

Rapport final

PER2024-050 - Flash audit automatisé pour la Cybersécurité Web : Conception d'un scanner de vulnérabilités proactif

Projet de développement

Amandine MARTIN

Maxime BILLY

Evan TRANVOUEZ

Mohamed BOUCHENGUOUR

Encadrant :

Christian DELETTRE

Karima BOUDAOUED

1. Introduction.....	4
2. Analyse de l'existant.....	5
2.1. Réglementation.....	5
2.1.1. Le RGPD : Règlement Général sur la Protection des Données.....	5
2.1.2. Directives NIS2 : sécurité des réseaux et des systèmes d'information.....	5
2.1.3. Cyber Resilience Act : Résilience des produits numériques.....	6
2.1.4. Digital Operation Resilience Act (DORA) : Résilience Opérationnelle Numérique.....	6
2.1.5. Solutions d'accompagnement.....	7
2.1.6. Synthèse.....	7
2.2. Référentiels.....	8
2.2.1. OWASP TOP 10.....	8
2.2.2. CWE top 25.....	9
2.2.3. Bilan sur les vulnérabilités.....	12
2.3. Détection des vulnérabilités.....	13
2.3.1. Scanners de vulnérabilités commerciaux.....	13
2.3.2. Scanners de vulnérabilités gratuits et open source.....	15
2.3.3. Outils de vulnérabilités gratuits et open source.....	16
2.3.4. Synthèse comparative.....	19
2.4. La Cybersécurité pour tous.....	20
2.4.1. Analyse du modèle SEO.....	20
2.4.2. Notre proposition : Weakspotter.....	23
2.5. Conclusion de l'analyse de l'existant.....	24
3. WeakSpotter.....	25
3.1. Cahier des Charges.....	25
3.2. Maquette et POC.....	26
3.3. Conception Logicielle.....	27
3.3.1. Architecture.....	27
3.3.2. Frontend.....	28
3.3.3. Backend.....	28
3.3.4. Microservice IA.....	33
3.3.5. Déploiement.....	36
3.4. Gestion de projet.....	37
3.4.1. Organisation du Travail.....	37
3.4.2. Licence.....	38
3.5. Résultats Obtenus.....	39
3.6. Problèmes rencontrés.....	45

3.6.1. Gestion de Projet.....	45
3.6.2. Technologies Utilisées.....	45
3.6.3. Microservice IA.....	45
3.7. Perspectives d'évolution.....	46
3.7.1. Frontend.....	46
3.7.2. Backend.....	46
3.7.3. Microservice IA.....	46
4. Conclusion.....	47
Annexes.....	48
Annexe 1 : Lexique.....	48
Annexe 2: Diagramme Logique.....	54
Annexe 3 : Maquette.....	56
Annexe 4: Preuve de Concept.....	61
Annexe 5: diagramme.....	63
Annexe 6: Interface graphique téléphone.....	64
Annexe 7: Outils open source.....	68
Annexe 8: Lien vers le projet.....	70
Bibliographie indicative.....	71
Scanners & Outils.....	71

1. Introduction

La sécurité des applications web constitue aujourd’hui un enjeu stratégique majeur pour les entreprises, qu’elles soient de grande envergure ou de petite et moyenne taille ([TPE/PME](#)). Les menaces se diversifient et se multiplient, incluant les [fuites de données](#), les [rançongiciels](#), les [défactements](#), ainsi que le vol de propriétés intellectuelles. En 2022, selon l'[ANSSI](#), 40 % des attaques par [rançongiciel](#) traitées ou rapportées en France ciblaient des TPE et PME. De plus, en 2023, Cybermalveillance.gouv.fr note une hausse globale des recherches d’assistance pour des attaques informatiques liées à des actes malveillants en ligne (piratage de compte (+26 %), [hameçonnage](#) (+21 %), [rançongiciel](#) (+17 %), etc.).

Les [cyberattaques](#) entraînent des répercussions lourdes sur leur victime, notamment pour les TPE et PME, souvent moins protégées, car l’exploitation des [vulnérabilités](#) a un impact direct sur ces structures. En moyenne, selon francenum.gouv.fr, une [cyberattaque](#) coûte 58 600 euros à une entreprise et entraîne une interruption du service informatique de 26 à 29 jours.

Des solutions existent déjà, mais celles-ci ne sont pas à la portée de ces entreprises. En effet, les solutions professionnelles de [test de pénétration](#) et de scan de vulnérabilités, bien que complètes et fiables, se révèlent généralement coûteuses, complexes à maîtriser et peu adaptées à un public non spécialiste. Les rendant donc non adaptées à ce public. De plus, des [outils gratuits](#) ou [open sources](#) existe également, pouvant être une solution peu coûteuse, mais celles-ci nécessitent fréquemment des compétences techniques avancées qui dépassent les capacités internes disponibles au sein des TPE/PME.

Par ailleurs, la législation européenne impose des exigences croissantes en cybersécurité, et ceci indépendamment de la taille des entreprises. Le RGPD (2018) exige des mesures de protection des données, tandis que la directive NIS2 (2022) élargit les obligations de cybersécurité à certaines PME et TPE opérant dans des secteurs critiques. Le Cyber Resilience Act et le Digital Operational Resilience Act obligent aussi des standards de sécurité sur les logiciels et infrastructures numériques. Face à ces obligations en matière de cybersécurité, les entreprises doivent adopter des solutions accessibles pour évaluer et corriger leurs vulnérabilités.

Ainsi, un écart significatif existe entre l’offre actuelle et les besoins réels des TPE et PME, lesquels souhaitent pouvoir effectuer un audit de sécurité à partir d’une simple URL, sans interventions techniques complexes et à un prix abordable.

C'est pour cela que la solution WeakSpotter, présentée par la suite, a été développée dans le but de permettre à ces entreprises de comprendre les enjeux de la cybersécurité et de pouvoir ainsi corriger les vulnérabilités qu'elles rencontrent sur leur site internet.

2. Analyse de l'existant

2.1. Réglementation

Les réglementations jouent un rôle clé pour protéger les systèmes et les données face aux cybermenaces croissantes. Ils obligent les entreprises à suivre des bonnes pratiques pour renforcer leur résilience et protéger les données personnelles.

Ces réglementations s'appliquent à toutes les entreprises, quelle que soit leur taille, et visent à garantir la protection des données, la sécurité des infrastructures numériques et la confiance entre les utilisateurs et les entreprises. Elles sont essentielles pour assurer la sécurité numérique et prévenir les cyberattaques.

2.1.1. Le RGPD : Règlement Général sur la Protection des Données

Le **RGPD** est entré en vigueur en mai 2018. Il impose des règles strictes pour la protection des données personnelles au sein de l'Union européenne. Ce règlement s'applique à toutes les entreprises traitant des données personnelles, quelle que soit sa taille.

Les entreprises doivent assurer la **confidentialité** et l'**intégrité** des données en mettant en place des mesures techniques et organisationnelles afin de prévenir tout accès ou traitement autorisé. En cas de violation de données, elles ont l'obligation de notifier les autorités dans un délai maximum de 72 h. Les organisations doivent également tenir un registre des traitements effectués et évaluer les risques qui y sont liés.

Le non-respect de ces obligations peut entraîner des sanctions financières sévères, pouvant atteindre **4 % du chiffre d'affaires mondial annuel ou 20 millions d'euros**.

2.1.2. Directives NIS2 : sécurité des réseaux et des systèmes d'information

Adoptée en 2022, la directive **NIS2** (Network and Information Security) renforce le cadre établi par la directive NIS de 2016. Elle élève les exigences en matière de cybersécurité pour les secteurs critiques et **élargit son champ d'application** à un plus grand nombre d'entreprises, **y compris certaines TPE et PME**. Les entreprises concernées doivent identifier et gérer les risques liés à leurs réseaux et systèmes d'information en mettant en place des mesures techniques et organisationnelles appropriées.

La directive exige aussi la notification de tout incident majeur susceptible de compromettre la continuité des services ou la sécurité des données. Une notification initiale doit être transmise dans un délai de 24 heures, suivie d'un rapport détaillé sous 72 heures. Les dirigeants d'entreprise sont directement responsables de la conformité à ces obligations, ce qui inclut la formation en cybersécurité et la réalisation d'audits réguliers.

Les entités concernées par la directive sont classées en deux catégories : les **entités essentielles** et les **entités importantes**.

2.1.2.1. *Entités Essentielles*

Les entités essentielles regroupent les organisations publiques et privées opérant dans des secteurs critiques comme **l'énergie, les transports, la finance, la santé et l'approvisionnement en eau**. Ces secteurs jouent un rôle clé dans le fonctionnement de la société et de l'économie, rendant impérative une protection renforcée contre les cybermenaces. En cas de non-conformité, ces entités risquent des sanctions pouvant atteindre **10 millions d'euros ou 2 % de leur chiffre d'affaires annuel global**.

2.1.2.2. *Entités Importantes*

Les entités importantes, quant à elles, couvrent des secteurs tels que la **production alimentaire, les services numériques, les fournisseurs de services postaux**, ou encore la **gestion des déchets**. Bien que ces secteurs soient considérés comme moins critiques que ceux des entités essentielles, leur bon fonctionnement reste vital pour la société. Les sanctions en cas de non-respect des obligations sont également significatives, atteignant jusqu'à **7 millions d'euros ou 1,4 % du chiffre d'affaires annuel général**.

2.1.3. Cyber Resilience Act : Résilience des produits numériques

Proposé par la Commission européenne en 2022, le **Cyber Resilience Act (CRA)** établit des normes de cybersécurité obligatoires pour tous les **produits numériques commercialisés** dans l'UE, qu'il s'agisse de logiciels, d'applications ou d'équipements connectés.

Les fabricants doivent garantir la sécurité de leurs produits dès la conception, ce qui inclut l'intégration de mesures préventives contre les vulnérabilités. Ils sont aussi tenus de fournir des mises à jour régulières pour corriger les failles identifiées après la mise sur le marché. Une documentation claire sur les risques et les mesures prises pour les atténuer doit être mise à disposition des utilisateurs.

Les exigences du CRA s'appliquent à toutes les entreprises, y compris les TPE et PME, qui doivent s'assurer de la conformité de leurs produits pour éviter des sanctions financières ou le retrait de leur offre du marché européen. En cas de non-respect, une amende pouvant atteindre **15 millions d'euros ou 2,5 % du chiffre d'affaires annuel mondial**, en fonction de l'infraction. Les entreprises ne respectant pas les normes de sécurité ou omettant de signaler les incidents dans les délais risquent également des sanctions supplémentaires.

2.1.4. Digital Operation Resilience Act (DORA) : Résilience Opérationnelle Numérique

Adopté en 2022, le **Digital Operational Resilience Act (DORA)** renforce la résilience numérique des entreprises opérant dans le secteur financier. Ce règlement impose des normes strictes en matière de gestion des risques technologiques, en exigeant l'identification des vulnérabilités liées aux technologies de l'information et de la communication (TIC).

Les entreprises doivent effectuer des tests réguliers de leurs systèmes pour évaluer leur robustesse face aux cybermenaces et gérer de manière proactive les risques associés à leurs fournisseurs de services numériques. De plus, les incidents significatifs doivent être signalés aux

autorités compétentes dans des délais précis, ce qui renforce la transparence et la réactivité des entreprises en cas de cyberattaques.

Le non-respect des obligations prévues par le DORA peut entraîner des sanctions financières importantes pouvant aller jusqu'à 1 % du chiffre d'affaires journalier global pour chaque jour non respecté. Cela souligne l'importance pour les acteurs financiers, y compris leurs prestataires, de se conformer à ces exigences.

2.1.5. Solutions d'accompagnement

En complément des cadres réglementaires européens, de nombreux États membres de l'UE ont mis en place des **initiatives nationales** pour renforcer la cybersécurité des entreprises, en particulier des TPE et PME.

Ces initiatives incluent des subventions pour l'acquisition de solutions de cybersécurité, des campagnes de sensibilisation et des guides de bonnes pratiques adaptés aux spécificités locales. Certains pays offrent également des outils d'évaluation gratuits ou des audits simplifiés pour aider les entreprises à identifier leurs faiblesses et à se conformer aux réglementations européennes.

En ce qui concerne la **France**, celle-ci a mis en place un programme d'appui et de conseil qui permet le diagnostic, la mise en œuvre d'un plan et d'achat de solutions.

Cyber PME est une action conjointe entre la **Direction générale des Entreprises** (DGE) et **Bpifrance**. Ainsi, pour la somme de 8 800 € (subventionné à 50 %), cet outil permet d'identifier et de prioriser les actions de sécurisation.

Ces mesures visent à pallier les difficultés financières et techniques des petites structures, en leur offrant un soutien concret pour améliorer leur résilience face aux cybermenaces.

2.1.6. Synthèse

Toutes ces législations convergent sur une conclusion : il est **nécessaire d'investir** dans la cybersécurité en France et plus généralement en Europe. Dans le cas où celle-ci serait ignorée, les entreprises s'exposent à de **lourdes peines** et à des **coûts d'interruption de services** qui représentent un coût bien plus élevé qu'une mise en conformité.

2.2. Référentiels

La sécurité des applications web repose sur la protection de trois piliers fondamentaux : la confidentialité, l'intégrité et la disponibilité (CIA). Ce principe vise à garantir que les données sensibles ne soient accessibles qu'aux personnes autorisées (confidentialité), que les informations et les systèmes soient protégés contre toute modification non autorisée (intégrité), et que les services soient disponibles pour les utilisateurs sans interruption (disponibilité).

Pour mieux comprendre ces risques, nous nous sommes appuyés sur des référentiels reconnus, tels que OWASP et CWE, qui recensent et classifient les vulnérabilités.

De plus, nous avons mis en place une légende afin de voir quel pilier est corrompu et pour faciliter la compréhension des risques encourus.

■ **Intégrité Corrompue:** une fois l'intégrité corrompue, les données et le système ne peuvent plus être considérés comme fiables.

■ **Confidentialité Corrompue:** une fois la confidentialité corrompue, les données et le système sont accessibles par des personnes qui ne sont pas censé y avoir accès.

■ **Disponibilité Corrompue:** une fois la disponibilité corrompue, les données et le système n'est plus accessible.

2.2.1. OWASP TOP 10

Le référentiel OWASP (Open Web Application Security Project) constitue une référence internationale incontournable. Son classement "Top 10" recense les failles de sécurité les plus critiques dans les applications web (OWASP, 2021).



Figure 1 - OWASP top 10 - 2021

Parmi ces vulnérabilités, on retrouve notamment :

- **Contrôle d'accès rompu [A01]**  : C'est quand un système ne protège pas correctement l'accès aux données ou aux fonctions. Cela peut porter atteinte à l'intégrité des données mais aussi à leur confidentialité.
- **Injections [A03] ([SQL](#), [XSS](#), etc.)**  : C'est quand un attaquant insère du code ou des commandes malveillantes dans une application via des failles de sécurité. Cela permet de manipuler des bases de données (suppression, modification ou ajout de données) ou d'exécuter du code malveillant dans les navigateurs (XSS).
- **Mauvaise configuration de sécurité [A05]**  : Cela arrive quand un système ou une application n'est pas bien protégé, par exemple avec des mots de passe par défaut, des permissions trop larges, ou des sécurités désactivées.
- **Composants vulnérables ou obsolètes [A06]**  : Les composants vulnérables ou obsolètes sont des logiciels ou matériels qui ne sont plus à jour et qui contiennent des failles de sécurité bien connues. Ces composants peuvent inclure des systèmes d'exploitation, des bibliothèques de code, des applications ou des dispositifs matériels. Si un attaquant exploite cette vulnérabilité, il peut tirer parti de failles de sécurité qui ont déjà été identifiées, ce qui facilite l'intrusion dans le système. Cela peut entraîner le vol ou la fuite de données sensibles, compromettant ainsi la confidentialité des informations. De plus, l'attaquant pourrait propager des attaques à d'autres systèmes connectés ou provoquer une interruption des services, rendant le système instable ou inutilisable.
- **Défaillances d'identification et d'authentification [A07]**  : elles se produisent lorsque les mécanismes qui vérifient l'identité des utilisateurs ne fonctionnent pas correctement. Cela peut permettre à des utilisateurs non autorisés d'accéder à des systèmes ou des données sensibles. Ces défaillances peuvent être le résultat de mots de passe faible, d'une absence de validation de l'identité ou de processus de connexion mal sécurisés.

2.2.2. CWE top 25

Un autre référentiel est le Common Weakness Enumeration (CWE) Top 25. Il s'agit d'un référentiel qui recense les 25 failles de sécurité critique dans les logiciels. Ce classement est basé sur la prévalence et la gravité des vulnérabilités, il aide les développeurs et les professionnels de la sécurité à se concentrer sur les problèmes de sécurité les plus fréquents.

2024 CWE Top 25					
Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2023
1	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	56.92	3	+1
2	CWE-787	Out-of-bounds Write	45.20	18	-1
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	35.88	4	0
4	CWE-352	Cross-Site Request Forgery (CSRF)	19.57	0	+5
5	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	12.74	4	+3
6	CWE-125	Out-of-bounds Read	11.42	3	+1
7	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.30	5	-2
8	CWE-416	Use After Free	10.19	5	-4
9	CWE-862	Missing Authorization	10.11	0	+2
10	CWE-434	Unrestricted Upload of File with Dangerous Type	10.03	0	0
11	CWE-94	Improper Control of Generation of Code ('Code Injection')	7.13	7	+12
12	CWE-20	Improper Input Validation	6.78	1	-6
13	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	6.74	4	+3
14	CWE-287	Improper Authentication	5.94	4	-1
15	CWE-269	Improper Privilege Management	5.22	0	+7
16	CWE-502	Deserialization of Untrusted Data	5.07	5	-1
17	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	5.07	0	+13
18	CWE-863	Incorrect Authorization	4.05	2	+6
19	CWE-918	Server-Side Request Forgery (SSRF)	4.05	2	0
20	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	3.69	2	-3
21	CWE-476	NULL Pointer Dereference	3.58	0	-9
22	CWE-798	Use of Hard-coded Credentials	3.46	2	-4
23	CWE-190	Integer Overflow or Wraparound	3.37	3	-9
24	CWE-400	Uncontrolled Resource Consumption	3.23	0	+13
25	CWE-306	Missing Authentication for Critical Function	2.73	5	-5

Figure 2 - CWE top 25

Parmi ces vulnérabilités, on retrouve notamment :

- **Injection de commandes OS (CWE-78) [7]** : Un attaquant peut envoyer des commandes malveillantes à un système. Ceci lui permettrait de prendre le contrôle du serveur, d'effacer ou de modifier des fichiers, d'installer des logiciels malveillants ou d'accéder à d'autres systèmes du réseau. Cela compromettrait la sécurité du système, entraînant des pertes de données et un accès non autorisé à des informations sensibles.
- **Injection SQL (CWE-89) [3]** : Technique ou des requêtes malveillantes sont envoyées à une base de données, exploitant des failles de sécurité. Ces requêtes sont ensuite exécutées par la base de données, ce qui peut permettre à l'attaquant d'accéder, de modifier ou de supprimer des données sans autorisation.
- **CSRF (Cross-Site Request Forgery) (CWE-352) [4]** : Attaque où un utilisateur est trompé pour effectuer des actions non autorisées sur un site web où il est déjà connecté. Cela peut inclure des actions telles que de changer de paramètre, effectuer des transactions, ou publier du contenu, tout cela sans que l'utilisateur s'en aperçoive.
- **Traversée de Répertoire (CWE-22) [5]** : Faille où un attaquant peut accéder à des fichiers ou des répertoires au-delà de leurs restrictions prévues. Cela est souvent possible en

manipulant les chemins de fichier dans les entrées utilisateur, comme en utilisant des séquences de caractère spécieux pour monter et descendre dans les répertoires (par exemple ..).

- **Lecture en dehors des limites (CWE-125) [6]**  : Faille où un programme lit des données en dehors des limites d'un tampon de mémoire, ce qui peut entraîner des comportements imprévus, et que des fuites d'information sensibles surviennent.
- **Utilisation après libération (CWE-416) [8]**   : Faille où un programme continue d'utiliser une mémoire qui a été libérée, ce qui peut entraîner des comportements imprévus, comme le plantage du programme ou des failles de sécurité exploitées par des attaquants.
- **Absence d'autorisation (CWE-862) [9]**   : Faille où un utilisateur non autorisé peut accéder à des ressources ou des fonctions pour lesquelles il n'a pas les permissions appropriées. Cela peut résulter de failles dans les mécanismes de contrôle d'accès ou de configuration incorrect, permettant ainsi à des individus non autorisés de consulter, modifier ou supprimer des données sensibles.
- **Téléchargement non restreint de fichiers dangereux (CWE-434) [10]**    : Faille où un attaquant peut télécharger des fichiers dangereux sur un serveur, exploitant des failles de sécurité. Cela peut permettre à des attaquants d'introduire des fichiers malveillants qui pourraient exécuter du code dangereux ou compromettre la sécurité du système.
- **Injection de Code (CWE-94) [11]**    : Technique où un attaquant introduit du code malveillant dans une application, en exploitant des failles de sécurité. Cela se produit souvent lorsqu'une application ne valide pas correctement les entrées utilisateur.
- **Validation d'entrée incorrecte (CWE-20) [12]**    : Faille où les entrées utilisateur ne sont pas correctement validées, permettant ainsi des attaques par injection de données malveillantes qui peuvent exploiter des failles de sécurité, comme des injections SQL, des scripts malveillants, ou d'autres attaques.

2.2.3. Bilan sur les vulnérabilités

On peut alors identifier des catégories de vulnérabilité critiques.

Catégorie	Impact principal
Problèmes liés aux Injections	Compromets l'intégrité des données et permets des actions non autorisées
Contrôle d'accès défaillant	Accès à des ressources sensibles sans autorisation
Mauvaise configuration de sécurité	Introduction de vulnérabilité exploitable par des erreurs de configuration
Défaillance d'authentification	Usurpation d'identité et compromission des comptes
Validation insuffisante des entrées	Exploitation via des données malveillantes
Vulnérabilité liée à la mémoire	Plantages ou exploitation des erreurs de programmation
Attaques spécifiques aux sessions	Actions non autorisées effectuées à l'insu des utilisateurs

Tableau 1- tableau récapitulatif des catégories des vulnérabilités les plus communes

Il existe aujourd'hui de nombreux types de vulnérabilités dans les applications web. Sans connaissances préalables ni outils de détection adaptés, il est difficile de les repérer efficacement. Pour les identifier toutes, il est essentiel de maîtriser un large éventail de domaines. Cependant, dans les TPE et PME, le dirigeant, souvent responsable de la gestion informatique, manque souvent de temps et des compétences nécessaires pour accomplir cette tâche. Cela est particulièrement crucial pour ces petites structures, car elles sont fréquemment les plus exposées aux cybermenaces en raison de leurs ressources limitées. D'où l'importance de proposer des outils adaptés, faciles à comprendre et à utiliser par leurs dirigeants.

Il est donc essentiel de pouvoir identifier et corriger ces vulnérabilités dès qu'elles se manifestent, et ce, dans toutes les applications web, quelle que soit la taille de l'entreprise. Toute entreprise doit se conformer aux exigences et réglementations européennes, indépendamment de ses ressources ou de sa taille.

2.3. Détection des vulnérabilités

Aujourd'hui, il existe plusieurs solutions pour détecter les vulnérabilités, mais celles-ci présentent certaines limites pour les dirigeants de TPE et PME. En effet, cette détection peut être classée en trois grandes catégories : les scanners de vulnérabilités commerciaux, les scanners open source et les outils open-source. Nous allons examiner leurs avantages et leurs inconvénients, en tenant compte des contraintes spécifiques des TPE et PME pour leur mise en application.

2.3.1. Scanners de vulnérabilités commerciaux

Des solutions, souvent proposées sous forme de services, sont conçues pour offrir une expérience clé en main, combinant efficacité, support technique et mises à jour régulières.

Cependant, leur adoption peut représenter un défi pour les TPE et PME. En raison de budgets généralement limités, ces entreprises peuvent rencontrer des difficultés à justifier l'investissement nécessaire pour ces outils payants. Selon un sondage réalisé par l'usine digital pendant l'été 2024, les TPE et PME ont un budget moyen de 2000 €. Et dans 80 % des entreprises de ce type, il s'agit du chef d'entreprise qui s'occupe de gérer l'informatique.

Il est essentiel de comprendre les avantages et les limites des scanners de vulnérabilités commerciales afin d'évaluer leur pertinence pour les petites structures. Cette section explore ces aspects pour mieux appréhender leur application dans un contexte de contraintes financières et opérationnelles.

OUTIL	PRIX	SIMPLICITÉ D'UTILISATION	COUVERTURE	SUPPORT
Acunetix (Invicti)	★★★★☆ 4 495 \$/an	★★★★★☆ Interface intuitive, rapports clairs, mais demande quelques bases en sécurité	★★★★★ Excellent couverture des vulnérabilités complexes, y compris applications dynamiques	★★★★☆ Support client réactif et bien documenté
Burp Suite Pro (PortSwigger)	★★★★★☆ 449 \$/an	★★★★☆☆ Moins intuitif, courbe d'apprentissage pour non-experts	★★★★★ Couverture très détaillée pour des tests approfondis	★★★★☆ Support et communauté très présents
Nessus (Tenable)	★★★★☆☆ 2 990 \$/an	★★★★☆☆ Interface accessible, mais exige des compétences techniques	★★★★★☆ Très bon pour les réseaux, configurations système, moins pour le web	★★★★☆ Support fiable et réactif
VRx (Vicarius)	★☆☆☆☆ 10 000 \$/an	★★★★☆☆ Accessible avec des outils avancés, mais nécessite une équipe IT	★★★★★ Focus exceptionnel sur la correction proactive	★★★★★ Support haut de gamme
Qualys TruRisk	★★★★★ Version gratuite ou 199 \$/mois	★★★★☆☆ Interface technique, nécessite un peu d'adaptation	★★★★★☆ Bonne couverture des vulnérabilités générales	★★★★☆ Support disponible, bonnes ressources
Intruder (Intruder System)	★★★★★☆ ~145 €/mois	★★★★★ Extrêmement simple d'utilisation	★★★★☆☆ Couverture correcte, mais limitée sur des failles complexes	★★★★☆ Support réactif, notifications automatiques utiles

Tableau 2 -tableau récapitulatif des scanners commerciaux

Nous avons réalisé un panorama des plus grosses solutions de cybersécurité proposées. Nous avons retenu six entreprises. (cf. Tableau 2).

Parmi les **avantages et inconvénients** qu'une solution propriétaire apporte à une alternative open source, nous avons les points suivants :

- **Support Client**

Une solution payante offre généralement un **support plus complet** qu'une solution Open Source. Certaines entreprises comme Acunetix et PortSwigger (Burp Suite) proposent également des **formations** pour enseigner les bonnes pratiques de cybersécurité aux professionnels. Un plus non négligeable pour une TPE/PME voulant se mettre à niveau dans ce domaine.

- **Simplicité d'utilisation**

Toutes ces plateformes ont pour but de simplifier la prise en main et l'utilisation de leur outil et, avec l'aide de professionnels, ils réussissent à **supprimer une grande partie de la complexité** de la mise en place initiale. Malgré tout, ce sont des outils professionnels, adressés à un personnel qualifié. C'est ce même personnel qui manque cruellement aux TPE/PME. Selon une étude réalisée en 2024 pour Cybemailveillance.gouv.fr, **72 % des TPE et PME** ne disposent pas de personnel qualifié dans ce domaine, montrant l'importance du support client. De plus, **56 %** des entreprises interrogées disent qu'elles ont un manque de connaissance et d'expertise

- **Couverture CVE**

Les outils payants ont généralement de plus grandes ressources leur permettant d'avoir une **plus grande couverture de vulnérabilités plus rapidement**. Malgré tout, en prenant l'exemple de Nessus et OpenVAS (une alternative open source à Nessus), selon intruder.io¹, Nessus couvrirait 41,8 % des vulnérabilités connues contre 37,4 % pour OpenVAS. Même si cela représente une différence notable, c'est une différence qui va plutôt intéresser les professionnels et non les petites entreprises en recherche d'une sécurité informatique basique.

- **Des coûts élevés**

Les licences des scanners commerciaux représentent une barrière financière importante. Elles varient traditionnellement entre **500** et **10 000 euros par an**, ce coût est souvent une barrière dans l'obtention et de leur mise en place. Alors que selon l'étude réalisée pour Cybermalveillance.gouv.fr, **68 %** des entreprises interrogées ont un budget alloué de moins de **2 000 euros par an**.

Bien que ces services offrent de nombreux avantages, tels que leur **simplicité d'utilisation** et une **couverture importante** des vulnérabilités, leur **coût élevé** les rend généralement inaccessibles pour les PME et TPE dans le cadre de leurs campagnes de sécurité.

¹ Intruder est un concurrent à Nessus, cette étude est donc à prendre avec du recul.

2.3.2. Scanners de vulnérabilités gratuits et open source

Face aux coûts élevés des solutions commerciales, de nombreuses alternatives gratuites et open source sont disponibles. Ces solutions sont entièrement gratuites et libres d'utilisation, mais généralement moins complètes que leurs homologues commerciaux.

De plus, ils sont aussi souvent plus difficiles d'utilisation, ou dans **72 % des TPE et PME**, aucun personnel n'est qualifié dans le domaine de la cybersécurité, selon l'étude faite pour Cybermalveillance.gouv.fr.

OUTIL	SIMPLICITÉ D'UTILISATION	COUVERTURE	SUPPORT
OWASP ZAP	★★★☆☆ Courbe d'apprentissage avec doc technique	★★★★★☆ Excellent couverture des vulnérabilités complexes, y compris applications dynamiques	★★★★☆☆ Support communautaire actif, mais pas de premium
Nikto	★☆☆☆☆ Moins intuitif, courbe d'apprentissage pour non-experts	★★★☆☆ Couverture très détaillée pour des tests approfondis	★★☆☆☆ Support limité à la communauté (non-officiel)
WPScan	★★★★★☆ Très simple si l'entreprise utilise WordPress	★★☆☆☆ Limité aux vulnérabilités WordPress (43,3 % de tous les sites web)	★★★★☆☆ Support communautaire ou premium limité
OpenVAS	★★★☆☆ Complexité élevée, demande une infrastructure dédiée (Fort malus pour l'interface graphique)	★★★★★ Large couverture des vulnérabilités réseau et web	★★★★☆☆ Support communautaire actif mais sans garantie

Tableau 3 -tableau récapitulatif des scanners

Pour notre analyse comparative, nous avons retenu quatre scanners (cf. Tableau 3) dont deux spécialisés dans le scan web.

Ces scanners/plateformes ont tous un point commun, ils publient leur code source et sont **libres de droits**. Créés par la communauté, ces outils sont mis à disposition gratuitement, **sans discriminer** le client mais aussi **sans garantie d'aucune sorte**.

Après analyse, voici les **avantages et inconvénients** que nous avons trouvés pour ces plateformes :

- **Complexité**

Là où les outils propriétaires étaient complexes seulement dans leur interprétation, les outils open source nécessitent aussi une **installation et configuration complexe** avant de pouvoir être utilisés. En effet, un travail préalable concernant l'installation de paquets spécifiques à chaque outil est nécessaire. Cela représente une barrière conséquente, nécessite l'intervention d'un personnel formé et d'une compréhension des besoins au préalable de la configuration. De plus, lors de l'utilisation de certains scanners comme nikto, tout se fait par ligne de commande, nécessitant une connaissance sur l'utilisation d'un terminal. La connaissance des commandes de l'outil et l'interprétation des résultats sont également nécessaires, car ces résultats sont souvent présentés dans un format non traditionnel.

- **Support**

Malgré cette complexité plus importante, le support lui aussi est souvent manquant. L'absence d'une structure de soutien formelle, combinée à une participation limitée de la communauté, et exacerbée par un manque de financement, peut rendre difficile pour les projets open-source de fournir un soutien adéquat à leurs utilisateurs. Les entreprises n'ont pour autre choix que de consulter la **documentation** écrite par la communauté, qui peut être **complexe** et/ou **lacunaire**. C'est une des raisons principales qui expliquent que les TPE/PME n'utilisent pas ce genre d'outils.

- **Couverture [CVE](#)**

Pour les mêmes raisons que le support vient à manquer, les scanneurs open source ont généralement un peu de **retard** sur les outils propriétaires en ce qui concerne la quantité de vulnérabilités prise en charge. Encore une fois, c'est un point plus susceptible de préoccuper les spécialistes que les petites entreprises qui recherchent uniquement une sécurité informatique de base.

2.3.3. Outils de vulnérabilités gratuits et open source

En plus des scanners de vulnérabilité, des outils pour pouvoir détecter des vulnérabilités existent. Ceux-ci sont spécifiques à une fonctionnalité ou un domaine très restreint.

Ces **outils** combinent les lacunes des scanners en ciblant des aspects précis, comme les ports ouverts, la configuration des serveurs ou les vulnérabilités propres à certains logiciels, permettant ainsi une analyse plus approfondie.

Parmi les outils existants, une partie sert à la détection de CMS. En effet, un **système de gestion de contenu (CMS)** est une plateforme logicielle permettant de créer, gérer et publier un site web sans nécessiter de compétences en programmation. Grâce à une interface intuitive, les utilisateurs peuvent ajouter du contenu, personnaliser l'apparence et intégrer des fonctionnalités via des extensions ou des modules.

Ces solutions sont particulièrement adoptées par les **TPE et PME**, qui ne disposent souvent ni des compétences techniques ni des moyens financiers pour développer un site sur mesure. Elles offrent une alternative accessible aux développements nécessitant des ressources spécialisées. Selon l'**AFNIC**, **83 %** des TPE/PME et **63 %** des microentreprises disposent d'un site web. ([AFNIC](#)), cela illustre l'importance pour ces entreprises d'avoir une présence en ligne, et le rôle essentiel des CMS pour y parvenir sans nécessiter de ressources techniques ou financières importantes.

En 2024, parmi l'ensemble des sites web, environ **68,7 % utilisent un CMS** ([WPADE](#)). WordPress domine largement avec **62,7 % de part de marché**, suivi par Shopify (6,4 %), Wix (3,9 %) et Squarespace (3 %). ([Bloggerspassion](#))

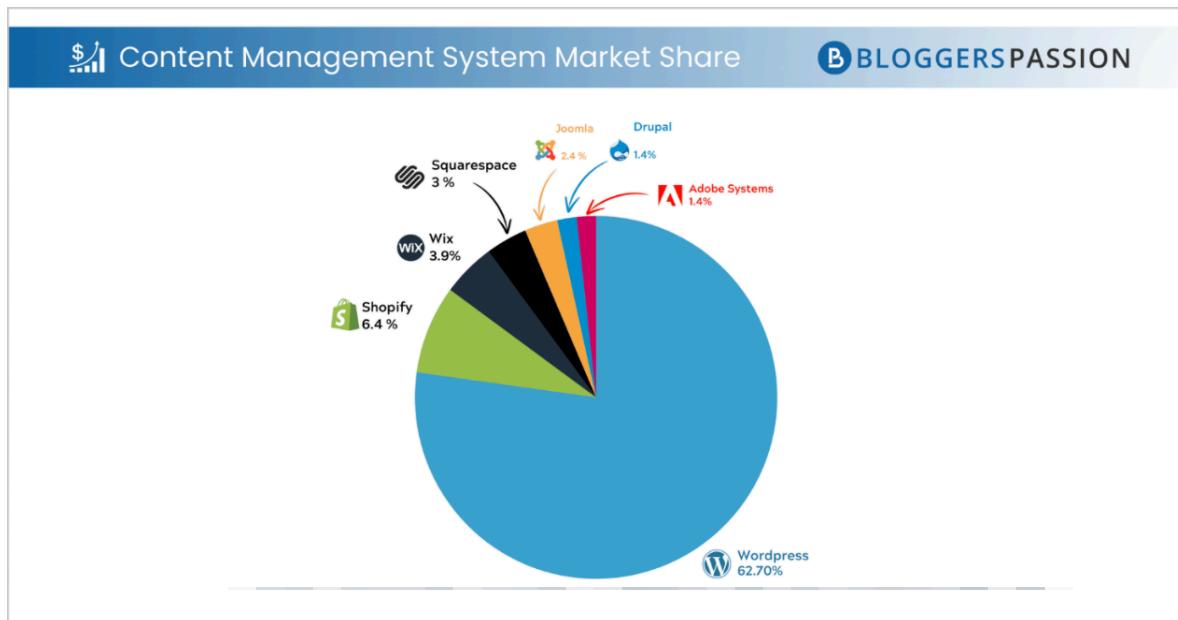


Figure 3 - part de marché des CMS

Cependant, cette popularité fait aussi des CMS une cible privilégiée des cyberattaques. Selon une étude menée par Sucuri en 2019 ([Sucuri](#)), **94 % des sites piratés utilisaient WordPress**, suivi par Joomla (2,5 %) et Drupal (1,2 %). Ces chiffres s'expliquent notamment par le fait que de nombreux sites WordPress sont créés et administrés par des personnes n'ayant pas nécessairement de connaissances en cybersécurité. Cela augmente ainsi le risque d'erreurs de configuration et d'utilisation de plugins vulnérables. Ce manque de sécurisation attire donc particulièrement les [cybercriminels](#), qui y voient une opportunité d'exploitation facile.

Étant donné leur large adoption et leur **forte exposition aux cyberattaques**, les CMS nécessitent une attention particulière en matière de sécurité. Pour mieux identifier les vulnérabilités spécifiques à ces plateformes, nous utiliserons des outils spécialisés qui viendront compléter les scanners précédents en ciblant des éléments propres à chaque système :

- **WPScan** : Analyse des sites WordPress pour identifier les vulnérabilités des extensions, des thèmes et des paramètres de configuration. Il aide également à repérer les versions obsolètes et les pratiques à risque.
- **Joomscan** : Conçu pour Joomla, il détecte les failles dans les composants du site, y compris les extensions vulnérables et les erreurs de configuration pouvant être exploitées.
- **Droopescan** : Spécialisé dans l'évaluation des sites Drupal, il permet d'identifier les modules obsolètes, les thèmes vulnérables et d'autres éléments pouvant compromettre la sécurité du site.

En revanche, des outils similaires n'existent pas pour des CMS comme Shopify, Wix ou Squarespace, ce qui limite les possibilités d'analyse spécifique. Dans ces cas, seuls les scanners généralistes seront utilisés, offrant un niveau de détection plus limité que pour les CMS bénéficiant d'outils spécialisés.

En plus de ces scanners, d'autres outils que les scanneurs de CMS existent

Catégorie	Outil	Description
Cartographie et découverte d'infrastructures	SUBLIST3R, CSmap, Whois, Dig, Oralyzer	Énumération des sous-domaines, identification des technologies, récupération des informations administratives (Whois), interrogation DNS et analyse des fichiers robots.txt.
Analyse de la configuration et des services	Nmap (NSE), SSH Audit, Wapiti	Détection des services actifs, audit de la sécurité SSH, analyse des configurations et des protections web (CSP, HSTS...).
Exploration et tests d'accès	GoBuster, FUZZ, Cloudflare Detect	Recherche de répertoires cachés, tests d'injection et d'accès, identification des protections Cloudflare.
Recherche de vulnérabilités et d'exploits	Searchsploit, Metasploit	Recherche d'exploits dans des bases de données publiques et exploitation automatisée des vulnérabilités.
Collecte d'informations (OSINT)	Email Harvester	Extraction automatique d'adresses email depuis des sources publiques.

Tableau 4 - tableau des outils par intermédiaire

Aucune automatisation n'existe pour l'utilisation de ces outils. À chaque fois qu'on souhaite en utiliser un, on doit alors exécuter une ligne de commande détaillée, et ce, pour chaque outil. De plus, ils ont tous des configurations spécifiques, ce qui est impossible pour un dirigeant de TPE et PME à suivre et à mettre en place. De plus, une fois chaque outil utilisé, ils ont chacun une façon différente de présenter les résultats, et de façon plus ou moins compréhensible. Il faut donc être aussi capable d'interpréter les résultats.

2.3.4. Synthèse comparative

Le tableau ci-dessous synthétise les caractéristiques des différentes catégories d'outils existants :

Catégorie	Coût approximatif (licence)	Complexité	prise en main	Adapté PME
Scanners commerciaux	de ~500 €/an à ~6000 \$/an	Moyenne	Interface professionnelle, configuration complexe	Faible (coût trop élevé)
Scanneurs gratuits/open source	0 €	Moyenne	configurations	Moyenne (compréhensibilité trop compliquée)
Outils	0 €	Élevée	ligne de commande et configuration technique	faible (mise en place des outils compliquée)

Tableau 5 - tableau récapitulatif des trois catégories

Ce tableau met en évidence le fossé existant entre les solutions commerciales, complètes, mais coûteuses et complexes, et les outils gratuits ou open sources, abordables, mais nécessitant des compétences techniques et des intégrations manuelles. Les PME, en quête d'une approche simplifiée (champ URL, un clic, rapport clair), ne trouvent pas de solution adéquate dans l'offre actuelle du marché.

2.4. La Cybersécurité pour tous

2.4.1. Analyse du modèle SEO

Les outils d'analyse [SEO](#), bien que centrés sur l'optimisation des moteurs de recherche, partagent des points communs avec les scanners de vulnérabilités en termes de fonctionnement et d'expérience utilisateur.

Une analyse [SEO](#) est traditionnellement **une analyse complète et complexe** à interpréter nécessitant l'intervention de personnel qualifié. Malgré cela, des outils ont été mis à disposition du public, gratuitement ou non, permettant la vulgarisation de ce sujet et l'analyse simplifiée d'un site web.

Dans cette partie, nous analyserons l'interface de **PageSpeed Insights** (Google Light house), un outil proposé gratuitement par Google à l'interface simple.



Figure 4 - Page d'accueil de PageSpeed Insights

Lors de la navigation vers le site de la plateforme, nous sommes accueillis par un simple champ de texte annoté de "Saisir l'URL d'une page Web" (cf. Figure 7). C'est un **design simple** qui attire l'attention via un **call to action** incitant à lancer l'analyse.

On remarque qu'il n'y a **pas d'étape d'installation** du logiciel et **pas d'étape de configuration**. C'est un grand avantage pour permettre l'accessibilité à tous. Quel que soit le budget, peu importe les compétences techniques, il est simple de créer une analyse.

Rapport du 26 janv. 2025, 14:52:12

<https://ozeliurs.com/>

Analyser

 Mobile  Bureau



Découvrez l'expérience de vos utilisateurs



Aucune donnée



Analysez les problèmes de performances

 67

Performances

 94

Accessibilité

 100

Bonnes pratiques

 90

SEO

 67

Performances

Les valeurs sont estimées et peuvent varier. Le [calcul du score lié aux performances](#) repose directement sur ces statistiques. [Affichez la calculatrice.](#)



 0–49  50–89  90–100

Figure 5 - Résultats d'une analyse

Après qu'une analyse soit terminée, une interface simple nous est présentée. Celle-ci comporte, à gauche, un simple **score global** accompagné d'un **code couleur tricolore**. À droite, un aperçu du site web scanné. Et au-dessus, quatre scores détaillant le résultat par catégorie.

Nous retiendrons que l'interface **ne surcharge pas l'utilisateur**, il lui présente le strict nécessaire quitte à manquer de détails.

DIAGNOSTIC

- ▲ Élément identifié comme "Largest Contentful Paint" — 6200 ms
- ▲ Diffusez des images aux formats nouvelle génération — Économies potentielles de 323 Kio
- ▲ Précharger l'image Largest Contentful Paint — Économies potentielles de 930 ms
- Diffusez des éléments statiques grâce à des règles de cache efficaces — 2 ressources trouvées
- Réduisez les ressources JavaScript inutilisées — Économies potentielles de 21 Kio
- Le temps de réponse initial du serveur était court — Le document racine a pris 50 ms
- Évitez d'enormes charges utiles de réseau — La taille totale était de 705 Kio
- ~ Évitez une taille excessive du DOM — 948 éléments

Figure 6 - Suite des résultats d'une analyse

Plus bas sur la page, la suite de l'analyse devient **plus technique** mais se concentre sur les éléments d'amélioration. Cette partie **garde le code tricolore et priorise le diagnostic**.

▲ Diffusez des images aux formats nouvelle génération — Économies potentielles de 323 Kio

Les formats d'image comme WebP et AVIF proposent souvent une meilleure compression que PNG et JPEG. Par conséquent, les téléchargements sont plus rapides et la consommation de données est réduite. [En savoir plus sur les formats d'image récents](#) [LCP] [FCP]

URL	Taille de la ressource	Économies potentielles
creddy.com	472,2 KiB	280,8 KiB
...20082fc1-94af-4773-9df0-28856b566748/image.png (images.credly.com)	195,1 KiB	128,2 KiB
...22a0ece5-ff05-4594-8320-25e55e9ae203/image.png (images.credly.com)	200,6 KiB	125,0 KiB
...70d71df5-.../CCNAITN__1_.png (images.credly.com)	37,4 KiB	14,0 KiB
...f4ccdba9-.../CCNASRWE__1_.png (images.credly.com)	39,2 KiB	13,6 KiB
picsum.photos	88,9 KiB	41,9 KiB
...1920/1080.jpg?blur=5&hmac=EcEtUQvcZ... (fastly.picsum.photos)	88,9 KiB	41,9 KiB

Figure 7 - Détail d'un élément de l'analyse

Finalement, l'utilisateur peut cliquer sur un élément de l'analyse pour en voir les détails et **consulter les liens** vers divers guides et **ressources en relation avec le problème**.

L'expérience utilisateur de PageSpeed Insights représente une source d'inspiration en termes de vulgarisation. En effet, cette simplicité de lecture et de compréhension nous semble être un élément primordial pour que le domaine de la cybersécurité soit accessible à tous. Notre cible utilisateur est peu qualifiée, comme peut l'être le personnel d'une TPE et PME.

2.4.2. Notre proposition : Weakspotter

Nous proposant une solution, Weakspotter, qui a pour but d'être facilement compréhensible, quel que soit le public et utilisable peu importe les compétences informatiques.

	Coût	Installation	Complexité	Utilisation Automatique
scanner commerciaux	entre 500 et 10 000 € euros par ans	Simple	à comprendre les rapports	Oui
scanner open-source	0 €	Complexe	à comprendre les rapports	Oui
outils open-source	0 €	Complexe	à comprendre les sorties	Non
Weakspotter	0 €	Simple	simple	Oui

Tableau 6 - Tableau récapitulatif des solutions envisageables pour une TPE et/ou PME

Notre solution **Weakspotter** répondra donc parfaitement au besoin des TPE et PME. Un service gratuit, composé de scanners et outils open-source, qui propose un scan complet et intuitif de n'importe quelle plateforme web.

Notre interface est fortement inspirée de sites d'analyse de SEO, avec le principal input de l'URL pour effectuer le scan. Elle offre ensuite une vue d'ensemble du site et des améliorations, avec possibilité d'aller plus en détails et de consulter les ressources externes.

Nous découpons chaque analyse en quatre catégories, la sécurité du **backend** web, la sécurité du **frontend** web, la sécurité du **serveur web sous-jacent** et finalement la sécurité du **DNS**. Ces catégories sont encore sujettes à changer au fil de nos recherches.

Pour rendre les analyses abordables par les chefs d'entreprise, nous comptons utiliser l'**IA** intelligemment pour **rédiger des descriptions des CVE** abordables par les non-initiés. Nous explorons également la possibilité de faire un résumé global IA.

Nous avons finalement réalisé un travail de **documentation** détaillé sur les méthodes de déploiement, dans le but d'assister les administrateurs système dans le déploiement de cet outil.

2.5. Conclusion de l'analyse de l'existant

L'analyse menée dans ce rapport met en évidence les défis majeurs rencontrés par les TPE et PME en matière de cybersécurité. Alors que les menaces numériques se multiplient et que les régulations, telles que le **RGPD** ou la directive **NIS2**, imposent des exigences croissantes, ces petites structures peinent à trouver des solutions adaptées à leurs besoins et à leurs moyens.

L'offre existante présente des **incompatibilités** par rapport aux attentes et aux capacités des petites entreprises. Les solutions commerciales, bien que performantes, restent souvent inaccessibles en raison de leur **coût élevé** et de leur **complexité**. De l'autre côté, les outils open source, bien qu'abordables, exigent des **compétences techniques** que ces structures ne possèdent généralement pas.

C'est dans ce contexte que nous créons le projet WeakSpotter, une solution conçue pour **démocratiser la cybersécurité** auprès des TPE et PME. En s'inspirant des approches simplifiées des outils SEO, WeakSpotter offre une expérience intuitive, basée sur un simple champ URL, et génère un **audit clair et exploitable** sans nécessiter d'expertise technique. Cette solution se positionne comme un pont entre la complexité des outils actuels et les besoins réels des petites entreprises.

3. WeakSpotter

WeakSpotter est une solution développée dans le cadre de notre PER afin d'apporter aux TPE et PME un audit de sécurité clair et accessible, quel que soit leur niveau technique. Notre objectif est de rendre l'audit de cybersécurité compréhensible et exploitable pour tous, afin de répondre aux besoins spécifiques de ces entreprises.

Le développement de **WeakSpotter** a débuté par une phase de **conception**, une étape essentielle pour garantir que la solution réponde parfaitement aux objectifs définis. Cette phase a été centrée autour du sujet suivant : **Flash audit automatisé pour la Cybersécurité Web : Conception d'un scanner de vulnérabilités proactif**.

3.1. Cahier des Charges

Le processus de conception a débuté par une présentation complète du projet par M. **DELETTRE**, qui a également précisé ses attentes. Ce projet de **PER** faisait également office de sujet pour les étudiants en **BUT (ancien DUT) Réseaux et Télécoms** de Sophia Antipolis. M. DELETTRE nous a ainsi présenté les maquettes développées précédemment par ces étudiants, servant de base à notre réflexion. À partir de ces éléments, nous avons élaboré le cahier des charges suivant.

La plateforme devra être conçue sous la forme de "[Software as a Service](#)" (SaaS), c'est-à-dire qu'elle devra être hébergée dans le cloud et être accessible de n'importe où, à n'importe quel moment, dès lors qu'il y a une connexion internet. Cela implique aussi que notre plateforme doit être capable de **stocker, isoler et sécuriser les données des clients** de manière fiable et conforme.

De plus, il a été spécifié que la plateforme se doit d'être développée et distribuée sous licence open source. Nous avons choisi la [licence GPL](#) (licence publique générale GNU) car nous dépendons d'autres produits sous cette même licence.

Finalement, notre application devra être facile à utiliser, avec un vocabulaire accessible et une présentation simplifiée des résultats. Elle devra également proposer deux types de scans : un scan simple et un scan complexe, ce dernier fournissant une analyse plus approfondie et détaillée des résultats.

3.2. Maquette et POC

Après s'être assuré que notre équipe avait une bonne compréhension du projet, nous avons poursuivi avec la phase de maquettage pour mieux identifier les besoins des utilisateurs ([Annexe 3: Maquette](#)). Cette étape nous a permis d'identifier et de valider les fonctionnalités essentielles avec M. DELETTRE (qui agit en tant que [Product Owner](#)). Bien que le design final ait évolué par rapport à la maquette initiale, cette phase de travail préparatoire a été déterminante pour établir l'architecture fonctionnelle du système et des fonctionnalités principales.

Une seconde maquette a par la suite été réalisée ([Annexe 4: Preuve de Concept](#)). Cette maquette est une Preuve de Concept ([POC](#)), un projet préliminaire qui démontre la faisabilité de WeakSpotter. Elle a, elle aussi, été présentée devant M. DELETTRE. Nous avons dans ce [POC](#) la présence de la partie scan et une ébauche de la partie du rapport. Le moyen de scan et l'ébauche de rapport sont les parties les plus importantes à réaliser sur WeakSpotter, le processus a donc été présenté au Product Owner ([PO](#)), puis validé par celui-ci.

Finalement, nous avons examiné les outils existants qui nous aideraient dans la réalisation de notre plateforme. Après avoir identifié ces outils (cf. [analyse de l'existant](#)) nous devions encore réfléchir à l'ordre d'exécution de ces outils. Nous avons donc commencé la réalisation d'un diagramme de logique ([Annexe 2 : diagramme de logique](#)) à la main avant de nous rendre compte que c'était un processus compliqué qu'il fallait automatiser.

À la fin de cette phase de conception (Maquette et [POC](#)) nous avons défini les fonctionnalités principales à implémenter dans **WeakSpotter**. Ces fonctionnalités sont les suivantes :

- **Interface d'analyse** : l'utilisateur dispose d'une page dédiée à la saisie d'URL, avec différents niveaux de complexité pour le scan.
- **Suivi en temps réel** : une interface de chargement dynamique est mise en place pour offrir une meilleure expérience utilisateur, elle permet de visualiser la progression du scan en cours, apportant une transparence sur les étapes en cours
- **Présentation des résultats** : le système propose deux niveaux de lecture des résultats :
 - un score global simplifié utile pour avoir une idée globale
 - une analyse détaillée présentant chaque vulnérabilité identifiée, accompagnée d'un score spécifique et de recommandations pour la remédiation
- **Gestion des utilisateurs** : système de gestion des utilisateurs avec
 - une interface d'authentification sécurisée
 - un historique des scans effectués

3.3. Conception Logicielle

Notre architecture pour WeakSpotter a été conçue en suivant le modèle client-serveur classique pour faciliter le développement et la maintenance future. Cette section détaille les choix techniques que nous avons faits, l'organisation des différentes parties, leur interaction et leur déploiement.

3.3.1. Architecture

Avant d'entrer plus en détail sur la conception de WeakSpotter, nous allons vous présenter son architecture.

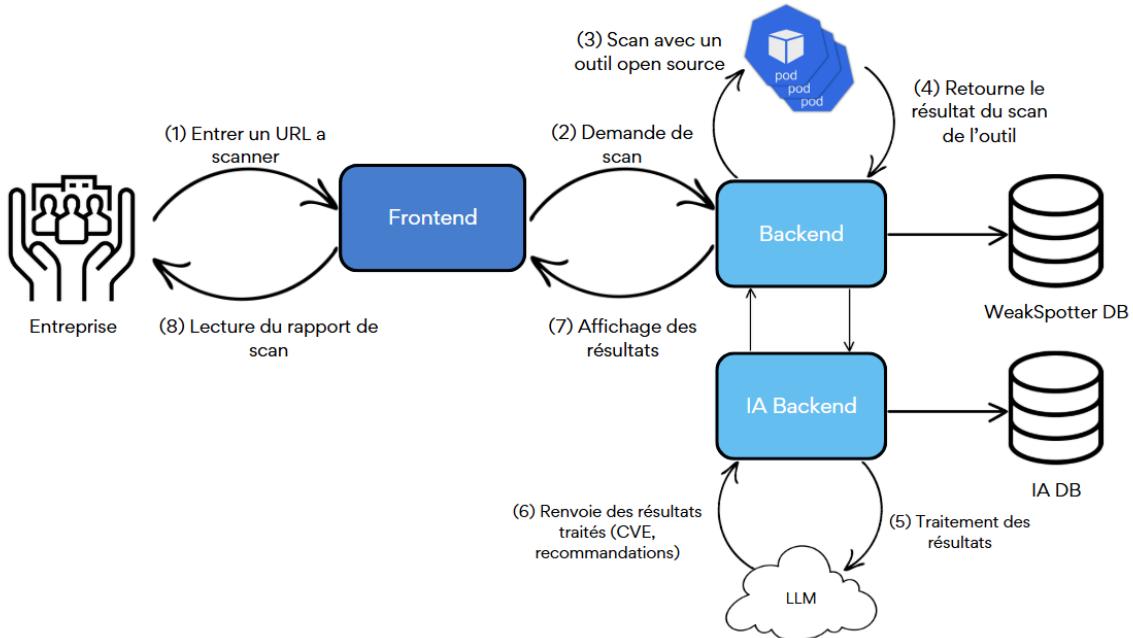


Figure 8 - Architecture globale de WeakSpotter

L'architecture repose sur plusieurs composants interconnectés :

- **Frontend** : Interface utilisateur permettant d'initier un scan en saisissant une URL.
- **Backend** : Gère les requêtes du frontend et orchestre l'exécution des scans.
- **Conteneurs OCI**: Exécute un outil open source pour scanner les vulnérabilités.
- **Backend IA** : Transmet les résultats du scan au modèle d'intelligence artificielle.
- **LLM (Large Language Model)** : Analyse les résultats et fournit des recommandations.

De plus, ce schéma d'architecture présente l'ordre de fonctionnement des différents composants :

- (1) **Entrez une URL à scanner** : l'utilisateur saisit l'URL cible via l'interface web.
- (2) **Demande de scan** : Le frontend envoie la requête au backend.

- (3) **Scan avec un outil open source** : Le backend exécute un conteneur Docker qui lance le scan de vulnérabilité sur l'URL.
- (4) **Retourne le résultat du scan de l'outil**: Le conteneur retourne les résultats bruts au backend.
- (5) **Traitement des résultats**: Le backend IA transmet les résultats au modèle LLM pour analyse.
- (6) **Renvoie des résultats traités (CVE, recommandations)**: Le LLM identifie les vulnérabilités (CVE) et génère des recommandations adaptées.
- (7) **Affichage des résultats**: Le backend renvoie les résultats analysés au frontend, où ils sont présentés à l'utilisateur.
- (8) **Lecture du rapport de scan**: L'utilisateur peut consulter un rapport détaillé des vulnérabilités détectées et des actions recommandées.

3.3.2. Frontend

La partie [frontend](#) de notre plateforme est responsable de la partie affichage au client, nous avons fait le choix de développer cette partie en utilisant le [TypeScript](#) et le framework [React](#). Cela nous a permis de développer plus rapidement. À cela, nous avons ajouté la bibliothèque de composants [DaisyUI](#) et le framework [CSS Tailwind](#), cela nous a permis de nous concentrer sur la partie cybersécurité tout en réalisant un [site web responsive](#) et esthétique rapidement. Tous ces éléments sont distribués sous une licence open source compatible avec la [GPL](#).

Notre plateforme est donc un [SPA](#) (Single Page Application) utilisant le React Router pour l'organisation et la navigation entre les pages. Nous avons organisé le frontend sous forme de composants React réutilisable, favorisant une architecture modulaire et maintenable.

Pour établir une connexion entre le frontend et le backend, nous utilisons la bibliothèque [Axios](#), qui nous permet de définir un client [API](#) avec des routes spécifiques, des méthodes, des paramètres et des types de retour. Nous appelons ensuite ces routes depuis les composants de page et transmettons les données sous forme d'état aux composants enfants pour une [hydratation](#) efficace. Cela garantit la synchronisation des données entre les différents composants de manière fluide.

En matière d'authentification, nous utilisons un [contexte React](#) pour gérer l'état global de l'utilisateur à travers toutes les pages. Cette approche centralisée est indispensable, car certaines interactions avec le backend nécessitent une authentification, et l'état de l'utilisateur doit être accessible depuis différentes pages.

3.3.3. Backend

La partie [backend](#) de notre plateforme est responsable de toute la logique derrière WeakSpotter. Elle comporte entre autres : l'authentification, l'orchestration d'un scan, l'exploitation des résultats, le stockage des scans, etc.

Nous avons choisi Python comme langage principal, en raison de sa simplicité d'utilisation et pratiqué par tous les membres de l'équipe. De plus, nous avons choisi le framework [FastAPI](#) puisqu'il est rapide, sécurisé par défaut et facile d'utilisation.

3.3.3.1. Persistance et Modèles

Pour la gestion de notre base de données, nous avons décidé d'utiliser [l'ORM sqlmodel](#), car il s'intègre particulièrement bien avec le framework [FastAPI](#). La base de données sous-jacente est une base [SQLite](#) mais peut être changé assez facilement avec une variable d'environnement pour une base [MariaDB](#) ou [PostgreSQL](#).

Nous avons défini trois entités pour notre base de données. L'entité “User” stocke les utilisateurs de notre site et leur mot de passe. Une entité “Scan” reliée à un ou plusieurs utilisateurs et une entité “Result” reliée à un “Scan”. Cela donne le schéma entités suivant :

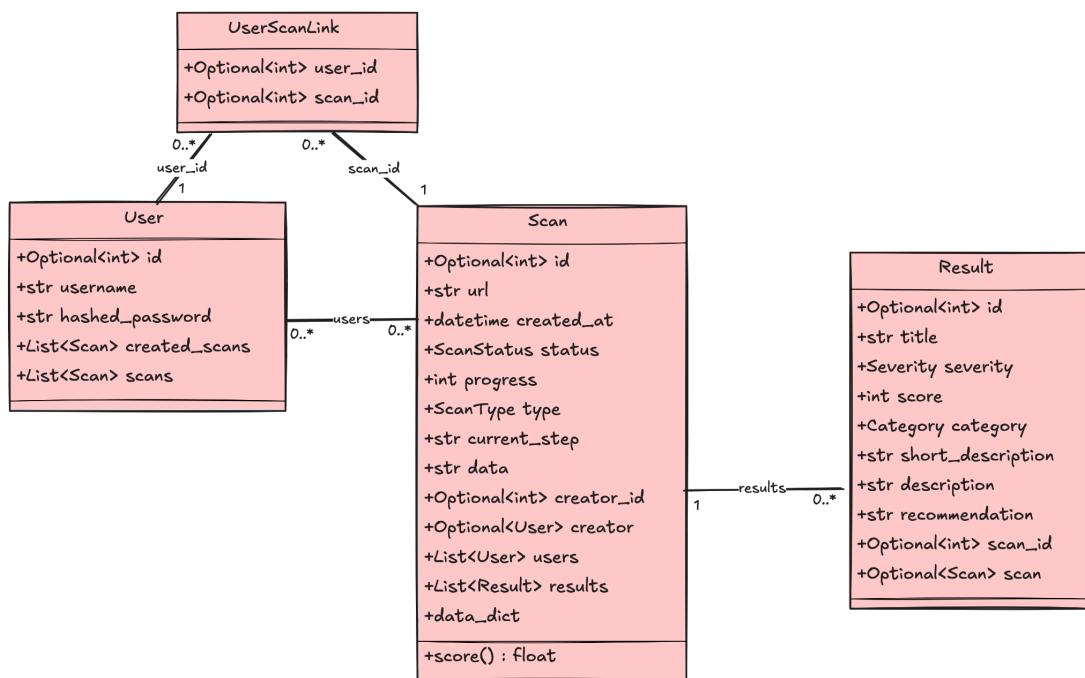


Figure 9 - Diagramme d'entités créé en base de données

3.3.3.2. REST API

Cette base de données a ensuite conditionné les routes api que nous avons créées. Chacune de ces routes ont une utilité distincte et répondent à des besoins spécifiques.

Gestion des scans

méthode	route	description
GET	/api/scans	Lister les scans d'un utilisateur
GET	/api/scans/{scan_id}	Lire un scan à partir de son identifiant
POST	/api/scans	Créer un scan et le lancer
DELETE	/api/scans/{scan_id}	Supprimer un scan de la base de donnée

Tableau 7 - Routes api pour la gestion des scans

Récupération des résultats à partir d'un scan :

méthode	route	description
GET	/api/scans/{scan_id}/results	Lire les résultats à partir d'un scan
GET	/api/scans/{scan_id}/report	Génération d'un rapport PDF pour impression/export

Tableau 8 - Routes api pour la récupération des résultats

Finalement, il fallait concevoir la méthode pour accéder aux résultats de manière authentifiée. Nous avons donc ajouté les routes concernant l'**authentification** suivantes à la liste préexistante :

méthode	route	description
POST	/api/users	Enregistrer un nouvel utilisateur
POST	/api/auth/login	Connecter un utilisateur

Tableau 9 - Routes api pour l'authentification

3.3.3.3. Système d'authentification

Pour le système d'authentification, nous avons choisi d'utiliser des [jetons web JSON \(JWT\)](#) une technologie standard répondant à nos besoins de sécurité. En utilisant [FastAPI](#) et en spécifiant les types des différents objets de [l'API](#), nous avons aussi une documentation automatiquement générée ainsi que le contrat [REST](#).

3.3.3.4. Orchestration des Scans

Après avoir géré les parties [API](#) de notre backend, il resterait à concevoir la partie orchestration des travaux de scans. Nous avons eu dès le début l'idée de séparer les différents outils dans des [conteneurs](#). Ceci nous simplifierait la gestion des dépendances et nous assurerait que les outils seraient indépendants les uns des autres, nous permettant de les tester unitairement.

Le backend est donc responsable de la création, de la configuration et de l'exécution de ces outils via un orchestrateur de tâches.

Cet orchestrateur prend comme source un fichier de configuration définissant les scans simples et complexes et ensuite produit des pipelines. Nous avons adopté un fonctionnement modulaire dans lequel les modules sont importés en runtime. Et pour que ce système fonctionne, nous avons dû mettre en place un format standard de Job à respecter. Tous les jobs héritent de la classe abstraite Job qui définit les fonctions de base et permet d'ajouter simplement un nouveau job sans modifier l'architecture de l'application.

Une contrainte apportée par le choix d'un déploiement Kubernetes et notre volonté d'utiliser des conteneurs pour mieux gérer les dépendances est qu'il est difficile de faire du développement

local avec un cluster [Kubernetes](#) (difficile, pas impossible). Nous avons donc choisi [Docker](#) pour le développement, mais cela nous a forcés à écrire une couche d'abstraction entre l'application et docker ou Kubernetes (en fonction des environnements). Donc en environnement de développement, l'orchestrateur utilise le [socket docker](#) pour lancer des conteneurs et en environnement de production et de staging, l'application s'interface avec l'api [Kubernetes](#).

Maintenant la question d'architecture résolue, nous avons dû adapter certaines applications à notre usage. Une liste exhaustive des outils utilisés et considérés est disponible dans l' [Annexe 7: Outils open source](#). En effet, certains projets que nous souhaitions utiliser étaient disponibles sur [GitHub](#), mais ne proposaient pas de [conteneurs](#). Nous avons donc effectué un travail de conteneurisation de certaines de ces applications. À ce jour, nous avons 13 [pull requests](#) d'ouvertes sur des projets open source concernant la [conteneurisation](#) automatique de ces projets. Nous avons proposé l'utilisation des [GitHub actions](#) et du [GitHub Container Registry](#) (GHCR) pour automatiser la publication d'images docker sur chaque commit de la branche principale. Nous avons la volonté de mettre notre travail (que nous aurions réalisé de toute façon) au service de la communauté en aidant ces projets dans leur développement.

3.3.3.5. *Interprétation des Résultats*

Le dernier élément dans la chaîne d'un scan reste l'interprétation et la vulgarisation des résultats fournis par les différents outils. Nous utilisons une vingtaine d'outils et tous ne fournissent pas un format de sortie standard (en [JSON](#), [YAML](#) ou [XML](#)). Nous avons donc dû écrire la logique pour analyser la sortie standard des outils et fournir une sortie en [JSON](#). Étant donné que l'analyseur de sortie standard est spécifique à chaque outil, cette logique a été intégrée dans le Job pour lequel il est responsable.

Enfin, il reste la tâche de vulgarisation des éléments de sortie du scan. Nous avons décidé qu'il fallait quatre éléments principaux à un résultat : le **titre** du résultat, son **score** (positif si cela améliore la sécurité du site, négatif sinon), une **description** et une **recommandation**.

Pour cela, nous avons eu une approche hybride. Certains outils (comme [wapiti](#)) produisent déjà une description et une recommandation que nous pouvons directement stocker et afficher aux clients. En revanche, certains outils (comme [Nikto](#) ou [Nmap](#)) produisent des sorties techniques, sans description. Nous utilisons donc une IA pour générer les descriptions et recommandations aux utilisateurs. Nous avons écrit un microservice IA, responsable de ce travail de vulgarisation qui lui aussi est modulaire et peut être remplacé par un module plus performant par exemple.

3.3.3.6. *Calcul du Score*

Au fur et à mesure du scan, un score est calculé. Nous avons décidé de partir d'une base de 100 points pour tout site web, 100 étant le maximum de points que nous pouvons donner à un scan.

Par la suite, une simple somme de tous les scores individuels des résultats est effectuée. Les scores individuels sont calculés directement en fonction d'une sortie d'un scanneur. Par exemple, si nmap détecte que le port 22 est ouvert, cela représente une mauvaise pratique en termes de cybersécurité, il lui est donc attribué le score de -3. Le score est choisi par les développeurs du module sachant que des directives générales en termes de score sont données : de -40 à -20 pour les vulnérabilités **critiques**, de -20 à -10 pour les **erreurs**, de -10 à -3 pour les **warnings** et de -3 à 10 pour les **infos**. Quant à lui, **débug** vaut 0 points en général.

Les scores n'étant pas donnés par de l'IA, ils sont déterministes et nous avons donc codé la fonctionnalité historique de scans qui permet de comparer les résultats de scans d'un même site en

fonction du temps. Nous espérons que cela peut encourager les clients à améliorer leurs pratiques en cybersécurité en leur montrant les effets concrets de leurs efforts.

3.3.3.7. *Export du rapport*

Le rapport de scan n'est pas seulement disponible sur la plateforme, nous avons également intégré une fonctionnalité d'**export** d'un scan au **format PDF** pour impression ou partage via mail... Nous avons fait le choix d'utiliser une bibliothèque python pour nous simplifier la tâche, pour cela, nous utilisons la bibliothèque **reportlab**.

3.3.3.8. *Conclusion*

Cette architecture constitue la base fonctionnelle et technique de **WeakSpotter**, bien qu'elle ne soit pas parfaite et qu'elle puisse être améliorée. Nous avons abordé ces améliorations dans la section **Perspectives d'évolutions**. Cette approche modulaire et flexible permet de répondre aux exigences du cahier des charges, tout en laissant de la place pour de futures évolutions.

3.3.4. Microservice IA

Une des parties les plus cruciales de notre projet est la **vulgarisation des vulnérabilités**, un élément central de notre cahier des charges. Étant donné son importance, nous avons voulu rendre cette fonction **modulaire**, afin qu'elle puisse évoluer indépendamment du backend principal.

Un problème s'est posé dès le début, la quantité colossale de vulnérabilités. Il existe plus de 200 000 [CVE](#) répertoriées, rendant leur vulgarisation manuelle irréaliste. Nous avons donc pris la décision d'automatiser cette tâche quitte à perdre en qualité des résultats. Notre choix s'est alors porté sur les [LLM](#), des modèles de langages généralistes, pas forcément experts dans le domaine de la cybersécurité, mais spécialisés dans l'art de la reformulation.

En plus d'utiliser les LLM, notre microservice s'appuie sur les bases de données du MITRE et du NVD pour compléter les données fournies par le backend principal. Ces bases de données contiennent des [CVE](#). Une [CVE](#) est une façon standard de faire référence à une vulnérabilité, et ce qui nous intéresse dans ces bases, c'est qu'elles contiennent une description technique et les différentes méthodes pour les corriger. Les LLM, combinés avec les informations des bases de données, forment une solution idéale pour répondre aux besoins de vulgarisation.

3.3.4.1. *Prompt Engineering*

Plusieurs techniques existent pour intégrer des données externes à une inférence d'IA (RAG, tool calling...). Nous avons choisi d'intégrer directement les données au prompt initial. C'est une approche naïve, mais qui semble fonctionner dans notre cas d'utilisation.

Voici un exemple de **prompt** utilisé pour générer une description simplifiée :

"Vulgarise cette vulnérabilité CVE en termes simples, compréhensibles pour une personne n'ayant aucune connaissance en cybersécurité. Utilise un langage clair et ne soit pas long dans ta réponse si cela n'est pas nécessaire. Ne renvoie que l'explication vulgarisée sous forme de texte brut, sans JSON ni informations techniques complexes."

Figure 10 - Exemple de prompt utilisé pour générer une description

Nous incluons dans ce prompt les données relatives à la CVE sous format **JSON**, comprenant l'identifiant de la CVE, sa description technique et son score de gravité.

3.3.4.2. *Interfaçage avec Groq*

En parallèle avec le travail de **prompt engineering**, nous avons fait une analyse comparative des différents LLM disponibles sur le marché et de la faisabilité de faire de l'inférence en local avec notre propre matériel. Les caractéristiques que nous avons comparées包含 la vitesse de réponse du service, la qualité de la réponse et le prix du service. Nous avons pris en compte les services de OpenAI, Mistral AI, Anthropic, Groq (à ne pas confondre avec Grok) et de Ollama (en local). Bien que les modèles de Groq ne donnaient pas forcément les meilleurs résultats, ils présentaient une rapidité sans comparaison et est proposé gratuitement jusqu'à 3 millions de [tokens](#) par mois. Faire de l'inférence en local était aussi gratuit, mais trop lent et manquant de qualité par rapport aux plus gros modèles.

Groq propose "*llama-3.3-70b-versatile*", un modèle d'IA créé par Meta, c'est ce modèle que nous avons choisi pour ce microservice. Notre [microservice](#) inclut donc un client interfaçant Groq et notre backend Python.

3.3.4.3. Base de Données et API REST

Pour optimiser notre [microservice](#) et éviter de gaspiller des [tokens](#) sur des vulnérabilités déjà traitées, nous avons mis en place un système de cache. Ce système stocke dans une base de données, les résultats des vulgarisations.

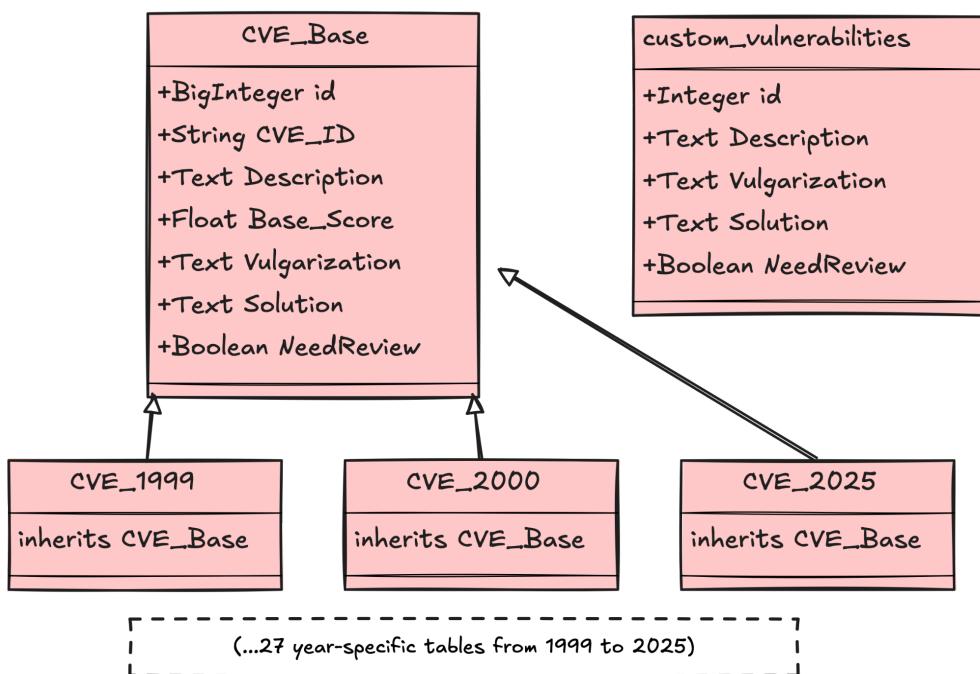


Figure 11 - Schéma d'Entités pour le microservice IA

Ce système nous permet aussi de pouvoir modifier des vulgarisations et de modérer notre plateforme dans le cas où le [LLM](#) générera des vulgarisations fausses ou invalides.

En suivant le modèle d'une [CVE](#), nous avons conçu une api [REST](#) composée de ces routes :

Gestion des CVE (Récupération et vulgarisation)

Méthode	Route	Description
GET	/cves/get_cve_by_id/{cve_id}	Récupérer une CVE avec sa vulgarisation et solution
GET	/cves/get_cves_by_ids	Récupérer plusieurs CVE avec leurs vulgarisations et solutions
GET	/cves/{year}	Récupérer toutes les CVE d'une année donnée (sans vulgarisation)

Tableau 10 - Routes api pour la gestion des CVE

Extraction de CVE depuis un texte

Méthode	Route	Description
POST	/cves/from_text	Extraire les CVE d'un texte et renvoyer leur vulgarisation

Tableau 11 - Routes api pour l'extraction de CVE

Gestion des vulnérabilités personnalisées

Méthode	Route	Description
POST	/add_custom_vulnerability	Ajouter une vulnérabilité personnalisée et générer sa vulgarisation
GET	/get_custom_vulnerabilities	Récupérer toutes les vulnérabilités personnalisées avec leurs vulgarisations

Tableau 12 - Routes api pour la gestion des vulnérabilités

Mise à jour des vulnérabilités

Méthode	Route	Description
PUT	/update	Mettre à jour une CVE ou une vulnérabilité personnalisée avec une nouvelle vulgarisation/solution

Tableau 13 - Routes api pour la mise a jours des vulnérabilités

Revue manuelle

Méthode	Route	Description
GET	/need_review	Récupérer toutes les CVE/vulnérabilités nécessitant une vérification manuelle

Tableau 14 - Routes api pour la revue manuelle

Et comme cette api avait pour vocation d'être aussi bien accessible aux développeurs qu'au backend principal, nous avons exposé cette api publiquement. Pour autoriser seulement notre équipe à utiliser ce service, nous avons mis en place un système d'authentification simple par clé api.

3.3.4.4. Conclusion

Bien que simple de l'extérieur, ce [microservice](#) à nécessité un travail important. Il a fallu faire des choix techniques, analyser les différentes solutions disponibles et les comparer.

Nous avons dû aussi faire des compromis, notamment sur la qualité des résultats, mais nous sommes satisfaits du résultat final.

Ce [microservice](#) est un élément clé de notre projet et en même temps un élément qui peut bénéficier le plus de l'ajout de nouvelles fonctionnalités.

3.3.5. Déploiement

Nous avons choisi arbitrairement d'utiliser [Kubernetes](#) pour déployer notre plateforme. Voulant adopter l'approche [GitOps](#) pour faciliter la collaboration au niveau opérationnel, nous y avons installé [ArgoCD](#). Notre déploiement est donc stocké sur GitHub et automatiquement synchronisé avec le [Cluster](#). Ce déploiement peut être vu en trois parties :

Le **Frontend** comprend un [déploiement](#), un [service](#) et un [ingress](#). Le **Backend** reprend ces éléments avec, en complément, un [volume persistant](#) et un [service account](#) avec des droits nécessaires pour créer des [pods](#) dans son [namespace](#).

Les **secrets** quant à eux sont stockés sur le serveur, mais nous aurions pu utiliser une solution comme "[sealed secrets](#)" pour les stocker sur [GitHub](#) et les versionner avec le reste du déploiement.

Pour garantir la sécurité des échanges entre nos clients et notre serveur, nous utilisons [HTTPS](#). Pour cela, nous avons Configuré les enregistrements DNS pour ce nom de domaine : "[*.weakspotter.ozeliurs.com](#)". Nous avons aussi installé un [opérateur](#) Kubernetes, "[certmanager](#)", qui permet l'obtention automatique de certificats pour le chiffrement HTTPS.

Finalement, nous servons le frontend et le backend sur le même domaine grâce à "[traefik](#)", un [reverse proxy](#) que nous avons configuré de la façon suivante :

- https://weakspotter.ozeliurs.com/api → http://backend/api
- https://weakspotter.ozeliurs.com → http://frontend

nb: l'ordre est important, la première règle est traitée en priorité.

3.4. Gestion de projet

3.4.1. Organisation du Travail

Dès le début de ce PER, nous avons dû gérer plusieurs dépôts et nous avons rapidement pris la décision de créer une organisation GitHub pour regrouper l'ensemble des dépôts sous une seule entité. Nous avons donc contribué sous l'organisation “WeakSpotter”, disponible sur GitHub: <https://github.com/WeakSpotter>.

Le Frontend principal et le Backend principal sont sur le dépôt “WeakSpotter/WeakSpotter”, aussi disponibles sur <https://github.com/WeakSpotter/WeakSpotter>. Ce dépôt centralisé est au cœur de notre développement et sera principalement évoqué dans cette partie de la documentation concernant la gestion de projet.

Nous avons adopté une stratégie de ramifications qui s'inspire des stratégies “Git Flow” et “GitHub Flow”. Notre stratégie repose fondamentalement sur deux branches : “main” qui représente notre version stable du projet, “develop” la partie instable de notre code. Nous allons plus en détail dans les détails de cette stratégie de ramifications dans notre documentation : <https://github.com/WeakSpotter/WeakSpotter/wiki/Contributing#branching-strategy->

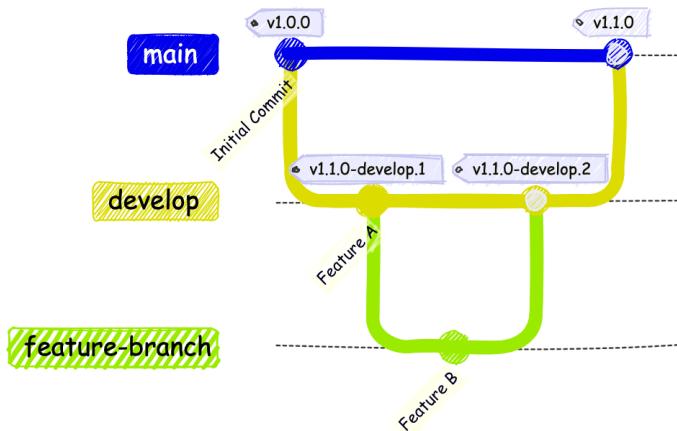


Figure 12 - Schéma de stratégie de ramifications

Concernant les “commits”, nous avons choisi de suivre la convention de commit Conventionnal Commits. Cette convention permet de décrire précisément les changements effectués dans chaque commit en suivant un format standardisé. Cela simplifie la collaboration, notamment lors de l'examen des modifications et permet une gestion efficace de l'historique des commits, tout en étant nécessaire pour notre flux d'intégration continue.

Notre intégration continue intègre les étapes classiques. Dans un premier temps, une étape de linting se lance, formatant le code de manière standard. Ensuite, notre flux de travail enchaîne avec SonarQube et inspecte le Python du backend et le TypeScript du frontend et nous assure une qualité de code. Et une fois cette étape de passée, Semantic Release package et fait une release du code sur notre GitHub, ayant généré la version et les notes de version en fonction des commits contenus dans cette release.

Toutes les versions gérées par [semantic release](#) sont accompagnées de deux [images docker](#) sur le GitHub Container Registry, taguées avec leur branche d'origine. Cela nous permet d'automatiser la partie déploiement continu avec [ArgoCD](#).

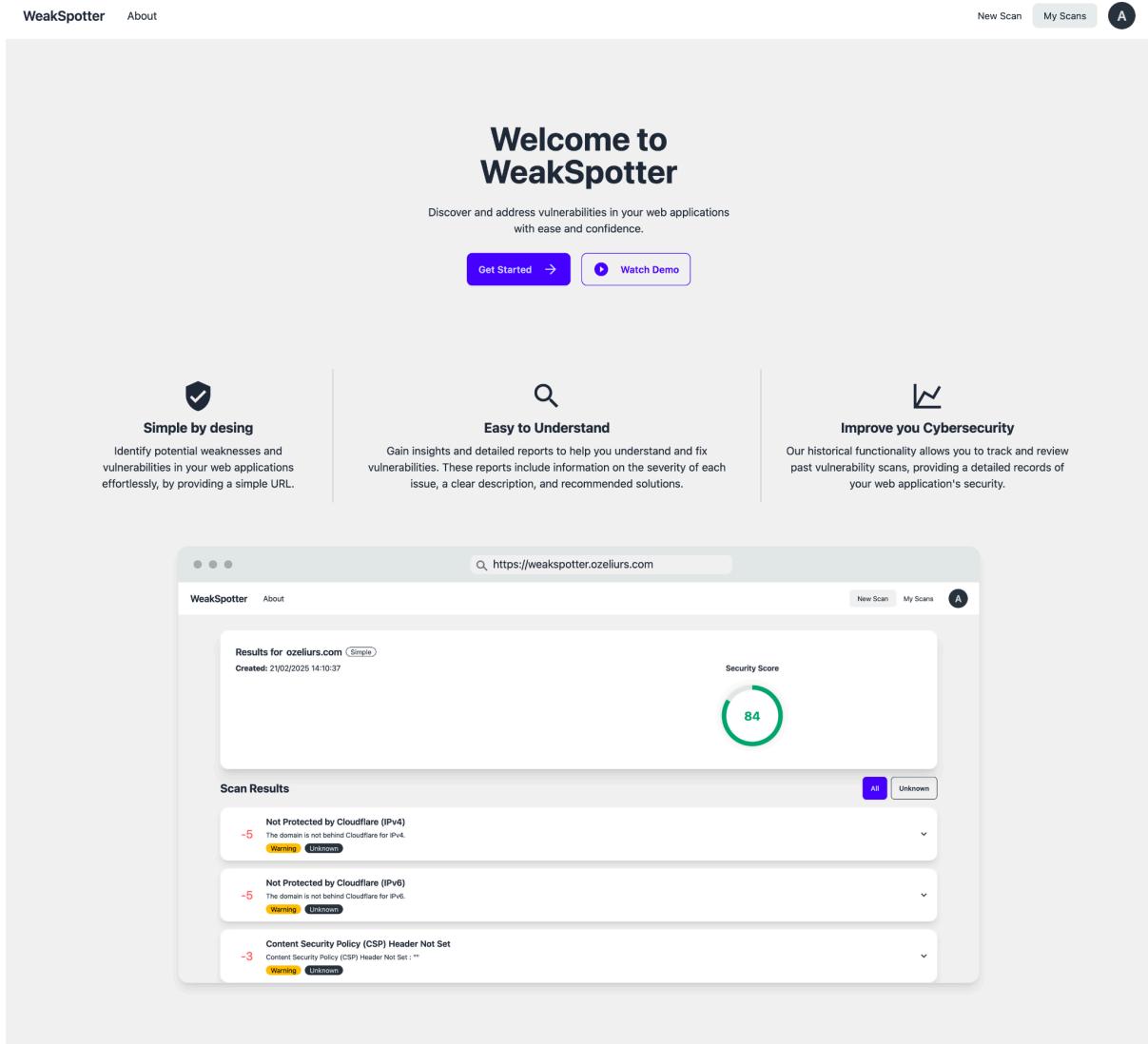
Pour notre [déploiement continu](#), nous avons fait le choix d'utiliser “[ArgoCD](#)” accompagné de “ArgoCD Image Updater”. Cette solution nous permet de gérer les deux environnements en place, l'environnement de production, qui est une représentation de la dernière version stable disponible sur notre dépôt et l'environnement de staging, une représentation de la branche develop. Ces environnements sont quasiment identiques, à quelques différences près, ce qui nous permet de tester notre code avant sa mise en production.

3.4.2. Licence

Pour la licence, nous avons choisi [GPLv3](#) (GNU General Public License Version 3), une licence open-source. Ce choix s'explique par sa forte présence parmi nos outils d'analyse et par sa simplicité, bien que la licence MIT offre davantage de flexibilité.

3.5. Résultats Obtenus

Nous avons ainsi développé plusieurs pages pour l'interface graphique, que nous détaillerons dans cette section en présentant les différents éléments qui la composent. Il est important de souligner que l'ensemble de l'interface est responsive (cf. [Annexe 6: interface graphique téléphone](#)).



The screenshot shows the WeakSpotter homepage. At the top, there is a navigation bar with "WeakSpotter" and "About" on the left, and "New Scan", "My Scans", and a user icon on the right. The main heading is "Welcome to WeakSpotter" with a subtitle "Discover and address vulnerabilities in your web applications with ease and confidence." Below this are two buttons: "Get Started →" and "Watch Demo". The page is divided into three sections:

- Simple by design**: Features a shield icon and text: "Identify potential weaknesses and vulnerabilities in your web applications effortlessly, by providing a simple URL." A note says: "Our historical functionality allows you to track and review past vulnerability scans, providing a detailed records of your web application's security."
- Easy to Understand**: Features a magnifying glass icon and text: "Gain insights and detailed reports to help you understand and fix vulnerabilities. These reports include information on the severity of each issue, a clear description, and recommended solutions."
- Improve your Cybersecurity**: Features a shield icon and text: "Our historical functionality allows you to track and review past vulnerability scans, providing a detailed records of your web application's security."

Below the homepage, a second screenshot shows the results for the domain "ozeliurs.com" (Simple scan). It displays a "Security Score" of 84 and a "Scan Results" section with three items:

- 5 Not Protected by Cloudflare (IPv4): "The domain is not behind Cloudflare for IPv4." Status: Warning, Unknown.
- 5 Not Protected by Cloudflare (IPv6): "The domain is not behind Cloudflare for IPv6." Status: Warning, Unknown.
- 3 Content Security Policy (CSP) Header Not Set: "Content Security Policy (CSP) Header Not Set : **" Status: Warning, Unknown.

Figure 13 - Page d'accueil WeakSpotter

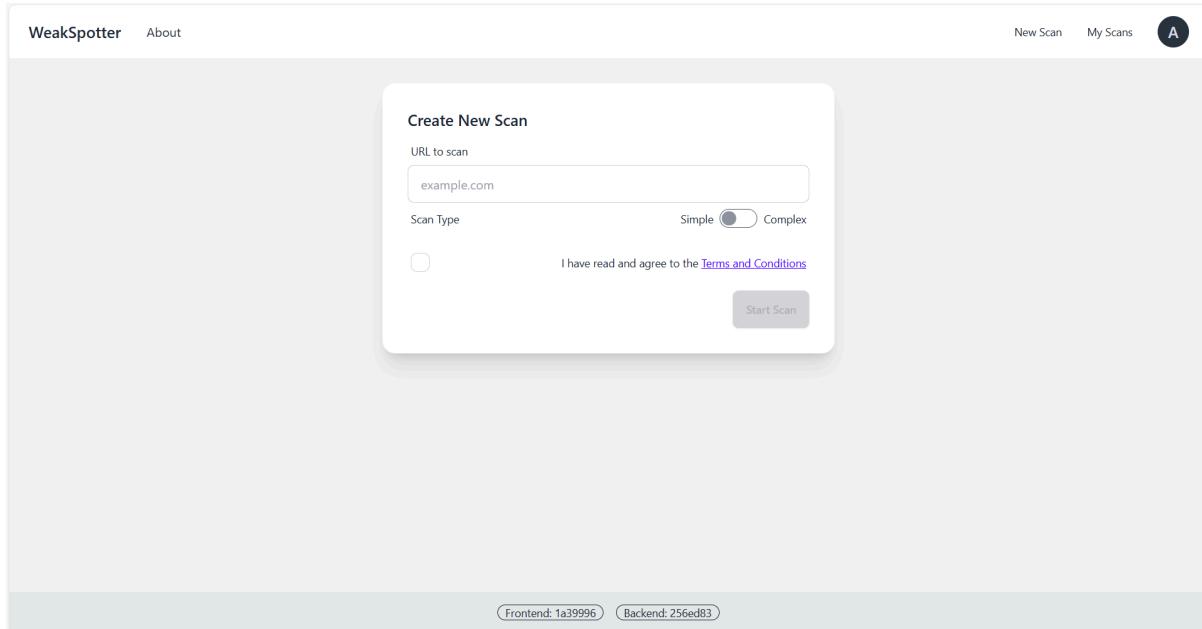
La page d'accueil présente une interface graphique typique d'une analyse SEO, comprenant un texte d'introduction qui décrit l'objectif de WeakSpotter et incite à créer le scan. Juste en dessous, les trois principales caractéristiques de WeakSpotter sont mises en avant : la simplicité d'utilisation, la facilité de compréhension et la fonctionnalité d'historique.

En dessous de ces éléments, un exemple de rapport de scan est affiché pour illustrer les résultats obtenus.

La page d'accueil joue un rôle central puisqu'elle permet d'accéder à toutes les autres sections. À tout moment, l'utilisateur peut y revenir en cliquant sur le logo **WeakSpotter**.

Avant de commencer à scanner un site web, il est nécessaire de s'authentifier ou de s'enregistrer par le système classique. Suite à cette authentification, l'utilisateur a la possibilité de pouvoir scanner un site web et d'accéder à l'historique des scans (et seulement ceux-ci) grâce au bouton **New Scan** et **My Scans** respectivement.

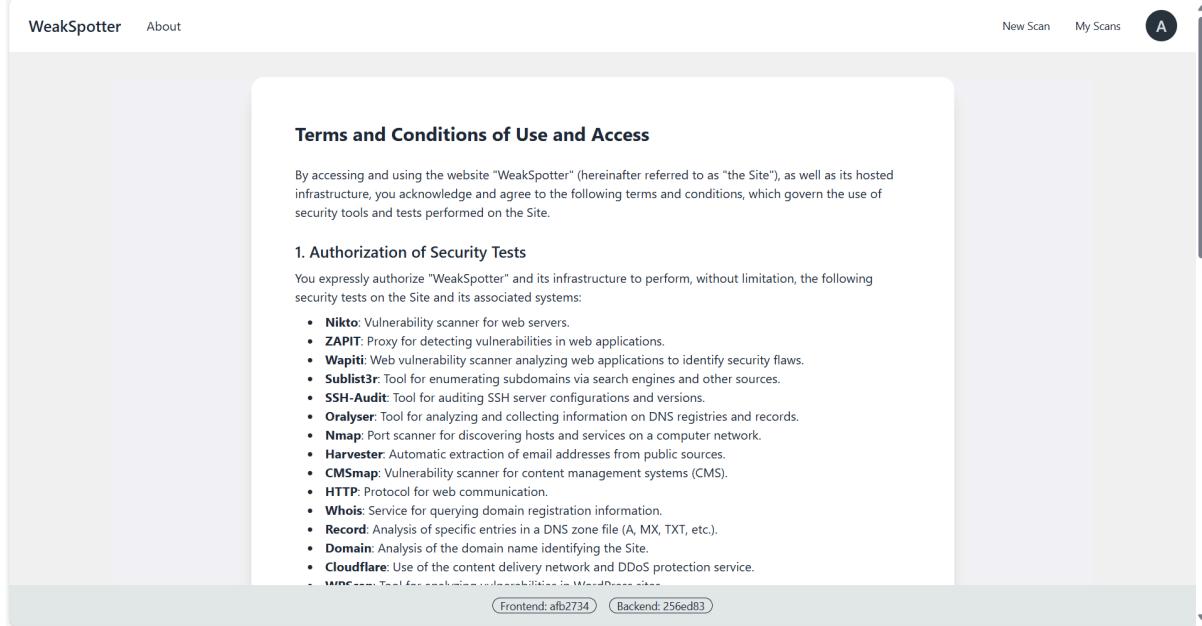
Il est alors possible de créer un scan depuis la page d'accueil en cliquant soit sur le bouton **New Scan** ou sur **Get Started**.



The screenshot shows the 'Create New Scan' interface. At the top, there's a navigation bar with 'WeakSpotter' and 'About' on the left, and 'New Scan' and 'My Scans' on the right. Below the navigation is a large input field labeled 'URL to scan' containing 'example.com'. To the right of this field is a 'Scan Type' section with a switch between 'Simple' (selected) and 'Complex'. Below the URL field is a checkbox followed by the text 'I have read and agree to the [Terms and Conditions](#)'. At the bottom right of the form is a 'Start Scan' button. The footer of the page includes the text 'Frontend: 1a39996 Backend: 256ed83'.

Figure 14 - Page début de scan sélection simple

Cette page permet de configurer un nouveau scan, il faut simplement entrer l'URL du site à analyser, sélectionner la complexité du scan (simple figure X ou complexe figure X). Par la suite, vu que le scan est invasif, il faut accepter les **conditions générales** (figure X).



The screenshot shows a web browser window with the URL <https://weakspotter.com>. The page title is "Terms and Conditions of Use and Access". At the top right, there are buttons for "New Scan" and "My Scans", and a user icon with the letter "A". The main content area contains the text of the terms and conditions, which is a standard legal document. At the bottom of the page, there is a footer bar with two status indicators: "Frontend: afb2734" and "Backend: 256ed83".

Figure 15 - Page des Termes et Conditions

Une fois l'URL renseignée et les conditions acceptées, l'analyse peut commencer automatiquement.

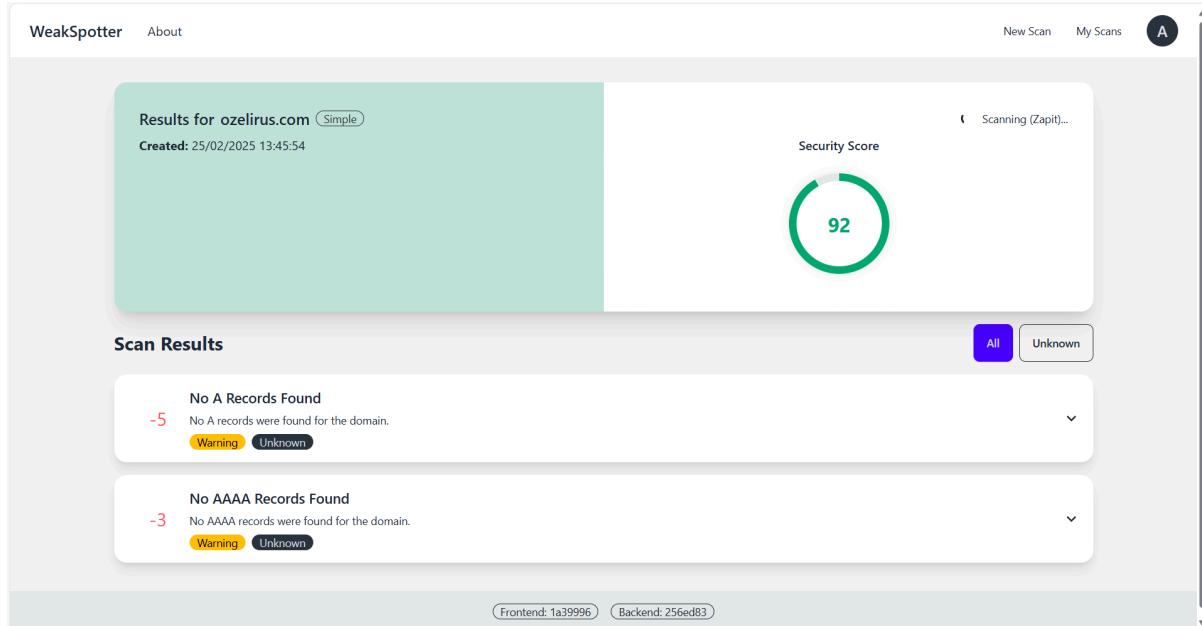


Figure 16 - Page de résultats de scan

Pendant l'analyse, une barre de progression verte indique l'avancement du scan en fonction du temps et des étapes réalisées. Il y a également une indication de l'outil en cours d'utilisation durant l'analyse. Chaque vulnérabilité détectée est affichée en temps réel et contribue au **calcul du score global**.

Chaque résultat permet ainsi de calculer le score global par un score déduit par vulnérabilité trouvée.

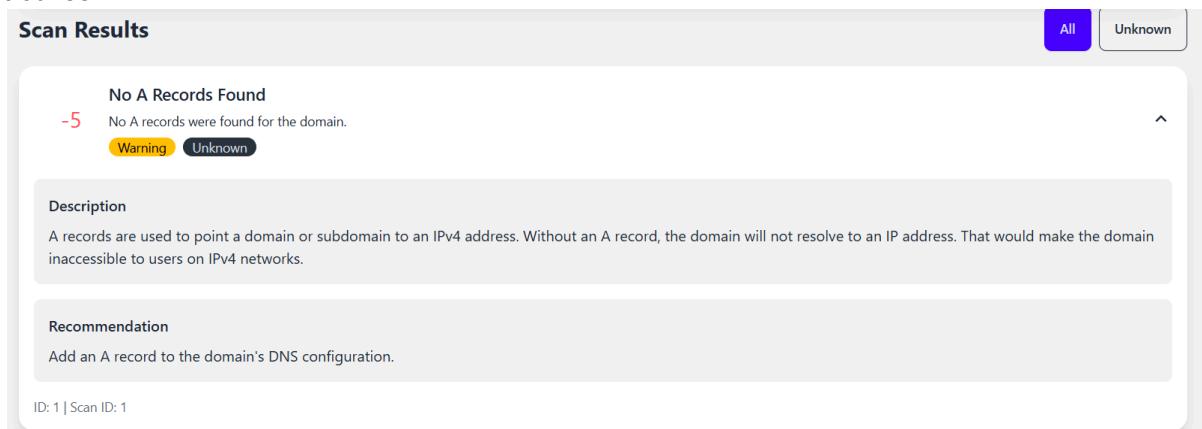
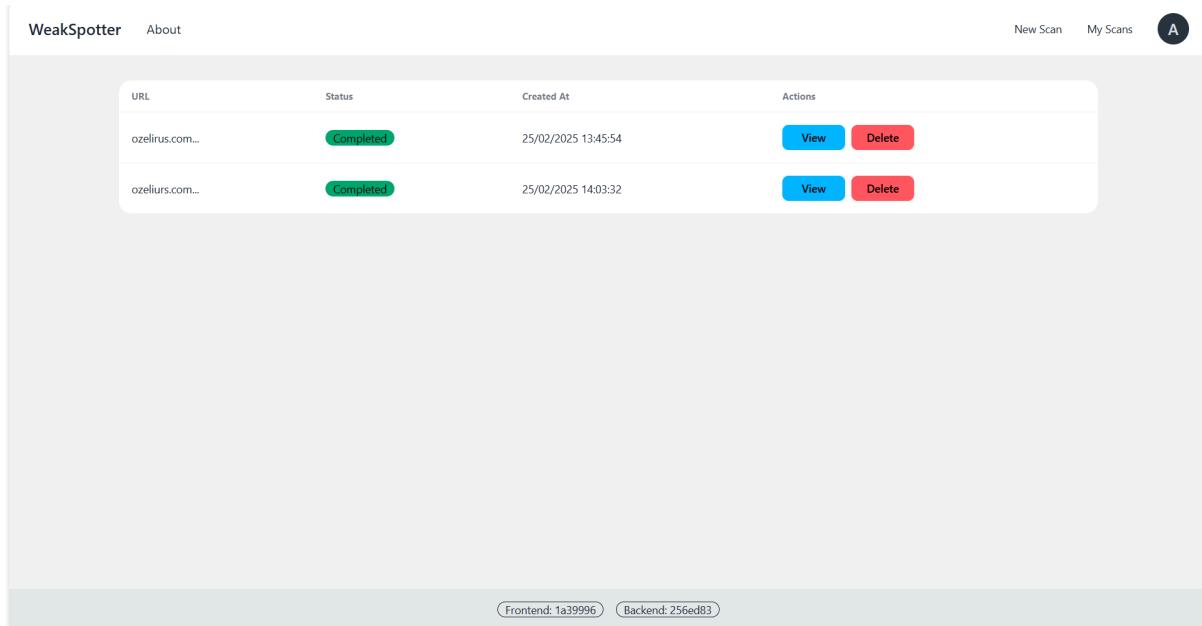


Figure 17 - détails sur un résultat de scan

Chaque vulnérabilité trouvée est présentée de la même façon :

- Le titre de la vulnérabilité
- un score
- une courte description
- des catégories de严重性 (info, debug, warning, error, critical)
- une description détaillée
- une recommandation

Il est possible d'avoir un historique des sites scannés en cliquant sur **My Scans**



URL	Status	Created At	Actions
ozelirus.com...	Completed	25/02/2025 13:45:54	View Delete
ozelirus.com...	Completed	25/02/2025 14:03:32	View Delete

Frontend: 1a39996 | Backend: 256ed83

Figure 18 - détails sur un résultat de scan

Il est alors possible d'accéder à tous les scans réalisés pour un utilisateur défini.

De plus, en cliquant sur **History**, nous avons une évolution du score d'un site web en fonction du temps.

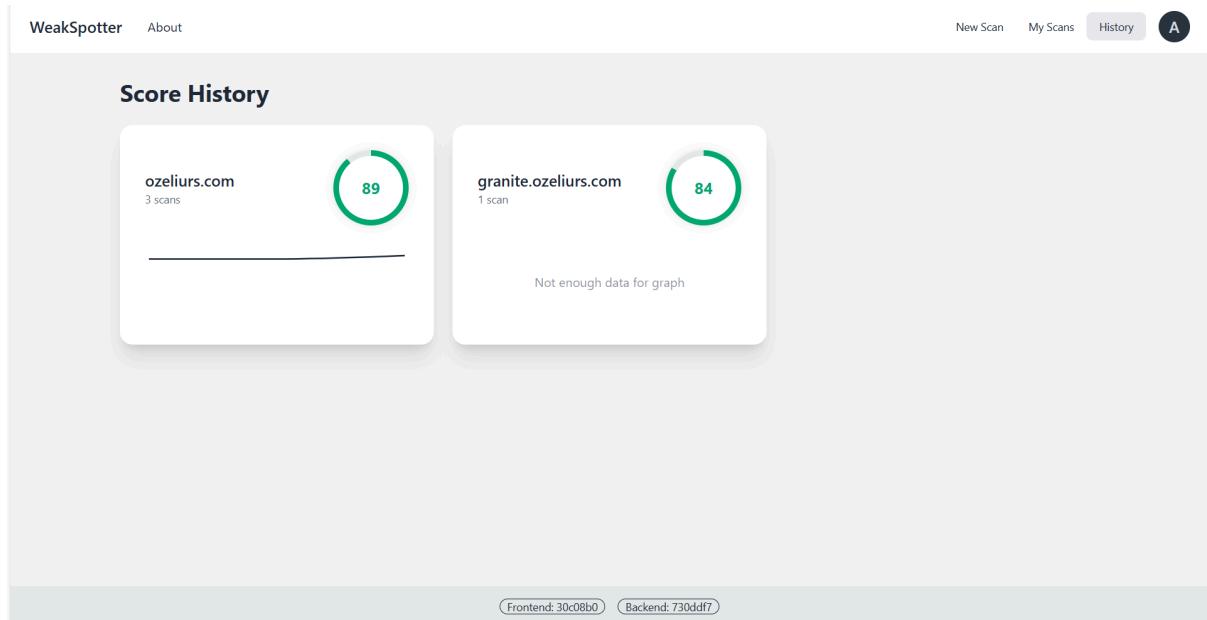


Figure 19 - historique de scan d'un site

Finalement, la page **About** regroupe diverses informations, notamment un récapitulatif des auteurs et leurs contacts, une mention des outils open-source utilisés ainsi qu'un tableau détaillant les outils employés en fonction du type de scan réalisé.

3.6. Problèmes rencontrés

3.6.1. Gestion de Projet

Au sein de notre équipe, la répartition du travail avait souvent été déséquilibrée, ce qui avait empêché une véritable collaboration entre les membres. Les tâches n'étaient pas réparties de manière égale, ce qui avait entraîné des retards et des erreurs. Cette situation avait eu un impact négatif sur la motivation et l'efficacité globale de l'équipe.

3.6.2. Technologies Utilisées

Bien que la technologie choisie pour le backend corresponde aux compétences de l'ensemble de l'équipe, ce n'était pas le cas pour toutes les autres technologies. Par exemple, seul un membre de l'équipe maîtrisait React avant ce projet, ce qui a ralenti le développement de l'interface. La même difficulté s'est posée avec Kubernetes.

3.6.3. Microservice IA

L'un des problèmes rencontrés concerne les fichiers JSON fournis par MITRE et NVD. Leur format évolue au fil des années, rendant l'extraction des informations plus complexe.

- **Changement de structure** : Les descriptions des CVE ne sont pas toujours situées au même endroit selon l'année ou l'organisation source.
- **Variabilité des scores** : Le score de criticité (CVSS) peut être stocké sous différentes clés ou suivre différentes versions (CVSSv2, CVSSv3).

Pour pallier ces incohérences, un travail de **prétraitement des données** a été nécessaire afin d'harmoniser les formats, extraire correctement les champs clés et structurer une base de données cohérente et exploitable pour l'API IA.

3.7. Perspectives d'évolution

Plusieurs éléments de ce projet sont destinés à évoluer, mais par manque de temps dans le cadre de ce PER, nous n'avons pas pu les modifier. Nous allons vous les présenter ci-dessous.

3.7.1. Frontend

Actuellement, nous utilisons la bibliothèque Axios dans notre frontend pour définir un client API. Une amélioration potentielle serait de générer automatiquement ce client API Axios à partir du contrat Open API généré par le backend, afin de garantir une cohérence et une synchronisation entre les deux parties de l'application. Car pour le moment ces routes sont générées manuellement. De plus, nous pourrions utiliser un store au lieu d'avoir plusieurs states au niveau de certains éléments comme le status d'un scan.

Il serait aussi intéressant de faire des tests utilisateurs. Dans le cadre de notre PER, nous n'avons pas eu le temps d'en effectuer, nous nous sommes concentrés sur son développement.

3.7.2. Backend

Pour le moment, nous utilisons des jetons web JSON (JWT) pour gérer l'authentification au sein de notre infrastructure, une technologie standard et largement adoptée. Cependant, il serait envisageable d'externaliser ce service plutôt que de le gérer nous-mêmes, car cela représente à la fois un risque en termes de sécurité et une charge de maintenance supplémentaire. Un outil comme Keycloak pourrait être une solution pertinente pour simplifier cette gestion tout en renforçant la sécurité.

Le manque de temps nous a empêché de réaliser des tests pour notre backend, ce qui signifie que les fonctionnalités de l'application n'ont pas été suffisamment validées. Il serait intéressant de tester unitairement et de bout en bout notre application.

3.7.3. Microservice IA

Même avec l'utilisation d'un **LLM** suivi d'une **revue manuelle**, certaines vulgarisations et solutions peuvent rester complexes pour un public non expert. Pour pallier ce problème, la mise en place d'un **système de signalement** permettrait aux utilisateurs d'indiquer lorsqu'une explication reste trop technique ou imprécise. Une fois signalée, la description pourrait être **retravaillée**, garantissant ainsi une meilleure accessibilité et une adaptation continue aux besoins des utilisateurs.

4. Conclusion

La cybersécurité est un enjeu majeur pour toutes les entreprises, y compris les TPE et PME, qui sont souvent les plus vulnérables face aux attaques informatiques. Notre analyse a mis en évidence un manque d'outils accessible, non adapté à leurs besoins et leurs moyens financiers.

Face à ce constat, nous avons développé WeakSpotter, un scanner de vulnérabilité automatisé et accessible à tous. Inspiré par les outils d'analyse SEO, il propose une interface intuitive permettant d'effectuer un audit de cybersécurité en un simple clic, sans nécessiter d'expertise technique. Grâce à l'intégration d'une IA pour vulgariser les résultats, WeakSpotter fournit des recommandations compréhensibles et exploitables par tous, rendant la cybersécurité plus accessible aux petites entreprises.

Les résultats obtenus montrent que notre solution répond efficacement aux attentes des TPE et PME, leur permettant d'identifier et de corriger leurs vulnérabilités avec un minimum d'effort. Toutefois

Ce projet de développement d'un scanner de vulnérabilités proactif pour la cybersécurité web a été une expérience enrichissante. Notre objectif principal était de concevoir et de développer une solution qui puisse aider les petites et moyennes entreprises (PME) à identifier et à corriger les faiblesses de sécurité dans leurs applications web. Nous avons respecté le cahier des charges et nous pensons vraiment avoir répondu à la problématique posée dans notre état de l'art.

Annexes

Annexe 1 : Lexique

Cette section présente les termes techniques et spécifiques utilisés dans ce rapport, ainsi que leurs définitions pour faciliter la compréhension

- **TPE** : Très Petite Entreprise, ce sont des entreprises de moins de 10 salariés et un chiffre d'affaires inférieur à 2 millions d'euros
- **PME** : Petit et Moyenne entreprise, ce sont des entreprises entre 10 et 250 salariés et un chiffre d'affaires inférieur à 50 millions d'euros
- **Fuite de données** : Une situation où des informations sensibles ou confidentielles (comme des données personnelles) sont accidentellement ou intentionnellement accessibles à des personnes non autorisées.
- **Rançongiciels (Ransomware)** : Un type de logiciel malveillant qui bloque l'accès aux fichiers ou systèmes d'une entreprise, en demandant une rançon pour les débloquer.
- **Défacement** : Une cyberattaque où un pirate modifie l'apparence d'un site web, souvent pour afficher un message ou un contenu non autorisé.
- **ANSSI** : Autorité française de cybersécurité qui protège les systèmes informatiques nationaux contre les cybermenaces. Elle conseille, prévient et régule la sécurité numérique des organisations publiques et privées en émettant des recommandations et en certifiant des produits de sécurité.
- **Hameçonnage** : L'hameçonnage est une technique de cyberattaque malveillante visant à tromper des individus pour voler des informations sensibles en se faisant passer pour une source légitime via des emails, messages ou sites web falsifiés.
- **Cyberattaque** : Une action malveillante visant à perturber, voler ou endommager un système informatique ou des données.
- **Vulnérabilité** : Une faiblesse dans un système, une application ou un réseau, que des pirates peuvent exploiter pour accéder ou nuire à des données ou des services.
- **Test de pénétration (Pentest)** : Une méthode pour évaluer la sécurité d'un système informatique en simulant des attaques pour identifier les failles.
- **Outils Open-source** : logiciels dont le code source est librement accessible, modifiable et redistribuable par tous.
- **Outils Libre** : logiciel dont l'utilisation, la modification et la duplication en vue de sa diffusion est permise.
- **IHM** : Interface Homme-Machine, domaine englobant l'UX (Expérience utilisateur) et UI (Interface utilisateur), améliorant le visuel et design d'une application.

- **Faille** : Une erreur ou une faiblesse dans un système informatique qui peut être exploitée par des pirates.
- **Cybercriminels** : Des individus ou groupes qui utilisent des technologies informatiques pour commettre des crimes, comme voler des données ou extorquer de l'argent.
- **Audit de sécurité** : Une analyse approfondie pour vérifier que les systèmes et données d'une entreprise sont bien protégés contre les menaces.
- **Flash audit** : Un audit rapide qui donne une vue d'ensemble de la sécurité d'une entreprise, pour identifier les points faibles en peu de temps.
- **SQL** : langage utilisé pour gérer les bases de données.
- **XSS** : Vulnérabilité qui permet d'injecter du code malveillant dans les pages web.
- **CVE** : système d'identification pour les vulnérabilités de sécurité connues.
- **Plugins** : Des extensions ou des modules ajoutés à un logiciel (souvent un CMS) pour lui ajouter des fonctionnalités spécifiques, par exemple un formulaire de contact ou un outil de sécurité.
- **LLM** : modèle de langage de grande taille est une IA entraînée sur de vaste quantité de texte pour comprendre et générer du langage naturel.
- **POC** : projet préliminaire qui démontre la faisabilité d'une idée, d'un concept ou d'une méthode.
- **SaaS** : SaaS est un modèle où les logiciels sont fournis et utilisés via Internet, généralement sur abonnement, sans installation locale.
- **Licence GPL** : licence libre créée par la Free Software Foundation. Elle permet aux utilisateurs de copier, modifier et redistribuer un logiciel sous certaines conditions.
- **Product Owner** : personne responsable de définir la vision du produit et de prioriser les fonctionnalités pour maximiser la valeur apportée par l'équipe de développement.
- **Frontend** : partie visible d'un site ou d'une application web, celle avec laquelle les utilisateurs interagissent, comme les boutons, les menus et les images.
- **TypeScript** : langage qui améliore JavaScript en ajoutant des règles pour éviter les erreurs et rendre le code plus clair et plus sûr.
- **React** : bibliothèque qui permet de créer des interfaces utilisateur avec des composants réutilisables.
- **DaisyUI** : bibliothèque de composants open-source.
- **CSS** : langage utilisé pour décrire l'apparence et la mise en forme d'un document écrit en HTML
- **Tailwind** : framework CSS qui permet de créer des styles directement dans le code HTML.

- **SPA** (*Single Page Application*) : application web qui fonctionne sur une seule page, en chargeant le contenu dynamiquement sans recharger toute la page, offrant ainsi une navigation plus fluide et rapide.
- **site web responsive** : site qui s'adapte automatiquement à tous les écrans (ordinateur, tablette, téléphone) pour offrir une expérience de navigation confortable.
- **Axios** : bibliothèque qui permet aux sites web de récupérer ou d'envoyer des données.
- **API** : (*Application Programming Interface*) interface qui permet à des applications de communiquer entre elles (ici entre le frontend et le backend) en échangeant des données de manière structurée
- **Hydratation** : processus par lequel une application web met à jour son état (donnée) côté client avec des données du côté serveur.
- **Backend** : partie invisible d'un site ou d'une application web, mais essentielle pour son bon fonctionnement. Il est responsable du traitement des requêtes et de la communication avec le frontend via des API.
- **Python** : langage de programmation connu pour sa simplicité, sa lisibilité et sa facilité d'apprentissage.
- **FastAPI**: framework de développement web qui permet de créer des API rapidement et efficacement, en offrant des performances élevées et en facilitant la gestion des données avec des validations automatiques.
- **ORM**: (*Object-Relation Mapping*) outils qui permettent de manipuler une base de données avec du code orienté objet, sans écrire directement des requêtes SQL.
- **sqlmodel**: bibliothèque python conçue pour interagir avec les bases de données SQL en utilisant des objets python.
- **SQLite**: base de données légère et intégrée qui ne nécessite pas de serveur.
- **MariaDB**: système de gestion des bases de données open source
- **PostgreSQL**: système de gestion de données relationnel open source.
- **JWT**: (*jetons Web JSON*) format de jeton utilisé pour représenter de manière sécurisée des informations entre deux parties sous forme de chaîne JSON.
- **REST**: style d'architecture qui permet aux applications de communiquer entre elles via des requêtes HTTP en utilisant des ressources identifiées par des URL.
- **conteneur Docker** : environnement léger, isolé et portable qui permet d'exécuter une application avec toutes ses dépendances (bibliothèques, configuration, ...) de manière cohérente, quel que soit l'endroit où il est déployé.
- **Kubernetes** : système open-source qui automatise le déploiement, la gestion et l'extensibilité des applications conteneurisées.

- **docker** : plateforme qui permet de créer, déployer et exécuter des applications dans des conteneurs, des environnements légers et isolés.
- **socket docker** : point de communication permettant à des applications ou des outils externes d'interagir avec le démon Docker pour gérer les conteneurs et les images
- **pull request** : demande de fusion de modifications dans un projet, permettant à d'autres de revoir et de valider le code avant son intégration.
- **conteneurisation** : technique qui permet d'emballer une application et ses dépendances dans un conteneur léger et isolé, afin de garantir son fonctionnement uniforme sur différents environnements.
- **GitHub Container Registry**: service de GitHub permettant d'héberger, gérer et distribuer des images de conteneurs Docker et OCI directement depuis GitHub.
- **GitHub** : plateforme en ligne qui permet de stocker, gérer et partager des projets de développement logiciel en utilisant Git.
- **Image Docker (Dockerfile)** : fichier qui contient tout ce dont une application a besoin pour s'exécuter : le code source, les bibliothèques, les dépendances, les configurations système.
- **JSON** : format de données léger et facile à lire, utilisé pour échanger des informations structurées sous forme de paires clé-valeur, souvent dans des applications web.
- **YAML**: format de données lisible par l'humain, habituellement utilisé pour la configuration de logiciels, qui représente des informations sous forme de paires, clé-valeur, listes et hiérarchies.
- **XML** : langage de balisage flexible utilisé pour structurer, stocker et échanger des données sous forme de texte, avec des balises définies par l'utilisateur pour organiser les informations.
- **SEO** : (Search Engine Optimization) ensemble des techniques visant à améliorer le référencement naturel d'un site web pour le rendre plus visible dans les résultats des moteurs de recherche.
- **Token** : unité de traitement utilisée par les modèles de langage, représentant un morceau de texte, comme un mot, une syllabe ou un caractère. Les LLM lisent et génèrent du texte en se basant sur ces unités.
- **GitOps**: branche de DevOps axée sur l'utilisation des dépôts de code Git pour gérer l'infrastructure et les déploiements de code d'application.
- **ArgoCD**: outil de déploiement continu (CD) pour Kubernetes, basé sur GitOps, qui permet d'automatiser, synchroniser et gérer les applications déclaratives en respectant les patterns GitOps.
- **Pods Kubernetes** : Plus petite unité déployable dans Kubernetes, qui encapsule un ou plusieurs conteneurs qui partagent les mêmes ressources réseau et de stockage et s'exécutent toujours ensemble sur le même nœud.

- **Cluster** : ensemble de pods connectées qui travaillent ensemble pour héberger et gérer des applications conteneurisées sous la coordination d'un système de contrôle central.
- **Déploiement Kubernetes** : Un outil qui permet de déployer et de gérer automatiquement plusieurs copies de votre application conteneurisée tout en assurant leur disponibilité continue.
- **Service Kubernetes** : Une ressource qui fournit une adresse réseau stable pour accéder à un groupe de pods, même si ces pods changent ou sont remplacés.
- **Ingress Kubernetes** : Un contrôleur de trafic qui agit comme un portail d'entrée pour diriger les requêtes externes vers les bons services à l'intérieur de votre cluster.
- **Volume persistant Kubernetes** : Ressource qui représente un espace de stockage durable dans le cluster, indépendant du cycle de vie des pods, permettant aux données de persister même après la suppression des conteneurs.
- **Service Account Kubernetes** : identité utilisée par le pods pour interagir avec l'API Kubernetes et accéder aux ressources du cluster selon des permissions définis.
- **Namespace Kubernetes** : mécanisme pour diviser un cluster en plusieurs environnements virtuels isolés, permettant de regrouper et de séparer les ressources par projet, équipe ou application.
- **sealed secrets** : méthode de chiffrement des secrets Kubernetes avant leur stockage dans des dépôts, garantissant ainsi leur sécurité tout au long du processus de déploiement.
- **HTTPS** : protocole de communication sécurisé qui chiffre les données échangées entre un navigateur et un serveur pour protéger la confidentialité et l'intégrité des informations.
- **microservice**: petite application autonome qui effectue une tâche spécifique et communique avec d'autres microservices pour former un système plus large.
- **certmanager**: opérateur Kubernetes open-source qui automatise la gestion, l'émission et le renouvellement des certificats TLS au sein d'un cluster.
- **opérateur** : extension de Kubernetes qui automatise le déploiement, la gestion et la mise à jour d'applications complexes en encapsulant les bonnes pratiques sous forme de contrôleurs personnalisés.
- **traefik**: reverse proxy et un load balancer open-source conçu pour s'intégrer facilement avec des environnements cloud et Kubernetes, permettant de gérer dynamiquement le routage du trafic.
- **reverse proxy** : serveur qui se place entre les clients et les serveurs backend pour gérer et optimiser les requêtes, améliorer la sécurité et répartir la charge.
- **dépôts** : espace de stockage où sont conservés, organisés et gérés des fichiers, souvent associé à un système de gestion de versions comme Git.

- **stratégie de ramification** : ensemble de règles définissant comment et quand créer, fusionner et gérer les branches dans un système de gestion de versions comme Git, afin d'organiser le développement collaboratif et de garantir la stabilité du code.
- **commits** : enregistrement d'une modification apportée à un dépôt dans un système de gestion de versions comme Git, incluant un message décrivant les changements effectués.
- **Conventional Commits** : ensemble de règles pour structurer les messages de commit de manière cohérente, facilitant la gestion des versions et l'automatisation des processus.
- **intégration continue** : pratique de développement logiciel où les modifications de code sont régulièrement fusionnées dans une branche principale et automatiquement testées pour détecter rapidement les erreurs et assurer la stabilité du projet.
- **linting**: processus d'analyse de code source pour détecter et signaler des erreurs, des bogues ou des incohérences selon des règles prédéfinies.
- **SonarQube**: outil d'analyse de qualité de code qui identifie les bogues, les vulnérabilités et les mauvaises pratiques dans le code source.
- **Semantic Release package**: outil d'automatisation de la gestion des versions et de la publication des packages.
- **OCI (Open Container Initiative)** : projet visant à créer des normes ouvertes pour les conteneurs de logiciels.
- **déploiement continu**: pratique de développement logiciel où les modifications de code sont automatiquement testées et déployées en production, garantissant des mises à jour rapides et fiables.

Annexe 2: Diagramme Logique

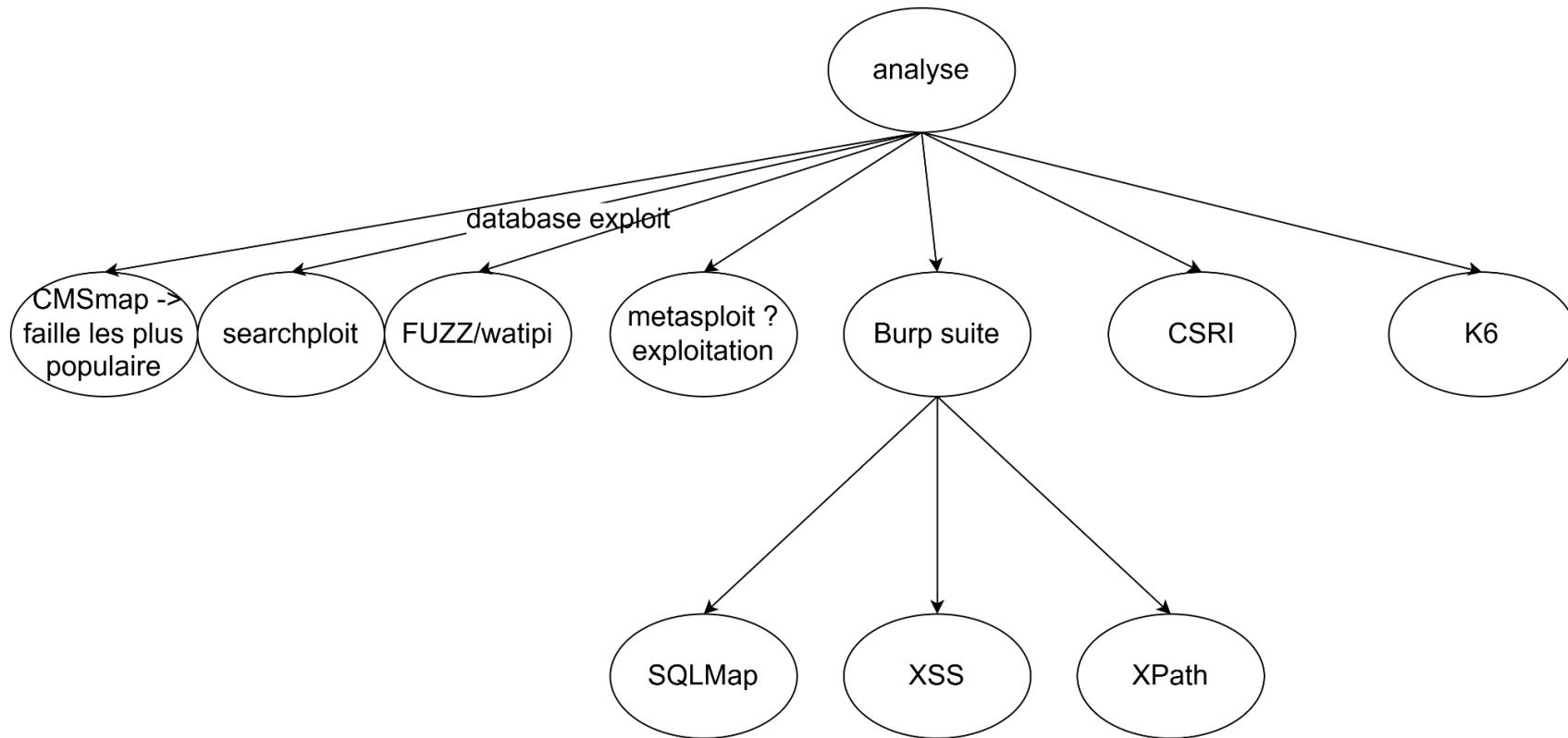


Figure 20 - diagramme Logique phase de reconnaissance

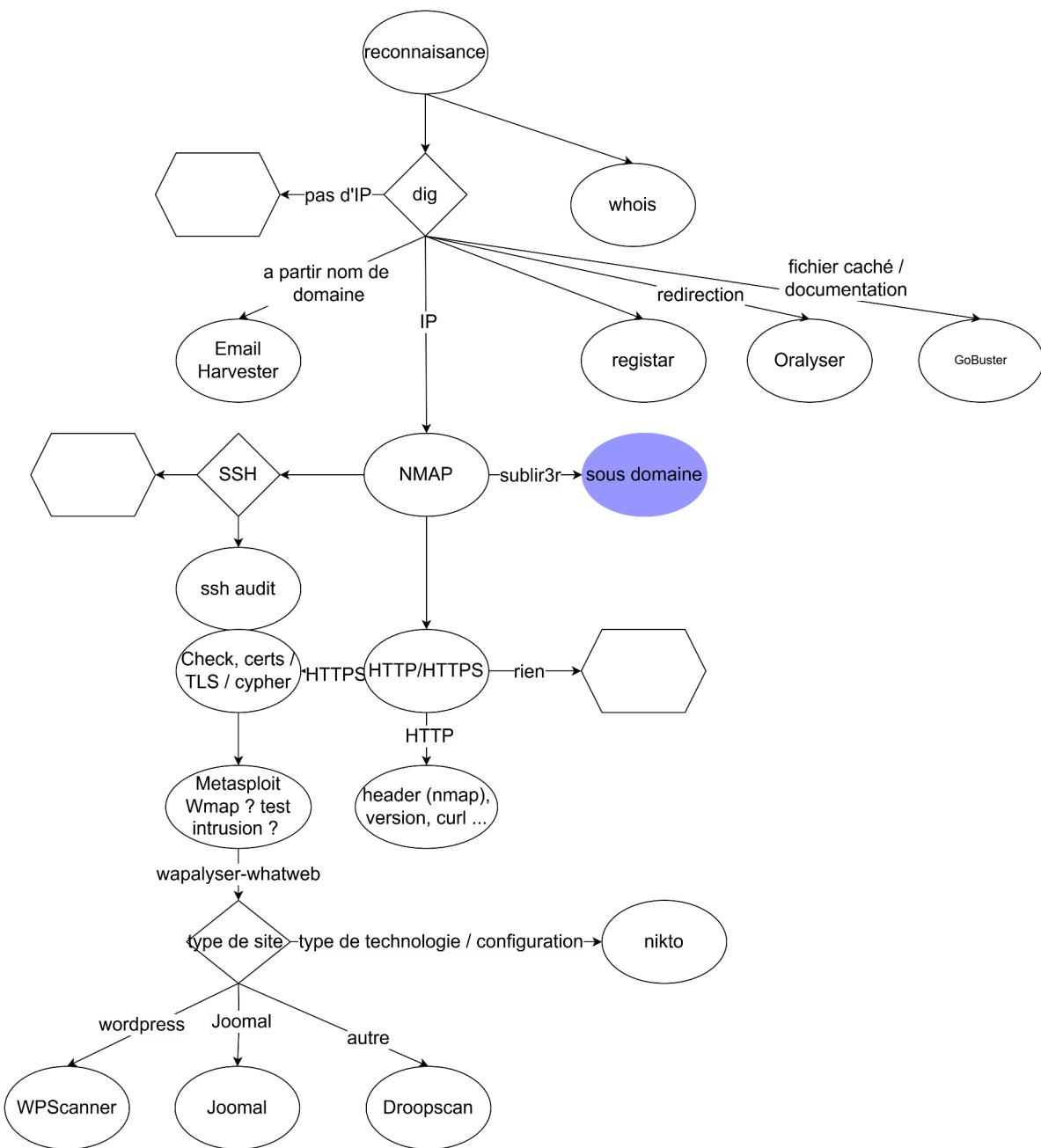


Figure 21 - diagramme Logique phase d'analyse

Annexe 3 : Maquette

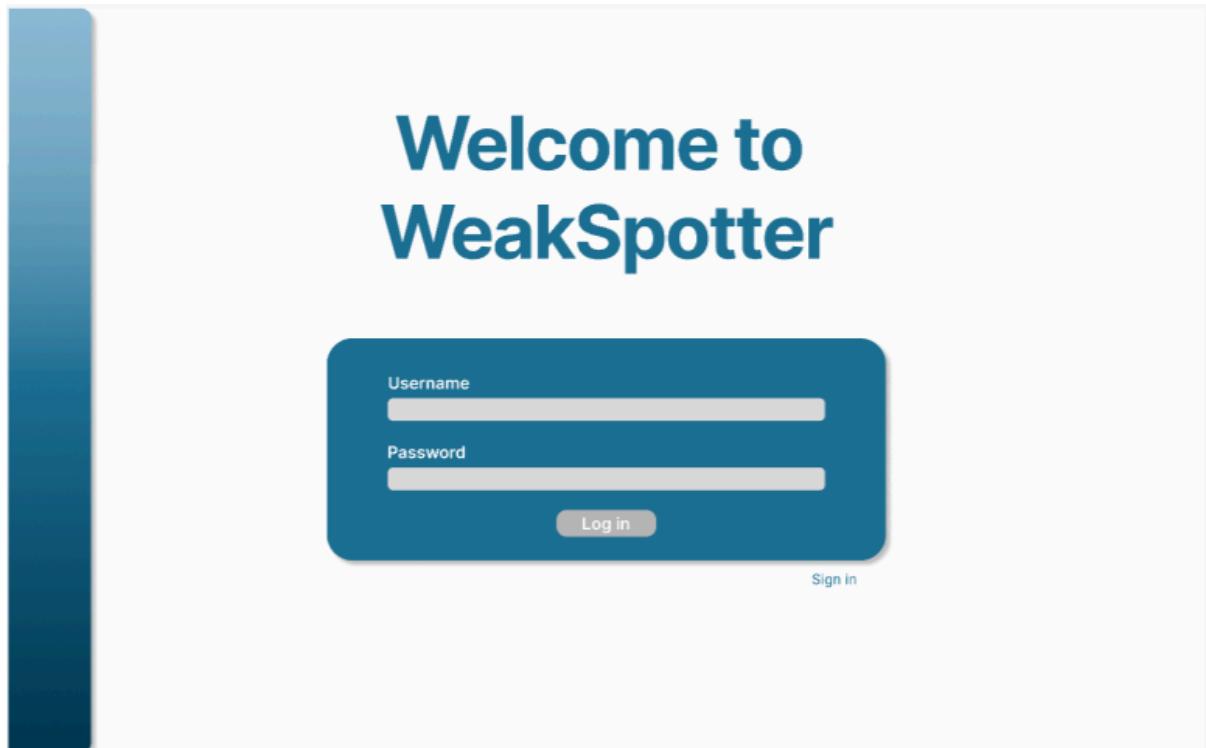


Figure 22 - maquette page d'authentification

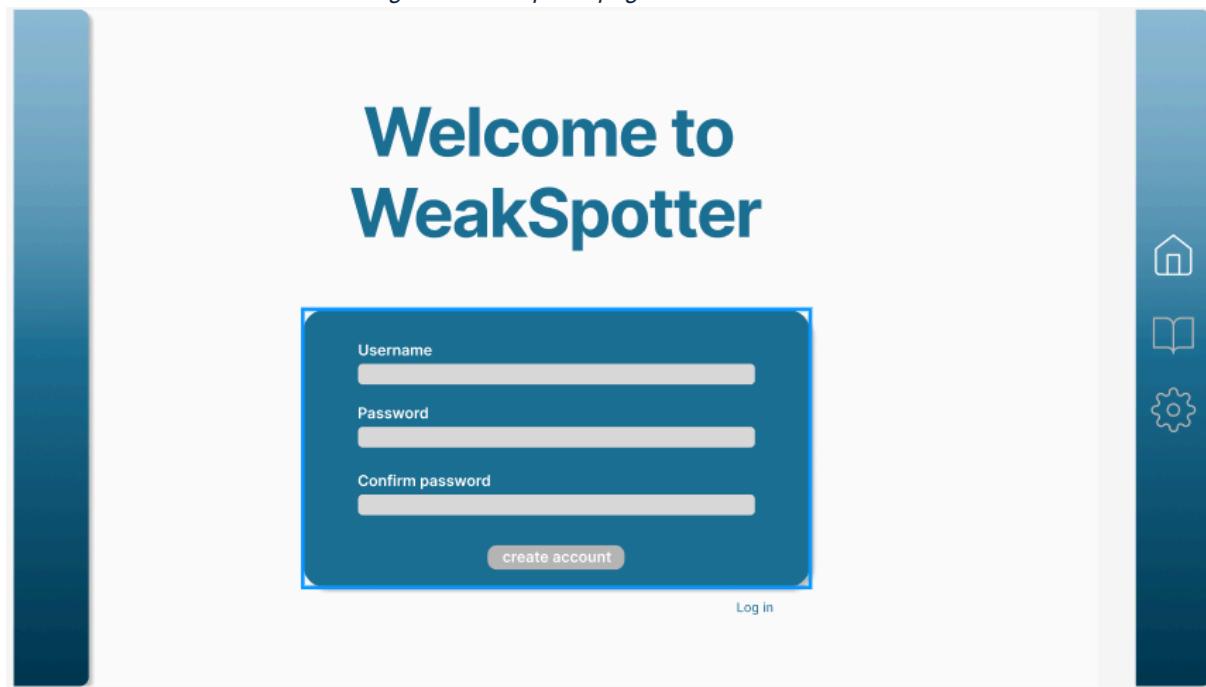


Figure 23 - maquette page de création de compte

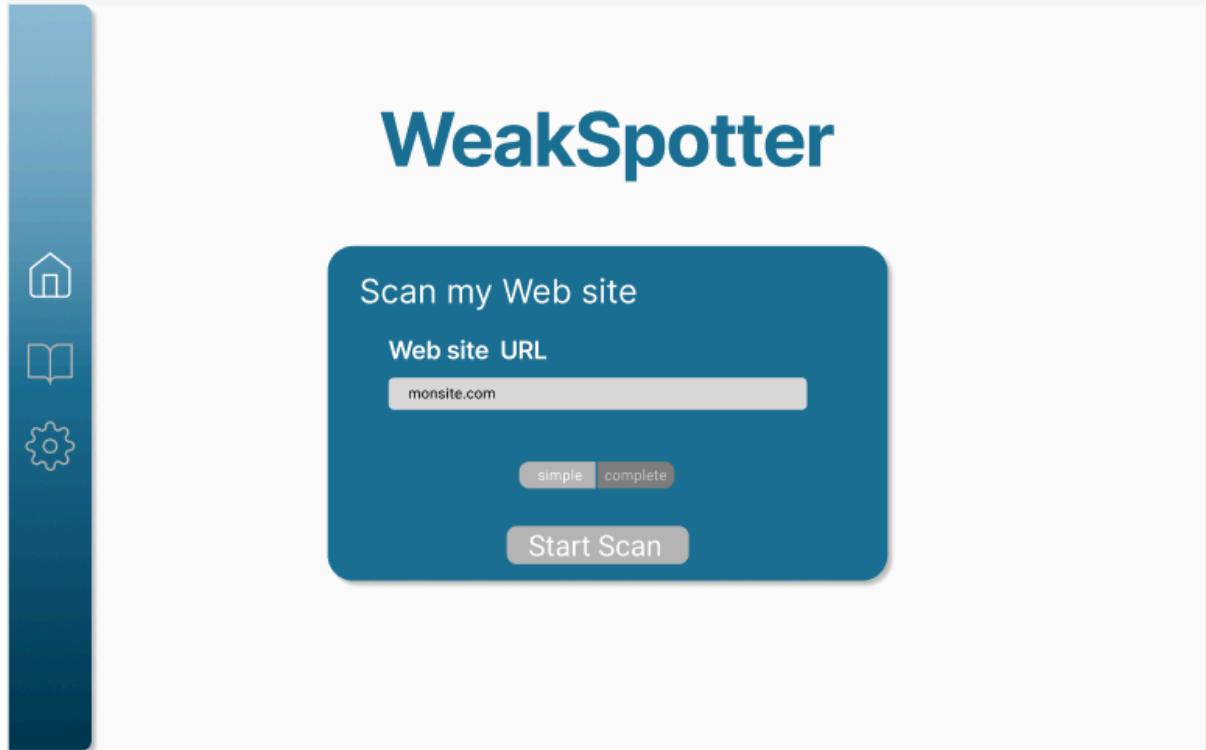


Figure 24 - maquette commencer un scan simple

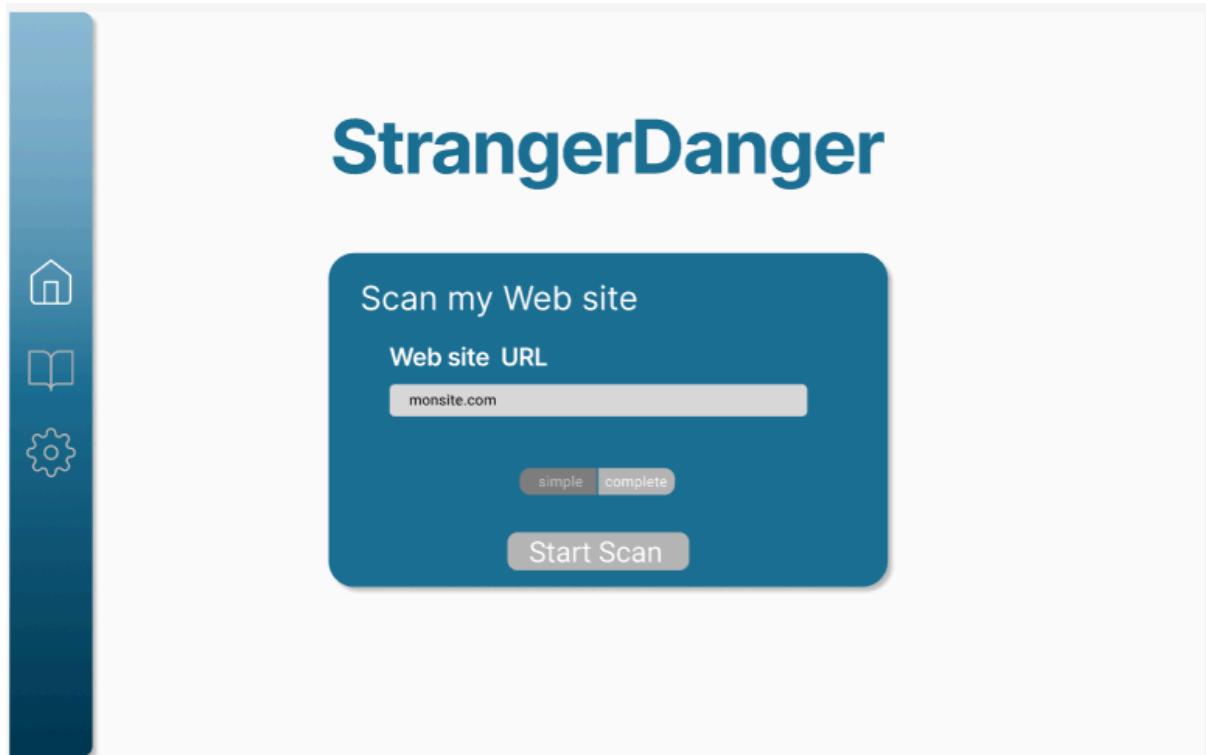


Figure 25 - maquette commencer scan complexe



Figure 26 - maquette écran de chargement

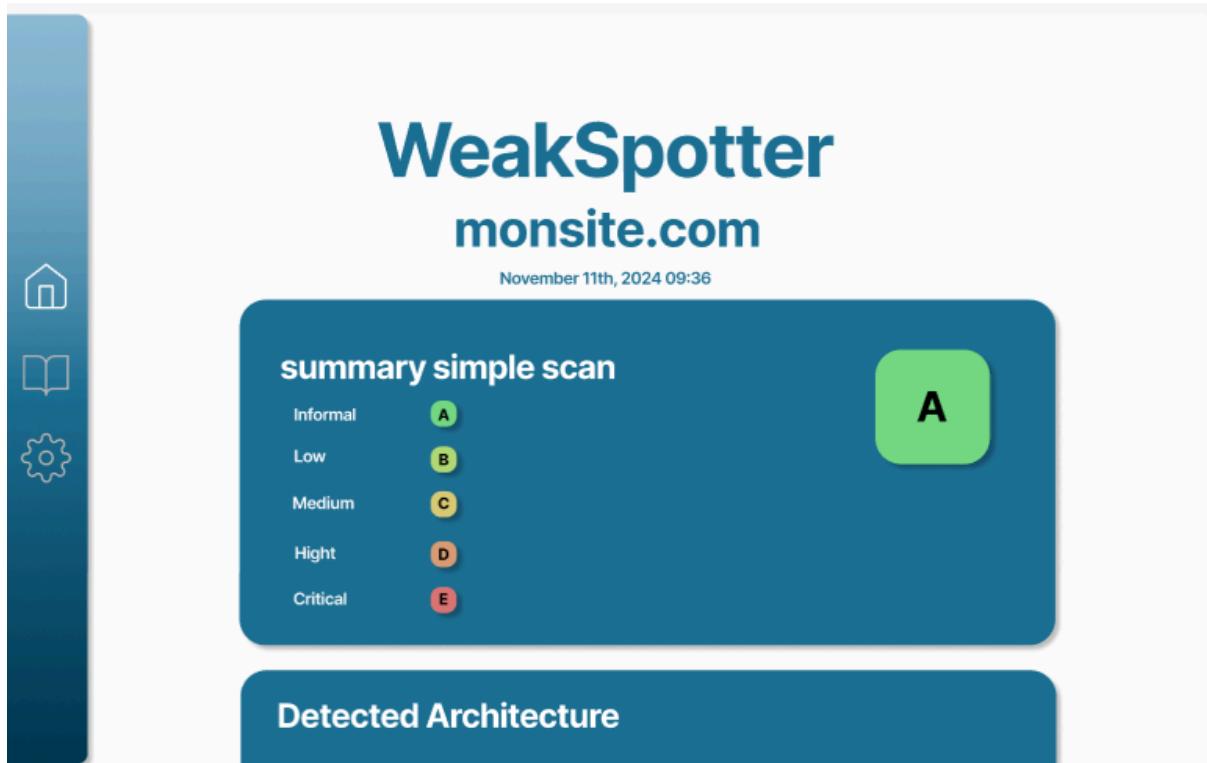


Figure 27 - maquette écran de résultats d'un scan

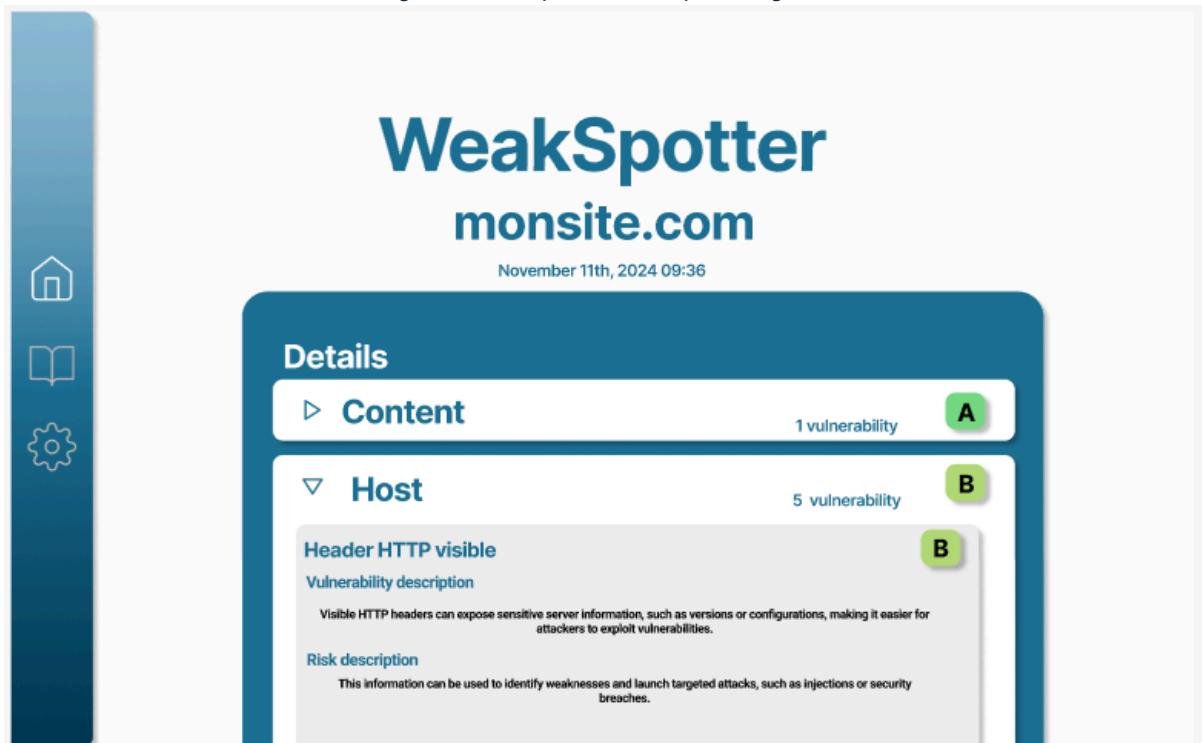


The dashboard displays the title "WeakSpotter" and the URL "monsite.com". Below this, the date "November 11th, 2024 09:36" is shown. A vertical sidebar on the left contains icons for home, book, and settings.

Details

- ▷ Content 1 vulnerability A
- ▷ Host 5 vulnerability B
- ▷ Code 8 vulnerability C
- ▷ Privacy 13 vulnerability D

Figure 28 - maquette détails par catégorie



The dashboard displays the title "WeakSpotter" and the URL "monsite.com". Below this, the date "November 11th, 2024 09:36" is shown. A vertical sidebar on the left contains icons for home, book, and settings.

Details

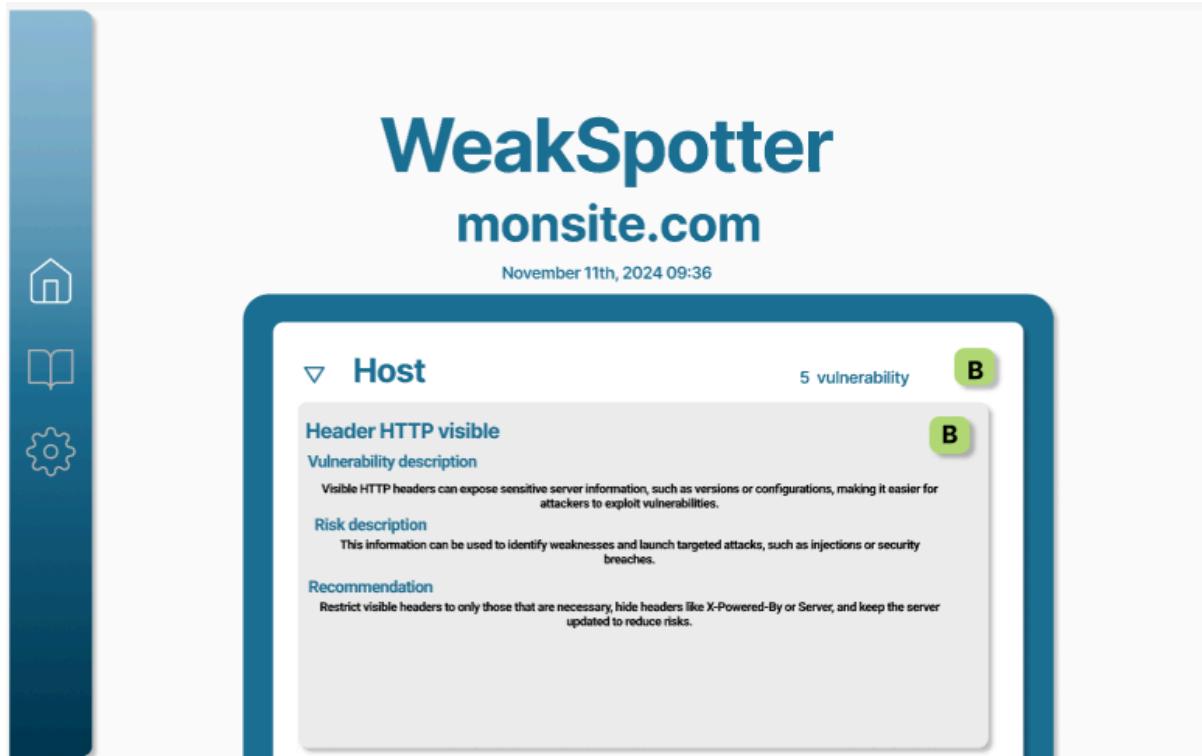
- ▷ Content 1 vulnerability A
- ▽ Host
 - Header HTTP visible**
 - Vulnerability description**

Visible HTTP headers can expose sensitive server information, such as versions or configurations, making it easier for attackers to exploit vulnerabilities.

 - Risk description**

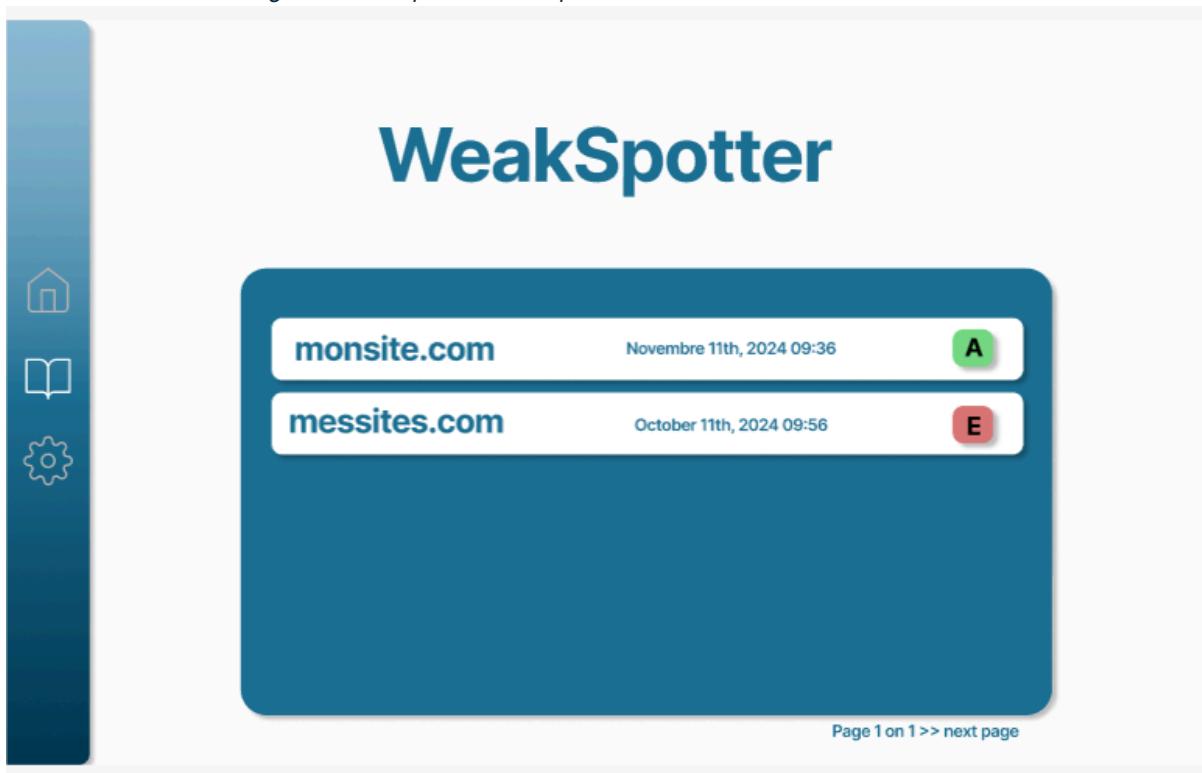
This information can be used to identify weaknesses and launch targeted attacks, such as injections or security breaches.B

Figure 29 - maquette explication des détails



The mockup shows a mobile application interface for 'WeakSpotter'. On the left is a vertical navigation bar with icons for Home, Book, and Settings. The main screen displays the title 'WeakSpotter' and the URL 'monsite.com'. Below this is the date 'November 11th, 2024 09:36'. A section titled 'Host' shows '5 vulnerability' with a green badge labeled 'B'. Under this, there's a 'Header HTTP visible' section with a green badge labeled 'B'. It includes a 'Vulnerability description' (Visible HTTP headers can expose sensitive server information), a 'Risk description' (Information can be used to identify weaknesses and launch targeted attacks), and a 'Recommendation' (Restrict visible headers to only those that are necessary, hide headers like X-Powered-By or Server, and keep the server updated to reduce risks).

Figure 30 - maquette conseil pour remédier au vulnérabilité trouvé



The mockup shows a mobile application interface for 'WeakSpotter'. On the left is a vertical navigation bar with icons for Home, Book, and Settings. The main screen displays the title 'WeakSpotter'. Below it are two cards: one for 'monsite.com' dated 'Novembre 11th, 2024 09:36' with a green badge labeled 'A', and another for 'messites.com' dated 'October 11th, 2024 09:56' with a red badge labeled 'E'. At the bottom right is the text 'Page 1 on 1 >> next page'.

Figure 31 - maquette historique de scan

Annexe 4: Preuve de Concept

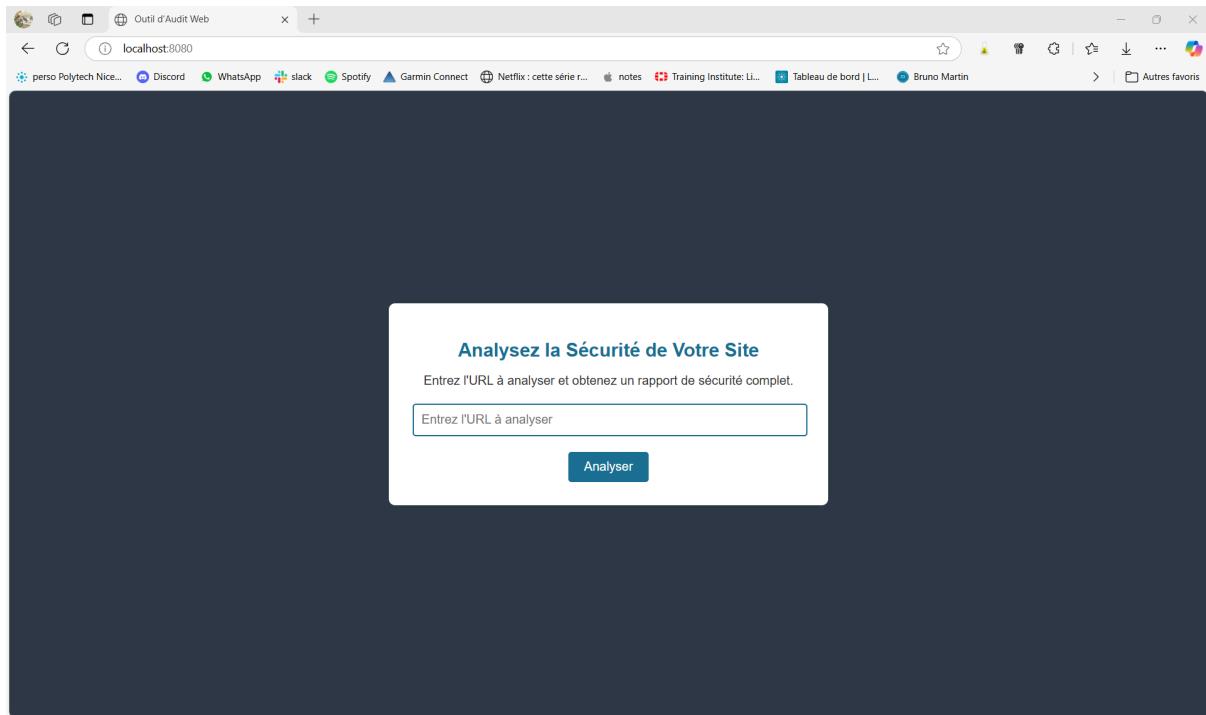


Figure 32 - POC début de scan

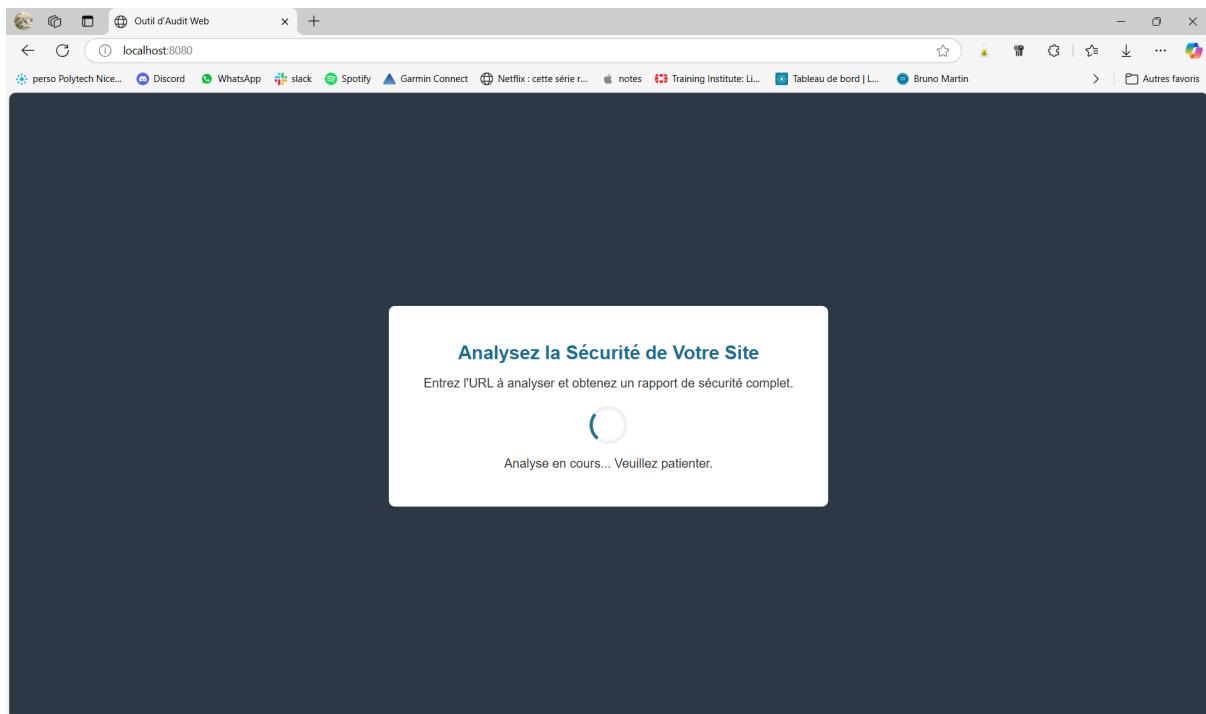


Figure 33 - POC chargement de scan

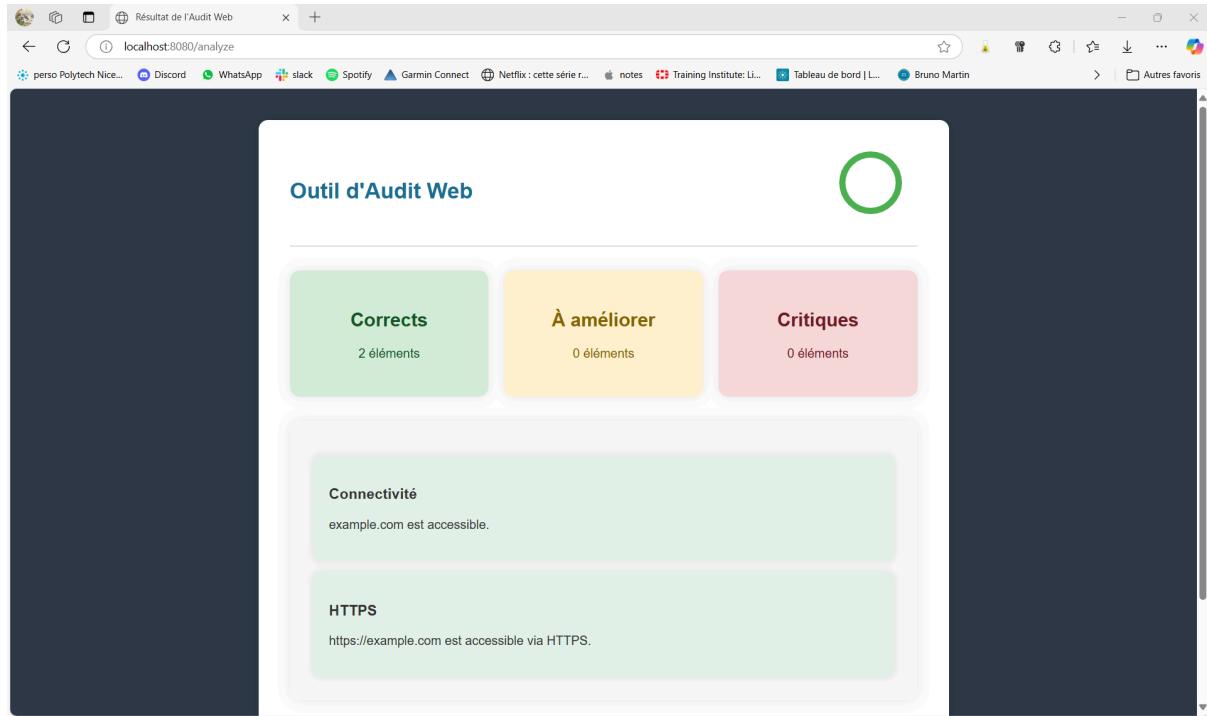


Figure 34 - POC résultat de scan

Annexe 5: diagramme

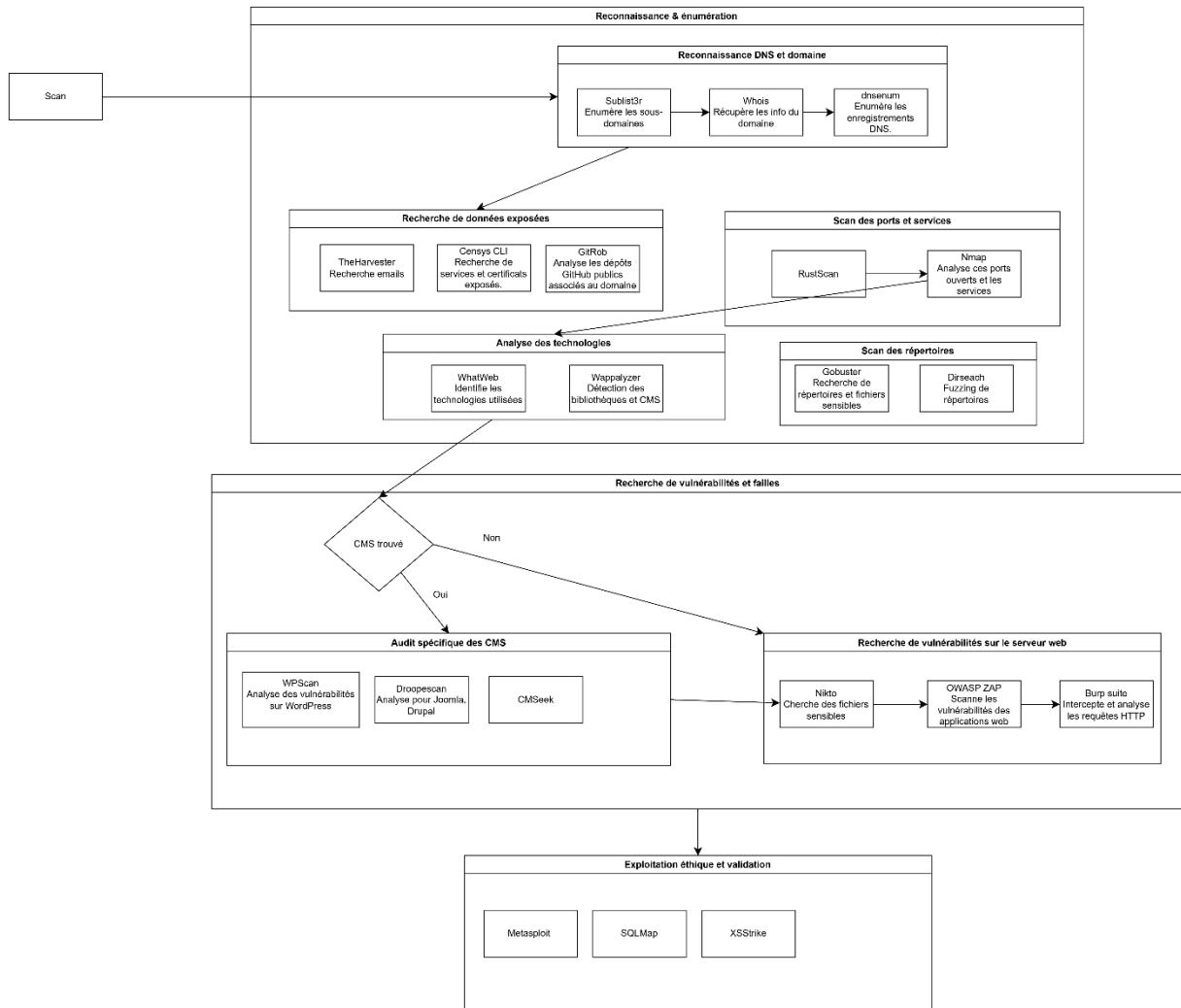


Figure 35 - diagramme logique avec container

Annexe 6: Interface graphique téléphone

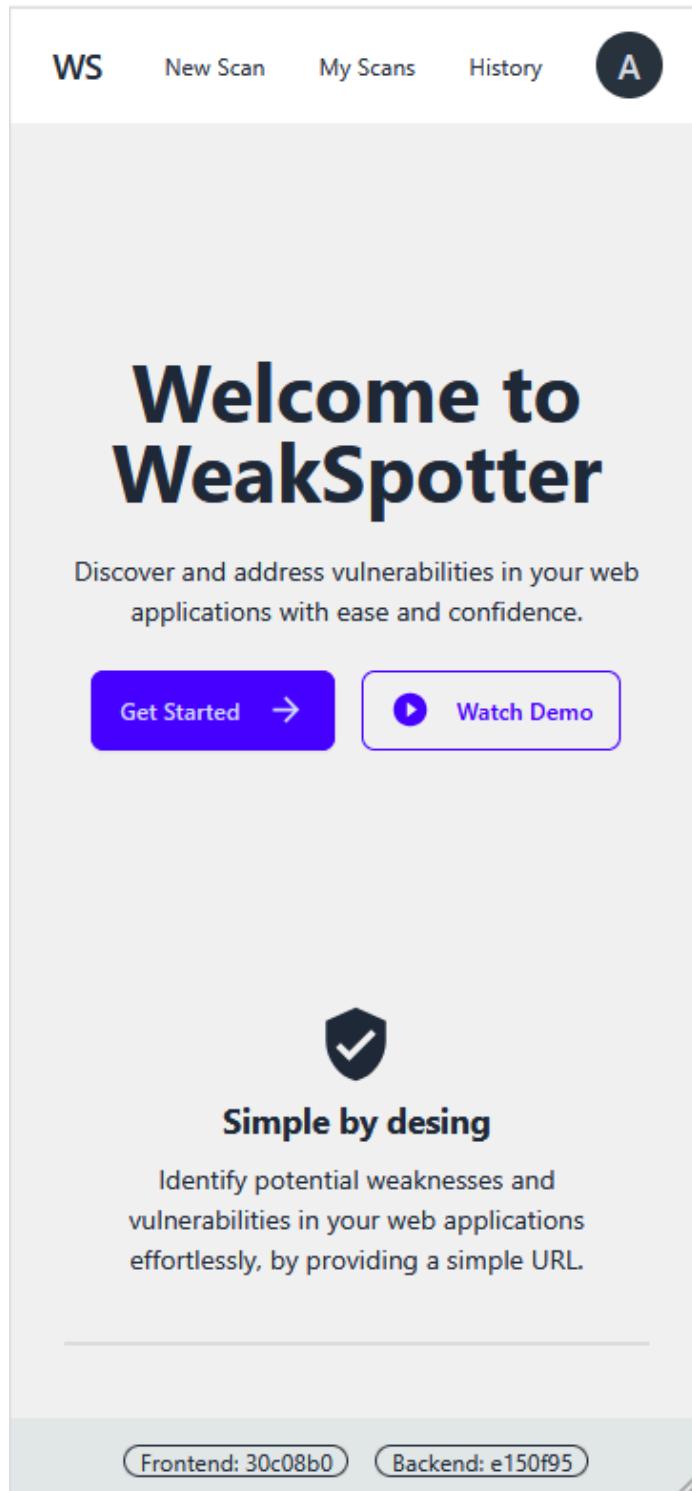


Figure 36 - Page d'accueil

WS [New Scan](#) [My Scans](#) [History](#) A

Create New Scan

URL to scan

Scan Type Simple Complex

I have read and agree to the [Terms and Conditions](#)

[Start Scan](#)

Frontend: 30c08b0 Backend: e150f95

Figure 37 - Page de début de scan

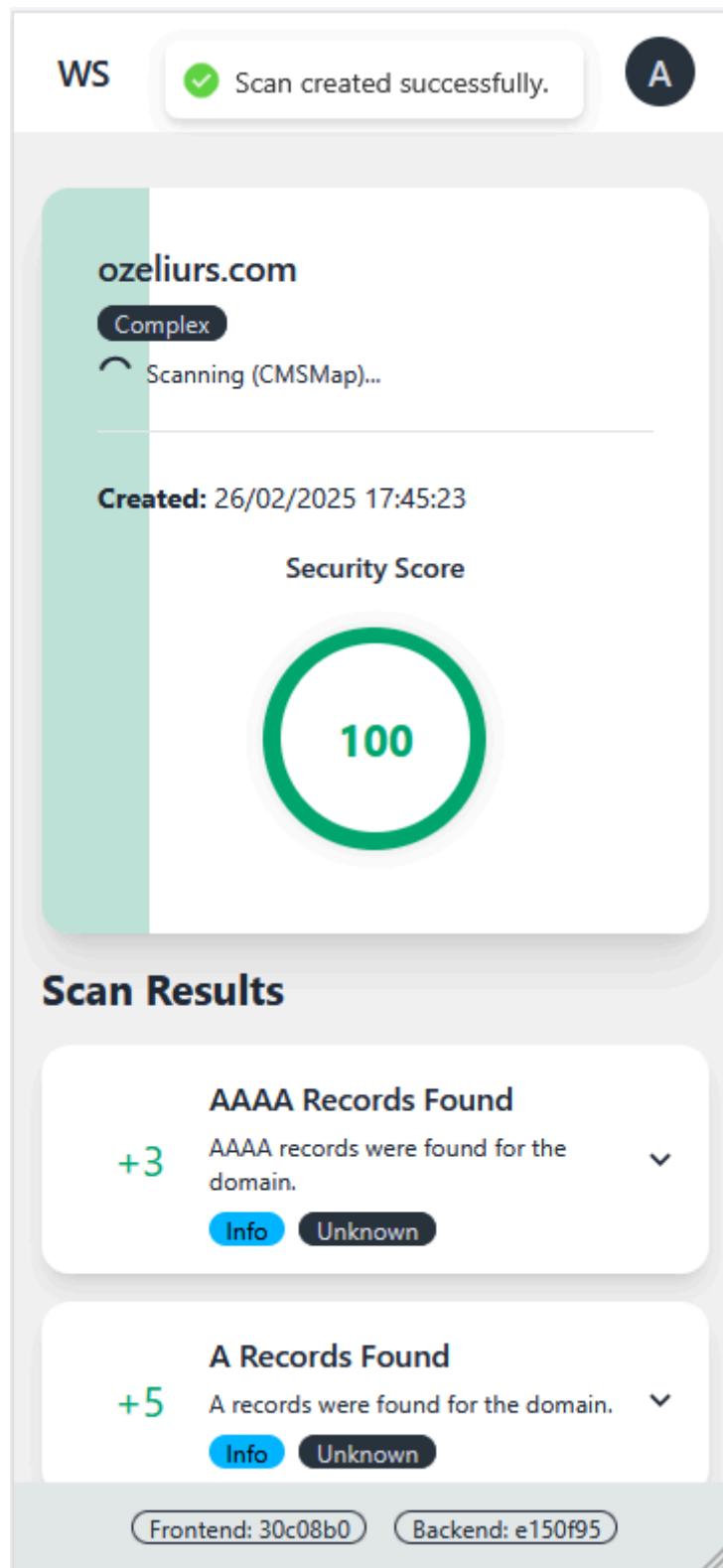
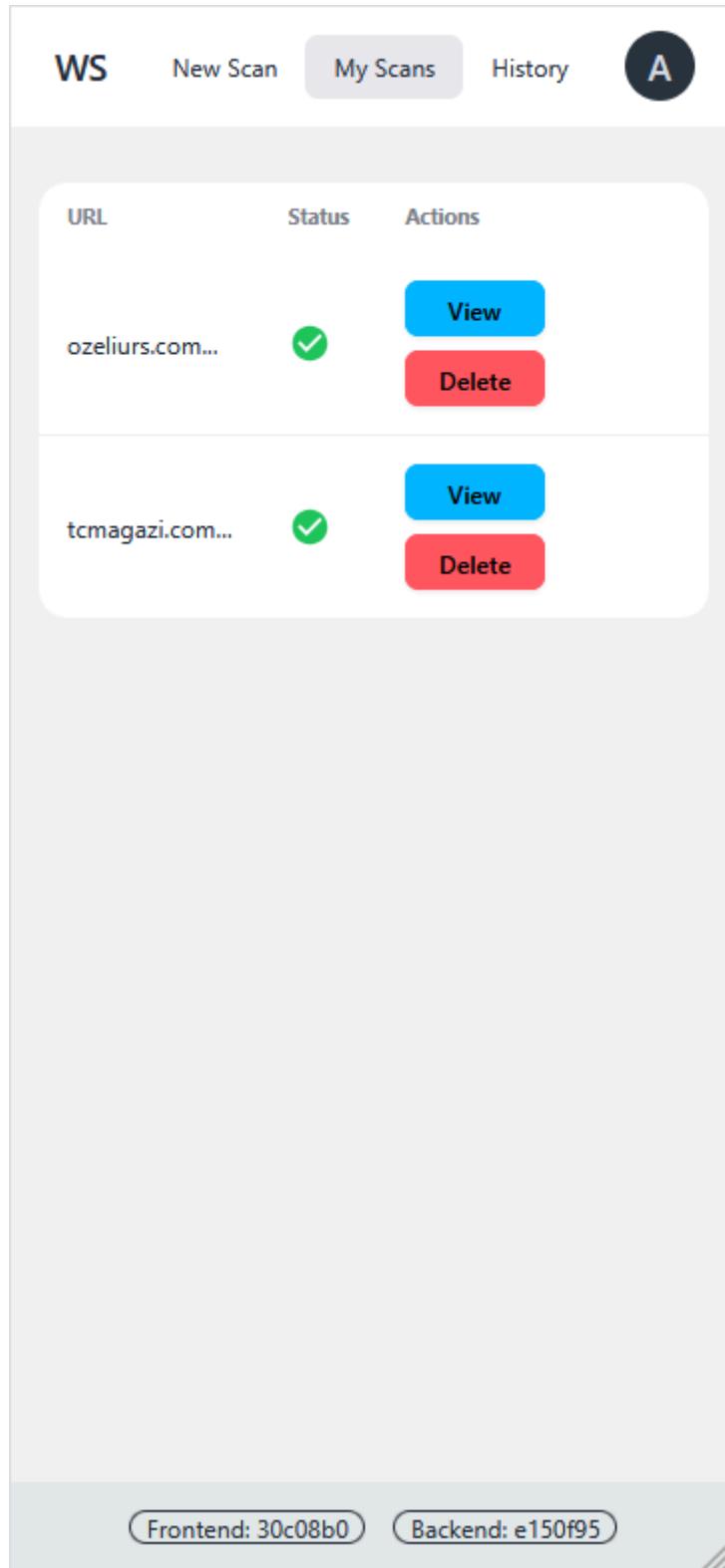


Figure 38 - Page de résultat de scan



The screenshot shows a web interface for managing scans. At the top, there is a navigation bar with tabs: 'WS', 'New Scan', 'My Scans' (which is selected and highlighted in grey), and 'History'. To the right of the tabs is a dark circular badge containing the letter 'A'. Below the navigation bar is a table listing two URLs:

URL	Status	Actions
ozeliurs.com...	✓	<button>View</button> <button>Delete</button>
tcmagazi.com...	✓	<button>View</button> <button>Delete</button>

At the bottom of the page, there are two small status indicators: 'Frontend: 30c08b0' and 'Backend: e150f95'.

Figure 39 - Page liste des scans effectué

Annexe 7: Outils open source

Nom	description	Utilisé	Licence
Domain extract	extraction du nom de domaine principale d'une URL ou d'une adresse e-mail	<input checked="" type="checkbox"/>	Other ▾
Whois	extraction d'information sur un nom de domaine ou d'une adresse IP comme le propriétaire, la date de création, l'expiration et l'hébergeur	<input checked="" type="checkbox"/>	MIT ▾
DNS Records	Récupération des enregistrements DNS d'un domaine, comme les adresses IP, les serveurs de messageries ou les alias	<input checked="" type="checkbox"/>	MPL ▾
Cloudflare Detect	Vérification de l'utilisation de Cloudflare comme service de protection et de gestion du trafic	<input checked="" type="checkbox"/>	Other ▾
Nmap	Scanner de réseau qui permet de détecter les appareils connectés, d'identifier les ports ouverts et de connaître les services actifs	<input checked="" type="checkbox"/>	NPSL ▾
HTTP Version	Détection de la version du protocole HTTP utilisée par un site web	<input checked="" type="checkbox"/>	MIT ▾
Email Harvester	Collection automatique des adresses e-mail à partir de sites web, de documents ou d'autre source en ligne	<input checked="" type="checkbox"/>	GPLv3 ▾
Nikto	scanner de vulnérabilité pour les serveurs web, pour la détection de failles, de mauvaise configuration et les logiciels obsolètes	<input checked="" type="checkbox"/>	GPLv3 ▾
zapit	scanner de vulnérabilité dans les applications web	<input checked="" type="checkbox"/>	Apach... ▾
ssh-audit	Analyse de la configuration d'un serveur SSH	<input checked="" type="checkbox"/>	MIT ▾
wapiti	scanner de vulnérabilité pour les applications web en "attaquant" les pages	<input checked="" type="checkbox"/>	GPLv3 ▾
sublist3r	Extraction des sous domaine d'un domaine en utilisant les sources publiques	<input checked="" type="checkbox"/>	GPLv3 ▾
oralyser	analyse et détection des vulnérabilités dans les applications web en utilisant des requêtes HTTP	<input checked="" type="checkbox"/>	GPLv3 ▾
cmsmap	Scanner de vulnérabilités conçu pour détecter les failles dans des CMS	<input checked="" type="checkbox"/>	GPLv3 ▾

vulnX	Détection de vulnérabilité dans les applications web en scannant pour identifier des failles courantes	<input checked="" type="checkbox"/>	GPLv3 ▾
Joomscan	scanner de vulnérabilité spécifiques aux sites webs utilisant le CMS Joomla	<input checked="" type="checkbox"/>	GPLv2 ▾
droopscans	scanner de vulnérabilité spécifiques aux sites webs utilisant le CMS Drupal	<input checked="" type="checkbox"/>	GPLv3 ▾
WPScan	scanner de vulnérabilité spécifique aux sites webs utilisant le CMS WordPress	pas GPL Compatible	WPScan ▾
The Harvester	Collecteur d'information sur des cibles, comme des adresses e-mail, des sous-domaines depuis des sources publiques	déjà réalisé par d'autre outil	Other ▾
PurpleTeam	Simulateur d'attaque et de test de défense d'un système	compliqué à déployer et licence trop restrictive	BSL et... ▾
OpenVAS	scanner de vulnérabilité open-source. Permet une analyse des systèmes et réseaux pour détecter des failles et fournir des recommandations.	Compliqué à déployer	GPLv2 ▾
cmseek	détection des CMS sur des sites webs	redondance avec d'autre outil	GPLv3 ▾
vbscan	détection des CMS	redondance avec d'autre outil	GPLv3 ▾
Nessus	Scanner de vulnérabilité qui analyse les systèmes, réseaux et applications pour détecter des failles de sécurité.	license trop restrictive	Propri... ▾
Metasploit	framework de test de pénétration qui permet de simuler des attaques pour détecter des vulnérabilités dans les systèmes	license trop restrictive	Other ▾

Tableau 15 - outil open source considéré

Annexe 8: Lien vers le projet

Lien de l'organisation git:

[WeakSpotter](#)

Lien du front et backend :

[WeakSpotter/WeakSpotter](#)

Lien du micro service IA:

[WeakSpotter/AI-CVE-Explainer](#)

Dernière release stable:

<https://weakspotter.ozeliurs.com/>

Dernière prerelease:

<https://staging.weakspotter.ozeliurs.com/>

Lien de l'api IA:

<https://ai-cve.weakspotter.ozeliurs.com/>

Bibliographie indicative

- Présentation des travaux effectués par des groupes étudiant SAE41 : WebCheck, Audifity, SpiderScan et Arrow, *encadré par Christian DELETTRE*
- [OWASP Top Ten \(2021\)](#), OWASP
- [PageSpeed Insights](#), Google Développeurs
- [Hacked Websites Trend Report 2019](#), Sucuri
- [ANSSI](#), ANSSI
- [Attaques par rançongiciels, tous concernés](#), ANSSI
- [Assistance aux victimes de cybermalveillance](#), Cybermalveillance
- [Portail de la transformation numérique](#), FranceNum
- [RGPD](#), Ministère de l'économie
- [La directive NIS 2](#), ANSSI
- [Homepage](#), Cyber Resilience Act
- [Homepage](#), Digital Operational Resilience Act
- [Part de marché des CMS 2024 : tendances et statistiques d'utilisation](#), WPADE
- [WordPress Statistics In 2024](#), Bloggers Passion
- [Cyber PME](#), Bpifrance
- [Présentation de Lighthouse | Chrome for Developers](#), Google Développeurs
- [CWE - 2024 CWE Top 25 Most Dangerous Software Weaknesses](#), MITRE
- [Plateforme ouverte, évolutive, sécurisée et orientée utilisateur pour l'e-commerce](#), Christian DELETTRE

Scanners & Outils

- [Acunetix](#), Invicti
- [Burp Suite Pro](#), PortSwigger
- [Nessus](#), Tenable
- [Intruder](#), Intruder System
- [vRx](#), Vicarius
- [OWASP ZAP](#), OWASP
- [Nikto](#), CIRT
- [WPScan](#), WPScan
- [OpenVAS](#), Greenbone
- [TruRisk](#), Qualys
- [SUBLIST3R](#), Ahmed Aboul-Ela
- [Ssh Audit](#), Joe Testa
- [GoBuster](#), OJ Reeves
- [Nmap](#), Gordon Lyon (Fyodor)
- [Whois](#), rfc1036, Ken Harrenstien
- [Dig](#), tigeli
- [Email Harvester](#), Christian Martorella
- [Oralyser](#), laramies
- [Homepage](#), Metasploit
- [Joomal](#), OWASP

- [Droopscan](#), *SamJoan*
- [Searchploit](#), *Exploit Database*
- [Watipi](#), *Wapiti-scanner*
- [Cloudflare detect](#), *christophetd*