Projet XXX

Dossier d’Architecture
volet sécurité

APPROBATION DU DOCUMENT			
Organisation	**Nom (fonction)**	**Date**	**Visa**

DIFFUSION	
Destinataire	**Organisation**

SUIVI DES MODIFICATIONS			
Version	**Date**	**Auteurs**	**Changements**

Sommaire

1. INTRODUCTION

Ce document fourni le point de vue sécurité de l'application. Se référer aux volets joints pour les points de vue applicatifs, infrastructure et développement. Un référentiel joint liste les points à statuer et les hypothèses prises.

A noter que la disponibilité est traitée dans le volet infrastructure.

1.1. DOCUMENTATION DE RÉFÉRENCE

Mentionner ici les documents d'architecture de référence (mutualisés). Ce document ne doit en aucun cas reprendre leur contenu sous peine de devenir rapidement obsolète et impossible à maintenir.

N°	Version	Titre/URL du document	Détail
1	*1.0*	*XXX_Dispositifs_securite*	*Catalogue de dispositifs de sécurité habilités*

1.2. GLOSSAIRE

Note : par souci de concision, nous ne détaillons ici que les termes et acronymes spécifiques à l'application. Pour les définitions générales, veuillez vous référer au glossaire d'entreprise.

Terme	Définition
SLO	*Single Log-Out*

2. EXIGENCES DE SÉCURITÉ

Présenter ici les exigences, pas les solutions. Celles-ci seront détaillées dans les chapitres ultérieurs de ce document.

Pour les projets particulièrement sensibles, prévoir un dossier d'analyse de risque. Pour cela, utiliser par exemple la méthode EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité[1]).

2.1. INTÉGRITÉ

L'intégrité concerne la durabilité, la justesse et le niveau de confiance dans les données de l'application. Gérer l'intégrité des données consiste à vérifier qu'elle ne peuvent être altérées ou supprimées (involontairement, suite à un crash disque par exemple) ou volontairement, par exemple dans le cadre d'une attaque de type "man in the middle" ou par une personne s'étant octroyé des droits indus.

Attention à ne pas multiplier les classes de données. Il est possible de ne définir qu'une seule classe de donnée pour l'ensemble de l'application (cas courant). Les dispositifs permettant de répondre à ces exigences seront définies au chapitre 2.

L'intégrité exigée pour chaque classe de donnée de l'application est la suivante :

Classe de données	**Niveau « Non intègre »**	**Niveau « Détectable »**	**Niveau « Maîtrisé »**	**Niveau « Intègre »**
	la donnée peut ne pas être intègre	la donnée peut ne pas être intègre si l'altération est identifiée	la donnée peut ne pas être intègre si l'altération est identifiée et corrigée en moins d'une heure	la donnée doit être rigoureusement intègre
Données de la base métier				*X*
Données archivées		*X*		
Données calculées XXX			*X*	
Silo NoSQL des données Big Data avant consolidation	*X*			
Sources de l'application				*X*
Avis d'imposition en PDF				*X*

1 *https://www.ssi.gouv.fr/guide/ebios-2010-expression-des-besoins-et-identification-des-objectifs-de-securite/*

2.2. CONFIDENTIALITÉ

« La confidentialité est le fait de s'assurer que l'information n'est accessible qu'à ceux dont l'accès est autorisé. » (ISO 27018) .

Attention à ne pas multiplier les classes de données. Il est possible de ne définir qu'une classe de donnée pour l'ensemble de l'application (cas courant). Les dispositifs permettant de répondre à ces exigences seront définies au chapitre 2.

La confidentialité exigée pour chaque classe de donnée de l'application est la suivante :

Classe de données	Niveau « Public »	Niveau « Limité »	Niveau « Réservé »	Niveau « Privé »
	tout le monde accéder à la donnée	la donnée n'est accessible qu'aux personnes habilitées	la donnée n'est accessible qu'au personnel interne habilité	la donnée n'est visible que par l'intéressé(e)
Contenu éditorial	*X*			
Profil du compte du site Web		*X*		
Historique du compte			*X*	
Logs des activités de l'internaute			*X*	
Données RH de type "aides sociales aux employés"				*X*

2.3. IDENTIFICATION

L'identification est l'ensemble des dispositifs permettant de différentier un utilisateur d'un autre (sans vérifier qu'il est bien celui qu'il prétend être).

Exemple 1 : Un utilisateur ne peut avoir qu'un identifiant et un identifiant ne peut être partagé par plusieurs utilisateurs.

Exemple 2 : l'identité d'un internaute fera l'objet d'un test d'existence avant tout appel de service.

Exemple 3 : un ID est non supprimable, non modifiable et non réutilisable

Le cas échéant, préciser également si l'application nécessite une fédération d'identité.

La fédération d'identité est l'utilisation d'une même identité géré par un « identity provider » depuis plusieurs entités différentes . Par exemple, France Connect très utilisé par les administrations et basé sur OpenId Connect permet de réutiliser le compte d'une administration pour se loguer sur le compte d'une autre (DGFiP et CNAM par exemple). Il y a aussi les « Connect with [Google| Twitter|...] » en technologie OpenId Connect. Contrairement au SSO, la fédération d'identité n'assure pas un login automatique à une application mais permet simplement de réutiliser les mêmes credentials (login/mot de passe).

Exemple : L'identification et l'authentification seront externalisés à un fournisseur d'identité pour simplifier la gestion de la sécurité et réduire les coûts de développement et d'exploitation.

2.4. AUTHENTIFICATION

L'authentification permet de vérifier la cohérence entre l'identité de l'utilisateur et la personne physique se connectant.

L'authentification peut être à un ou plusieurs facteurs (dans ce dernier cas, on parle d'authentification forte). Ces facteurs peuvent être : quelque chose que l'on connaît (mot de passe), quelque chose qu'on est (biométrie), quelque chose qu'on a (token, générateur de mot de passe unique, pièce d'identité avec photo...).

Penser à décrire le système d'authentification une fois inscrit mais également lors de l'inscription (authentification initiale).

La délégation d'authentification s'appuie sur une technologie de fédération d'identité pour authentifier l'utilisateur.

Il est bien sûr possible d'ajouter au besoin dans le tableau ci-dessous des facteurs d'authentification spécifiques à votre organisation.

Les facteurs d'authentification requis en fonction des situations sont (on peut exiger plusieurs occurrences du même facteur, utiliser autant de croix) :

Cas d'authentification	Mot de passe respectant la politique de mot de passe XXX	Clé publique ssh connue	OTP par Token	Biométrie	Connaissance de données métier	E-mail d'activation	Délégation d'authentification
Utilisateur déjà inscrit	*X*						
Création d'un compte					*X X*	*X*	
Modification du mot de passe	*X*					*X*	
Accès aux logs		*X*					
Ajout d'un bénéficiaire de virement	*X*		*X*				
Application mobile YYY							*X*

2.5. SSO, SLO

Décrire les besoin en terme de Single Sign On et Single Log Out.

*Nous entendons ici SSO dans son sens complet : une authentification **automatique** à une application d'un utilisateur déjà authentifié depuis une autre application du même domaine de confiance.*

Attention, la mise en place de SSO est souvent complexe, surtout si l'infrastructure (ID provider…) n'existe pas encore. Elle nécessite souvent une adaptation des applications. L'exigence en SSO doit être justifiée. Une application périphérique ou un outil rarement utilisé n'a en général pas besoin de SSO (une simple authentification centralisée au sein d'un annuaire LDAP suffit souvent). Attention également à évaluer l'impact qu'aurait une authentification faible (mauvais mot de passe par exemple) sur la sécurité de l'ensemble du SI.

Exemple 1 : aucun SSO n'est exigé puisque toutes les IHM de l'application sont exposées au sein d'un portail JSR352 qui gère déjà l'authentification.

Exemple 2 : aucun besoin de SSO ou SLO n'est identifié

Exemple 3 : cette application Web métier devra fournir une authentification unique mutualisée avec celle des autres applications de l'intranet : une fois authentifié sur l'une des applications, l'agent ne doit pas avoir à se reconnecter (sauf bien sûr suite à l'expiration de sa session). De même, une déconnexion depuis l'une des applications doit assurer la déconnexion de toutes les applications de l'intranet.

2.6. PREUVE

Lister ici les données à conserver car pouvant servir de preuve en cas de contestation ainsi que leur durée de rétention :

Donnée	Objectif	Durée de rétention
Log complet (IP, heure GMT, détail) des commandes passées sur le site	*Prouver que la commande a bien été passée*	*1 an*
Date et contenu du mail de confirmation	*Prouver que le mail de confirmation a bien été envoyé*	*2 ans*
Contrat d'assurance signé et numérisé en PDF	*Prouver que le contrat a bien été signé*	*5 ans*
Avis d'imposition primitif avec signature numérique	*Conserver le montant et de l'impôt.*	*5 ans*

2.7. NON RÉPUDIATION

Lister ici les actions métiers possédant une exigence de non-répudiation, c'est à dire un dispositif permettant de rendre impossible la remise en cause d'un contrat en prouvant l'identité des deux parties et l'intégrité du document par signature numérique comme décrit dans le texte n°2000-230 du 13 mars 2000 du code civil.

Donnée signée	Origine du certificat client	Origine du certificat serveur
Déclaration d'impôt sur le revenu (données X, Y et Z)	*PKI de l'administration fiscale*	*Verisign*

2.8. ANONYMAT ET VIE PRIVÉE

Lister les contraintes d'anonymat et de vie privée légale (exigée par la CNIL) + les contraintes supplémentaires de l'organisation. Voir aussi https://www.cnil.fr/fr/comprendre-vos-obligations .

Exemple 1 : Aucun consolidation de donnée de pourra être faite entre les données du domaine PERSONNE et du domaine SANTE.

Exemple 2 : Par soucis de sécurité en cas d'intrusion informatique, certaines données des personnes seront expurgées avant réplication vers la zone publique : le taux de cholestérol et le poids.

Exemple 3 : aucune donnée raciale, politique, syndicales, religieuse ou d'orientation sexuelle ne pourra être stockée sous quelque forme que ce soit dans le SI.

Exemple 4 : Les données OpenData issues du domaine « logement » ne contiendront que des données consolidées de niveau commune, pas plus précise.

Exemple 5 : En application de la directive européenne « paquet telecom », un bandeau devra informer l'usager de la présence de cookies.

Exemple 6 : En application du RGPD, un consentement explicite des utilisateurs dans la conservation de leurs données personnelles de santé est à mettre en place.

2.9. AUTORISATIONS

Une autorisation (ou habilitation) permet de donner l'accès à une fonction applicative (ou « privilège » ou « permission ») à un utilisateur ou un groupe d'utilisateur.

Par exemple :

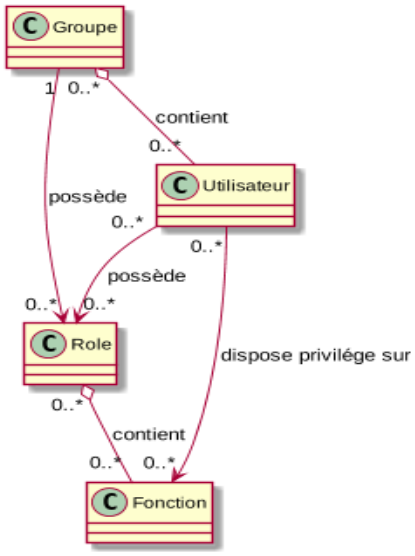
- *Fonction F1 : faire un virement inter-bancaire*
- *Fonction H2 : voir l'historique de son compte*
- *Fonction E2 : supprimer un utilisateur*

Attention à ne pas multiplier le nombre de fonctions et de rôles pour éviter une explosion combinatoire et des coûts de gestion associés.

Pour simplifier la gestion des autorisations par factorisation, on peut :

- *regrouper les utilisateurs dans des groupes (G_chef_service = Untel, Lui, Elle)*
- *regrouper les fonctions dans des rôles (R_Administrateur, R_banquier_niv1, R_chef_service…) qu'on peut affecter à une personne ou un groupe*

Exemple de gestion complète des autorisations :



Penser à spécifier les éventuels pseudos-utilisateurs et leurs rôles :

- *@anonyme : les personnes non connectées*
- *@connecte : les personnes connectées*

Préciser si l'application doit utiliser de la délégation d'autorisation (type OAuth2) et si oui, l'application est-elle fournisseur ou consommateur d'autorisations ? Quelles sont les autorisations concernées ?

Exemple 1 : les personnes non connectées auront accès à tous les privilèges en lecture seule, c'est à dire : ….

Exemple 2 : l'application s'appuiera sur une gestion des autorisations matricielle de type rôles → groupes et utilisateurs . Le détail des autorisations sera donnée dans les SFD.

Groupe / utilisateur	Rôle « suppression »	Rôle administration	Rôle consultation des données de base
Groupe g_usagers			*X*
@anonyme			
Groupe g_admin	*X*	*X*	*X*
Utilisateur XXX	*X*		*X*

Exemple 3 : L'application aura besoin de façon obligatoire du carnet d'adresse Facebook de l'usager

2.10. TRAÇABILITÉ, AUDITABILITÉ

Lister ici les besoins en traces permettant de détecter par exemple :

- *un usage abusif des applications Back Office par des employés*
- *des intrusions informatiques*
- *des modifications de données*

Les traces sont des données nominatives et complètes pour permettre l'audit. Elles sont donc elles-mêmes sensibles et nécessitent souvent un bon niveau de confidentialité (voir 2.2)

Différentier :

- *les traces métier (bilan d'un acte de gestion complet comme « l'agent XXX a consulté le dossier de Mme YYY »)*
- *et les traces applicatives (logs) comme dans un fichier de log : « [INFO] 2016/12/23 11:14 Appel par XXX du service consulter » qui sont de niveau technique.*

Pour les données les plus sensibles, prévoir une traçabilité à deux niveaux (tracer la consultation des traces) pour éviter une traçabilité hiérarchique abusive.

La traçabilité des données des référentiels (PERSONNE typiquement) nécessite une historisation complète, ce qui est de toute façon une bonne pratique (voir par exemple Longépé « Le projet d'Urbanisation du SI », règles applicatives 1, 2 et 3). Pour cela, prévoir un MCD permettant d'ajouter un enregistrement à chaque changement de la donnée avec une date de modification et une date d'effet.

Exemple 1 : pour le module XXX, toute action métier (en mise à jour comme en consultation) devra faire l'objet d'une trace métier contenant a minima l'agent, la date et en cas de modification l'ancienne et la nouvelle valeur.

Exemple 2 : Toute intrusion dans le SI devra être détectée (dans la mesure du possible).

Exemple 3 : nous devons pouvoir reconstituer l'historique du dossier de tout patient à n'importe quelle date.

3. INTÉGRITÉ

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.1.

Classe de données	Niveau exigé	Mesures
Données de la base métier	*Intègre*	*- Utilisation du SGBDR SGBD PostgreSQL avec un niveau d'isolation transactionnelle SERIALIZABLE* *- Les entités seront référencées uniquement par des ID techniques issues de séquences PostgreSQL.*
Données archivées	*Détecté*	*Génération de checksums SHA-256 des backups*
Données calculées XXX	*Maîtrisé*	*Stockage d'un checksum SHA1, relance du calcul automatiquement par batch dans les 24H.*
Silo NoSQL des données Big Data avant consolidation	*Non intègre*	*Pas de mesure particulière, pas de backup*
Sources	*Intègre*	*Utilisation du SCM Git*
Avis d'imposition PDF	*Intègre*	*Signature numérique par la clé privée de l'administration des données D de l'avis au format PKCS#7 (RSA, SHA256) avec horodatage. D= base64(montant net + date+nom). La signature résultante sera intégrée a posteriori au format hexa en pied de page du PDF.*

4. CONFIDENTIALITÉ

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.2.

Classe de données	Niveau exigé	Mesures
Contenu éditorial	*Public*	*Aucune, contenu en HTTP et HTTPS, pas d'authentification.*
Profil du compte du site Web	*Limité*	*L'accès à ce contenu nécessite une authentification réussie par login/mot de passe.*
Historique du compte	*Réservé*	*L'accès à ce contenu est réservé aux exploitants habilités, uniquement via des requêtes PL/SQL de la base de données.*
Logs des activités de l'internaute	*Réservé*	*L'accès aux fichiers de log est réservé aux exploitants habilités (accès SSH à la machine XXX et mot de passe Unix)*
Données RH aides sociales aux employés	*Privé*	*Ces données sont chiffrées en AES 256 sous forme d'un BLOB en base, remontées au client Web via le service REST YYY puis déchiffrées au sein du navigateur dans l'application Angular (librairie forge.js) via un mot de passe complémentaire de l'utilisateur (non stocké coté serveur). Il s'agit donc d'un chiffrement client uniquement. Une perte de mot de passe rend les données irrécupérables. Les données modifiées sur le client sont chiffrées et enregistrées à nouveau dans le BLOB via le service REST XXX.*

Penser aussi à la confidentialité de données dérivées :

- *Chiffrement des backups*
- *Chiffrement des données clientes pour les applications lourdes. Cela peut être un chiffrement matériel en SED (Self Encryption Disk), un chiffrement logiciel de niveau partition (SafeGuard, dm-crypt) ou de niveau fichier (encfs, TrueCrypt,…)*

5. IDENTIFICATION

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.3.

En cas d'utilisation de fédération d'identité, renvoyer le lecteur au chapitre 7.

Exemple 1 : L'Id des usagers de l'application sera l'attribut uid des DN `cn=XXX,ou=service1,dc=entreprise,dc=com` dans l'annuaire LDAP central. Un filtre sera également appliqué sur l'appartenance au groupe `ou=monapplication,dc=entreprise,dc=com`

Exemple 2 : Pour assurer la non réutilisation des ID des comptes supprimés, une table d'historique sera ajoutée dans l'application et requêtée avant toute création de nouveau compte.

6. AUTHENTIFICATION

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.4.

Pour les authentifications par mot de passe, décrire le mode de stockage et de vérification. Penser également à décrire les solutions de changement de mot de passe.

En cas d'utilisation de fédération d'identité, renvoyer le lecteur au chapitre 7.

Exemple 1 : L'authentification des internautes inscrits se fera par login/mot de passe (respectant la politique de mot de passe XXX)

Exemple 2 : L'authentification des internautes à l'inscription se fera par la saisie du code internaute figurant sur les factures + la valeur de la dernière facture puis par l'activation du compte via un lien figurant dans un e-mail de vérification.

Exemple 3 : lors de la création d'un nouveau bénéficiaire de virement dans l'espace internet, l'utilisateur devra fournir un mot de passe unique issu de son token OTP en plus d'être authentifié.

Exemple 4 : Les mots de passe ne seront en aucun cas conservés mais stockés sous la forme de digest bcrypt.

7. FÉDÉRATION D'IDENTITÉ

En cas de fédération d'identité, détailler la technologie choisie et son intégration. Les solutions les plus courantes sont actuellement : France Connect, OpenId Connect, SAML ou Oauth 2.0 (pseudo-authentification seulement pour cette dernière). Pour les applications Web, préciser les contraintes navigateur (activation des cookies en particulier).

Exemple : L'IHM grand public permettra une identification et authentification France Connect de sorte que les utilisateurs puissent utiliser leur compte DGFiP ou CNAM pour s'identifier et s'authentifier. La cinématique d'authentification sera la suivante : <schéma>

8. SSO, SLO

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.5.

En cas de SSO/SLO, détailler la technologie choisie et son intégration. Quelques solutions courantes : CAS, OpenAM, LemonLDAP::NG. Pour les applications Web, préciser les contraintes navigateur (activation des cookies en particulier).

Exemple 1 : L'IHM XXX intégrera un client CAS spring-security pour le SSO. Le serveur CAS utilisé sera YYY. Son royaume d'authentification (realm) sera l'annuaire AD XXX. <schéma>

Exemple 2 : Comme toutes les applications du portail métier, l'IHM XXX devra gérer les callbacks de déconnexion provenant du serveur CAS suite à une demande de SLO. <schéma>

9. PREUVE

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.6.
Exemple 1 : tous les documents servant de preuve seront archivés dans la GED Alfresco.

Exemple 2 : Les logs contenant le terme [PREUVE] et issu de l'ensemble des composants seront centralisés via le système de centralisation de log Elastic Search puis insérés avec traitement Logstash de façon journalière vers la base MongoDB « preuves ».

10. NON-RÉPUDIATION

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.7.

Exemple : La déclaration d'impôt sera signée par le certificat client de l'usager (certificat X509, RSA, SHA-256) qui lui a été fourni par le composant XXX suivant l'architecture suivante : <schéma>.

11. ANONYMAT ET VIE PRIVÉE

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.8.

Exemple 1 : un audit interne sera mené une fois par an sur le contenu des données en base et les extractions à destination des partenaires.

Exemple 2 : les données à destination de la zone publique seront exportées partiellement via un `COPY (SELECT …) TO <fichier>`. Les colonnes sensibles seront ainsi exclues de la réplication.

Exemple 3 : le bandeau d'acceptation des cookies sera mis en ouvre sur toutes les pages de l'application Angular via le module angular-cookie-law.

12. AUTORISATIONS

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.9.

Exemple 1 : la gestion des autorisations sera gérée applicativement et stockée au sein de tables `Utilisateur`, `Groupe`, `Role` et `Privilege` de la base de donnée PostgreSQL. Ces tables seront décrites dans le dossier de spécification.

Exemple 2 : L'obtention du carnet d'adresse Facebook sera en OAuth2. On utilisera l'API Java Google Oauth2.

13. TRACABILITÉ, AUDITABILITÉ

Décrire ici les mesures répondant aux exigences exprimées au chapitre 2.10.

Exemple 1 : à la fin de chaque action métier, l'application ReactJS appellera dans une action asynchrone un service REST de trace métier. Ce service stockera les traces dans une base Elastic Search pour consultation en Kibana. <schéma>

Exemple 2 : l'outil d'IDS hybride (réseau + host) OSSEC sera installé sur l'ensemble des machines utilisées par l'application.

Exemple 3 : Les tables XXX, YYY, .. seront historisées suivant le principe suivant : … <diagramme de classe>

14. AUTO-CONTRÔLE DES VULNÉRABILITÉS

La gestion des vulnérabilités dépasse largement le cadre de ce document mais il est bon de s'auto-contrôler pour s'assurer que les failles les plus courantes sont bien prises en compte et comment. Cette liste est en partie basée sur le TOP 10 OWASP. Pour le TOP 10 des application mobiles, adapter cette liste avec le TOP 10 mobile .

Vulnérabilité	Pris en compte ?	Mesure technique entreprises
Accès à des ports privés	*X*	*Configuration du pare-feu iptables sur la machine exposée à Internet. Seul les ports 80 et 443 sont ouverts. Le pare-feu sera configuré en mode stateful (avec extension conntrack) .*
Attaque de mot de passe par force brute	*X*	*Utilisation de fail2ban, mise en prison de 1h au bout de 3 tentatives de connexion ssh.*
Visibilité des URLs directes	*X*	*Centralisation de tous les accès depuis Internet via un reverse proxy Apache + mod_proxy. Réécriture d'URLs pour masquer les URL internes.*
Contournement du contrôle d'accès	*X*	*Utilisation du SSO CAS, voir 8.*
Injection SQL	*X*	*Utilisation de `PreparedStatement` uniquement, audit des requêtes SQL.*
Injection OS	*X*	*Vérification qu'il n'y a aucun appel de commandes systèmes dans le code (type `Runtime.exec()`)*
Violation de gestion d'authentification et de session	*X*	*Traité avec le dispositif anti-CSRF, voir plus bas. On logue l'IP à fin d'audit.*
XSS	*X*	*- Utilisation de librairie d'échappement. Pour les modules Java, nous utiliserons `StringEscapeUtils.escapeHtml4()` de commons-lang.* *- Utilisation des headers HTTP :* `X-Frame-Options SAMEORIGIN` `X-XSS-Protection 1;mode=block` `X-Content-Type-Options nosniff` `Content-Security-Policy` `X-XSS-PROTECTION` *(pour parer les détournements de dispositifs anti-XSS des navigateurs)* *- Spécification systématique de l'encoding dans le header de réponse Content-Type (ex : `text/html;charset=UTF-8`) pour parer les attaques basées sur des caractères spéciaux contournant l'anti-XSS.*
ReDOS	*X*	*Vérification que les expressions régulières utilisées par les dispositifs anti-XSS ne sont pas éligibles à ce type d'attaque, voir https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS .*
Référence directe à un objet	*X*	*Vérification à chaque requête que les arguments passés correspondent bien à la personne identifiée. Par exemple, toute requête contient son ID et on vérifie par une requête que le dossier qu'elle tente de consulter lui appartient bien avant de poursuivre la requête initiale.*

Planification des mises à jour de sécurité	*X*	*- Les mises à jour Centos seront planifiées tous les premiers mercredi du mois.* *- Les mises à jour Wildfly sont appliqué au plus deux semaines après leur sortie.*
Exposition de données sensibles	*X*	*- Tous les algorithmes de sécurité sont à jour : au minimum SHA-256, AES 256* *- Le SSL V2 et V3 est désactivé coté Apache suite à la faille DROWN (* ***SSLProtocol all -SSLv2 -SSLv3****)* *- L'application ne fonctionne qu'en HTTPS* *- Le serveur Web fixera le header HSTS avec includeSubDomains sur toutes les resources*
CSRF	*X*	*- Utilisation du dispositif anti-CSRF d'AngularJS (https://docs.angularjs.org/api/ng/service/$http)*
Manque de contrôle d'accès au niveau fonctionnel	*X*	*- Mise en place de la politique d'autorisation décrite en 2.9* *- Campagne de tests fonctionnels*
Log injection	*X*	*- Échappement des logs avant de les transmettre à log4j* *- Vérification des outils de consultation de logs*
Attaques HTTPS + compression CRIME/BREACH	*X*	*- Désactivation de la compression HTTPS au niveau de l'Apache : SSLCompression off* *- Dispositif anti-CSRF*
Upload de fichiers malicieux	*X*	*- Validation des pièces jointes par l'anti-virus clamav*
...		

(il existe bien sûr de nombreux autres points de contrôle dépendants du contexte de l'application)