UQAC
Université du Québec
à Chicoutimi

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

Par : Jean-Luc Cyr

Université du Québec à Chicoutimi

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 l'autorisation/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.

Une prépondérance de l'utilisation du serveur Apache pour délivrer 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 pour ce serveur capable de 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 élaborée.

Dans un premier temps, tel que présenté par tous les gabarits de gestion de la sécurité informationnelle, il est important d'identifier et classifier les différentes ressources contenues dans les applications.

Dans un second temps, il faut définir les intervenants qui interagiront avec ces ressources et les permissions associées aux relations entre les intervenants et les ressources. Plusieurs méthodologies sont proposées pour définir ces relations.

Sur ces bases, un modèle est défini et appliqué pour conserver l'identification des ressources, des intervenants et les relations entre eux. Ces informations servent de base pour développer un module de gestion des accès aux systèmes et de protection des ressources. La gestion de ces informations est documentée et déléguée aux développeurs de l'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

RÉSUMÉ ii
Table des matières iii
Liste des figures vi
Liste des tableaux vii
Introduction 1
Présentation de la problématique 2
Failles de sécurité dans les médias 2
Développement de produits 2
Systèmes utilisés 2
Constatations de l'OWASP 3
Systèmes existants 3
Objectifs du projet 5
Méthodologie 6
Échéancier 7
Financement 7
État de l'art 8
Systèmes utilisés sur le marché 9
Les standards en sécurité informatique 11
Les systèmes Web 13
L'environnement 13
Le protocole 14
Méthodes d'authentifications 14
Les serveurs HTTP 15
Les systèmes d'exploitation 16
Gestion des authentifications (usagers) 17
Comparaison des systèmes 17
Gestion des autorisations (objets) 19
Définition des ressources 19
Système d'exploitation (Unix) 20
Système d'exploitation (Windows) 21
Applications web 22

Gestion de la sécurité serveur web (Apache) 22
Développement / expérimentation 24
Choix de l'environnement de travail 24
Mise en place de l'environnement de travail 24
Phases de développement et d'implémentation 25
Phase 0 – Préparation du système et observations 26
État du système avant manipulation 26
Configuration par défaut du serveur web Apache HTTP 26
Installation des bibliothèques et outils de développement 26
Phase I – Gestion de l'authentification de base 28
Objectifs et suppositions 28
Réalisation 29
Conclusions 29
Phase II – Gestion de l'authentification avec PAM 31
Objectifs et suppositions 31
Réalisation 32
Observations et conclusions 32
Phase III – Gestion des autorisations avec le système de fichiers 33
Objectifs et suppositions 33
Détails techniques 33
Réalisation 34
Observations et conclusions 34
Phase IV – Gestion des autorisations avec une base de données (et journalisation) 35
Objectifs et suppositions 35
Réalisation 35
Observations et conclusions 35
Phase V – Outils de gestion 36
Objectifs et suppositions 36
Réalisation 36
Observations et conclusions 36
Phase VI – Filtre de contenu 37
Objectifs et suppositions 37
Réalisation 37

Observations et conclusions 37
Mise en production 38
Objectifs et suppositions 38
Réalisation 38
Collecte des résultats 38
Analyse des résultats 38
Observations et conclusions 38
Références 39
Annexe I – Articles concernant des piratages informatique 41
Annexe II – Fichiers de configuration par défaut d'Apache HTTP 45
Annexe III – OWASP Top 10 2017 47

Liste des figures

Figure 1 - Échéancier préliminaire 7
Figure 2 - BuiltWith - Server statistics 9
Figure 3 - w3techs - Server statistics 9
Figure 4 - BuiltWith - Language statistics 9
Figure 5 - w3techs - Language statistics 10
Figure 6 - w3techs - CMS statistics 10
Figure 7 - builtwith - CMS statistics 10
Figure 8 - Environnement web client-serveur 13
Figure 9 – w3techs - Adoption de http/2 13
Figure 10 - Apache architecture - P.22 15
Figure 11 - Request processing – P.44 15
Figure 12 - Apache 2 New Data Axis – P.47 16
Figure 13 - w3techs - Operating Systems statistics 16
Figure 14 - Ajax/Ajaj requêtes 19
Figure 15 - Http Basic Autentication 28
Figure 16 - Mécanisme de PAM 31
Figure 17 - Action HTTP vs filesystem permissions 33

Liste des tableaux

Tableau 1 - Estimation préliminaire 7
Tableau 2 - Méthodes HTTP et actions 20
Tableau 3 - Permissions de base système de fichier Unix 21
Tableau 4 - Exemple d'attribution de permission de fichier Unix 21
Tableau 5 - Modules de sécurité Apache 2.4 22
Tableau 6 - Modules de sécurité non-officiels Apache 2.4 23
Tableau 7 - Fichiers unix de base pour l'authentification 26

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 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.

Plusieurs solutions sont mises de l'avant pour essayer de sécuriser les systèmes mais chacune apporte ses problématiques particulières. La question « serait-il 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? » a donc été soulevée.

Le présent projet tente d'y répondre en proposant des outils et des guides pour séparer le plus possible la gestion de la sécurité du code applicatif tout en conservant une relation forte entre l'application gérant les données et le système gérant la sécurité.

À la suite de l'étude des technologies les plus répandues, l'environnement de développement de ce projet a été orienté vers 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.

[1] Ref: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Présentation de la problématique

Failles de sécurité dans les médias

Les médias nous rapportent régulièrement des attaques de pirates contre des propriétés Internet. Ces attaques touchent de multiples sites, de plus ou moins grande envergure, appartenant à toute sorte d'entreprises. 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 de transport ontarienne 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 sans doute pas de les révélées au grand public. C'est pourquoi de plus en plus les organismes réglementaires tente de légiférer pour forcer la divulgation de ces évènements afin d'en minimiser les conséquences et de permettre aux victimes d'en être informés afin qu'elles puissent agir de manière éclairée.

Développement de produits

Un grand nombre de site web sont développés à l'aide de produits tel que Wordpress[2], Joomlà[3], Drupal[4]; d'autres sont réalisés sous forme de projets autonomes en divers langages tel que PHP[5], Ruby[6], Python[7], ou autre. La plupart des systèmes disponibles autant 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 mis en place derrière un serveur web. Un faible nombre utilise plutôt des bibliothèques de code tel que Flask[8] et Django[9] en Python, à même l'application, pour implémenter le protocole HTTP plutôt que d'utiliser un serveur web tel qu'Apache HTTP.

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

Systèmes utilisés

Parmi les serveurs web les plus répandus l'on retrouve

- Apache[10]

[2] https://wordpress.org
[3] https://www.joomla.org
[4] https://www.drupal.org
[5] http://www.php.net
[6] https://www.ruby-lang.org/
[7] https://www.python.org
[8] http://flask.pocoo.org
[9] https://www.djangoproject.com
[10] http://httpd.apache.org

- Nginx[11]
- IIS[12]

En regardant les statistiques d'utilisation des serveurs, l'on peut se rendre compte que dans la majorité des systèmes utilisent Apache HTTP.

Ce serveur offre plusieurs modules pour la gestion de la sécurité tel que mod_auth_basic. Cependant, mon expérience sur le marché du travail m'a permis de constater que peu de ces fonctions sont couramment utilisés dans des projets. Seul l'utilisation de mod_auth_basic, conjointement avec des fichiers locaux « .htaccess » sont parfois utilisés en cours de développement.

Constatations de l'OWASP

L'OWASP[13] (Open Web Application Security Project) une organisation internationale sans but lucratif, qui a pour but de mettre en avant plan la sécurité des applications, a diffusé à trois reprises un top 10 des éléments de sécurité les plus critiques des applications web.

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.

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 eux, 3 des points (A3, A4 et A8 2013 et A2, A5 et A6 2017) sont directement relies à la gestion des authentifications et autorisation. Le détail de chacun pour 2017 se retrouve en Annexe III.

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 WAF (Web Application Firewall) « Appliance » (Imperva[14], BigIP F5[15], Cisco[16])

[11] https://www.nginx.com
[12] https://www.iis.net
[13] https://www.owasp.org/
[14] https://www.imperva.com
[15] https://f5.com/products/big-ip
[16] https://www.cisco.com/c/en/us/products/security/index.html

- Les WAF « Logiciel » (ModSecurity[17])
- Les proxy (Squid[18], Varnish[19])
- Des modules serveur (Apache : mod_auth_xxxx, nginx : naxsi, modsecurity)
- Des bibliothèques de code

Ces produits pour la plupart ajoutent une couche de sécurité devant les systèmes applicatifs. Dans certains cas, notamment les WAF, le but premier est de détecter les attaques (Owasp 2013 A1, A2, A5, 2017 A1, A7) envers les systèmes plutôt que d'en gérer l'accès. Les « 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 et fonctionnent de manière totalement indépendante. La plupart engendre des coûts supplémentaires en matériel ou logiciel et même qu'une équipe spécialisée pour leur mise en place et leur gestion.

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

[17] http://www.modsecurity.org
[18] http://www.squid-cache.org
[19] https://varnish-cache.org

Objectifs du projet

Développer et diffuser des outils pour permettre une meilleure gestion de l'autorisation et l'authentification dans les projets de développement Web.

La base de ces outils sera la réutilisation des pratiques dans les domaines traditionnels (non-web), l'abstraction de la couche 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.

Le projet proposé est de développer un module Apache incorporant plusieurs fonctionnalités des modules existant (autorisation, authentification, journalisation) en utilisation 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 meilleures et les plus répandues pratiques 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.

Méthodologie

Le projet a été réalisées selon 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
4. Développement et documentation du système (pistes de solutions)
5. Réalisation du module et outils
6. Application à un environnement de production (validation)
7. Statistiques de performance et diagnostique (validation)
8. Livraison publique

L'environnement de développement utilisé est basé sur « LAMP » soit Linux (Ubuntu 16.04 LTS Serveur), Apache (Version 2.4.xx) avec le module mod_php (PHP Version 7.xx) et Mysql (Mysql Server version 5.7).

Le développement du module a été 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 » basée sur une distribution standard de Linux incluant Apache, Mysql et PHP.

Échéancier

Le programme de Maîtrise en ingénierie (3708) comprend un volet mémoire (6RECHER) de 27 crédits. (En comparaison avec les projets de synthèse en ingénierie I et II pour lequel on estime la charge de travail à un total de 225h pour 5 crédits, cela représente donc environ 1215h de travail normalement réparti sur 6 sessions).

L'ampleur du projet a donc été aligné avec cette charge de travail. Cependant pour conserver une marge de manœuvre, la planification a été effectuée sur cinq sessions plutôt que six. Voici la répartition approximative pour les principaux éléments.

Tableau 1 - Estimation préliminaire

Éléments	**Nb Heures**	**Session de travail**
État de l'art	300h	2017-2 / 2017-3
Élaboration des pistes de solutions	300h	2017-3 / 2018-1 / 2018-2
Développement	200h	2018-2
Validation	150h	2018-2
Documentation (guide d'utilisation)	150h	2018-2 / 2018-3
Rédaction et présentation	200h	2018-2 / 2018-3
Total	**1200h**	

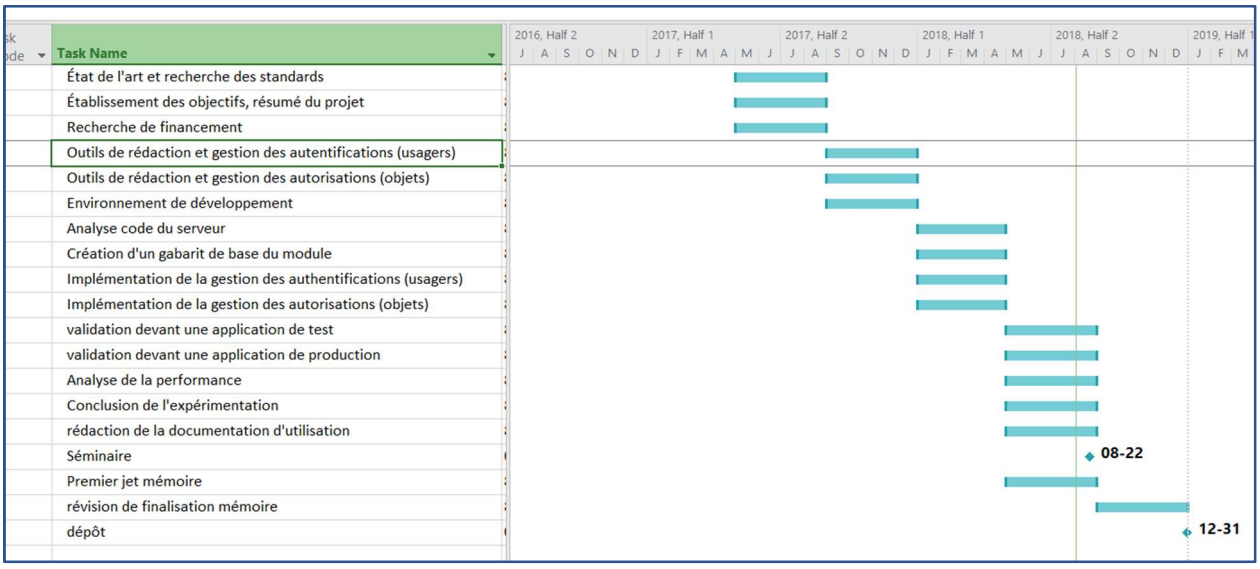

Figure 1 - Échéancier préliminaire

Financement

La réalisation de ce projet a été rendu possible par un apport financier de l'UQAC par l'entremise des budgets de perfectionnement long des chargé(e)s de cours et d'ADN Informatique entreprise privée intéressée à la commercialisation du support auprès d'entreprises pour l'implémentation du produit.

État de l'art

Le premier point à valider est la prédominance de certains systèmes sur le marché pour le développement 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.

Par la suite, la revue des standards, l'analyse des technologies existantes et le ciblage des éléments pertinent au projet a été effectué. Pour ce faire, une revue des projets Open Source sur le sujet de même que certains code source a également été réalisée.

En complément d'information, les pages de manuels électroniques de Unix (Unix ManPage) ont été utilisés 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.

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, Logiciel bancaires (bnc, tangerine, desjardins), Paiement ApplePay, ainsi que diverses applications selon les plateformes et système d'exploitation.

Note : *Plusieurs éléments du texte font référence à des informations dont la source première est un site Internet. Afin de simplifier l'accès à l'information complète, les liens vers ces sites ont été placés en note de bas de page. L'ensemble des adresses des sites est également présente dans les références à la fin du document*.

Systèmes utilisés sur le marché

Comme l'on peut le constater sur le rapport statistique de BuiltWith[20], 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[21] corroborent ces informations attribuant notamment 46% du marché à Apache suivi de Nginx avec 39% en août 2018.

L'on peut donc envisager que le développement d'un module et d'un guide pour la sécurisation des sites web utilisant Apache comme serveur aurait donc un marché potentiel intéressant.

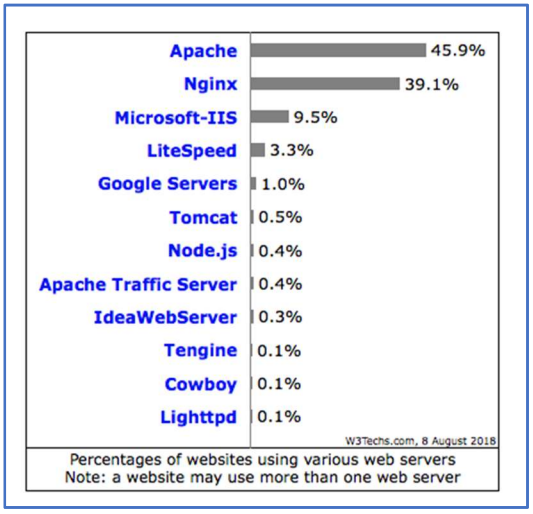

Figure 3 - w3techs - Server statistics

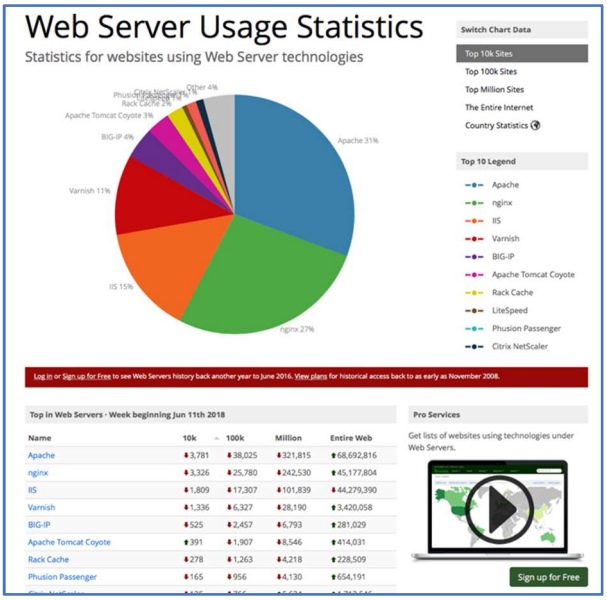

Figure 2 - BuiltWith - Server statistics

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

De même, le langage de programmation le plus répandu est PHP avec 54%, suivi par les technologies ASP.NET avec 37%. Les données de w3techs ont 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 serveur, mod_php particulièrement sous Apache, confirme probablement l'exactitude des statistiques de serveurs.

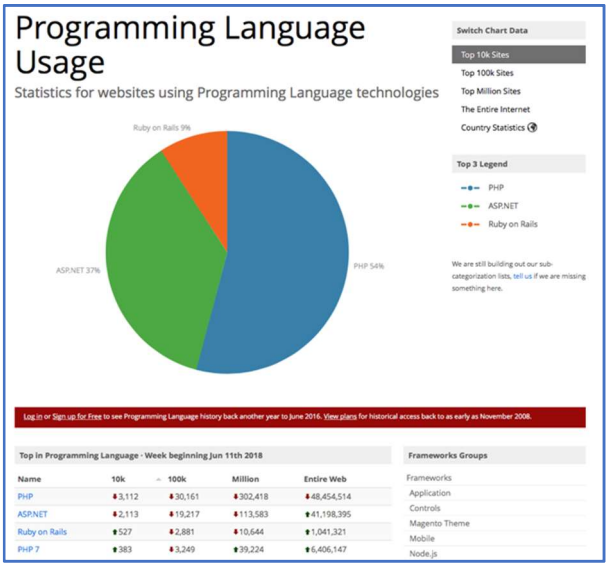

Figure 4 - BuiltWith - Language statistics

[20] https://trends.builtwith.com

[21] https://w3techs.com/technologies/overview/web_server/all

Toujours dans la même lignée, si l'on regarde l'utilisation d'application web disponibles pour la mise en ligne de sites, le joueur majeur est Wordpress avec 36% du marché. Ce dernier étant en langage PHP et généralement utilisé sous Apache confirme à nouveau la tendance.

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.

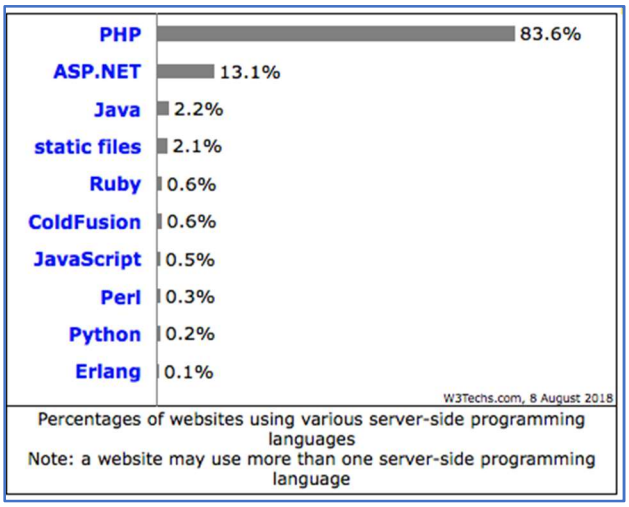

Figure 5 - w3techs - Language statistics

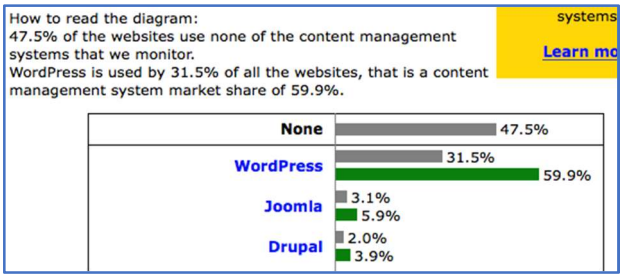

Figure 6 - w3techs - CMS statistics

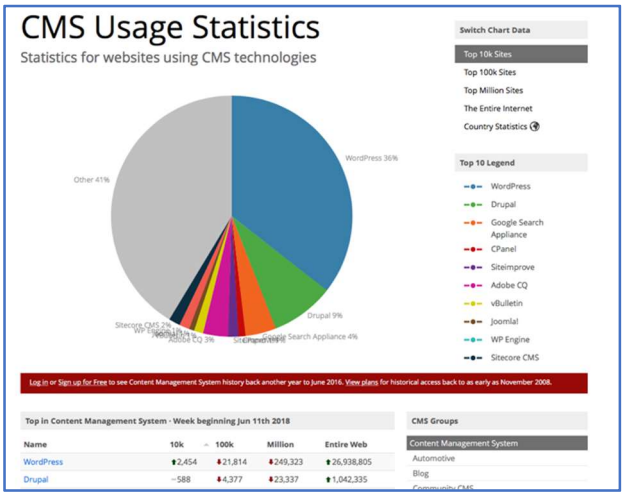

Figure 7 - builtwith - CMS statistics

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 dernière grande initiative mise en place est la GDPR (General Data Protection Regulation) Européenne[22] qui tente de responsabiliser les systèmes sur lesquels transigent ou sont traités des informations personnelles.

Cependant l'adoption des normes est souvent laissée à la bonne volonté des parties. Dans quelque 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'informations n'appartient à aucun groupe de professionnel. L'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 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[23] : 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[24] : Gestion me manière responsable et sécuritaire des information (financières, propriété intellectuelles, informations personnelles des employés ou informations sensibles fournies par des tiers) en possession de l'entreprise.

ITIL (Information Technology Infrastructure Library) Security management[25] : 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 relié 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)[26] : qui ont plusieurs publications sur la sécurité informatique.

La série de publication « NIST Special Publication 800 / 1800 series » qui traite de sécurité informatique et cyber sécurité.

[22] https://www.eugdpr.org
[23] https://www.pcisecuritystandards.org/
[24] https://www.iso.org/isoiec-27001-information-security.html
[25] https://www.axelos.com/best-practice-solutions/itil
[26] https://csrc.nist.gov/publications

IETF (Internet Engineering Task Force)[27] : Organisme pour le développement des standards de l'Internet dans un processus ouvert. Il est composé de nombreux groupes de travail sur différents dont l'IAB (Internet Architecture Board) traitant de l'architecture de l'Internet.

IISP (Institute of Information Security Professionals)[28] : Organisation indépendante à but non lucrative pour l'avancement du professionalisme dans le domaine de la sécurité de l'information.

CERT (Computer Emergency Response Team)[29] : Division de l'institut d'ingénierie logiciel de l'université Carnegie Mellon. Ils maintiennent notamment une base de données des vulnérabilités des systèmes informatiques[30]. Ils ont également développé la méthode OCTAVE (Operationally Critical Threat Asset and Vulnerabilite Evaluation) pour le classement des actifs et la détermination des risques associés.

De même, chaque institution de grande envergure, propose généralement des guides d'opérations, des règlements ou autre documents traitants des responsabilités relativement à l'obtention, au traitement et à l'entreposage de données sensibles. Par exemple, l'on retrouve ces directives pour le gouvernement du Québec sur le site su secrétariat du Conseil du trésor à l'adresse : https://www.tresor.gouv.qc.ca/ressources-informationnelles/securite-de-linformation/directive-sur-la-securite-de-linformation-gouvernementale/

Toutes ces normes présentent des éléments essentiels à prendre en compte lors de transaction, entreposage ou manipulations de données sensibles. 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, il a été remarqué 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 proposer. Sauf 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.

[27] https://www.ietf.org/
[28] https://www.iisp.org/
[29] https://www.sei.cmu.edu/about/divisions/cert/index.cfm
[30] https://www.kb.cert.org/vuls

Les systèmes Web

L'environnement

Le fonctionnement du web est une interaction entre un client (navigateur) et un serveur HTTP. 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. Ce sauf dans de rare cas où des exceptions de sécurité doivent être acceptés par l'utilisateur pour l'utilisation des ressources locales (par exemple par des scripts Java, JavaScript ou ActiveX contenu 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.

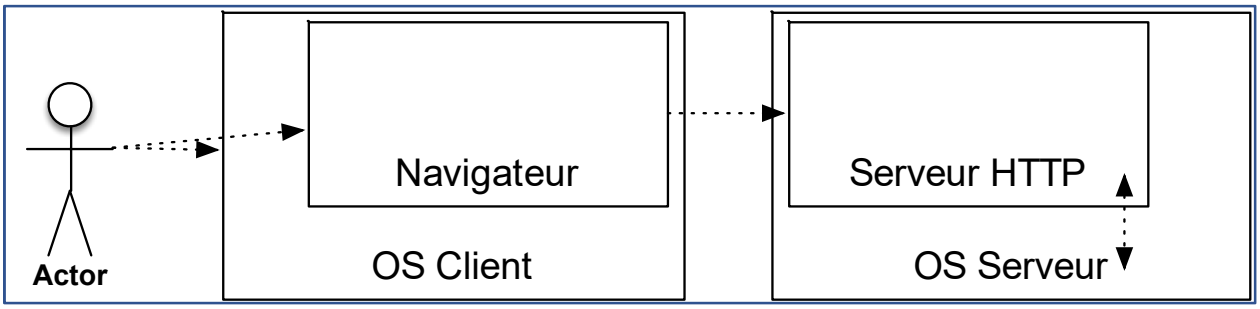

Figure 8 - Environnement web client-serveur

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, c'est à dire que par défaut, il n'y a pas de session de travail. L'utilisation par défaut du protocole HTTP n'utilise pas non plus par défaut de connexions persistantes (« keep-alive ») pour la réalisation de plusieurs transferts d'objets[31]. Tous les échanges sont initiés par le client. Cela tend à changer avec l'adoption de HTTP/2 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 fait par le logiciel client et nécessite plusieurs connexion ou échanges avec le serveur afin d'obtenir tous les éléments nécessaires.

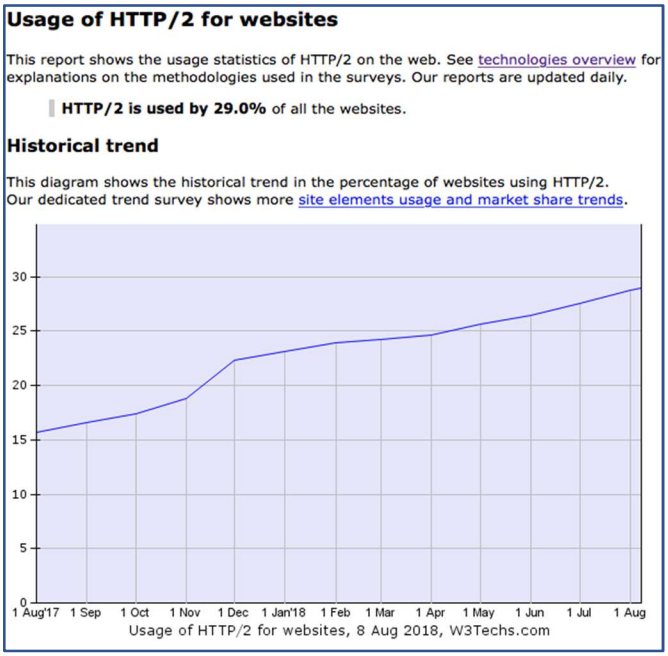

Figure 9 – w3techs - Adoption de http/2

[31] Plus de details : https://www.oreilly.com/library/view/http-the-definitive/1565925092/ch04s05.html

Le protocole

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[32]. Ces spécifications sont en constante révision sous la supervision du IETF HTTP Working Group[33]. 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. Les plus couramment utilisés sont la transmission par le serveur d'une réponse « 401 » et la transmission par le client d'un entête d'autorisation « Authorization ». Plusieurs autres éléments tel que des entêtes supplémentaires, des statuts de réponse, l'utilisation de cookies et la transmission de paramètres peuvent également servir à implémenter une couche d'autorisation et authentification.

Il faut également mentionner que le protocole HTTP que ce soit 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. Cette information est importante car elle influence la sécurité lors de l'utilisation de « basic authentication » du protocole. Déjà en 2015[34], 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.

Méthodes d'authentifications

L'authentification de l'utilisateur peut se faire de plusieurs manières. Le protocole HTTP inclus des fonctionnalités de supportées par la plupart des navigateurs basés sur la transmission d'un code de statut 401 et de l'entête WWW-Authenticate[35].

Les mécanismes les plus utilisés sont: Basic, Digest et NTLM.

D'autres mécanismes plus complets et complexes peuvent également être utilisés tel que:

Utilisation de certificats, Utilisation d'un jeton (« token ») (en cookies ou en paramètre) conjointement avec une authentification directe sur le serveur ou délégation à un tier (SSO ou Single Sign-on[36]).

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. Dû aux restrictions de l'environnement du navigateur, l'identité de l'utilisateur sur le système ne sont généralement pas accessibles pour une transmission directe sauf dans des cas comme l'utilisation de NTLM/Kerberos sur des postes Windows avec des sessions ouvertes sur un domaine.

[32] https://ietf.org/standards/rfcs/
[33] https://httpwg.org
[34] https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/
[35] https://www.httpwatch.com/httpgallery/authentication/
[36] https://en.wikipedia.org/wiki/Single_sign-on

Les serveurs HTTP

Le serveur Apache HTTP[37] de « Apache Foundation[38] » est un logiciel Open Source. 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 est maintenant un Unix basé sur BSD). Le livre « The Apache Modules Book – Application Development with Apache » 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 trois illustrations ci-dessous extraites du livre cité ci-dessus illustrent ces informations.

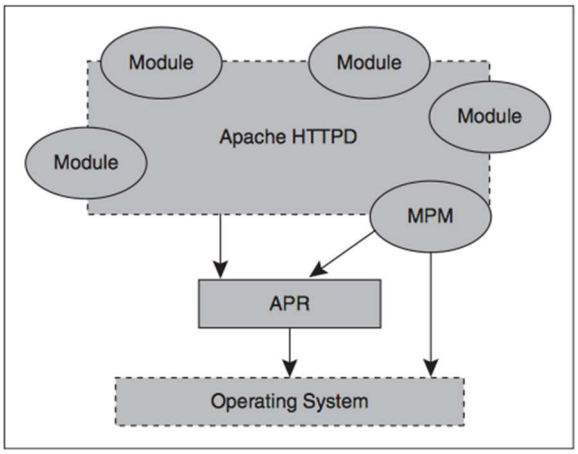

Figure 10 - Apache architecture - P.22

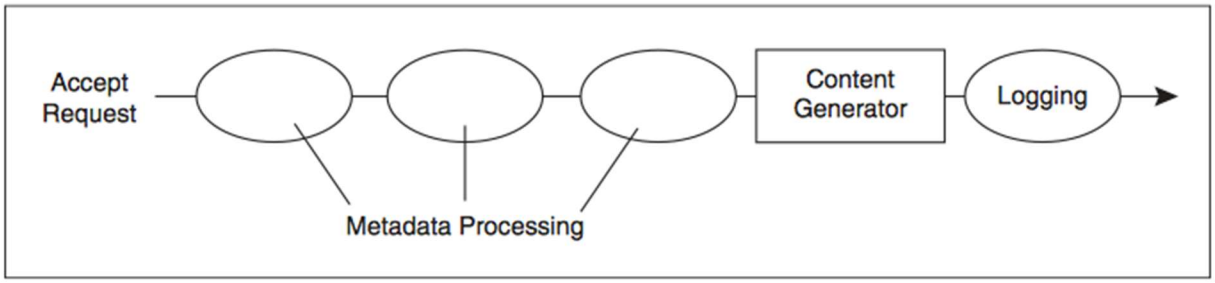

Figure 11 - Request processing – P.44

[37] http://httpd.apache.org
[38] http://apache.org

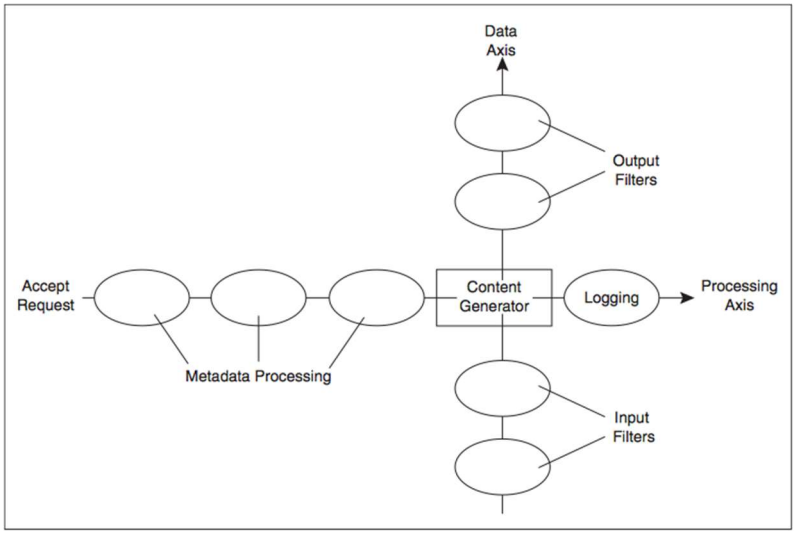

Figure 12 - Apache 2 New Data Axis – P.47

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 (Output Filters). De plus, plusieurs modules sont déjà disponibles et pourraient servir de base au projet[39].

Les systèmes d'exploitation

Selon w3Techs[40] (Web Technology Surveys) le système d'exploitation le plus utilisé pour opérer des sites web est « Unix ».

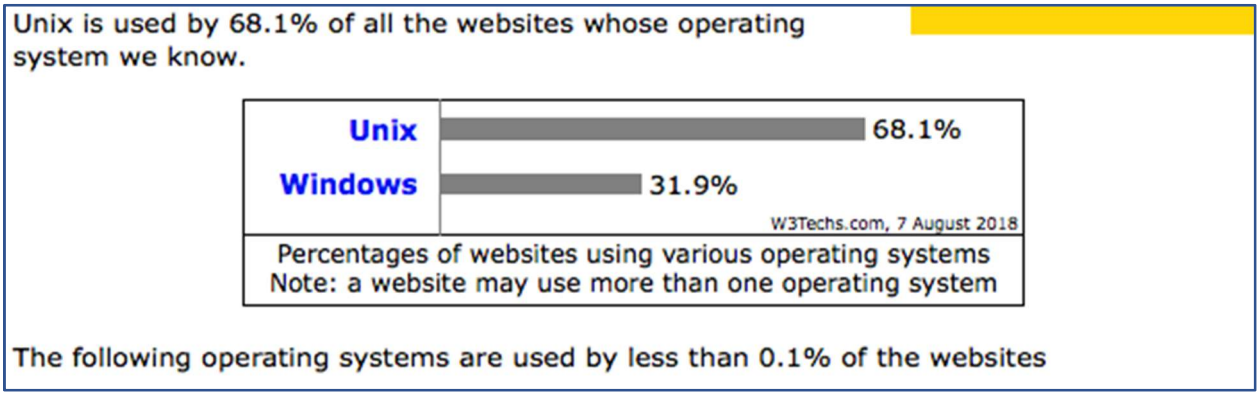

Figure 13 - w3techs - Operating Systems statistics

Il serait donc pertinent de s'attarder à ce système d'exploitation en premier.

[39] Liste des modules apache https://en.wikipedia.org/wiki/List_of_Apache_modules
[40] https://w3techs.com/technologies/overview/operating_system/all

Gestion des authentifications (usagers)

L'identification d'un utilisateur peut être fait de diverses manières. La plus conventionnelle et répandue est l'utilisation d'un identifiant conjointement avec un mot de passe. Des systèmes de clé public/privé et de certificats sont également répandus. De plus en plus de nouvelles technologies émergent tel que l'utilisation de données biométriques mais ne sont pas applicables facilement pour le moment dans le contexte du web.

Comparaison des systèmes

Système d'exploitation (Unix)

L'ouverture d'une session de travail dans l'environnement Unix, se fait traditionnellement à partir d'une liste d'utilisateurs et de mots de passe contenu 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, NIS+, LDAP ou autre. Dans ce cas, les logiciels de base utilisés pour l'authentification (login par exemple) détectent dans la configuration du système la stratégie à utiliser pour obtenir les informations nécessaires.

Ce mécanisme permettant d'ajustement automatique de plusieurs logiciels au bon système de permission peut être fait via le système PAM (Pluggable Autentification Module)[41].

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

Chaque ligne comprend :

- L'identifiant texte de l'utilisateur (login name)
- Un mot de passe optionnel encrypté
- L'identifiant numérique de l'utilisateur (UID)
- L'identifiant numérique du groupe de base de l'utilisateur (GID)
- 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 encryptés

- L'identifiant texte de l'utilisateur (doit exister dans /etc/passwd)
- Un mot de passe encrypté 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 changer le mot de passe)
- L'âge maximal du mot de passe (nb de jour avant la nécessité de changer le mot de passe)
- Le nombre de jour avant l'âge maximal durant lequel l'utilisateur reçoit un avertissement
- Le nombre de jour 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

[41] http://www.linux-pam.org/

- 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 tel que : useradd, usermod, passwd, groupadd, groupmod. Des fonctions et structures de données sont également disponibles pour leur manipulation en langage C.

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 tel que l'Active Directory pourraient peut-être apporter des éléments intéressants.

Applications natives

Les application 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.

Applications mobiles

Les applications mobiles (iOS, Android, …) se comparent aux applications natives de bureau. 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 digitales ou faciale. En supplément, certaines applications demandent une identification supplémentaire gérée localement ou sur un serveur distant (identification sur un service web, un service nuagique ou autre).

Applications web

Outres 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 local ou en réseau. D'autres délèguent même ces fonctionnalités en partie à des tiers (utilisation de SSO par exemple).

Gestion des autorisations (objets)

Définition des ressources

Avant de définir la méthode pour accorder des autorisations (ou privilèges d'accès) il faut d'abord déterminer les ressources (« assets »). Le système se voulant générique, seul leur identification sera présentée ici et non la valeur qui leur est attribué car celle-ci est propre à chaque système.

La composition des ressources web associés à des URL était à prime abord une relation 1 :1, dans la version 0.9 de HTTP qui n'avait que la méthode GET et retournait toujours un fichier HTML, avec des fichiers présents sur le système. Déjà, à ce point, par la nature des documents transmis qui sont généralement en HTML (hyper text markup language), chacun de ces documents nécessite une étape de composition par le client pour être affiché. Cette étape comprend la récupération d'autres éléments (images par exemple) associés à d'autres URL qui doivent également être protégés.

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, généralement à partir d'une base de données. Ce contenu étant généralement du HTML et incluant la nécessité d'accéder à d'autres ressources statiques ou également dynamique.

Pour complexifier d'avantage, 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)[42] pour la modification dynamique de ce document de base rend donc inutilisable les permissions de base accordées au document.

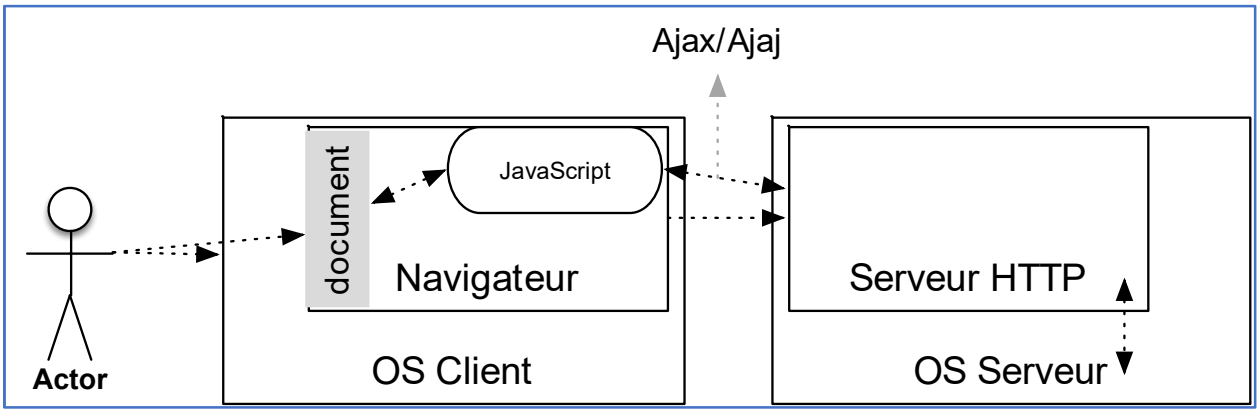

Figure 14 - Ajax/Ajaj requêtes

Cependant un point demeure, chaque ressource accédée (page de base, objets inclus tel que les images, scripts inclus pour exécution, objets récupérés via Ajax) est récupérée sur le serveur et peut être soumise indépendamment à une gestion des privilèges.

Le suivi de la relation entre la donnée et son emplacement dans une ou plusieurs pages et la correspondance avec l'URL est pratiquement impossible. Une méthodologie pour y arriver sera proposé

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

dans ce projet. (Inclusion d'attribut sur les objets du HTML représentant les UID, GID, PERMS provenant de la bd).

L'association des droits avec les données est donc de plus en plus complexe.

L'utilisation de REST tel que développé par Roy Fielding dans son mémoire « Architectural Styles and the Design of Network-based Software Architectures »[43] pourrait être un point de départ pour simplifier l'identification et l'accès aux ressources. Cependant nous ne pouvons pas présumer que les systèmes applicatifs développeront des services REST pour délivrer les objets.

Tableau 2 - Méthodes HTTP et actions

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

L'utilisation de REST attribut 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, cela conjointement avec les URLs qui identifient les ressources de manière précise. Cette attribution unique des URLs combiné avec des définitions des actions possibles, nous donne une piste intéressante sur les privilèges qui doivent être accordés aux utilisateurs pour interagir avec les différents objets.

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 s'attarder au mécanisme de permission du système de fichier.

Le système de base de permission de Unix, date des premiers jours de la création de ce système d'exploitation. Les permissions permettent le contrôle sur la possibilité de lire, modifier, naviguer et exécuter le contenu du système de fichier.

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,

[43] https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

l'écriture et l'exécution (R,W,X). L'on retrouve donc trois triples d'information RWXRWXRWX représentant respectivement les permissions du propriétaire, du groupe et du public.

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

Tableau 3 - Permissions de base système de fichier Unix

Permissions textuelles	**valeur 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.

Voici quelques exemples de permissions possibles

Tableau 4 - Exemple d'attribution de permission de fichier 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 tel que : ls, chown, chmod. Des fonctions et structures de données sont également disponibles pour leur manipulation en langage C.

Ref : https://en.wikipedia.org/wiki/File_system_permissions

Système d'exploitation (Windows)

La protection des ressources dans les systèmes Windows comme pour les Unix sont 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 sous Unix. Il est basé sur une liste de contrôle (ACL) développée plus récemment que le système de base de permission de 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 de Unix.

Applications web

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 faisait partie de l'application qui gère de manière autonome, c'est-à-dire sans reposer sur le serveur HTTP, l'accès aux ressources. L'application gère donc, à l'interne, une liste de contrôle (ACL) pour les objets/URL qu'il délivre.

Gestion de la sécurité serveur web (Apache)

De base, la définition des accès aux fichiers au travers d'un serveur web, particulièrement Apache HTTP est très rudimentaire.

Le premier niveau est la gestion des fichiers par le système d'exploitation. En effet, le serveur est configuré pour délivrer du contenu à partir d'un répertoire racine () à l'intérieur duquel les règles de sécurité du système d'exploitation sont appliquées. Seuls les fichiers pour lequel l'utilisateur exécutant le serveur (règle générale sous Apache HTTP, l'utilisateur www-data) peuvent être délivrés aux clients.

Le second niveau est défini dans la configuration du serveur. En effet, Apache permet de définir des droits d'accès particuliers par une définition en arborescence des répertoires et fichiers qui peuvent être délivrés. Cette configuration peut contenir des définitions d'utilisateur, de groupe ainsi que des permissions spéciales par exemple pour l'exécutions d'un script et la délivrance du résultat plutôt que la transmission directe du contenu du fichier ou script.

Les éléments qui peuvent être utilisés pour construire cette configuration proviennent du cœur du serveur ainsi que des modules supplémentaires qui sont disponibles et peuvent y être incorporé tel que :

Tableau 5 - Modules de sécurité Apache 2.4

Modules de sécurité disponibles sous Apache 2.4 directement de l'Apache Fundation	
mod_access_compat	Group authorizations based on host (name or IP address)
mod_auth_basic	Basic HTTP authentication
mod_auth_digest	User authentication using MD5 Digest Authentication
mod_auth_form	Form authentication
mod_authn_anon	Allows "anonymous" user access to authenticated areas
mod_authn_core	Core Authentication
mod_authn_dbd	User authentication using an SQL database
mod_authn_dbm	User authentication using DBM files
mod_authn_file	User authentication using text files
mod_authn_socache	Manages a cache of authentication credentials to relieve the load on backends
mod_authnz_fcgi	Allows a FastCGI authorizer application to handle Apache httpd authentication and authorization
mod_authnz_ldap	Allows an LDAP directory to be used to store the database for HTTP Basic authentication.
mod_authz_core	Core Authorization
mod_authz_dbd	Group Authorization and Login using SQL

mod_authz_dbm	Group authorization using DBM files
mod_authz_groupfile	Group authorization using plaintext files
mod_authz_host	Group authorizations based on host (name or IP address)
mod_authz_owner	Authorization based on file ownership
mod_authz_user	User Authorization

Tableau 6 - Modules de sécurité non-officiels Apache 2.4

Modules de sécurité disponibles sous Apache 2.4 d'autres sources	
mod_auth_pam	http://tldp.org/HOWTO/User-Authentication-HOWTO/x316.html
mod_authnz_pam	https://www.adelton.com/apache/mod_authnz_pam/
mod_authnz_external	http://icephoenix.us/linuxunix/apache-and-http-authentication-with-pam/ (Pwauth)
mod_auth_shadow	http://mod-auth-shadow.sourceforge.net
mod_AuthMemCookie	https://zenprojects.github.io/Apache-Authmemcookie-Module/

Une autre extension intéressante, qui n'est pas directement reliée à la sécurité mais en implémente une, est l'extension pour le protocole WebDav[44] [45] [46] définie dans le RFC 4918. Son but est de permettre la collaboration pour la création de contenu (fichiers) au travers du Web, tel un système de fichiers en réseau (nfs ou autre).

D'autres serveurs web utilisent des stratégies différentes, reliées aux systèmes d'exploitation pour lequel ils ont été principalement conçu, comme par exemple Internet Information Server (IIS) de Microsoft, qui utilise les particularités de Windows tel que le système d'ACL. Il en va de même pour d'autres serveurs particuliers tel qu'Apache Tomcat[47] qui repose sur les technologies disponibles en langage Java.

[44] https://httpd.apache.org/docs/2.4/mod/mod_dav.html
[45] https://en.wikipedia.org/wiki/WebDAV
[46] https://tools.ietf.org/html/rfc4918
[47] https://tomcat.apache.org

Développement / expérimentation

Choix de l'environnement de travail

Basé sur les données recueillies, il a été établi que pour une plus grande portée du projet, les technologies LAMP devraient être ciblées.

L'environnement de production visé sera Linux Ubuntu Serveur, Apache HTTP 2.4 et MySql. Ce dans le but que le projet soit utilisable par les principaux sites en place notamment ceux réalisés avec WordPress.

L'environnement de développement devra inclure les versions sources des produits mais le résultat (module produit) devra pouvoir s'installer en production avec un minimum de dépendances particulières. (Autres que celle du OS, Apache et Mysql).

Mise en place de l'environnement de travail

Des machines virtuelles ont donc été créées pour le développement et les tests en production. Les spécifications sont les suivantes :

La **machine virtuelle de développement** a été créée avec le logiciel Parallel 13[48] sous MacOS 10.13 HighSierra sur un iMac Intel Core i5 3.2Ghz, 24G Mémoire DDR3, disque SSD.

Deux processeurs virtuels, 1G de mémoire vive, 64G de disque ont été assigné à cette machine.

Le système d'exploitation Linux Ubuntu 16.04 LTS Serveur y a été installé.

La **machine virtuelle de test/production** sera créée avec une configuration similaire. Cependant les outils de développement n'y seront pas installés afin d'être le plus fidèle possible à un environnement de production standard.

Un serveur GIT a été mis en place pour la gestion des codes sources (Logiciel Kalithea[49], installé sur un serveur d'ADN Informatique). À la fin du projet, les codes sources seront déplacés sur un dépôt public tel que GitHub[50] ou BitBucket[51].

L'environnement développement utilisé est en ligne de commande directement sur la machine virtuelle pour la compilation et l'installation et la programmation a été réalisé dans l'environnement Visual Studio Code[52], projet ouvert de Microsoft.

[48] https://www.parallels.com
[49] https://kallithea-scm.org
[50] https://github.com
[51] https://bitbucket.org/product
[52] https://code.visualstudio.com

Phases de développement et d'implémentation

Le développement sera effectué en phases. Chacune de ces phases comprendra un volet d'analyse pour le développement d'une solution à un volet de la problématique suivi d'une implémentation sous forme de prototype pour en valider le fonctionnement. La conclusion de chaque phase apportera des éléments pour l'élaboration de la phase suivante, sous forme de correctifs pour la phase terminée et d'ajouts pour la phase en cours afin de se rapprocher d'une solution plus complète.

La phase 0 ou préparatoire a été l'analyse d'un système « vierge » afin de déterminer les paramètres par défaut du système et de valider les outils de sécurité déjà en place et disponibles.

Les phases I et II ont été mise en place en se basant sur la relation 1 :1 entre les URL et les fichiers sur le système d'exploitation. Aucun contenu dynamique n'a été utilisé durant cette phase. Le but étant de garder le focus sur l'authentification de l'utilisateur et non sur l'association des privilèges aux objets.

La phase III intègre une première association d'autorisation entre les objets et l'utilisateur au moyen des permissions associés au niveau du système d'exploitation.

La phase IV tente de généraliser ce système pour son utilisation avec des ressources crées dynamiquement. L'incorporation de code PHP pour la génération de contenu est utilisée.

La phase V mettra en place des outils de base pour l'administration et la configuration du système.

La phase VI explorera l'ajout d'un traitement après la génération de contenu.

S'adjoint au développement principal, la rédaction de guide pour la mise en place et l'utilisation du système.

Phase 0 – Préparation du système et observations

État du système avant manipulation

Cette première phase, consiste à constater la configuration de base du système d'exploitation pour la gestion des authentifications et autorisations.

Immédiatement après l'installation, nous notons donc les permissions associées aux différents fichiers et les composantes disponibles.

Les fichiers de base du système identifié précédemment ont les attributs de sécurité suivants :

Tableau 7 - Fichiers unix de base pour l'authentification

Fichier	Propriétaire	Groupe	Permissions
/etc/passwd	root	root	-rw-r--r--
/etc/shadow	root	shadow	-rw-r-----
/etc/group	root	root	-rw-r--r--

D'autres observations ont été effectuées et seront présentées aux étapes suivantes.

Configuration par défaut du serveur web Apache HTTP

Après ces observations, l'installation d'un environnement web a été réalisée. L'installation par défaut du serveur Apache HTTP comprend différents modules et configurations de base.

Le répertoire contenant les fichiers de configuration du serveur Apache HTTP sous Ubuntu se retrouvent dans le répertoire /etc/apache2. L'ensemble des fichiers est réparti dans une série de répertoires ayant les suffixes -available et -enabled de même que quatre fichiers à la racine. Seuls les fichiers à la racine et ceux contenus dans les répertoires suffixés -enabled sont actifs dès l'installation.

La liste de ces fichiers actifs par défaut se retrouve en annexe II.

Le fichier 000-default dans le sous-répertoire sites-enabled définit le traitement par défaut du serveur indépendamment du nom reçu dans les requêtes. Il assigne le répertoire /var/www/html comme répertoire par défaut du système.

Après analyse de ces différents fichiers, l'on sait donc que l'utilisateur exécutant le serveur est par défaut www-data et le groupe est www-data. On peut constater que la configuration de base permet la livraison de tous les fichiers contenus dans le répertoire /var/www/html du système de fichier sans aucune autre restriction que les permissions du système de fichier qui doivent allouer le droit en accès et en lecture à l'utilisateur ou au groupe www-data à son contenu.

Installation des bibliothèques et outils de développement

Pour être en mesure de développer un module complémentaire pour le serveur Apache HTTP, certaines bibliothèques de composants doivent être installées de même que les outils de développement en langage C. L'installation minimale se fait par l'ajout du paquet « apache2-dev ».

Parmi les composants installés, l'on retrouve les bibliothèques du « Apache Portable Runtime[53] » et des outils pour la compilation, l'assemblage et l'installation de modules. Parmi ceux-ci un outil spécifique « apxs[54] » qui permet de créer des gabarits de développement pour les modules, en fait la compilation, l'assemblage et l'installation dans l'environnement Apache HTTP installé.

[53] https://apr.apache.org
[54] https://httpd.apache.org/docs/2.4/programs/apxs.html

Phase I – Gestion de l'authentification de base

Objectifs et suppositions

La première étape de développement est la mise en œuvre d'une méthode permettant de déterminer qui tente d'accéder à une ressource peu importe sa nature. Pour ce faire, on doit identifier l'utilisateur du navigateur. Le protocole HTTP nous permet d'effectuer cette tâche de plusieurs manières.

La plus simple est l'utilisation des entêtes WWW-Authenticate et du statut 401. Lors de l'utilisation du Basic Autentication, le navigateur doit à chaque requête transmettre un entête contenant le nom d'utilisateur et le mot de passe au serveur. Ce dernier peut alors valider l'information. Si cette l'information est absente ou erronée, il retourne le statut 401 demandant une nouvelle tentative avec des informations correctes. Le navigateur fait alors apparaître une boite de dialogue à l'utilisateur lui demandant de saisir l'information demandée, puis une nouvelle requête est effectuée sur le serveur avec les informations saisies.

La validation de l'informations reçue (identifiant/mot de passe) est validée avec une liste à la disposition du serveur. Le module de base d'Apache HTTP effectuant cette opération utilise un fichier texte contenant la liste des identifiants et mot de passe. Nous allons donc ici reproduire le même comportement. Cependant nous allons tenter d'utiliser la liste d'identifiants et de mots de passe du système d'exploitation pour faire la validation. Cette première itération du module devrait ressembler au projet existant « mod_auth_shadow[55] » à l'exception de ne pas utiliser de programme distinct. Nous accéderons aux informations du système directement dans les fichiers /etc/passwd, /etc/group et /etc/shadow.

Nous assumerons également à ce stade que toutes les ressources correspondent à un objet sur le système de fichier, soit une relation 1 :1 et qu'une identification valide de l'utilisateur entraine automatiquement un droit d'accès aux ressources. Pour ce faire, toutes les ressources doivent donc être se trouver dans le répertoire du serveur /var/www/html et être accessible par ce dernier.

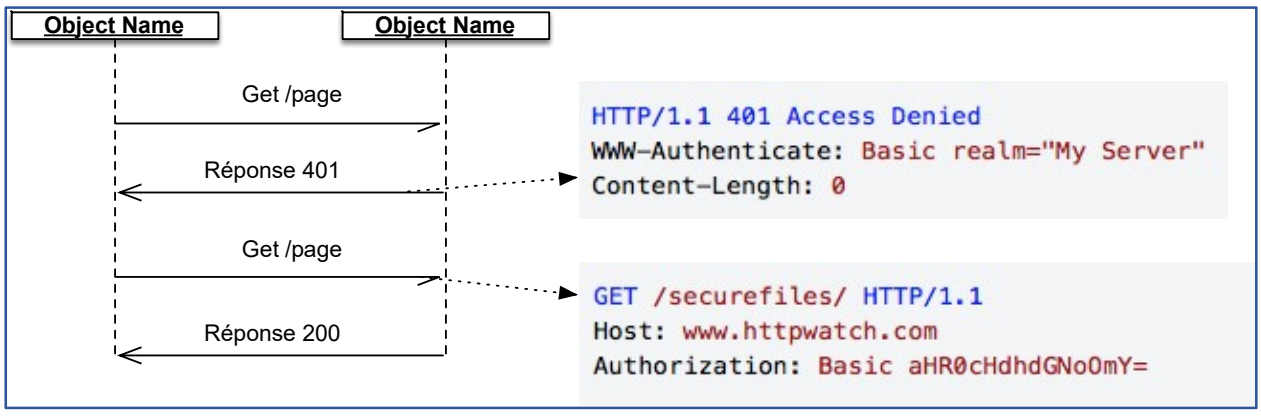

Figure 15 - Http Basic Autentication

[55] http://mod-auth-shadow.sourceforge.net

Réalisation

La familiarisation avec l'environnement de développement d'Apache HTTP, Apache apr a été relativement aisée. La compilation et l'installation du gabarit s'est déroulée sans aucune problématique. L'incorporation du code permettant d'accéder de manière conforme aux fichiers /etc/passwd, /etc/group et /etc/shadow incluant les algorithmes d'encryptions nécessaire étaient bien documentés[56] dans la littérature et n'a pas apporté de défi supplémentaire.

Une problématique a été rencontrée au niveau des droits d'accès au fichier /etc/shadow. Les instructions fournies avec le module « mod_auth_shadow » nous informaient déjà que les permissions d'accès de base à ces fichiers ne permettaient pas aux serveurs Apache HTTP exécuté avec un compte non privilégié d'y avoir accès.

Différentes alternatives ont été soulevées pour tenter de solutionner cette problématique. Chacune apportant à la fois une solution mais aussi divers inconvénients.

La première solution consistait à donner accès à l'utilisateur www-data ou au groupe www-data aux fichiers du système par la modification de la propriété de ces fichiers. Cependant, modifier la propriété des fichiers pour un autre utilisateur ou groupe avait de fortes chances d'altérer le fonctionnement normal du système en plus de possiblement créer des brèches de sécurité.

La seconde solution était d'ajouter des droits à l'utilisateur www-data en le déclarant membre du groupe shadow. En inspectant les droits de ce groupe sur l'ensemble des fichiers du système cette option semblait moins risquée que la première.

La troisième solution consistait à exécuter le serveur avec les privilèges du super utilisateur (root). En fait, au démarrage le serveur Apache HTTP est exécuté avec ces privilèges afin de pouvoir accéder aux ports réseaux réservés (sous 1024) 80 et 443 pour accepter les connexions. Il semblait donc intéressant de tirer profit de ce fait.

À l'expérimentation, lors de la modification de la configuration pour utiliser le super utilisateur plutôt que www-data, il a été constaté que le code du serveur en version compilé (distribué de manière standard dans les paquets) ne permet pas d'utiliser le super utilisateur. Il a donc été décidé de recompiler le serveur en mettant hors-fonction ce mécanisme de validation. Une fois cette tâche complétée, nous avons pu valider le bon fonctionnement du module développé.

Cette solution a cependant des considérations importantes. Malgré le fait que le logiciel Apache HTTP lui-même soit relativement sécuritaire, sont exécution complète (tout processus et thread) en tant que super utilisateur peut rendre accessible tous les objets du système de fichier car il surpasse les permissions allouées. De plus, l'utilisation de modules pour exécuter du code tel que mod_php se ferait également en mode privilégié ce qui rendrait accessible l'ensemble du système aux programmes ainsi déployés et exécutés.

Conclusions

Puisque pour des raisons de sécurité le fichier /etc/shadow n'est accessible qu'au super utilisateur et aux membres du groupe shadow, la seconde solution semble la plus approprié, à défaut de trouver un

[56] http://pubs.opengroup.org/onlinepubs/9699919799/functions/contents.html

mécanisme indirect pour accéder aux informations du système. La première solution étant clairement non sécuritaire de même que la troisième.

La seconde étape sera donc de valider si l'utilisation d'un système alternatif offrirait une manière adéquate d'accéder aux informations du système de manière sécuritaire.

Phase II – Gestion de l'authentification avec PAM

Objectifs et suppositions

Suite aux conclusions de la phase I, une recherche a été effectué afin de déterminer si d'autres méthode d'authentification sur les systèmes Linux étaient disponible. Le « Plugable Authentication Module » est ressorti parmi les solutions trouvées comme une interface possible avec les fichiers du système, particulièrement /etc/shadow qui causait un problème.

L'utilisation de PAM avec ses fonctions de plus haut niveau (permettra également de supporter des systèmes utilisant d'autres mécanisme d'authentification tel que NIS ou autre plutôt que seulement les fichiers de base de Unix.

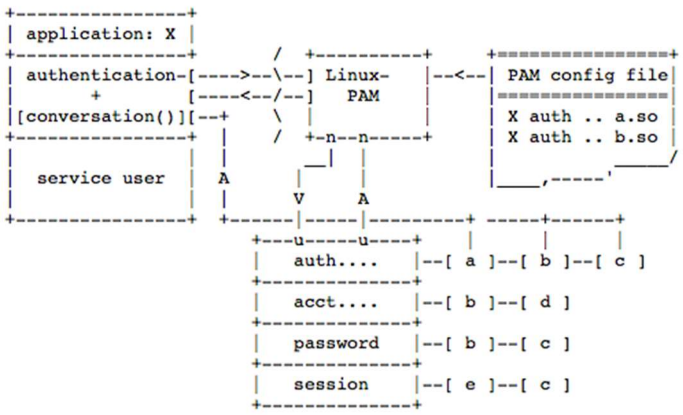

Figure 16 - Mécanisme de PAM[57]

Nous avons donc regardé l'installation standard du système PAM dans l'environnement Ubuntu 16.04. Sur le système initial, le système « PAM » était déjà en place par défaut. La configuration initiale comprenait des informations pour les systèmes/fonctions suivantes :

- Chfn
- Chpasswd
- Chsh
- Common-account
- Common-auth
- Common-password
- Common-session
- Common-session-noninteractive
- Cron
- Login
- Newusers
- Other
- Passwd
- Runuser
- Runuser-l
- Sshd
- Su
- Sudo
- System-user

Cette itération du module devrait ressembler au projet existant « mod_auth_pam[58] » à l'exception de tenter d'utiliser également le fichier /etc/shadow pour les mots de passe et /etc/group pour les groupes

[57] http://www.linux-pam.org/Linux-PAM-html/sag-overview.html
[58] http://pam.sourceforge.net/mod_auth_pam/index.html

supplémentaires. Nous ne gèrerons toujours pas à ce stade la cartographie des URLs et objets/fichiers et les autorisations afférentes.

Réalisation

Le remplacement de l'accès direct aux fichiers /etc/passwd et /etc/shadow par les fonctions de base PAM s'est fait relativement aisément.

Nous avons constaté, avec la configuration utilisée de PAM, que malgré l'utilisation de PAM, qui est en soit une amélioration à la portabilité du module, nous avons toujours la contrainte d'avoir des droits d'accès en lecture sur le fichier /etc/shadow. Pour arriver à un résultat intéressant il a dont fallu mettre d'avant la solution #2 proposé à la phase précédente soit d'inclure l'usager www-data dans le groupe shadow.

Une recherche plus précise sur le système nous a révélé que les fichiers suivants seraient affectés par cet accord de permission

- /etc/gshadow
- /etc/shadow
- /sbin/unix_chkpwd
- /sbin/pam_extrausers_chkpwd
- /usr/bin/chage
- /usr/bin/expiry

Nous avons donc statué que l'accès à ces fichiers par le module Apache ne devrait pas compromettre la sécurité du système.

Observations et conclusions

Suite à la réalisation de cette phase, nous constatons que l'utilisation de PAM ne permet pas de résoudre le problème d'accès au fichier /etc/shadow. Cependant l'apport au niveau de la portabilité et la possibilité d'utiliser d'autres système d'authentification sous Linux est tout de même un avancement.

Il serait intéressant ultérieurement de valider si une modification à la configuration de PAM ou l'utilisation d'une autre technologie permettrait d'éviter la modification des permissions tout en conservant les avantages de portabilité.

Phase III – Gestion des autorisations avec le système de fichiers

Objectifs et suppositions

Avec le résultat probant des phases I et II pour identifier l'utilisateur désirant accéder aux ressources en combinant l'« HTTP basic autentication » et l'authenfication du système d'exploitation, nous allons maintenant procéder à la validation des permissions associés aux fichiers sur le système de fichiers(toujours dans une relation 1 :1 entre les URLs et les objets sur le système de fichier).

Les deux points majeurs à considérer ici sont la relation entre l'URL et les objets/documents retournés et l'association des méthodes HTTP aux actions effectuées sur ces objets.

Nous assumons ici l'utilisation des actions du protocole HTTP en correspondance avec leur définition première et l'association d'un URL avec un document ou objets précis même si ce dernier est généré dynamiquement.

La représentation des permissions devra correspondre aux actions possibles au travers du protocole HTTP.

HTTP Verb	**CRUD Action**	**Permission système de fichier**
POST	Create	w (2)
GET	Read	r (1)
PUT	Update	r
DELETE	Delete	w
		x (4)

Figure 17 - Action HTTP vs filesystem permissions

Ces permissions seront validées tel que le permet le système pour le propriétaire de la ressource, son group et le public. Chaque URL correspondra à un fichier sur le disque. Les ressources créées dynamiquement (ne correspondant pas à un objet réel sur le disque) se verront créer un objet correspondant sur le disque pour l'attribution des permissions.

Cette étape se rapproche du comportement de WebDav mais sans l'utilisation d'un protocole différent de HTTP.

Détails techniques

Les UID, GID et groupes supplémentaires sont récupérés à partir de l'identification du client. La méthode HTTP est récupérée dans les entêtes de même que l'URL demandé. L'URL est traduite en nom de fichier

Le protocole HTTP comprend plusieurs méthodes, parmi celles-ci tel que cité précédemment, GET, POST, PUT et DELETE peuvent correspondre aux actions possibles sur les objets/documents correspondants aux URLs demandés.

Les permissions accordées aux objets en lecture, écriture et exécution sur le système de fichier unix seront donc équivalentes à ces actions.

La problématique de relation entre les URLs et le contenu délivrer dynamiquement devra aussi être abordée.

Réalisation

Les fonctions de bas niveau en C des bibliothèques Unix ont rendu relativement trivial l'implémentation de la gestion des fichiers et permission sur le système de fichier. Cependant une problématique a été rencontrée si le serveur n'a pas accès aux fichiers alors que l'utilisateur devrait y avoir accès.

De plus, lors de la génération dynamique du contenu, une multitude de fichiers vides ont été créés. Pour la représentation d'un catalogue contenant plusieurs données, la limite du nombre de fichiers pouvant coexister dans le même répertoire et la lourdeur du système de fichiers peut devenir une problématique.

Observations et conclusions

L'utilisation du système de fichiers sans l'exécution d'Apache HTTP avec les droits de super utilisateur contraint les permissions que l'on peut accorder aux objets. En effet, l'utilisateur exécutant Apache HTTP doit obligatoirement avoir accès aux objets pour en vérifier les permissions. Cela impose que www-data soit propriétaire des objets, membre du groupe auquel appartient les objets ou que le public puisse accéder aux objets. Le dernier est impossible puisqu'aucune restriction ne s'appliquerait aux fichiers, le premier est inadéquat puisque la propriété de l'objet ne pourrait être à son créateur ou propriétaire. La seule alternative serait donc l'utilisation des groupes.

Les craintes quant à la sécurité soulevés en phase I si le serveur est exécuté avec les privilèges de super-utilisateur nous font rejeter cette solution.

De plus, avec la génération dynamique de contenu, un fichier ou répertoire devrait exister pour chaque objet ou contenu généré ce qui encombrerait inutilement le système de fichier, particulièrement, par exemple, lors d'utilisation d'un service web pour accéder à tous les enregistrements d'une base de données.

Phase IV – Gestion des autorisations avec une base de données (et journalisation)

Objectifs et suppositions

À ce stade, avec, en place, le volet authentification et une prémisse du volet « autorisation », nous allons tenter d'améliorer ce dernier. En effet tel que nous l'avons déjà démontré, dans les systèmes la relation 1 :1 entre l'URL et un objet sur le système de fichiers ne représente pas la réalité. De plus, l'utilisation du système de fichier comme dictionnaire des règles d'accès impose l'attribution de permission spécialement pour permettre au serveur d'accéder à ces objets.

Parmi les améliorations que nous tenterons d'apporter nous retrouvons :

- Facilité de création et suivi de la carte des URLs
- Facilité de gestion des propriétés et permissions
- Journalisation des dernières requêtes
- Permettre d'exécuter Apache HTTP avec les droits d'un utilisateur normal et non du super-utilisateur

Pour ce faire, nous allons donc mettre en place une base de données pour gérer ces éléments qui sera accessible à la fois par le serveur, les outils pour sa gestion et le code applicatif (en PHP ou autre).

Deux possibilités ont été analysées soit l'utilisation d'une base de données en mémoire (memcache) ou d'un moteur de base de données (mysql ou autre). Le premier malgré ses performances plus élevées en temps de réponse a été rejeté dû à la volatilité des informations et l'impossibilité de les conserver pour analyse en cas de problématique. Parmi les bases de données considérées, MySQL s'est démarquée par son fort taux d'adoption par les développeurs dans le modèle « LAMP ».

Réalisation

La réalisation de cette phase n'est pas complètement terminée.

Des améliorations sont en cours pour que le modèle de base de données se rapproche le plus possible de la structure du système de fichiers, incluant non seulement les permissions mais également des informations sur le dernier accès à la ressources et d'autres informations. De plus, l'intégration d'informations sur le statut de la ressource lors de la dernière consultation (200 ok, 404 not found, …) devra être intégré pour permettre un meilleur suivi.

Observations et conclusions

–

À venir

Phase V – Outils de gestion

Objectifs et suppositions

Puisque l'administration des objets sur le système de fichier est connue des administrateurs de système et a servi de prémisse à ce développement, nous allons tenter de reproduire les outils administratifs disponibles tel que : ls, chown, chmod pour permettre la gestion des informations relatives aux permissions conservées dans la base de données.

Cela permettra un gestion facile incluse dans le projet et éliminant la nécessite de développer de tels outils à même le système administratif.

Réalisation

À venir

Observations et conclusions

À venir

Phase VI – Filtre de contenu

Objectifs et suppositions

Comme certains documents sont générés dynamiquement, le contenu peut avoir des restrictions particulières. De plus, il est difficile de prédéterminer l'association URL – contenu. Pour ce faire, il faut que le document soit réalisé avant d'y attribuer les permissions. Apache HTTP permet d'ajouter un module de filtre après la génération. En ajouter un composant à cet endroit, il sera donc possible non seulement de filtrer sur les URLs mais également sur le contenu des documents.

Cette stratégie permettra également d'avoir une granularité plus grande sur le contenu retourné. Cela permettrait en effet de « censurer » certaines parties d'un document selon les permissions associées à chacune des parties.

Réalisation

À venir

Observations et conclusions

À venir

Mise en production

Objectifs et suppositions

Lors de la mise en production devant une applications web, nous allons analyser l'impact sur la performance (temps de réponse traiter les requêtes) et la détection des objets pour lesquels les permissions sont mal définies.

Réalisation

À venir

Collecte des résultats

À venir

Analyse des résultats

À venir

Observations et conclusions

À venir

Références

Belshe, M., et al. (2015). Hypertext transfer protocol version 2 (http/2).

Fielding, R., et al. Hypertext transfer protocol–http/1.1. RFC2616, June 1999.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures, UNIVERSITY OF CALIFORNIA, IRVINE.

Kew, N. (2007). The Apache modules book: application development with Apache, Prentice Hall Professional.

Pate, S. D. (2003). UNIX filesystems: evolution, design, and implementation, John Wiley & Sons.

Reschke, J. (2015). "RFC 7617: The 'Basic'HTTP Authentication Scheme." published on September.

Sites Internet

- https://wordpress.org
- https://www.joomla.org
- https://www.drupal.org
- http://www.php.net
- https://www.ruby-lang.org/
- https://www.python.org
- http://flask.pocoo.org
- https://www.djangoproject.com
- http://httpd.apache.org
- https://www.nginx.com
- https://www.iis.net
- https://www.owasp.org/
- https://www.imperva.com
- https://f5.com/products/big-ip
- https://www.cisco.com/c/en/us/products/security/index.html
- http://www.modsecurity.org
- http://www.squid-cache.org
- https://varnish-cache.org
- https://trends.builtwith.com
- https://w3techs.com/technologies/overview/web_server/all
- https://www.eugdpr.org
- https://csrc.nist.gov/publications
- https://www.oreilly.com/library/view/http-the-definitive/1565925092/ch04s05.html
- https://ietf.org/standards/rfcs/
- https://httpwg.org
- https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/
- https://www.httpwatch.com/httpgallery/authentication/

- https://en.wikipedia.org/wiki/Single_sign-on
- http://httpd.apache.org
- http://apache.org
- https://en.wikipedia.org/wiki/List_of_Apache_modules
- https://w3techs.com/technologies/overview/operating_system/all
- http://www.linux-pam.org/
- https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- https://httpd.apache.org/docs/2.4/mod/mod_dav.html
- https://en.wikipedia.org/wiki/WebDAV
- https://tools.ietf.org/html/rfc4918
- https://tomcat.apache.org
- https://www.parallels.com
- https://kallithea-scm.org
- https://github.com
- https://bitbucket.org/product
- https://code.visualstudio.com
- https://apr.apache.org
- https://httpd.apache.org/docs/2.4/programs/apxs.html
- http://mod-auth-shadow.sourceforge.net
- http://www.linux-pam.org/Linux-PAM-html/sag-overview.html
- http://pam.sourceforge.net/mod_auth_pam/index.html

Annexe I – Articles concernant des piratages informatique

NOTE : Recherche de « piratage equifax » sur Eureka.

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

ÉCONOMIE
Les données de clients auraient été piratées
Une grosse commande pour des avions CSeries de Bombardier
La rareté de main-d'œuvre favorisera-t-elle le sursaut de la productivité tant espéré?
page b1

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 -

Presse
Presse web

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 – Fichiers de configuration par défaut d'Apache HTTP

Fichiers à la racine :

- apache2.conf – définitions de base et inclusions des autres fichiers.
- envvars – définition des variables d'environnement pour l'exécution dont l'utilisateur/groupe.
- magic – définition des types de fichiers en concordance avec Mime.
- ports.conf – définition des ports d'écoute par défaut (HTTP 80, HTTPS 443).

L'on retrouve dans le répertoire conf-enabled les fichiers :

- charset.conf -
- localized-error-pages.conf -
- other-vhosts-acces-log.conf -
- security.conf – Informations reliés aux entêtes transmises par défaut à propos du système
- server-cgi-bin.conf

L'on retrouve dans le répertoire mod-enabled les fichiers:

- access_compat.load
- alias.conf/.load
- auth_basic.load
- authn_core.load
- authn_file.load
- authz_core.load
- authz_host.load
- authz_user.load
- autoindex.conf/load
- deflate.conf/load
- dir.conf/load
- env.load
- filter.load
- mime.conf/load
- mpm_event.conf/load
- negotiation.conf/load
- setenvif.conf/load
- status.conf/load

Et pour terminer dans le répertoire sites-enabled le fichier

- 000-default.conf – Définition du site par défaut

D'autres modules et configurations sont également disponibles dans les répertoires avec un suffix -available. Pour les activer il suffit de faire un lien symbolique entre les répertoires -enabled et -available ou d'utiliser les fonctions d'Apache tel que

- a2disconf – désactive une configuration
- a2dismod – désactive un module
- a2dissite – désactive un site
- a2enconf – active une configuration
- a2enmod – active un module
- a2ensite – active un site

Annexe III – 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