WAVESTONE

Security & Agility

Card game

Cards explanation

The game is played with cards, each with 2 sides:

RECTO

/ **Evil User Stories** describe in a very educational way what can go wrong in the scope of study, using which vulnerabilities (e.g. elevation of privileges on a Web server, brute force attack, XSS,…).

VERSO

/ **Security User Stories** describe the security measures to be implemented to ensure that the Evil User Story does not occur (e.g. use of a strong AES 256/512 encryption algorithm, ...).

How to play ?

/ It is necessary to **gather at least the members of the Squad with a functional knowledge** of the solution (Product Owner) and a technical knowledge and risks (**security advisers, developers, architects**).
/ The architects **draw the application architecture** on a large A3 poster on a table by showing the data flows and the data classification and the Product Owner lists the next User Stories that will have to be developed.
/ The Security Champion and the developers **distribute the exploitable vulnerabilities** on the architecture diagram.
/ The Product Owner and the Security Champion **qualify the impact** that each vulnerability can have.
/ The Security Champion and the developers check whether the security measures to counter the identified vulnerabilities are already implemented.
/ If security measures are not yet in production: the Security Champion **prioritizes the technical measures to be implemented** to cover the induced risks (risk for the company, not just at the business level).
/ The Product Owner **prioritizes other security measures with regard to business risks and the means of the team**.

example

RECTO

1

As an attacker, I want to **inject malicious code in the application** using unsecured fields

EXAMPLE

- **XSS (Cross-site scripting) injections,** that may allow an attacker to prompt malicious messages to other users to trick them, or steal their session cookie if not sufficiently secured
- **SQL injections, that may allow** an attacker to interact directly with the database and access / modify data without control

APPLIES TO

- User Stories that involve the creation of a new input/user search field, or a new parameter sent within an HTTP request

VERSO

As a developer, I want to ensure **that code injection attacks** are **prevented**

BACKLOG TASK(S)

- Before inserting user input in a request – or display the data back to the user – always encode special characters, using *htmlspecialchars* for instance, or one of the many available libraries that prevent code injection. Else, create and enforce a single way of filtering input for code injection.
- Always use cookies (authentication/session) with HttpOnly attribute to ensure they cannot be stolen by Javascript code

1

As an attacker, I want to **inject malicious code in the application** using unsecured fields

EXAMPLE

/ **XSS (Cross-site scripting) injections,** that may allow an attacker to prompt malicious messages to other users to trick them, or steal their session cookie if not sufficiently secured

/ **SQL injections, that may allow** an attacker to interact directly with the database and access / modify data without control

APPLIES TO

/ User Stories that involve the creation of a new input/user search field, or a new parameter sent within an HTTP request

As a developer, I want to ensure **that code injection attacks** are **prevented**

BACKLOG TASK(S)

/ Before inserting user input in a request – or display the data back to the user – always encode special characters, using *htmlspecialchars* for instance, or one of the many available libraries that prevent code injection. Else, create and enforce a single way of filtering input for code injection.

/ Always use cookies (authentication/session) with HttpOnly attribute to ensure they cannot be stolen by Javascript code

2

As an attacker, I want to perform **applicative privilege escalation** to access functionalities and/or data I'm not allowed to see

EXAMPLE

/ **Vertical privilege escalation:** gaining access to specific functionalities such as administrative parameter, or write rights when using a read-only account

/ **Horizontal privilege escalation:** gaining access to data that are not in my account's perimeter (other users' data for instance)

APPLIES TO

/ User Stories that involve the creation of a new functionality, or a new interface used to access data compartmented by perimeter

As a developer, I want to ensure that users can **only access functionalities / data that fit their privileges**

BACKLOG TASK(S)

/ Manage the privileges server-side based on session cookie, and systematically check the cookie for each request sent to the server. If the user the cookie belong to does not have the privileges to access a functionality / data set, deny the access and kill the session (invalidate the cookie)

/ When using content ID in an URL (downloaded file, item ID, etc.), avoid using a numeric identifier that can be enumerated

3

As an attacker, I want to **steal another user's credentials** (login and password)

EXAMPLE

- **Online brute force** may allow an attacker to try to guess numerous times the password of another user and, if not blocked, can ultimately success
- **Spear phishing** is widely used these days by attacker to steal credentials from targeted users, with fine-crafted emails

APPLIES TO

- User Stories that involve the creation of a new authentication interface

As a developer, I want to ensure that online **brute force cannot be performed**, and that sensitive accounts have a **strong authentication mean set in place**

BACKLOG TASK(S)

- Block user accounts following numerous connection attempts. For instance, 3, 5 or 10 attempts based on the application sensitivity. The blockage can be based on the user ID, or on the IP address. Additionally, display a generic message when the connection fails, so the attacker can't know if a login is valid or not
- For sensitive accounts, set a strong authentication mean (two-factor authentication). It will prevent an attacker that performed successful spear-phishing to gain access to the application

As a developer, I want to ensure that **session cookies are sufficiently secured**

BACKLOG TASK(S)

- Systematically set security parameters for every cookie created : *httpOnly* and *secure.* The first one will prevent client-side code to access the cookie, and the second will prevent the cookie to be sent across an unencrypted channel
- Revoke the session cookie when the user disconnect from the application, or when his session has timed-out

4

As an attacker, I want to **usurp another user session**

EXAMPLE

- **Session cookie theft:** if an attacker is able to steal the cookie of another user, he would be able to connect with the user's account (without knowing his password) and perform action as the user, without any mean of tracking it back to the attacker

APPLIES TO

- User Stories that involve the creation of a new authentication interface / a new session cookie

5

As an attacker, I want to **trick another user** and make him perform an action **without knowing**

EXAMPLE

/ **CSRF attacks (Cross Site Request Forgery):** if the application modification requests are predictable (without any random parameter), an attacker could send a forged request to another user, that will execute it with his session cookie (hence his privileges) without knowing

APPLIES TO

/ User Stories that involve the creation of a new modification functionality

As a developer, I want to ensure that **applicative requests are not predictable**

BACKLOG TASK(S)

/ Systematically add a random parameter for every modification request, that cannot be guessed. The server should check the value of this random parameter before accepting any modification request

/ The random parameter (or token) could be generated by the server based on the user ID, the timestamp, the form ID, and a salt known only by the server

As a developer, I want to ensure **that all uploaded files are sanitized**

BACKLOG TASK(S)

- Define a white-list of file extensions that are allowed, and deny all other file type
- Run a signature-based check on the file before uploading to the server
- For XML files, disable XML external entities processing and DTD for all XML engines, to prevent XXE (XML eXternal Entities) attacks

6

As an attacker, I want to **gain control over the underlying server by sending an infected file**

EXAMPLE

- **File injection:** if the file type is not properly checked, of if the file does not run through an antivirus tool before being uploaded on the server, an attacker could send an infected file that may allow him to gain direct access to the server with high privileges (for instance, a reverse-shell)

APPLIES TO

- User Stories that involve the creation of a file-upload functionality

7

As an attacker, I want to **steal unsecured data in transit**

EXAMPLE

/ **Unencrypted flows interception:** if the applicative flow are not encrypted using an SSL certificate (HTTPS), anyone on the same subnetwork than the user could read in cleartext all information transmitted (password, business data, etc.). If the SSL certificate support obsolete algorithms (such as SSLv3), the result could be the same

APPLIES TO

/ All User Stories

As an architect / server operator, I want to ensure that **all communications run through an encrypted channel**

BACKLOG TASK(S)

/ Allow only encrypted communication with the web server, by deploying an SSL certificate and activating the HSTS option

/ Redirect port 80 to 443

/ Regularly check encryption algorithms supported by the server, to ensure that they are all up-to-date according the best security practice (no SSLv2/v3 and TLSv1.0, no 3DES, etc.)

As an architect / server operator, I want to ensure that **only necessary services (ports) are exposed to the user**

BACKLOG TASK(S)

- Ensure that port exposure is limited to what the user needs to see (for instance, only port 80 and 443 for a web-server)
- Design the internal network to ensure that databases & applicative servers are placed on dedicated network zones, which standard users can't access
- Administrative interfaces such as RDP and SSH should be limited to a dedicated administrative network.

8

As an attacker, I want to **gain control over the underlying server by using an over-exposed service**

EXAMPLE

- **Unnecessary service exposure:** if the web server presents services (ports) that are not necessary for standard users, such as administrative services, an attacker could exploit them to compromise the server

APPLIES TO

- User Stories that involve network flows opening or a change in the application exposure (intranet / internet)

9

As an attacker, I want to
compromise the application by exploiting a known vulnerability

EXAMPLE

/ **Outdated component:** if the components are not up-to-date (OS, middleware, library), they may present public vulnerabilities. Such vulnerabilities often have public exploitation script, that can be used by script-kiddies without advanced technical knowledge

APPLIES TO

/ All User Stories that involve the deployment of a new component – but this must be follow-up overtime and not just once

As a server operator, I want to ensure that
all components are up-to-date with the latest security patches

BACKLOG TASK(S)

/ Create and maintain an exhaustive list of all technologies used by the application, and their versions

/ Subscribe to an alerting service for those technologies, that will alert you if new vulnerabilities are published

As an architect / developer, I want to ensure that **my application is protected against denial of service (at network & code level)**

BACKLOG TASK(S)

- Depending on the Availability needs for the application, ensure that network-layer and application-layer anti-DDOS measures are in place
- For instance, at network-level, a service provider can prevent such attacks. At application-layer, mitigation measures such as Captcha or Web Application Firewall can reduce the impact of the attack

10

As an attacker, I want to **perform a denial of service to prevent legitimate users from accessing the application**

EXAMPLE

- **Denial of Service (DOS) or Distributed DOS:** an attacker that would want to impact the availability of the application could craft numerous demanding requests, that would overload the processing capacity of the application. This way, legitimate users would not be able to access the application

APPLIES TO

- User Stories that involve Internet exposition change

11

As an attacker, I want to **access technical server files through the application**

EXAMPLE

/ **Path Traversal:** by manipulating variables that reference files with "../.." sequences and its variations or by using absolute file paths, it may be possible to access arbitrary files and directories stored on file system. Ultimately, the attacker could access files and directories that are stored outside the web root folder

APPLIES TO

/ All User Stories that involve the access to files on the server

As a developer, I want to ensure that **technical files out of the web server directory cannot be accessed by the application**

BACKLOG TASK(S)

/ Prefer working without user input when using file system calls

/ Ensure the user cannot supply all parts of the path – surround it with your path code

/ Validate the user's input by only accepting known good – do not sanitize the data

As a server operator / developer, I want to **ensure that default account are disabled and user passwords are strong**

BACKLOG TASK(S)

- For each technologies deployed (OS, middleware), disable default account and – if not possible – change the default password for a stronger one
- Define a strong password policy for applicative users, with 8 characters minimum and a mix of upper/lower case, special char, numbers. The password must be different from the login

12

As an attacker, I want to **gain access to the application or the components using a default or guessable account**

EXAMPLE

- **Default technical account:** on most components, default technical account with predefined password are activated at the installation. If not changed, an attacker could get access using one of them
- **Low password policy:** on the application, if users can set low passwords (password = login), the attacker could get access guessing one of them

APPLIES TO

- User Stories that involve installing a new component or defining a user password policy

13

As an attacker, I want to **gather technical information** that could help me to better **understand how the application / server works**

EXAMPLE

- **Applicative error display:** by generating errors within the application (sending incorrect type of data in a field, for instance), the attacker may be able to get the stack trace of the error
- **Server error display:** by generating errors server-side (asking a page that does not exist for instance), the attacker can get the webserver version

APPLIES TO

- All User Stories that involve the creation of new error pages (applicative & server)

As a server operator /developer, I want to ensure that **all error messages are generic**

BACKLOG TASK(S)

- Application side, define generic errors pages to be displayed when the application encounter an unknown behaviour. The stacktrace should not be prompted on the users' browser (or in the source code)
- Server side, activate the standard 404, 500, etc. error messages, without displaying the webserver version. Additionally, in HTTP headers, does not display the components versions.

WAVESTONE

Security & Agility

Example – developing application: WaveCare

Use case: WaveCare app

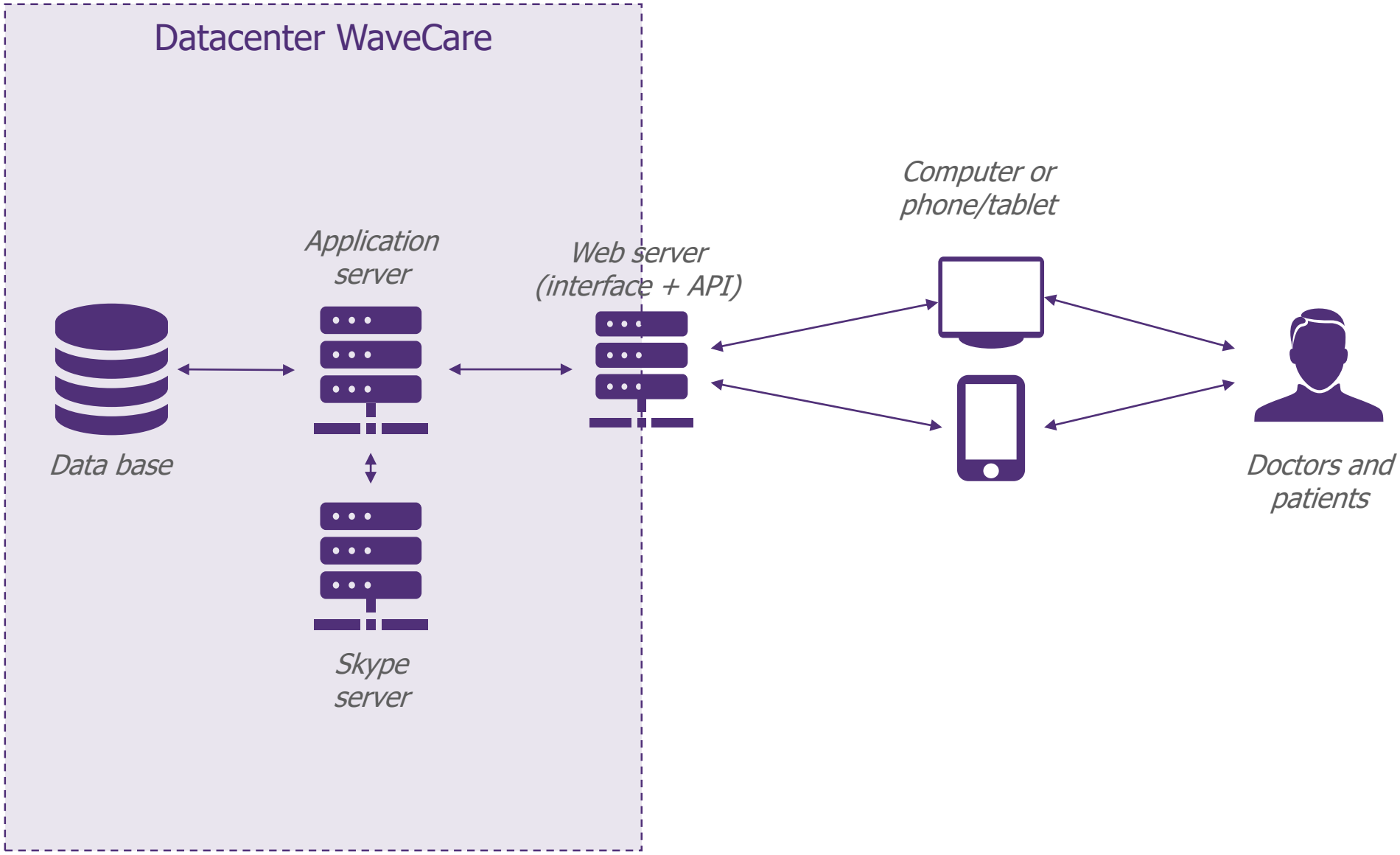
Check practitioners availability near you

Real time health data transfer thanks to your connected watch

Remote consultation by Visio (Skype conference)

Prescription reception after the appointment in dematerialized format

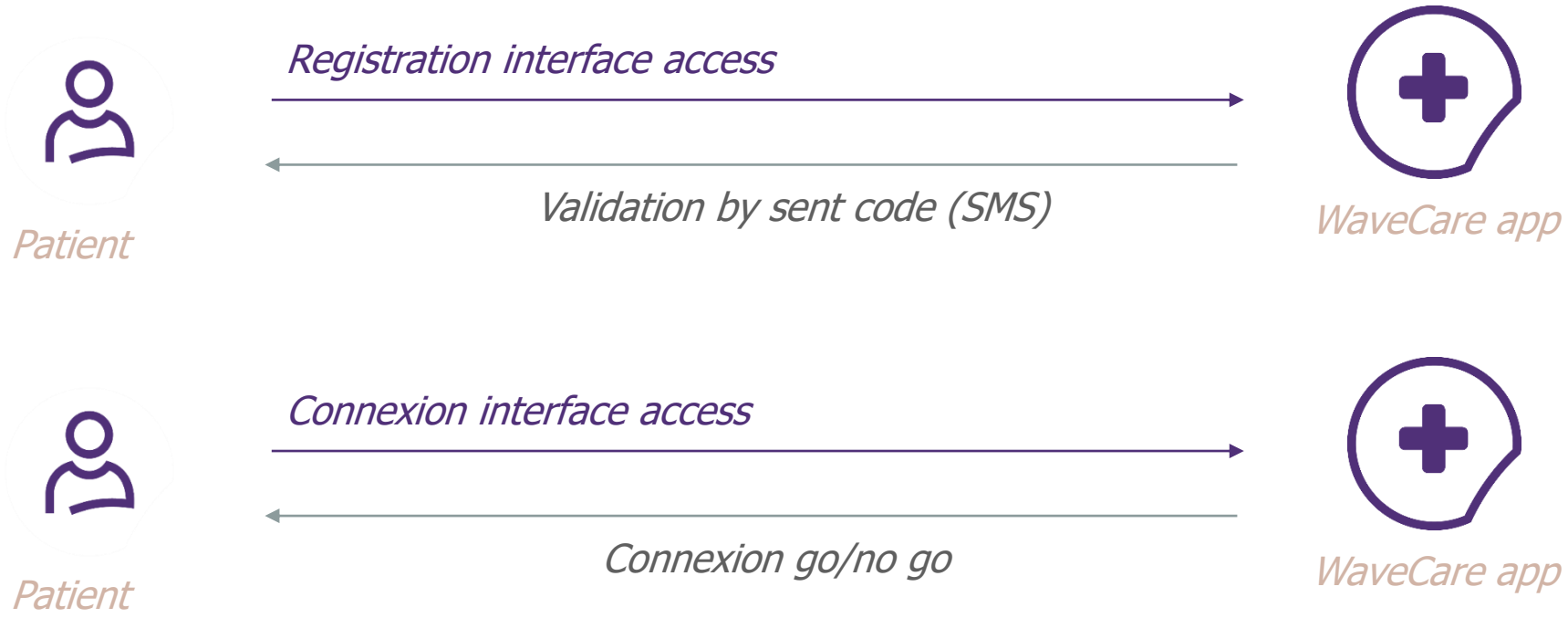
Architecture simplified diagram



Architecture diagram: solution

Card	Explanation
3 : brute force	For the API
7 : unencrypted flows	Can be configured globally at Web server level (forcing HTTPS everywhere is easier than doing it in small increments)
8 : exposed services	Minimize the web server exposed services. No services should be exposed on other servers.
9 : components updates	Monitor components vulnerabilities: database, application server, web server, Skype version, etc.
10 : DDoS	If useful, consider putting anti DDOS equipment on the Internet Front server

Feature 1 : patient registration and connexion



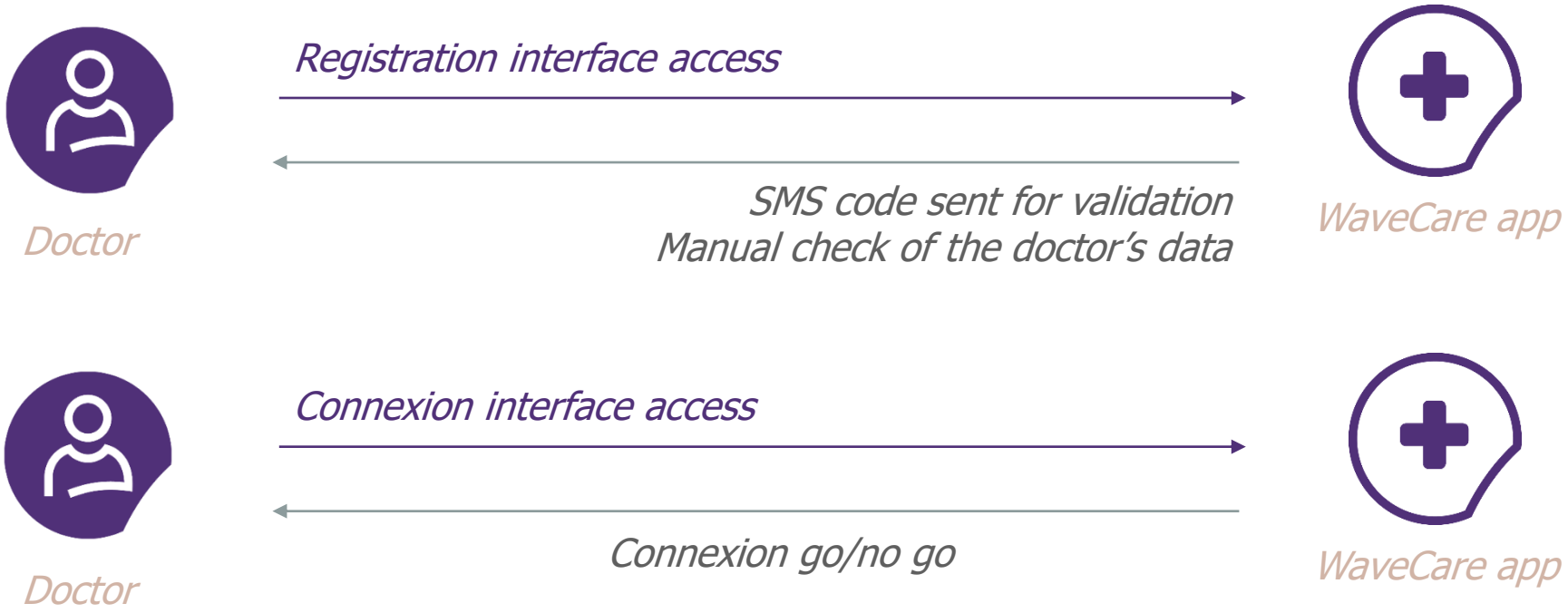
Data used by the feature

Inscription	Connexion
- Last name, First name - Birthdate - Postal address - Email address - Password - Phone number - Social security number	- Email address or phone number used for sign-up - Password

Feature 1 : solution

Card	Explanation
1 : code injection	All registration form fields may be facing code injections. The same goes for sign-up fields.
3 : brute force	Login interface = mandatory anti brute force
4 : cookie theft	When connexion is made, then a cookie is created. Anti cookie theft must be implemented
7 : unencrypted flows	Connexion or registration flows must be encrypted (they can show passwords)
12 : By default password/accounts	Ask for a complex password at sign-up / changing the default password is mandatory
13 : wordy error messages	Having wordy error messages on non authenticated interface is where someone without account can find useful information (to attack the application)

Feature 2 : registration and connection of a doctor



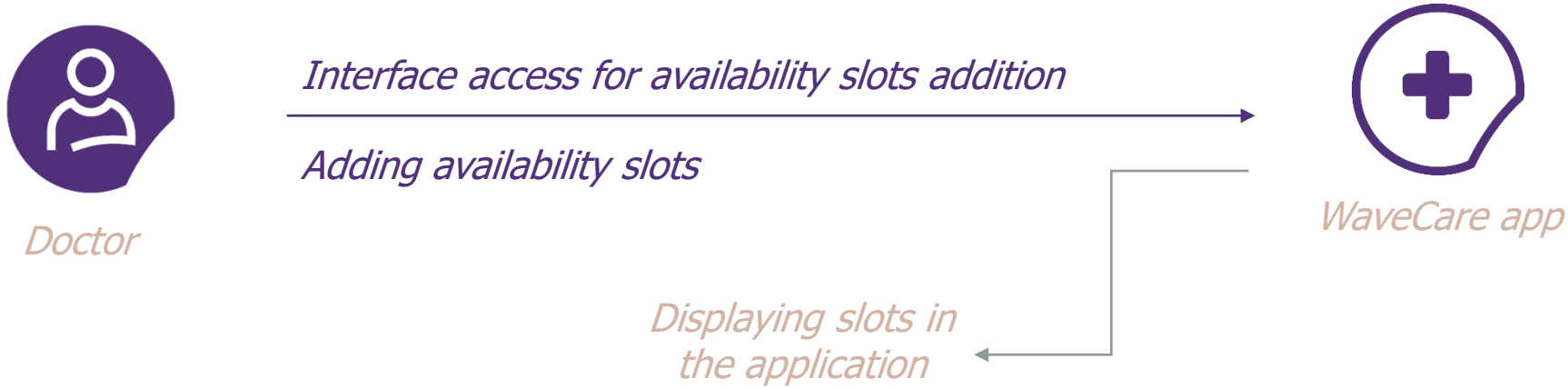
Data used by the feature

Registration	Connexion
- Last name, First name - Birthdate - Postal address - Email address - Password - Phone number - Doctor number - Practice address and practice telephone number - Specialization the Doctor offers - For each specialty, copy of the diploma in attachment	- Email address or phone number used for sign-up - Password - Doctor number

Feature 2 : solution

Card	Explanation
1 : code injection	All registration form fields may be facing code injections. The same goes for sign-up fields.
3 : brute force	Login interface = mandatory anti brute force It's even more sensitive for doctors, strong authentication could be implemented (if they can edit prescription for example)
4 : cookie theft	When connexion is made, then a cookie is created. Anti cookie theft must be implemented
6 : file injection	Copy of the diploma for doctors is to be taken into account
7 : unencrypted flows	Connexion or registration flows must be encrypted (they can show passwords)
11 : Path traversal	For doctors only, as they can upload and view files
12 : default password/account	Ask for a complex password at sign-up / changing the default password is mandatory
13 : wordy error messages	Having wordy error messages on non authenticated interface is where someone without account can find useful information (to attack the application)

Feature 3 : adding availability slots by a doctor



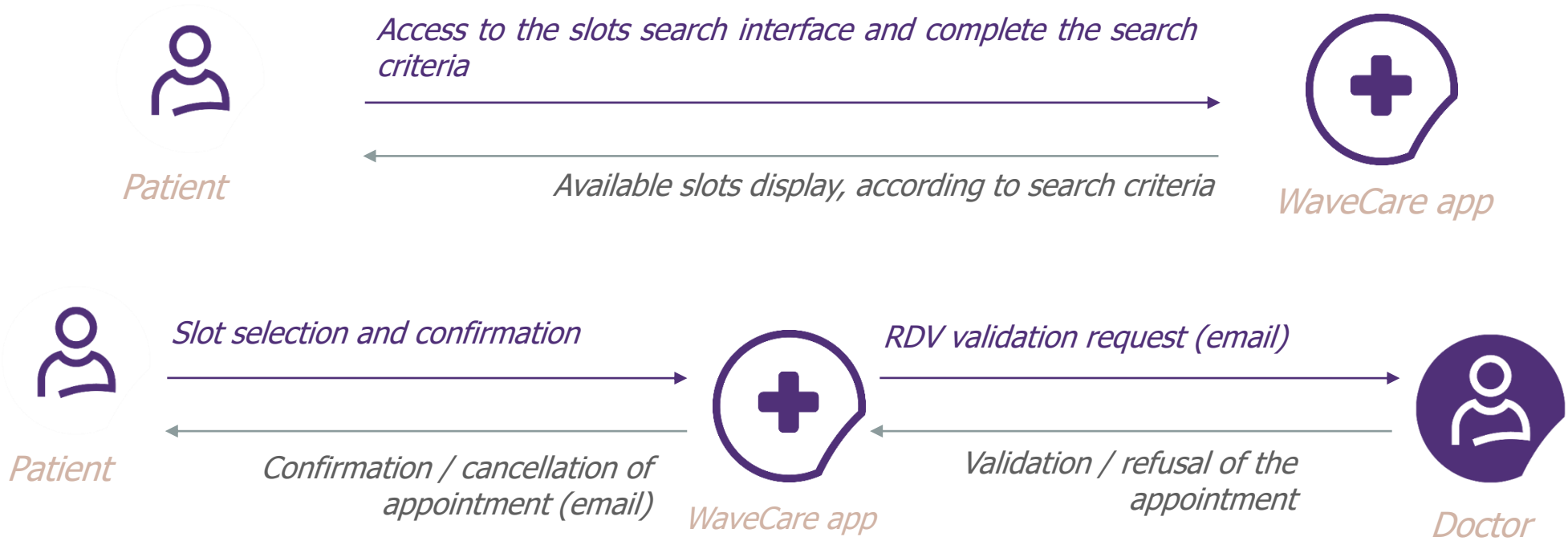
Data used by the feature

Availability slots addition
- Slots date and schedule - For each slots, indication of the doctor speciality (for which he/she attached a diploma) - Consultation rates

Feature 3 : solution

Card	Explanation
1 : code injection	Adding-slots fields must be secured
2 : escalation of privileges	It is not necessary that a Doctor can create slots for another Doctor, or that he/she can create slots for a specialty which he/she has not proof (with a state diploma)
5 : CSRF (request replay)	The requests should not be predictable, otherwise it could trap a Doctor and make him/her create slots without his/her knowledge.
13 : wordy error messages	Having wordy error messages on non authenticated interface is where someone without account can find useful information (to attack the application)

Feature 4 : search and reservation of a slot by a patient



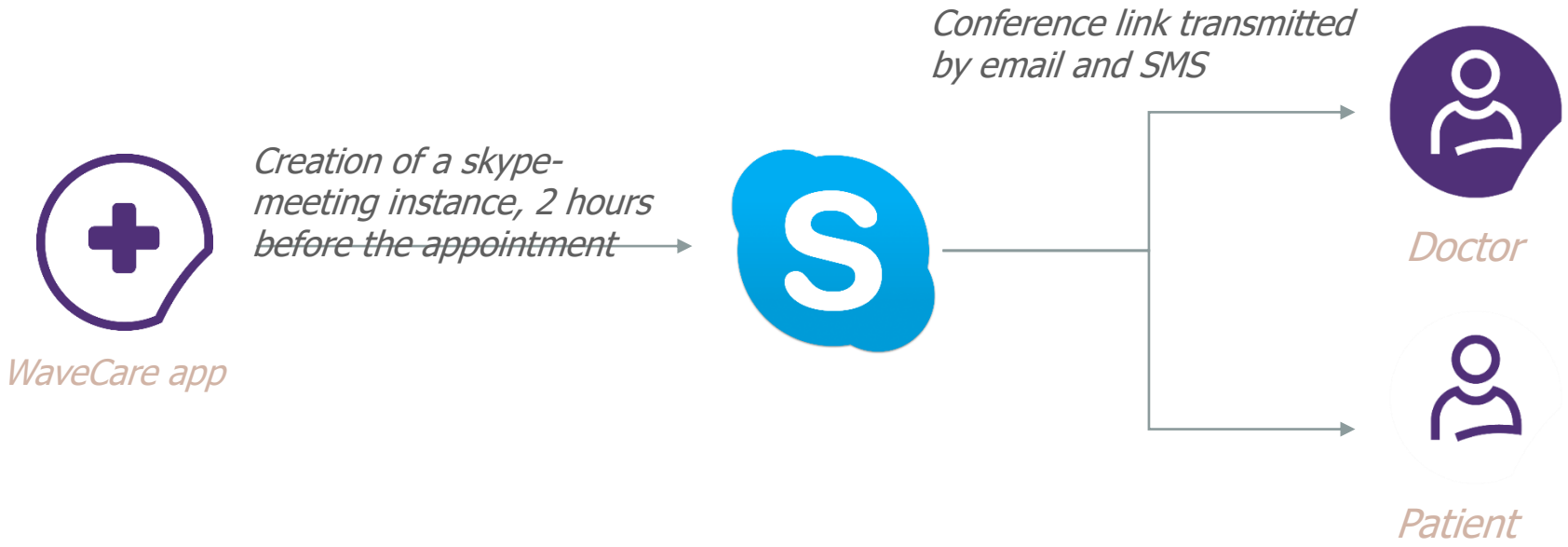
Data used by the feature

Search criteria	Booking	Doctor validation	Confirmation/cancellation
- Type of specialty sought - Date sought - Search geographic area (postal code or department) - Type of agreement (sector 1, sector 2, free fees)	- N/A	- Mail summarizing the slot & specialty requested - Acceptance or refusal link in the email (URL)	- If confirmation, summary of information for the appointment (date, specialty, Doctor, etc.) - If cancellation, N / A

Feature 4 : solution

Card	Explanation
1 : code injection	Slot search fields must be secured
2 : escalation of privileges	A user must not be able to book a slot pretending to be someone else. Likewise, a Doctor must not be able to validate a request for another doctor (the link sent by email must have a random & non-iterable token)
5 : CSRF (request replay)	The requests should not be predictable, otherwise it could trap a Doctor and make him/her create slots without his/her knowledge.
13 : wordy error messages	Having wordy error messages on non authenticated interface is where someone without account can find useful information (to attack the application)

Feature 5 : D-day, remote appointment



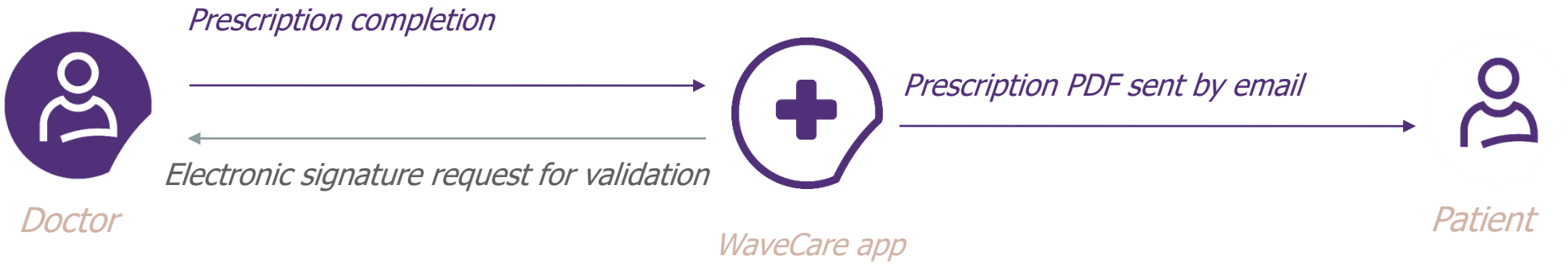
Data used by the feature

Skype-meeting creation
- Skype-meeting dedicated URL

Feature 5 : solution

Card	Explanation
1 : code injection	Configuration fields for the Skype meeting must be secured
2 : escalation of privileges	A patient can't join another skype meeting of another patient
7 : encryption	Be sure that the skype meeting flows are encrypted (if possible)
10: DDoS	On the infrastructure side. It could prevent patient and doctor from having the meeting (very sensitive point)

Feature 6 : sending a post-appointment prescription



Data used by the feature

Prescription creation	Prescription signature	Prescription sending
- Type of medication prescribed - Dosage indication	- Signature of doctor (following multi factor validation)	- PDF document sent by email

Feature 5 : solution

Card	Explanation
1 : code injection	Fields used for prescription completion must be secured
2 : escalation of privileges	No doctor or patient can fill a prescription for another doctor
3: password theft	To protect this very sensitive feature, opt for multi factor authentication
5 : CSRF (request replay)	The requests should not be predictable, otherwise it could trap a Doctor and make him/her create prescriptions without his/her knowledge.
11 : Path traversal	File retrieval in the application

PARIS

LONDON

NEW YORK

HONG KONG

SINGAPORE *

DUBAI *

SAO PAULO *

LUXEMBOURG

MADRID *

MILANO *

BRUSSELS

GENEVA

CASABLANCA

ISTANBUL *

LYON

MARSEILLE

NANTES

* Partnerships

WAVESTONE