WAVESTONE

Sécurité & Agilité

Jeu de cartes

Présentation

Fournir des outils ludiques pour favoriser la prise en compte des enjeux SSI

Présentation du jeu

Le jeu est composé de cartes, chacune ayant 2 faces :

RECTO

/ **Les Evil User Stories** représentent les scénarios d'attaque que pourrait réaliser un utilisateur malveillant, en détournant les fonctionnalités d'une User Story (ex: élévation de privilèges, injection de code XSS, etc.)

VERSO

/ **Les Security User Stories** décrivent les mesures de sécurité à implémenter pour s'assurer que l'Evil User Story ne se produise pas (ex : encodage des caractères spéciaux pour tous les champs de saisie, etc.).

Comment jouer ?

/ Il faut rassembler à minima les **membres de la Squad ayant une connaissance fonctionnelle de la solution** (Product Owner) et une connaissance **technique** et des **risques** (référents sécurité, développeurs, architectes).
/ Les architectes **dessinent l'architecture applicative** sur une grande affiche A3 sur une table en faisant apparaitre les flux de données et la classification des données et le **Product Owner liste les prochaines User Stories qui devront être développées.**
/ Le référent sécurité (Security Champion) et les développeurs **répartissent les vulnérabilités exploitables sur le schéma d'architecture**.
/ Le Product Owner et le Security Champion **qualifient l'impact que peut avoir chaque vulnérabilité.**
/ Le Security Champion et les développeurs vérifient si les **mesures de sécurité** permettant de contrer les vulnérabilités identifiées sont déjà **implémentées**.
/ Si des mesures de sécurité ne sont pas encore en production : Le Security Champion **priorise les mesures techniques** à implémenter permettant de couvrir les **risques induits** (risque pour l'entreprise, pas seulement au niveau business).
/ Le Product Owner priorise les autres mesures de sécurité au regard des **risques business** / et des moyens de l'équipe.

RECTO

En tant qu'attaquant, je veux **injecter du code malveillant** dans les champs de saisie non sécurisés de l'application

EXEMPLES

- **Les injections XSS (Cross-site scripting),** qui peuvent permettre à un attaquant d'afficher des messages malveillants à d'autres utilisateurs pour les piéger, ou voler leur cookie de session s'il n'est pas suffisamment sécurisé
- **Injections SQL**, qui peuvent permettre à un attaquant d'interagir directement avec la base de données et d'accéder/modifier les données sans contrôle

S'APPLIQUE AUX

- User Stories qui impliquent la création d'un nouveau champ de saisie/de recherche utilisateur, ou d'un nouveau paramètre envoyé dans une requête HTTP

VERSO

En tant que développeur, je veux m'assurer que les attaques par **injection de code** sont **évitées**

TÂCHE(S) A INTEGRER AU BACKLOG

- Avant d'insérer des données saisies par un utilisateur dans une requête, ou avant de les réafficher à l'utilisateur, il est nécessaire d'échapper tous les caractères spéciaux (avec la fonction *htmlspecialchars* par exemple)
- Utiliser l'une des nombreuses bibliothèques disponibles qui empêchent les attaques par injection de code, ou créer et appliquer une méthode unique de filtrage des entrées utilisateurs
- Toujours créer les cookies de session avec l'attribut HttpOnly pour s'assurer qu'ils ne peuvent pas être volés par le code Javascript

RECTO

En tant qu'attaquant, je veux **injecter du code malveillant** dans les champs de saisie non sécurisés de l'application

EXEMPLES

- **Les injections XSS (Cross-site scripting),** qui peuvent permettre à un attaquant d'afficher des messages malveillants à d'autres utilisateurs pour les piéger, ou voler leur cookie de session s'il n'est pas suffisamment sécurisé
- **Injections SQL**, qui peuvent permettre à un attaquant d'interagir directement avec la base de données et d'accéder/modifier les données sans contrôle

S'APPLIQUE AUX

- User Stories qui impliquent la création d'un nouveau champ de saisie/de recherche utilisateur, ou d'un nouveau paramètre envoyé dans une requête HTTP

VERSO

En tant que développeur, je veux m'assurer que les attaques par **injection de code** sont **évitées**

TÂCHE(S) A INTEGRER AU BACKLOG

- Avant d'insérer des données saisies par un utilisateur dans une requête, ou avant de les réafficher à l'utilisateur, il est nécessaire d'échapper tous les caractères spéciaux (avec la fonction *htmlspecialchars* par exemple)
- Utiliser l'une des nombreuses bibliothèques disponibles qui empêchent les attaques par injection de code, ou créer et appliquer une méthode unique de filtrage des entrées utilisateurs
- Toujours créer les cookies de session avec l'attribut HttpOnly pour s'assurer qu'ils ne peuvent pas être volés par le code Javascript

Cas d'usage : *Application WaveCare*

Description :

- Consultez les disponibilités de praticiens
- Transmettez en temps réel vos données de santé grâce à votre montre connectée
- Réalisez la consultation à distance en Visio
- Recevez votre ordonnance après le RDV en format dématérialisé

Exemples de fonctionnalités concernées par cette carte :

- Inscription et connexion d'un patient / médecin
- Ajout de créneaux de disponibilités d'un médecin
- Envoi d'une ordonnance post-RDV

/ **02**

Jeu de cartes

RECTO

1

En tant qu'attaquant, je veux **injecter du code malveillant** dans les champs de saisie non sécurisés de l'application

EXEMPLES

- **Les injections XSS (Cross-site scripting),** qui peuvent permettre à un attaquant d'afficher des messages malveillants à d'autres utilisateurs pour les piéger, ou voler leur cookie de session s'il n'est pas suffisamment sécurisé
- **Injections SQL**, qui peuvent permettre à un attaquant d'interagir directement avec la base de données et d'accéder/modifier les données sans contrôle

S'APPLIQUE AUX

- User Stories qui impliquent la création d'un nouveau champ de saisie/de recherche utilisateur, ou d'un nouveau paramètre envoyé dans une requête HTTP

2

En tant qu'attaquant, je veux effectuer **une élévation de privilèges** pour accéder à des fonctionnalités et/ou données que je ne suis pas autorisé à voir

EXEMPLES

- **Elévation verticale des privilèges :** accès à des fonctionnalités non autorisées depuis un compte à faibles privilèges (ex: lecture seule): fonctions d'administration ou droits d'écriture / validation
- **Elévation horizontale des privilèges :** accès à des données qui ne sont pas dans le périmètre du compte utilisé (données d'autres utilisateurs par exemple, ou d'une autre entité métier)

S'APPLIQUE AUX

- User Stories qui impliquent la création d'une nouvelle fonctionnalité, ou d'une nouvelle interface utilisée pour accéder à des données cloisonnées par périmètres

VERSO

En tant que développeur, je veux m'assurer que les attaques par **injection de code** sont **évitées**

TÂCHE(S) A INTEGRER AU BACKLOG

- Avant d'insérer des données saisies par un utilisateur dans une requête, ou avant de les réafficher à l'utilisateur, il est nécessaire d'échapper tous les caractères spéciaux (avec la fonction *htmlspecialchars* par exemple)
- Utiliser l'une des nombreuses bibliothèques disponibles qui empêchent les attaques par injection de code, ou créer et appliquer une méthode unique de filtrage des entrées utilisateurs
- Toujours créer les cookies de session avec l'attribut HttpOnly pour s'assurer qu'ils ne peuvent pas être volés par le code Javascript

En tant que développeur, je veux m'assurer que les utilisateurs ne peuvent **accéder qu'aux fonctionnalités / données qui correspondent à leurs privilèges**

TÂCHE(S) A INTEGRER AU BACKLOG

- Vérifier systématiquement le cookie de session de l'utilisateur pour chaque requête envoyée au serveur. Si l'utilisateur auquel le cookie appartient n'a pas les privilèges pour accéder à une fonctionnalité / un périmètre, refuser l'accès et terminer la session (invalider le cookie de session)
- Si un identifiant est utilisé dans une URL pour accéder à une ressource (fichier, donnée, etc.), ne pas utiliser d'identifiant numérique qui peut être deviné / incrémenté

RECTO

3

En tant qu'attaquant, je veux **voler les identifiants d'un autre utilisateur** (login et mot de passe)

EXEMPLES

- **Le brute force** peut permettre à un attaquant d'essayer de deviner plusieurs fois le mot de passe d'un autre utilisateur et, s'il n'est pas bloqué, il peut réussir
- **Le Spear phishing** peut permettre de piéger des utilisateurs ciblés, avec des mails malveillants construits spécialement pour la cible, incitant à cliquer sur un lien malveillant

S'APPLIQUE AUX

- User Stories qui impliquent la création d'une nouvelle interface d'authentification

4

En tant qu'attaquant, je veux **usurper la session d'un utilisateur**

EXEMPLE

- **Vol de cookie de session :** si un attaquant est capable de voler le cookie d'un autre utilisateur, il pourra accéder au compte de l'utilisateur (sans connaître son mot de passe) et effectuer des actions en son nom, sans qu'il soit possible de remonter à l'attaquant

S'APPLIQUE AUX

- User Stories qui impliquent la création d'une nouvelle interface d'authentification / d'un nouveau cookie de session

VERSO

En tant que développeur, je veux m'assurer que le **brute force ne peut pas être utilisé** et que l'**authentification forte** est mise en place

TÂCHE(S) A INTEGRER AU BACKLOG

/ Bloquer les comptes utilisateurs suite à de nombreuses tentatives de connexion infructueuses. Par exemple, 3, 5 ou 10 tentatives en fonction de la sensibilité de l'application. Le blocage peut être basé sur l'identifiant de l'utilisateur ou sur l'adresse IP. En outre, afficher un message générique lorsque la connexion échoue, afin que l'attaquant ne puisse pas savoir si un login est valide ou non

/ Pour les comptes sensibles, implémenter l'authentification forte (authentification à deux facteurs). Cela empêchera un attaquant qui a réussi un spear-phishing d'accéder à l'application

En tant que développeur, je veux m'assurer que **les cookies de session sont suffisamment sécurisés**

TÂCHE(S) A INTEGRER AU BACKLOG

/ Définir systématiquement les paramètres de sécurité pour chaque cookie créé : httpOnly et secure. Le premier empêchera le code côté client d'accéder au cookie, et le second empêchera le cookie d'être envoyé sur un canal non chiffré

/ Révoquer le cookie de session lorsque l'utilisateur se déconnecte de l'application, ou lorsque sa session a expiré

RECTO

5

En tant qu'attaquant, je veux **piéger un autre utilisateur** et lui faire exécuter une action **sans qu'il s'en aperçoive**

EXEMPLE

/ **CSRF attacks (Cross Site Request Forgery):** si les requêtes de modification de sont prévisibles (sans aucun paramètre aléatoire), un attaquant pourrait envoyer une requête forgée (prédictible) à un autre utilisateur, qui l'exécutera avec son cookie de session (donc avec ses privilèges) sans s'en apercevoir

S'APPLIQUE AUX

/ User Stories qui impliquent la création d'une nouvelle fonctionnalité de modification de données dans l'application

6

En tant qu'attaquant, je veux **prendre le contrôle du serveur sous-jacent en envoyant un fichier infecté**

EXEMPLE

/ **Injection de fichiers :** si le type de fichier n'est pas correctement vérifié, ou si le fichier ne passe pas par un outil antivirus avant d'être téléchargé sur le serveur, un attaquant pourrait envoyer un fichier infecté qui pourrait lui permettre d'accéder directement au serveur avec des privilèges élevés (par exemple, un reverse-shell)

S'APPLIQUE AUX

/ User Stories qui impliquent la création d'une fonctionnalité de téléchargement de fichiers

VERSO

En tant que développeur, je veux m'assurer que **les requêtes de l'application ne sont pas prédictibles**

TÂCHE(S) A INTEGRER AU BACKLOG

- Ajouter systématiquement un paramètre aléatoire pour chaque demande de modification, qui ne peut être deviné. Le serveur doit vérifier la valeur de ce paramètre aléatoire avant d'accepter toute demande de modification
- Le paramètre aléatoire (ou token) pourrait être généré par le serveur sur la base de l'ID utilisateur, de l'horodatage, de l'ID formulaire et d'un sel connu uniquement par le serveur

En tant que développeur, je veux m'assurer que **tous les fichiers téléchargés sont assainis**

TÂCHE(S) A INTEGRER AU BACKLOG

- Définir une liste blanche des extensions de fichier autorisées et refuser tout autre type de fichier
- Réaliser un scan antivirus basé sur la signature du fichier avant de le télécharger sur le serveur
- Pour les fichiers XML, désactiver le traitement des entités externes XML et la DTD pour tous les moteurs XML, afin de prévenir les attaques XXE (XML eXternal Entities)

RECTO

7

En tant qu'attaquant, je veux **voler des données non sécurisées en transit**

EXEMPLE

/ **Interception de flux non chiffrés :** si les flux applicatifs ne sont pas chiffrés à l'aide d'un certificat SSL (HTTPS), toute personne se trouvant sur le même sous-réseau que l'utilisateur pourrait lire en clair toutes les informations transmises (mot de passe, données commerciales, etc.). Si le certificat SSL prend en charge des algorithmes obsolètes (tels que SSLv3), le résultat pourrait être le même

S'APPLIQUE À

/ Toutes les User Stories

8

En tant qu'attaquant, je veux **prendre le contrôle du serveur sous-jacent en utilisant un service surexposé**

EXEMPLE

/ **Exposition inutile des services :** si le serveur web présente des services (ports) qui ne sont pas nécessaires aux utilisateurs standard, tels que des services administratifs (SSH, RDP), un attaquant pourrait les exploiter pour compromettre le serveur

S'APPLIQUE AUX

/ User Stories qui impliquent l'ouverture de flux de réseau ou un changement dans l'exposition des applications (intranet / internet)

VERSO

En tant qu'administrateur du serveur web, je veux m'assurer que **toutes les communications passent par un canal chiffré**

TÂCHE(S) A INTEGRER AU BACKLOG

- N'autoriser que les communications chiffrées avec le serveur web, en déployant un certificat SSL et en activant l'option HSTS
- Rediriger le port 80 vers 443
- Vérifier régulièrement les algorithmes de chiffrement pris en charge par le serveur, afin de s'assurer qu'ils sont tous à jour selon les meilleures pratiques de sécurité (pas de SSLv2/v3 et TLSv1.0, pas de 3DES, etc.)

En tant qu'architecte réseau, je veux m'assurer que **seuls les services nécessaires (ports) sont exposés à l'utilisateur**

TÂCHE(S) A INTEGRER AU BACKLOG

- Veiller à ce que l'exposition des services réseaux soit limitée à ce que l'utilisateur a besoin de voir (par exemple, uniquement les ports 80 et 443 pour un serveur web)
- Concevoir le réseau interne de manière à ce que les bases de données et les serveurs applicatifs soient placés sur des zones de réseau dédiées, auxquelles les utilisateurs standard ne peuvent pas accéder
- Les interfaces administratives telles que RDP et SSH devraient être limitées à un réseau administratif dédié

RECTO

9

En tant qu'attaquant, je veux **compromettre l'application en exploitant une vulnérabilité connue**

EXEMPLE

- **Composant obsolète :** si les composants ne sont pas à jour (OS, middleware, etc.), ils peuvent présenter des vulnérabilités publiques. Ces vulnérabilités ont souvent un script d'exploitation public disponible sur internet, qui peut être utilisé facilement par des script-kiddies

S'APPLIQUE AUX

- Toutes les User Stories qui impliquent le déploiement d'un nouveau composant – mais il faut assurer un suivi dans le temps

10

En tant qu'attaquant, je veux **effectuer un déni de service pour empêcher les utilisateurs légitimes d'accéder à l'application**

EXEMPLE

- **Déni de service (DOS) ou DOS distribué :** un attaquant qui voudrait avoir un impact sur la disponibilité de l'application pourrait soumettre de nombreuses requêtes, qui surchargeraient la capacité de traitement de l'application. De cette façon, les utilisateurs légitimes ne pourraient pas accéder à l'application

S'APPLIQUE AUX

- User Stories qui impliquent une exposition sur Internet

VERSO

En tant qu'administrateur, je veux m'assurer que **tous les composants sont à jour avec les derniers correctifs de sécurité**

TÂCHE(S) A INTEGRER AU BACKLOG

- Créer et maintenir une liste exhaustive de toutes les technologies utilisées par l'application, et de leurs versions
- S'inscrire à un service d'alerte pour ces technologies, qui permet d'être averti si de nouvelles vulnérabilités sont publiées

En tant qu'architecte / développeur, je veux m'assurer que **mon application est protégée contre le déni de service (au niveau du réseau et du code)**

TÂCHE(S) A INTEGRER AU BACKLOG

- En fonction des besoins de disponibilité de l'application, veiller à ce que des mesures anti-DDOS soient mises en place pour la couche réseau et la couche applicative
- Par exemple, au niveau du réseau, un fournisseur de services peut empêcher de telles attaques. Au niveau de la couche applicative, des mesures d'atténuation telles que la mise en place d'un Captcha ou d'un Web Application Firewall (WAF) peuvent réduire l'impact de l'attaque

RECTO

11

En tant qu'attaquant, je veux **accéder aux fichiers techniques du serveur par l'intermédiaire de l'application**

EXEMPLE

/ **Path Traversal:** en manipulant des variables qui référencent des fichiers stockés sur le serveur, et en remontant l'arborescence via des séquences "../..." ou en utilisant des chemins de fichier absolus, il peut être possible d'accéder à des fichiers et des répertoires sensibles qui sont stockés en dehors du répertoire du site web (ex: récupérer le /etc./passwd)

S'APPLIQUE À

/ Toutes les User Stories qui impliquent l'accès à des fichiers sur le serveur

12

En tant qu'attaquant, je veux **accéder à l'application ou à ses composants en utilisant un compte par défaut ou un compte dont le mot de passe peut être deviné**

EXEMPLES

/ **Compte technique par défaut :** sur la plupart des composants, des comptes techniques par défaut avec mot de passe prédéfini sont activés à l'installation. S'il n'est pas modifié, un attaquant pourrait obtenir l'accès en utilisant l'un d'eux

/ **Politique de mots de passe faibles** : sur l'application, si les utilisateurs peuvent définir des mots de passe faibles (mot de passe = login), l'attaquant pourrait obtenir l'accès en devinant l'un d'entre eux

S'APPLIQUE AUX

/ User Stories qui impliquent l'installation d'un nouveau composant ou la définition d'une politique de mot de passe pour l'utilisateur

VERSO

En tant que développeur, je veux m'assurer que **les fichiers techniques hors du répertoire du serveur web ne sont pas accessibles par l'application**

TÂCHE(S) A INTEGRER AU BACKLOG

- Ne pas se baser sur les entrées utilisateur pour réaliser d'appel de fichier système
- S'assurer que l'utilisateur ne peut pas fournir tous les éléments du chemin
- Valider la saisie de l'utilisateur en n'acceptant que le format attendu

En tant qu'administrateur, je veux **m'assurer que les comptes par défaut sont désactivés et que les mots de passe des utilisateurs sont robustes**

TÂCHE(S) A INTEGRER AU BACKLOG

- Pour chaque technologie déployée (OS, middleware), désactiver le compte par défaut et - si ce n'est pas possible - changer le mot de passe par défaut pour un mot de passe plus fort
- Définir une politique de mot de passe forte pour les utilisateurs applicatifs, avec 8 caractères minimum et un mélange de majuscules et minuscules, de caractères spéciaux et de chiffres. Le mot de passe doit être différent du login

RECTO

13

En tant qu'attaquant, je souhaite **recueillir des informations techniques** qui pourraient m'aider à mieux **comprendre le fonctionnement de l'application / du serveur**

EXEMPLES

- **Affichage des erreurs applicatives :** en générant des erreurs dans l'application (envoi d'un type de données incorrect dans un champ, par exemple), l'attaquant peut être en mesure d'obtenir la trace détaillée de l'erreur
- **Affichage des erreurs du serveur :** en générant des erreurs côté serveur (en demandant une page qui n'existe pas par exemple), l'attaquant peut obtenir la version du serveur web

S'APPLIQUE À

- Toutes les User Stories qui impliquent la création de nouvelles pages d'erreur (applicative et serveur)

VERSO

En tant qu'opérateur serveur/développeur, je veux m'assurer que **tous les messages d'erreur sont génériques**

TÂCHE(S) A INTEGRER AU BACKLOG

/ Côté application, définir les pages d'erreurs génériques à afficher lorsque l'application rencontre un comportement inconnu. Le détail de l'erreur (stacktrace) ne doit pas être affiché sur le navigateur de l'utilisateur (ou dans le code source)

/ Côté serveur, activez les messages d'erreur standard 404, 500, etc., sans afficher la version du serveur web. De plus, dans les en-têtes HTTP, n'affichez pas les versions des composants.

/ **03**

Exemple d’application à développer : WaveCare

Cas d’usage : application WaveCare

Consultez les disponibilités de praticiens près de chez vous

Transmettez en temps réel vos données de santé grâce à votre montre connectée

Réalisez la consultation à distance en Visio (conférence Skype)

Recevez votre ordonnance après le RDV en format dématérialisé

Schéma d'architecture simplifié de la solution

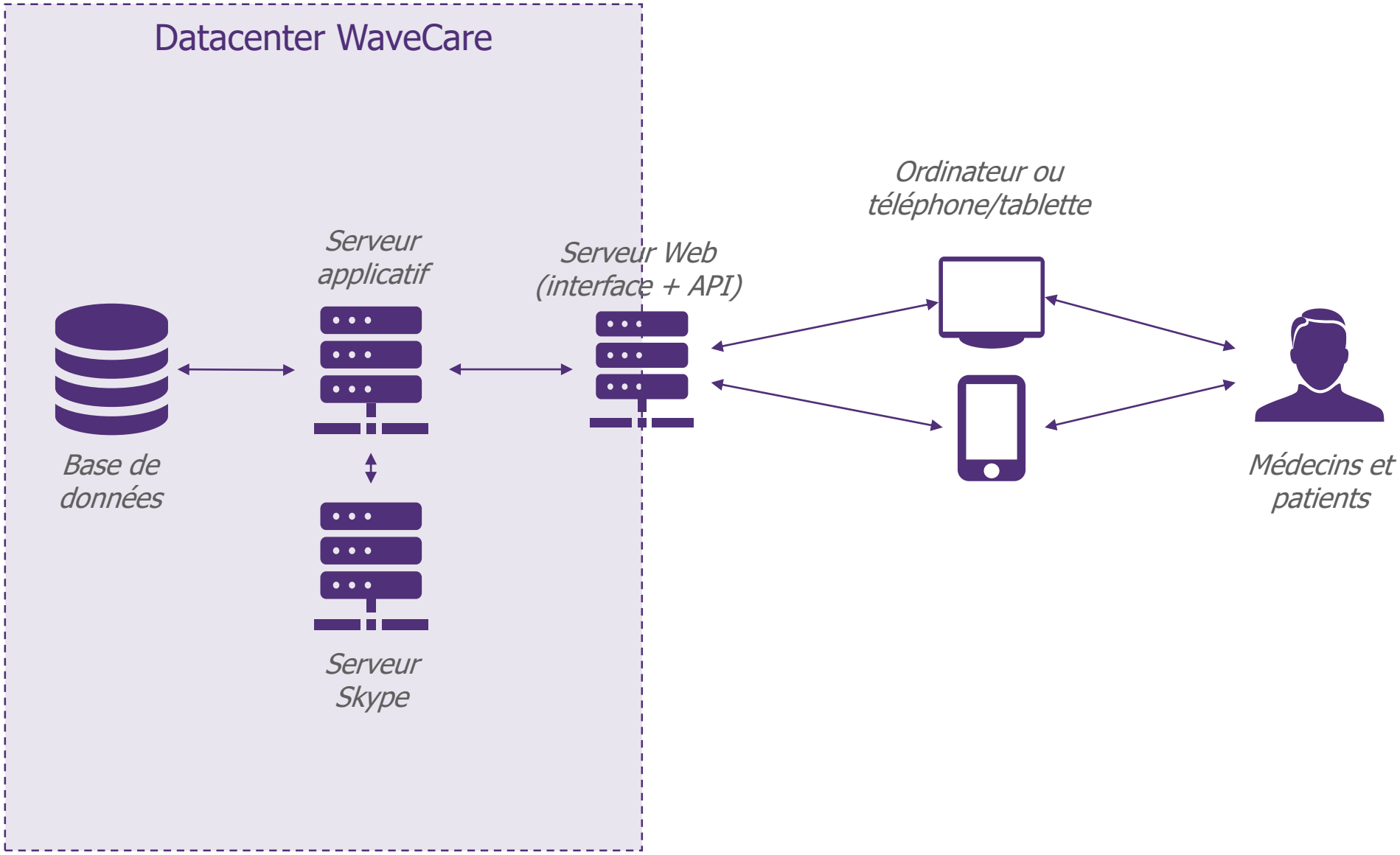
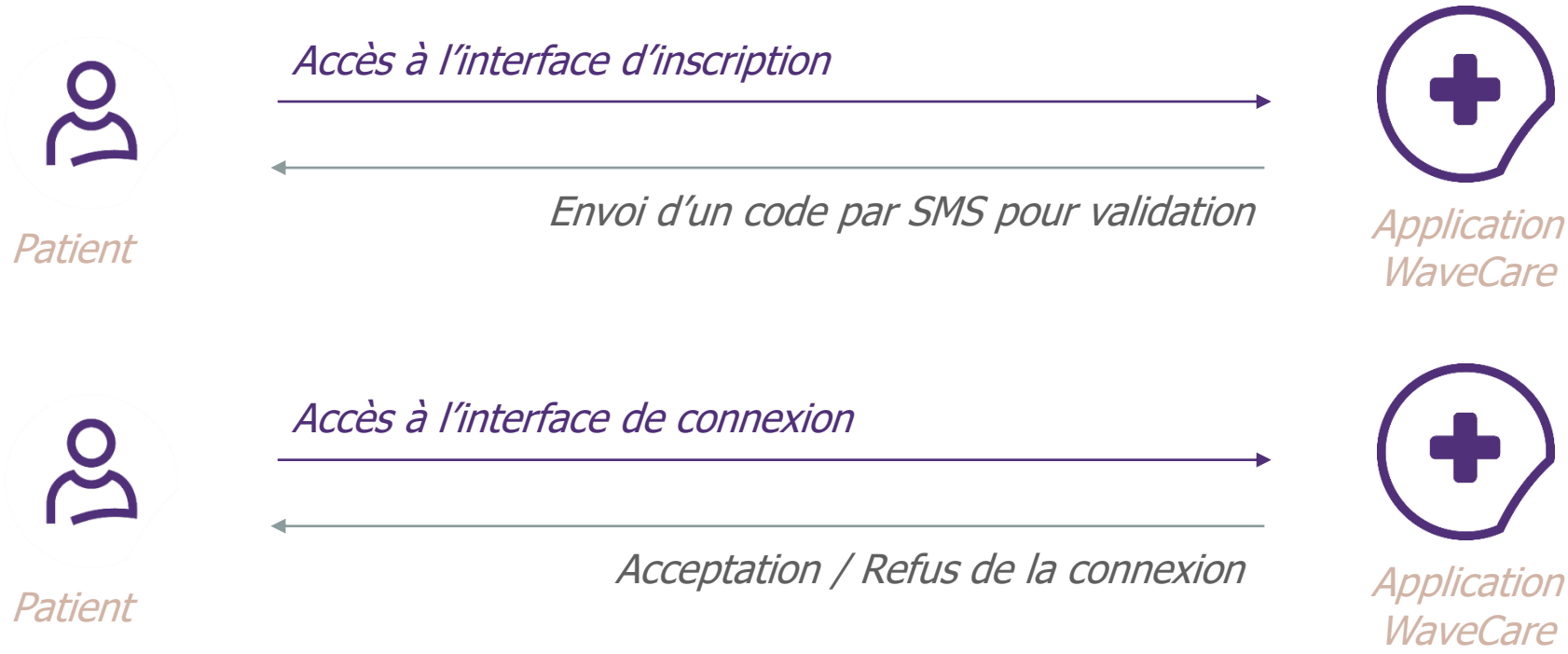


Schéma d’architecture : cartes qui s’appliquent

Carte	Explication
3 : brute force	Pour l’API notamment
7 : flux non chiffrés	Se configure globalement au niveau du serveur Web (forcer HTTPS partout, à l’origine, c’est plus simple que de le faire par petit bout)
8 : services exposés	Limiter au maximum les services exposés sur le serveur Web. N’avoir aucun service exposé sur les autres serveurs.
9 : mise à jour des composants	Surveiller les vulnérabilités qui apparaissent sur l’ensemble des composants : base de données, serveur applicatif, serveur web, version de Skype, etc.
10 : DDoS	Penser, si cela paraît utile, à mettre des équipements anti DDOS en Frontal d’Internet

Fonctionnalité 1 : inscription et connexion d'un patient



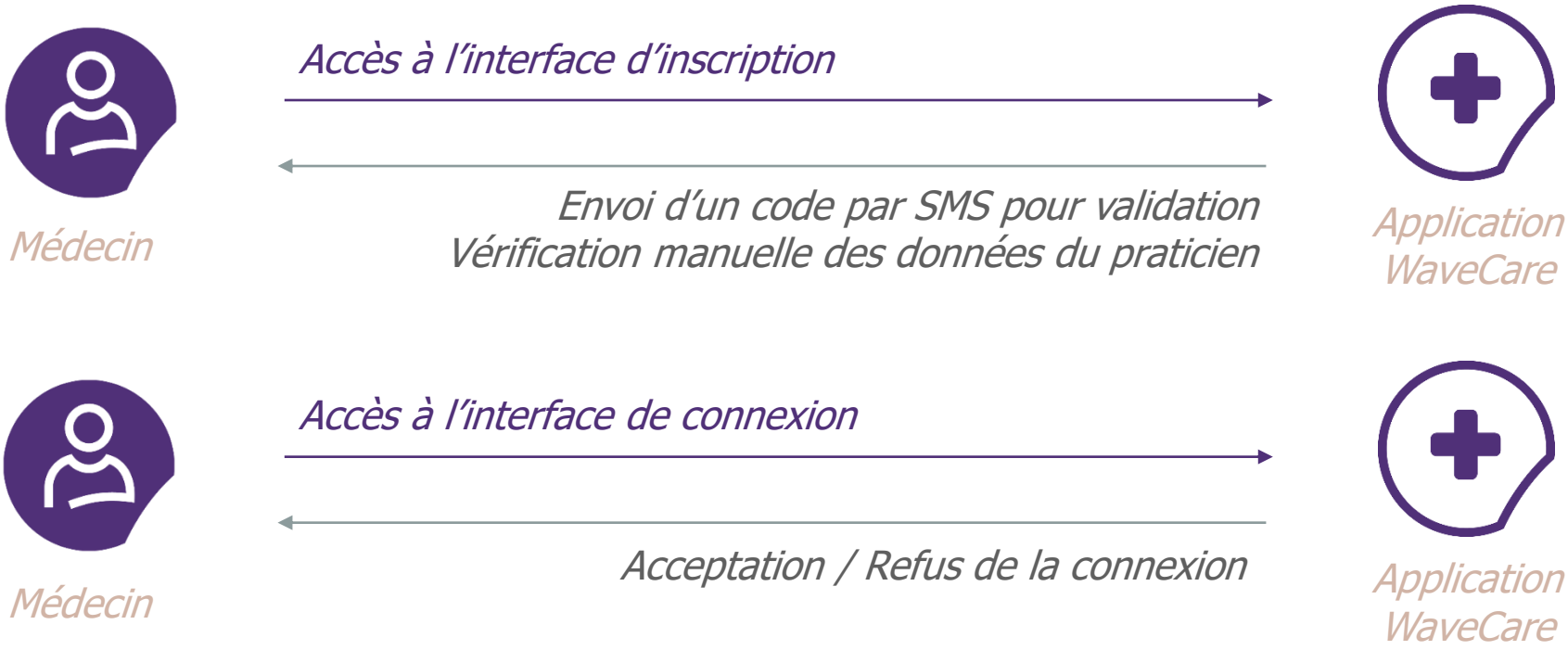
Données traitées par les fonctionnalités

Inscription	Connexion
- Nom, Prénom - Date de Naissance - Adresse - Adresse mail - Mot de passe - Numéro de téléphone portable - Numéro de sécurité Sociale	- Adresse mail d'inscription ou numéro de téléphone - Mot de passe

Fonctionnalité 1 : cartes qui s'appliquent

Carte	Explication
1 : injection de code	Tous les champs du formulaire d'inscription peuvent être sujets à de l'injection de code. De même pour les champs de login.
3 : brute force	Interface de login = anti-bruteforce nécessaire
4 : vol de cookie	Lorsque connexion = création du cookie, donc applicable
7 : flux non chiffrés	Les flux de connexion / d'inscription (qui contiennent le mot de passe) doivent être absolument chiffrés!
12 : Mot de passe / comptes par défaut	Définition du mot de passe lors de l'inscription : complexité du mot de passe demandé !
13 : Messages d'erreur verbeux	Message d'erreur sur les interfaces non authentifiées (inscription) = c'est là que quelqu'un sans compte va chercher des infos

Fonctionnalité 2 : inscription et connexion d'un médecin



Données traitées par les fonctionnalités

Inscription	Connexion
- Nom, Prénom - Date de Naissance - Adresse - Adresse mail - Mot de passe - Numéro de téléphone portable - Numéro de Praticien - Adresse du cabinet et numéro de téléphone du cabinet - Spécialisation de médecine proposée - Pour chaque spécialité, copie du diplôme en pièce jointe	- Adresse mail d'inscription ou numéro de téléphone - Numéro de praticien - Mot de passe

Fonctionnalité 2 : cartes qui s'appliquent

Carte	Explication
1 : injection de code	Tous les champs du formulaire d'inscription peuvent être sujets à de l'injection de code. De même pour les champs de login.
3 : brute force	Interface de login = anti-bruteforce nécessaire Pour les médecins c'est plus sensible, on pourrait aussi penser à mettre de l'authentification forte (car ils peuvent éditer des prescriptions notamment)
4 : vol de cookie	Lorsque connexion = création du cookie, donc applicable
6 : injection de fichier	Pour les médecins, on dépose une copie du diplômé d'état, donc on a ça en plus ;)
7 : flux non chiffrés	Les flux de connexion / d'inscription (qui contiennent le mot de passe) doivent être absolument chiffrés!
11 : Path traversal	Pour les médecins il y a upload de fichier, donc consultation du fichier uploadé. Donc il faut faire attention à ça aussi.
12 : Mot de passe / comptes par défaut	Définition du mot de passe lors de l'inscription : complexité du mot de passe demandé !
13 : Messages d'erreur verbeux	Message d'erreur sur les interfaces non authentifiées (inscription) = c'est là que quelqu'un sans compte va chercher des infos

Fonctionnalité 3 : ajout de créneaux de disponibilités d'un médecin

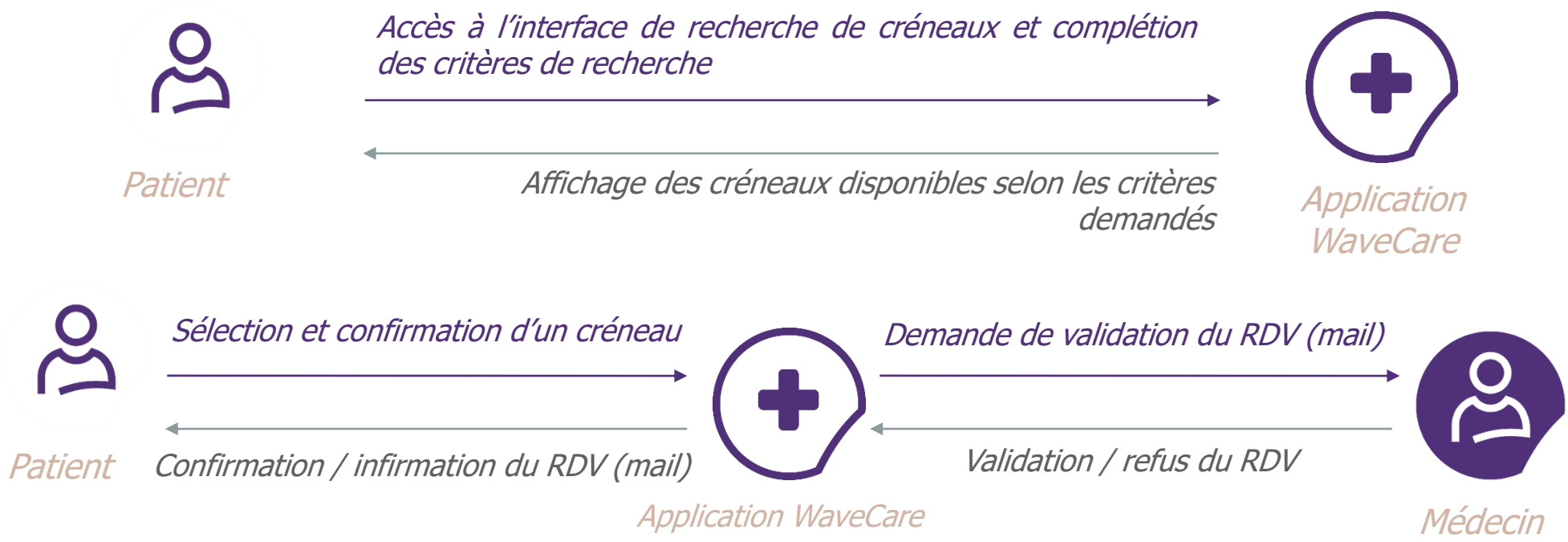
Données traitées par les fonctionnalités

Ajout de créneaux
- Date et heures des créneaux - Pour chaque créneaux, spécialités proposée (parmi celles pour lesquelles le médecin a fourni un diplôme - Tarif de la consultation

Fonctionnalité 3 : cartes qui s’appliquent

Carte	Explication
1 : injection de code	Les champs d’ajouts de créneau doivent être sécurisés
2 : escalade de privilèges	Il ne faut pas qu’un médecin puisse créer des créneaux pour un autre médecin, ou qu’il puisse créer des créneaux pour une spécialité qu’il n’a pas prouvée (avec un diplôme d’état)
5 : CSRF (rejeu de requêtes)	Idem, il ne faut pas que les requêtes soient prédictibles, ce qui pourrait permettre de piéger un médecin et de lui faire créer des créneaux à son insu.
13 : Messages d’erreur verbeux	Messages d’erreurs de la fonction de création de créneaux à prendre en compte

Fonctionnalité 4 : recherche et réservation d'un créneau par un patient



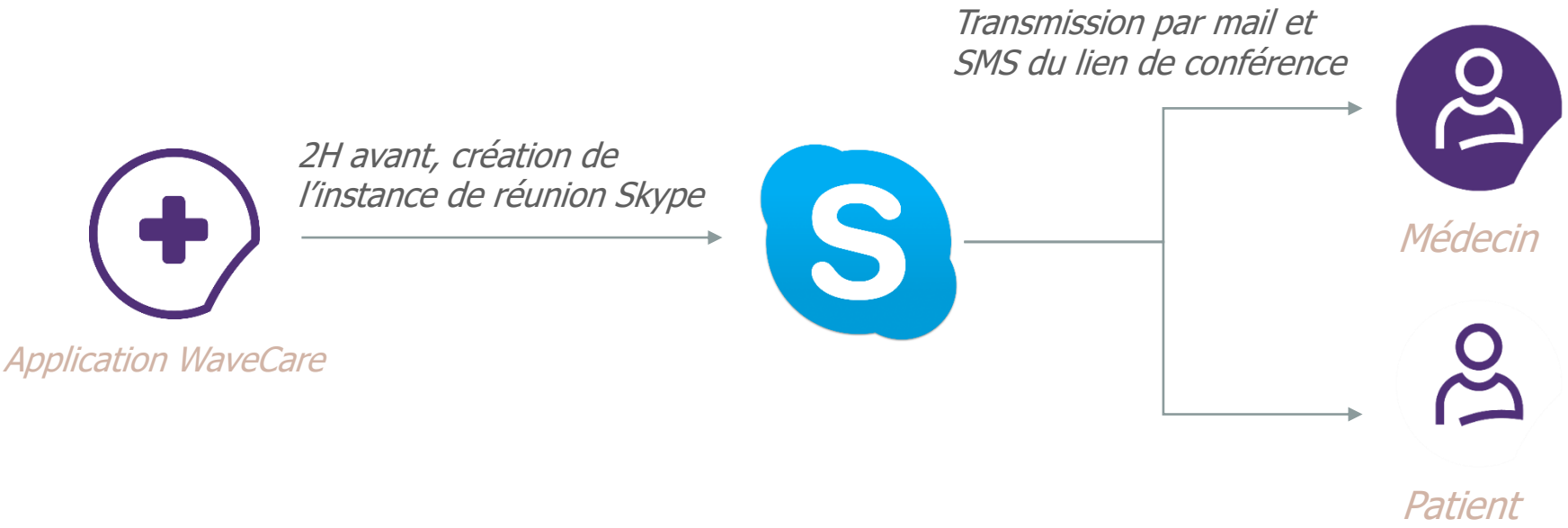
Données traitées par les fonctionnalités

Recherche	Réservation	Validation du médecin	Confirmation / Infirmation
- Type de spécialité recherchée - Date recherchée - Zone géographique de recherche (code postal ou département) - Type de convention (secteur 1, secteur 2, honoraires libres)	- N/A	- Mail récapitulant le créneau & spécialité demandée - Lien d'acceptation ou de refus dans le mail (URL)	- Si confirmation, récapitulatif des informations du RDV (date, spécialité, médecin, etc.) - Si infirmation, N/A

Fonctionnalité 4 : cartes qui s’appliquent

Carte	Explication
1 : injection de code	Les champs de recherche de créneau doivent être protégés
2 : escalade de privilèges	Il ne faut pas qu’un utilisateur puisse réserver un créneau à la place d’un autre. De même, il ne faut pas qu’un médecin puisse valider une demande à la place d’un autre (le lien envoyé par mail doit avoir un token aléatoire & non itérable)
5 : CSRF (rejeu de requêtes)	Idem, il ne faut pas que les requêtes soient prédictibles, ce qui pourrait permettre de piéger un utilisateur ou un médecin
13 : Messages d’erreur verbeux	Messages d’erreurs de la fonction de création de créneaux à prendre en compte

Fonctionnalité 5 : le jour J, réalisation de RDV à distance



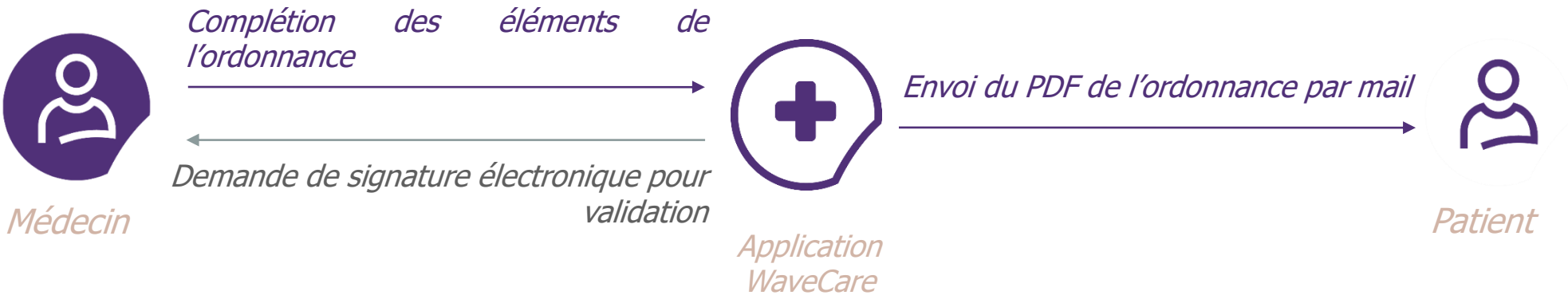
Données traitées par les fonctionnalités

Création de conférence Skype
- URL de la conférence Skype dédiée

Fonctionnalité 5 : cartes qui s’appliquent

Carte	Explication
1 : injection de code	Les champs de paramètre de la réunion Skype doivent être protégés
2 : escalade de privilèges	Il ne faut pas qu’il soit possible de se connecter à la réunion Skype d’un autre patient
7 : chiffrement	S’assurer que la conférence Skype (cas un peu spécifique) transite bien sur un flux chiffré.
10: DDoS	Côté infra mais c’est LE point le plus sensible de l’application : rendre une consultation prévue indisponible.

Fonctionnalité 6 : envoi d'une ordonnance post-RDV



Données traitées par les fonctionnalités

Création de l'ordonnance	Signature de l'ordonnance	Envoi de l'ordonnance
- Type de médicament prescrit - Indication de posologie	- Signature du médecin (suite à validation double facteur)	- Document PDF envoyé par mail

Fonctionnalité 5 : cartes qui s’appliquent

Carte	Explication
1 : injection de code	Les champs utilisés pour remplir l’ordonnance doivent être sécurisés
2 : escalade de privilèges	Il ne faut pas qu’un médecin ou qu’un patient puisse remplir une ordonnance pour un autre médecin
3: vol de mot de passe	Pour protéger cette fonction ultra-sensible, opter pour de la double authentification
5 : CSRF (rejeu de requêtes)	Il ne faut pas que les requêtes soient prédictibles, ce qui pourrait permettre de piéger un médecin et de lui faire créer des ordonnances à son insu.
11 : Path traversal	Il y a récupération d’un fichier dans l’application

PARIS

LONDON

NEW YORK

HONG KONG

SINGAPORE *

DUBAI *

SAO PAULO *

LUXEMBOURG

MADRID *

MILANO *

BRUSSELS

GENEVA

CASABLANCA

ISTANBUL *

LYON

MARSEILLE

NANTES

* Partnerships

WAVESTONE