, const void *parsed_require_args)
1177:        {
1178:            char *user = r->user;
1179:            ap_log_rerror("mod_absec.c", 342, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : user : %s", user);
1180:
1181:            ////////
1182:            /* http method validate the perm asked (r/w vs get/post,put) */
1183:            //ap_rprintf(r, "Before Method: %s<br/>\r\n", r->method);
1184:
1185:            int permmask = 0;
1186:            if (strcmp(r->method,"GET")==0) permmask=0444;     // r
1187:            if (strcmp(r->method,"PUT")==0) permmask=0222;     // w
1188:            if (strcmp(r->method,"POST")==0) permmask=0222;    // w
1189:            if (strcmp(r->method,"DELETE")==0) permmask=0111;  // x
1190:
1191:            authnz_config_rec *cfg = ap_get_module_config(r->per_dir_config,
&absec_module);
1192:            struct stat fperm;
1193:            fperm.st_mode = cfg->default_perms;
1194:            ap_log_rerror("mod_absec.c", 329, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : default perms : %o", cfg->default_perms);
1195:            fperm.st_uid = cfg->default_uid;
1196:            ap_log_rerror("mod_absec.c", 329, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : default uid : %o", cfg->default_uid);
1197:            fperm.st_gid = cfg->default_gid;
1198:            ap_log_rerror("mod_absec.c", 329, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : default gid : %o", cfg->default_gid);
1199:
1200:            ////////
1201:            /* check file permission on filesystem */
1202:            /* should include <sys/stat.h> */
1203:            int status;
1204:            //status = stat(r->filename, &fperm);
1205:            status = perms_lookup(r, &fperm);
1206:            //ap_rprintf(r, "Result mysql: %d<br/>\r\n",);
1207:            if (status==-1) {
1208:                ap_rprintf(r, "stat erreur %d", errno);
```

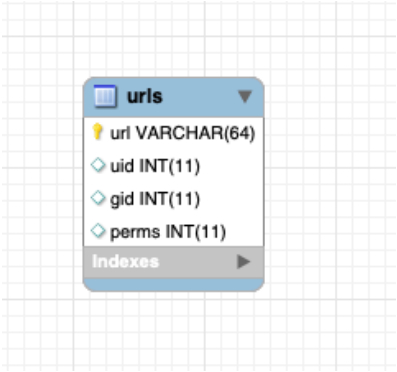
```
1209:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : stat erreur %d", errno);
1210:                return (OK);
1211:            }
1212:
1213:            //ap_rprintf(r, "File perms %o, owner %d, group %d (status
%d)<br/>\r\n", fperm.st_mode, fperm.st_uid, fperm.st_gid, status);
1214:
1215:            /* check if any permission (ogw) match method (get r, put/post w,
delete x) */
1216:            if ((fperm.st_mode & permmask)==0) {
1217:                /* no permission match, return don't even have to check user perms
*/
1218:                ap_rprintf(r, "Aucune permission pour la méthode %s (%o, %o)", r-
>method, fperm.st_mode, permmask);
1219:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Aucune permission pour la méthode %s (%o, %o)", r->method,
fperm.st_mode, permmask);
1220:                return AUTHZ_DENIED;
1221:            }
1222:
1223:            // If file is world accessible for asked method return content
1224:            if (fperm.st_mode & 0x7 & permmask) {
1225:                /* if so, return, no need to check user perms */
1226:                //ap_rprintf(r, "Fichier public<br/>\r\n");
1227:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier accessible a tous");
1228:                return AUTHZ_GRANTED;
1229:            }
1230:
1231:            if (!user) {
1232:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : No user");
1233:                return AUTHZ_DENIED_NO_USER;
1234:            }
1235:
1236:            struct passwd *pw;
1237:            pw = getpwnam(user);
1238:
1239:            // If file is user readable and user match return content
1240:            if ((fperm.st_uid==pw->pw_uid) && (fperm.st_mode & 0700 & permmask)) {
1241:                //ap_rprintf(r, "Fichier propriétaire<br/>\r\n");
1242:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier propriétaire<br/>\r\n");
1243:                return AUTHZ_GRANTED;
1244:            } else {
1245:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : No user access %d", pw->pw_uid);
1246:            }
1247:
1248:            // If file is group readable and primary group match return content
1249:            if ((fperm.st_gid==pw->pw_gid) && (fperm.st_mode & 0070 & permmask)) {
1250:                //ap_rprintf(r, "Fichier groupe<br/>\r\n");
1251:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : Fichier groupe<br/>\r\n");
1252:                return AUTHZ_GRANTED;
1253:            } else {
1254:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : No group access %d", pw->pw_gid);
1255:            }
1256:
1257:            // check supplemental groups
1258:            gid_t grouplist[16];
1259:            int grouplistsize = 16;
1260:            int groupreturn;
1261:            groupreturn = getgrouplist(user, pw->pw_gid, grouplist,
&grouplistsize);
1262:            if (groupreturn >= 0) {
1263:                ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : liste groups (%d)", grouplistsize);
1264:                for (int i=0; i<grouplistsize; i++) {
1265:                    ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS,
r, "authz_check_absec : group %d", grouplist[i]);
```

```
1266:                   // If file is group readable and match a supplemental group
return content
1267:                   if ((fperm.st_gid==grouplist[i]) && (fperm.st_mode & 0070 &
permmask)) {
1268:                       ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR,
APR_SUCCESS, r, "authz_check_absec : Fichier group supp");
1269:                       return AUTHZ_GRANTED;
1270:                   }
1271:               }
1272:           }
1273:
1274:           ap_log_rerror("mod_absec.c", 282, 1, APLOG_ERR, APR_SUCCESS, r,
"authz_check_absec : DENIED<br/>\r\n");
1275:           return AUTHZ_DENIED;
1276:
1277:           if (r->user==NULL) {
1278:               return AUTHZ_DENIED;
1279:           }
1280:
1281:           return AUTHZ_GRANTED;
1282:       }
1283:
1284:       ////////////////////////////////////////////////////////////////
1285:       static const authn_provider authn_absec_provider =
1286:       {
1287:           &authn_check_absec,
1288:           NULL
1289:       };
1290:
1291:       ////////////////////////////////////////////////////////////////
1292:       static const authz_provider authz_absec_provider =
1293:       {
1294:           &authz_check_absec,
1295:           NULL
1296:       };
1297:
1298:       ////////////////////////////////////////////////////////////////
1299:       static void absec_register_hooks(apr_pool_t *p)
1300:       {
1301:           ap_register_auth_provider(p, AUTHN_PROVIDER_GROUP, "absec", "0",
&authn_absec_provider, AP_AUTH_INTERNAL_PER_CONF);
1302:           ap_register_auth_provider(p, AUTHZ_PROVIDER_GROUP, "absec", "0",
&authz_absec_provider, AP_AUTH_INTERNAL_PER_CONF);
1303:       }
1304:
1305:       ////////////////////////////////////////////////////////////////
1306:       /* Dispatch list for API hooks */
1307:       AP_DECLARE_MODULE(absec) = {
1308:           STANDARD20_MODULE_STUFF,
1309:           authnz_absec_config,   /* create per-dir    config structures */
1310:           NULL,                  /* merge  per-dir    config structures */
1311:           NULL,                  /* create per-server config structures */
1312:           NULL,                  /* merge  per-server config structures */
1313:           absec_auth_basic_cmds, /* table of config file commands       */
1314:           absec_register_hooks   /* register hooks                      */
1315:       };
```

ANNEXE V

Schémas base de données

La base de données minimale crée pour la réalisation du module et des tests a la structure suivante :



ANNEXE VI

Compilation d'Apache HTTPD

Installation d'Apache - à partir des sources, avec l'option pour l'exécuter comme superutilisateur (« root »)

L'installation par défaut d'apache ne permet pas de l'exécuter avec le super utilisateur ``root''. Faire une installation manuelle, en ajoutant l'option de compilation.

```
1316:       CFLAGS=-DBIG_SECURITY_HOLE
```

Téléchargement des sources d'Apache

```
1317:       wget "http://apache.mirror.rafal.ca//httpd/httpd-2.4.29.tar.gz"
1318:       tar -zxvf httpd-2.4.29.tar.gz
1319:
1320:       cd srclib
1321:       wget "http://httpd-mirror.sergal.org/apache//apr/apr-1.6.3.tar.gz"
1322:       tar -zxvf apr-1.6.3.tar.gz
1323:       ln -s apr-1.6.3 apr
1324:
1325:       wget "http://httpd-mirror.sergal.org/apache//apr/apr-util-1.6.1.tar.gz"
1326:       tar -zxvf apr-util-1.6.1.tar.gz
1327:       ln -s apr-util-1.6.1 apr-util
1328:
1329:       wget "http://httpd-mirror.sergal.org/apache//apr/apr-iconv-1.2.2.tar.gz"
1330:       tar -zxvf apr-iconv-1.2.2.tar.gz
1331:       ln -s apr-iconv-1.2.2 apr-iconv
1332:
1333:       https://github.com/libexpat/libexpat/archive/R_2_2_5.tar.gz
1334:       apr-util/XML/expat/lib
```

Note: libapr1-dev et libaprutil1-dev peuvent être installées manuellement avec:

Configuration, compilation et installation

```
1335:       cd httpd-2.4.29/
1336:       env CFLAGS="-DBIG_SECURITY_HOLE"
1337:       ./configure --with-included-apr --with-included-apr-util --
prefix=/usr/local/apache2 (ou /home/jlcyr/apache2) CFLAGS=-DBIG_SECURITY_HOLE
1338:       make
1339:       sudo make install
```

Démarrage

```
1340:       sudo vi /usr/local/apache2/conf/httpd.conf (si nécessaire)
1341:       /usr/local/apache2/bin/apachectl start (utiliser sudo si port 80)
```

Pour rouler apache comme root, le recompiler avec les options suivantes:

```
1342:       ./configure --prefix=/usr/local/apache2 CFLAGS=-DBIG_SECURITY_HOLE
1343:
1344:       env CFLAGS="-DBIG_SECURITY_HOLE"
1345:       ./configure --prefix=/usr/local/apache2 CFLAGS=-DBIG_SECURITY_HOLE
1346:       make clean; sudo make install
1347:       sudo /usr/local/apache2/bin/apachectl start
```

ANNEXE VII

Tests MatLab

```
% Description: Matlab to benchmark web request
% Date: 2018-02-05
% By: Jean-Luc Cyr

% Test configuration
server = "http://10.211.55.15";
file = "/absec/index.html";
user = "jlcyr";
pass = "passwordp";
hits = 10;

% Load python interpretor
% shoud have pymysql in the lib path
[v, e, loaded] = pyversion;
if loaded
    disp('To change the Python version, restart MATLAB, then call pyversion.')
else
    pyversion /usr/local/bin/python
end

% Parameters
url = server+file;

% Denied don't check anything
options = weboptions('username',user,'password',pass);
set_perms(file,000);
x = run_test(url, hits, options);

% Allowed for user, check user/pass
options = weboptions('username',user,'password',pass);
set_perms(file,700);
y = run_test(url, hits, options);

% Allowed for all, don't check anything
options = weboptions('username',user,'password',pass);
set_perms(file,777);
z = run_test(url, hits, options);

perm = get_perms(file);

% Calculate mean request time
mx = mean(x);
my = mean(y);
mz = mean(z);

% Plot scatter results
l(1) = scatter(1:hits, x);
l(2) = refline([0 mx]);
hold on;
l(3) = scatter(1:hits, y);
l(4) = refline([0 my]);
l(5) = scatter(1:hits, z);
l(6) = refline([0 mz]);
hold off;

l=legend('perms 000',string(mx),'perms 700 invalid',string(my),'perms
777',string(mz));
l.FontSize = 14;
t=title('request: '+url);
t.FontSize = 14;
xlabel('essai');
ylabel('ms');

figHandles = findobj('Type','figure');
for p=1:length(figHandles)
    figure(figHandles(p));
end
```

```
disp('Test done');
beep;

function results = run_test(url, hits, options)
    % Clear previous results
    clear y;
    % Make the requests and time them
    disp('Running queries');
    for x = 1:hits
        display('itÈration: '+string(x))
        tic;
        for c = 1:100
            tcwebread(url,options);
        end
        y(x) = toc / 100.0;
        pause(1);
    end
    % Convert time in second to miliseconds
    results = y * 1000;
end

function perm = get_perms(url)
    py.importlib.import_module('pymysql');
    conn = py.pymysql.connect('10.211.55.15','jlcyr','password','absec');
    cur = conn.cursor;
    cur.execute('select * from urls where url="'+url+'"');
    res = cur.fetchone();
    perm = string(dec2base(res{4},8));
end

function set_perms(URL, perm)
    perm = floor(perm/100)*8^2+floor(mod(perm,100)/10)*8+floor(mod(perm,10));
    py.importlib.import_module('pymysql');
    conn = py.pymysql.connect('10.211.55.15','jlcyr','password','absec');
    cur = conn.cursor;
    cur.execute('update urls set perms='+string(perm)+' where url="'+url+'"');
    conn.commit();
end

function tcwebread(url, options)
    try
        webread(url,options);
    catch
        disp('.');
    end
end
```

ANNEXE VIII

Tests PHP

```
<?PHP
/*
   Description: Simulate authentification / autorisation page
   Author: Jean-Luc Cyr
   Project: ABSEC - Tests
   Date: 2019-03
*/

/* Validation du mot de passe avec le fichier /etc./shafow */
/* SRC : https://stackoverflow.com/questions/7940775/how-can-i-calculate-shadow-
password-for-sha-512-using-php-shell */
function authenticate($user, $pass){
  $shad =  preg_split("/[$:]/",`cat /etc/shadow | grep "^$user\:"`);
  if (!isset($shad[2]) || !isset($shad[3])) return false;
  $mkps = preg_split("/[$:]/",crypt($pass, '$'.$shad[2].'$'.$shad[3].'$'));
  return ($shad[4] == $mkps[3]);
}

/* get needed vars from server */
$user = $_SERVER["PHP_AUTH_USER"];
$pass = $_SERVER["PHP_AUTH_PW"];
$uri = $_SERVER["REQUEST_URI"];
$method = $_SERVER["REQUEST_METHOD"];
$file = $_SERVER["DOCUMENT_ROOT"].$uri;
/* Database connection info */
$servername = "localhost";
$username = "jlcyr";
$password = "password";
/* Default perms values */
$default_perms = 0700; // Octal value
$default_uid = 100;
$default_gid = 1000;

/* Check permission for asked URL */
// Create connection
$conn = new mysqli($servername, $username, $password);
//echo "<pre>"; var_dump($conn);echo "</pre>";
// Check connection
if (mysqli_connect_error()!=0) {
    header("HTTP/1.1 500 Internal Server Error");
    echo "No database connexion";
    echo "Connection failed: " . mysqli_connect_error();
    exit();
}

$sql = "SELECT * FROM absec.urls WHERE url='".$uri."'";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
    $row = mysqli_fetch_assoc($result);
} else {
    $row = array();
    $row['url'] = $uri;
    $row['uid'] = $default_uid;
    $row['gid'] = $default_gid;
    $row['perms'] = $default_perms;
}

mysqli_close($conn);

/* Calculate perms mask associated with method */
switch($method) {
    case 'GET':
        $perms_mask = 0444;
    break;
    case 'POST':
```

```
    case 'PUT':
        $perms_mask = 0222;
    break;
    case 'DELETE':
        $perms_mask = 0111;
    break;
    default:
        $perms_mask = 0000;
    break;
}

/* Check basic cases */
// Allowed for all
if (($row['perms'] & $perms_mask & 07) > 0) {
    header("HTTP/1.1 200 OK");
    echo "Permis pour tous\r\n<br/>";
    exit();
}
// // Denied for all
if (($row['perms'] & $perms_mask) == 0) {
    header("HTTP/1.1 403 Unauthorized");
    echo "Aucune permission, refusé à tous\r\n<br/>";
    exit();
}
// Other cases need authentication
// We don't have auth so ask for it
if (!$_SERVER["PHP_AUTH_USER"]){
    header('WWW-Authenticate: Basic realm="My Realm"');
    header("HTTP/1.1 401 Unauthorized");
    exit();
}
// We have en auth, check if user valid
$user_info = posix_getpwnam($user);
// We don't find user in system file
if (!$user_info) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header("HTTP/1.1 401 Unauthorized");
    exit();
}
// Check if password is valid (against /etc/shadow)
if (!authenticate($user, $pass)){
    header('WWW-Authenticate: Basic realm="My Realm"');
    header("HTTP/1.1 401 Unauthorized");
    exit();
}
// Check user access
if (($row['perms'] & $perms_mask & 0700) && ($row['uid'] == $user_info['uid'])) {
    header("HTTP/1.1 200 OK");
    echo "Permission à l'utilisateur\r\n<br/>";
    exit();
}
// Check group access
if (($row['perms'] & $perms_mask & 0070) && ($row['gid'] == $user_info['gid'])) {
    header("HTTP/1.1 200 OK");
    echo "Permission au groupe\r\n<br/>";
    exit();
}
// Check supplementary groups


// We don't have any valid permission
header('WWW-Authenticate: Basic realm="My Realm"');
header("HTTP/1.1 403 Unauthorized");
echo "Aucune permission valide\r\n<br/>";
exit();

?>
```

ANNEXE IX

Base de données WordPress