МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА СИСТЕМ ІНФОРМАЦІЙНОГО ТА КІБЕРНЕТИЧНОГО ЗАХИСТУ

«На правах рукопису»
УДК 681.3.06

«До захисту допущено»
Завідуючий кафедрою СІКЗ
_____к.т.н. Г.В. Шуклін
«____» _____2023 р.

БАКАЛАВРСЬКА АТЕСТАЦІЙНА РОБОТА

зі спеціальності 125 “Кібербезпека”

на тему: **Фреймворк для розробки веб–застосунків з підвищеною безпекою**

Студент групи СЗД–41 Козеренко Дмитро Олександрович _____
(підпис)

Науковий керівник: д.т.н**.** Ахрамович Володимир Миколайович _____
(підпис)

Нормоконтроль Зозуля Сергій Анатолійович _____
(підпис)

КИЇВ – 2023

«ЗАТВЕРДЖУЮ»

Завідувач кафедри СІКЗ

_____ к.т.н. Г.В. Шуклін

(підпис)

«_____»_____ 2023р.

ЗАВДАННЯ

на атестаційну роботу бакалавра

студенту: Козеренку Дмитру Олександровичу

1. Тема роботи: Фреймворк для розробки веб–застосунків з підвищеною безпекою від «16» 02 2023р.

2. Термін здачі студентом оформленої роботи «25» 05 2023р.

3. Об'єкт дослідження: дослідження сучасних вимог до захищеного веб–додатку. Розробка вимог до фреймворку для розробки веб–додатків.

4. Предмет дослідження: системи захисту інформації.

5. Мета роботи: імплементація вимог та розробка фреймворку.

6. Перелік питань, які мають бути розроблені:

- реалізація фреймворку для розробки веб–додатка з підвищеною безпекою
- економічне обґрунтування

7. Перелік ілюстрованого матеріалу: презентація матеріалу на слайдах виконана в Microsoft PowerPoint.

8. Дата видачі завдання «16» 02 2023р.

Науковий керівник _____ Ахрамович В.М.

(підпис)

Завдання прийняв до виконання _____ Козеренко Д.О.

(підпис)

КАЛЕНДАРНИЙ ПЛАН

Дата видачі завдання «16» 02 2023р.

№ п/п	Найменування робіт	Термін виконання
1	Огляд літератури на тему роботи	до 20.02.23
2	Дослідження процесу розробки та вимог безпеки	до 27.02.23
3	Імплементація вимог до захищеного ПЗ	до 16.03.23
4	Реалізація фреймворку для розробки веб–додатків з підвищеною безпекою	до 12.04.23
5	Оформлення пояснювальної записки	до 02.05.23
6	Оформлення ілюстративного матеріалу	до 15.05.23

Студент: СЗД – 41 Козеренко Д.О. _____

(підпис)

Науковий керівник: д.т.н. Ахрамович В.М. _____

(підпис)

Нормоконтроль: Зозуля С.А. _____

(підпис)

РЕФЕРАТ

Пояснювальна записка 69 стор., 7 рис., 9 табл., 20 посилання.

Об'єктом розробки є фреймворк для розробки веб-додатку.

Метою роботи є створення базового каркасу для веб-програми з урахуванням вимог безпеки та автоматизації реалізації цих вимог на старті проекту.

Дипломна робота присвчена дослідженню процесу інтеграції вимог безпеки в гнучку методологію розробки веб-додатку, дослідженню основних вимог безпеки веб-додатку, розробці та реалізації вимог безпеки у веб-додатку та оптимальній інтеграції реалізації цих вимог у процес розробки програмного забезпечення.

ФРЕЙМВОРК, ВЕБ ДОДАТОК, ГІТ, РОЗРОБКА, КРИПТОГРАФІЯ, ЛОГУВАННЯ

ABSTRACT

Explanatory note 69 pages, 7 figures, 9 tables, 20 references.

The object of development is a framework for developing a web application.

The goal of the work is to create a basic framework for a web application taking into account security requirements and automating the implementation of these requirements at the start of the project.

The thesis is devoted to the study of the process of integrating security requirements into a flexible web application development methodology, the study of the main security requirements of a web application, the development and implementation of security requirements in a web application, and the optimal integration of the implementation of these requirements into the software development process.

FRAMEWORK, WEB APP, GIT, DEVELOPMENT, CRYPTOGRAPHY, LOGGING

ЗМІСТ

стор.

ВИЗНАЧЕННЯ, ПОЗНАЧЕННЯ І СКОРОЧЕННЯ

CSRF – Cross Site Request Forgery – Міжсайтова підробка запиту]

XSS – Міжсайтовий скриптинг

Framework – програмна платформа, що визначає структуру програмної системи; програмне забезпечення, що полегшує розробку та об'єднання різних компонентів великого програмного проекта.

MVC – Модель–Подання–Контролер

OWASP – Open Web Application Security Project

REST – Representational State Transfer

ВСТУП

Загалом framework — це реальна або концептуальна структура, призначена служити опорою або керівництвом для створення чогось, що розширює структуру на щось корисне.

У разі використання водоспадної моделі розробки вимоги безпеки вбудовуються на етапі розробки документації. Цей підхід має мінуси як стандартні для цієї моделі розробки так і особливі для вимог безпеки.

Наприклад, виявлення нової вразливості у технології, що використовується на останніх стадіях розробки, може коштувати досить дорого.

Сучасні проблеми

Нині ринку переважають гнучкі методології розробки – Agile. І рівень виконання вимог безпеки вкрай залежить від кількох факторів:

- досвідченість команди. Досвід розробки захищених систем. Наявність Project Backlog.
- присутність у проекті спеціаліста з комп'ютерної безпеки.
- рівень зрілості підприємства. Наявність корпоративних правил забезпечення безпеки у проектах.
- досвід підтримки веб–додатків

Інша проблема полягає в малій цінності вимог безпеки для замовника порівняно з функціональністю, що продає. Таким чином вимоги безпеки можуть бути відсунуті реалізацією функціональності, що продає. Це одна з основних причин низького ступеня захищеності більшості сучасних веб–додатків. Проблема малої цінності вимог безпеки щодо функціональності може вирішитися зменшенням необхідного для впровадження вимог безпеки часу.

Таким чином, найбільш раціональним буде реалізація фреймворку для реалізації вимог безпеки у веб–додатках.

Актуальність теми дипломної роботи

Завдання захисту веб–додатків стоїть вкрай гостро. Переважна більшість сайтів схильна до різних уразливостей. Їх частка представлена у табл. 1.1. Незважаючи на існування різних інструментів реалізації захисту веб–додатків — їх використання часом занадто дорого, якщо важливо якнайшвидше вивести програму на ринок.

Таблиця 1.1.

Найбільш поширені вразливості в Java веб–програми за даними компанії Positive Technologies

Тип уразливості	Частка сайтів
Cross–Site Scripting	77%
XML External Entities	54%
Brute Force	46%
Path Traversal	31%
Information Leakage	31%
URL Redirection Abuse	31%
SQL Injection	23%
Cross–Site Request Forgery	23%
Application Misconfiguration	23%
HTTP Response Splitting	23%

Метою дипломної роботи є розробка фреймворку для розробки веб–додатків із підвищеною безпекою.

Поставлена мета обумовила наступні завдання дипломної роботи:

- зібрати інформацію про методи захисту веб–додатків;
- розробити методику розробки захищеного веб–додатку;
- оцінити ефективність розробленого фреймворку;

Ця теза базується на дослідженні, проведеному Open Web Application Security Project (OWASP). OWASP — це глобальна некомерційна організація, яка займається підвищенням безпеки програмного забезпечення. Основною метою організації є підвищення обізнаності про безпеку програмного забезпечення, щоб люди та організації могли приймати обґрунтовані рішення. OWASP має унікальну можливість пропонувати неупереджену та практичну інформацію про інформаційну безпеку різним особам, таким як окремі особи, університети, корпорації, державні установи та інші організації по всьому світу.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Розвиток веб–технологій і зміни в бізнес–середовищі призвели до збільшення використання веб–додатків у різних секторах, таких як корпоративні, державні та державні послуги. Хоча веб–програми пропонують зручність і ефективність, вони також створюють нові загрози безпеці, які потенційно можуть поставити під загрозу IT–інфраструктуру організації, якщо їх не врахувати належним чином.

Організації покладаються на заходи безпеки для захисту своєї IT–інфраструктури вже більше десяти років. Однак традиційних заходів безпеки та технологій може бути недостатньо для захисту мереж від нових загроз, оскільки зараз зловмисники націлені на недоліки безпеки в дизайні веб–додатків. Щоб зменшити ці ризики, надзвичайно важливо впровадити нові заходи безпеки, як технічні, так і адміністративні, разом із розробкою веб–додатків.

Для боротьби з ризиками, пов’язаними зі службами веб–додатків, необхідно розуміти загальні вразливості, виявлені у веб–додатках, і процес розробки веб–додатків.

1.1. Фреймворки

В інформатиці програмне середовище являє собою абстракцію, в якій програмне забезпечення, що забезпечує загальні функції, може бути вибірково змінено за допомогою додаткового коду користувача, таким чином надаючи програмне забезпечення для конкретних додатків. Програмне середовище забезпечує стандартний спосіб створення та розгортання додатків.

Програмне середовище – універсальне, багаторазове програмне середовище, яке забезпечує певну функціональність як частину більшої програмної платформи для полегшення розробки програмних додатків, продуктів та рішень.

Таке програмне середовище може включати програми підтримки, компілятори, бібліотеки коду, набори інструментів та інтерфейси прикладного програмування (API), які об'єднують всі різні компоненти, щоб забезпечити розробку проекту або системи.

В структурах є ключові відмінні риси, які відокремлюють їх від звичайних бібліотек:

● інверсія управління: у фреймворках, на відміну бібліотек чи стандартних користувацьких додатків, управління потоком всієї програми не продиктовано зухвалим, а структурой.

● розширюваність: користувач може розширити структуру зазвичай шляхом вибіркового перевизначення; або програмісти можуть додавати спеціалізований код користувача для забезпечення певної функціональності.

● немодифицируемый код структури: фреймворк, загалом, повинен змінюватися, приймаючи у своїй користувальницькі розширення. Іншими словами, користувачі можуть розширювати структуру, але не повинні змінювати свій код.

Розробники фреймворків прагнуть полегшити розробку програмного забезпечення, дозволяючи розробникам та програмістам приділяти час виконанню вимог до програмного забезпечення, а не працювати зі стандартними низькорівневими деталями, тим самим скорочуючи загальний час розробки. Наприклад, команда, яка використовує веб–фреймворк для розробки банківського веб–сайту, може зосередитися на написанні коду, специфічного для банківської справи, а не на механізмі обробки запитів та керування сесіями.

Фреймворки часто значно збільшують обсяг програмних продуктів, викликаючи явище, зване «роздування коду». Через потреби додатків, орієнтованих на потреби клієнтів, як паралельні, так і додаткові структури іноді виявляються в продукт.

Складність API у фреймворках може призвести до недостатнього скорочення загального часу розробки через додатковий час, необхідний для вивчення його структури. Ця критика особливо справедлива, коли розробники вперше стикаються з новою або незнайомою структурою. Якщо фреймворк не використовується в наступних проектах, час, витрачений на вивчення фреймворку, може бути дорожчим, ніж написання коду без нього. Однак, коли структура буде вивчена, майбутні проекти можна буде завершити, використовуючи менше ресурсів. Концепція фреймворку полягає в компіляції набору стандартних рішень, які підходять для більшості типових завдань, що може підвищити продуктивність розробника при використанні фреймворку. Це твердження не стосується кількості коду, залученого до кінцевого результату, або його відносної ефективності та стислості.

Використання будь–якого рішення у формі бібліотеки включає додаткові функції та компоненти сторонніх розробників, які не використовуються, якщо програмне забезпечення не є компонувальником–компілятором, який створює жорсткий виконуваний модульЦя тенденція у суперечці порушує важливе питання про структуру програмного забезпечення. Використання каркасу, який є елегантним, порівняно з кодом, який просто вирішує проблему, як і раніше, є мистецтвом, а не наукою. А також відповідає принципу DRY «Програмна елегантність» має на увазі ясність, стислість і невеликі відходи (додаткові або сторонні функції, більшість яких визначається користувачем). Наприклад, для тих фреймворків, які генерують код, «елегантність» має на увазі створення коду, який є чистим і зрозумілим для досить обізнаного програміста (і тому легко модифікується), порівняно з тим, який просто генерує правильний код.

Через елегантність проблема полягає в тому, що відносно невелика кількість програмних фреймворків витримала випробування часом: кращі фреймворки змогли витончено розвиватися, оскільки основна технологія, на якій вони були побудовані, була вдосконалена. Навіть там, розвинувшись, багато таких пакетів збережуть успадковані можливості, що роздмухують остаточне програмне забезпечення, оскільки в іншому випадку методи, що замінялися, були збережені паралельно з більш новими методами.

Далі представлені основні існуючі фреймворки безпеки для веб–додатків.

JAAS(Служби автентифікації та авторизації Java) – це API безпеки, який складається з набору пакетів Java, призначених для автентифікації користувачів та авторизації. Він був представлений як додатковий пакет Java SE 1.3, а потім інтегрований в JDK, починаючи з JDK 1.4.

Позитивні якості: є стандартним засобом для реалізації авторизації і легко піддається розширенню. Може бути використаний з іншими інструментами

Недоліки: малий функціонал.

Spring Security– це фреймворк з широкими можливостями для налаштування, який широко використовується для вирішення питань автентифікації та авторизації у веб–застосунках, побудованих на Java.

Переваги: реалізація веб–запитів та викликів методів із використанням технології аспектно–орієнтованого програмування.

Недоліки: сильна залежність від великовагового контексту Spring

Apache Shiroвважається високоефективним фреймворком безпеки для Java, яка виконує криптографію, авторизацію та

керування сеансом на всіх типах програм Java незалежно від їх розміру. Був розроблений як інтуїтивно зрозумілий і простий у використанні, але забезпечує надійні функції безпеки.

OACC(Java Application Security Framework) – це інфраструктура безпеки програм для Java, призначена для управління доступом на рівні об'єктів. У ньому основна увага приділяється наданню повнофункціонального API для забезпечення дотримання та управління вимогами автентифікації та авторизації програми – це повна реалізація потужної та гнучкої моделі безпеки.

Висновок

Вищезазначені рішення пропонують інструменти, які можуть забезпечити безпеку на всіх рівнях. Однак їх слабким місцем є перевірка даних, і вони не реалізують брандмауер або механізм репутації для запитів ресурсів. Ці структури зосереджені виключно на аспектах безпеки та не забезпечують швидкий запуск програми.

1.2. Методологія.

Замкова модель. Ця модель має на увазі, що всі ризики мають бути визначені, можливі збитки – оцінені, а на кожну несподівану подію має бути заготовлений план. Відповідно до загроз і ризиків повинні бути сформовані організаційні та технічні заходи щодо захисту компанії від загроз ІБ. Цей підхід у деяких джерелах називається Castle model of cyber security або замкова модель.

Його основні принципи – поділ всього інформаційного простору на внутрішній, «безпечний» і зовнішній, повний загроз і зловмисників. Ці два сегменти розділені «стінами», можливо,

побудованими пошарово, а зв'язок між зовнішнім та внутрішнім світом забезпечують шлюзи – «gateways».

Agile Cybersecurity Action Plan– Гнучкий спосіб управління кібербезпекою. Він протиставляється «замковій моделі» побудови кібербезпеки та покликаний допомогти CISO максимально ефективно «відпрацьовувати» зміни ландшафту загроз та ризиків. Процес передбачає активну спільну роботу крос–функціональних та крос–організаційних команд для:

- створення та оновлення профілів ризиків;
- оцінки ефективності роботи ІБ–інфраструктури компанії та відповідності її існуючим профілям ризиків;
- виявлення потенційних проблем та недоліків до того, як вони перетворяться на інциденти;
- створення плану усунення виявлених проблем та недоліків;

Agile концепція пропонує кілька методів для визначення дисфункційних вимог у рухомих користувачами історіями процесах розробки. Нефункціональні вимоги поділяються на два великі типи:

- Нефункціональні історії користувача: Блоки тестованої функціональності, написані у форматі історій користувача. Афторами цих історіях може бути внутрішній ІТ персонал. Наприклад: “Як аналітик безпеки, я хочу, щоб система припиняля невдалі спроби автентифікації, так щоб додаток не був схильний до атак”;
- Обмеження: Це спільні питання, які можуть відбиватися на декількох історіях користувача. Це свого роду податок на відповідні роботи. Наприклад, вимога щоб усі розробники валідували дані із полів HTTP форм у web додатках – це обмеження;

Визначивши дані нефункціональні вимоги, існує можливість зробити безпеку видимою і виправдати час, виділений на ті поліпшення, які зазвичай не покращують взаємодію користувача з системою.

Впровадження процесів інформаційної безпеки в Agile

У нинішньому конкурентному середовищі своєчасні випуски продуктів мають вирішальне значення, провідні IT-компанії тепер використовують стратегії безперервних змін, що дозволяють оновлювати кілька продуктів за один день. Незважаючи на те, що гнучкі методології скорочують час, необхідний для доставки початкової версії продукту, ефективність команди розробників з точки зору доставленої вартості за одиницю часу може негативно вплинути. На відміну від моделі водоспаду, яка значною мірою залежить від детальної документації та зусиль команди розробників у кодуванні, гнучкі методології вимагають значного часу, присвяченого спілкуванню. Гнучкі методології передають відповідальність за прийняття рішень команді розробників, яка бере на себе повну відповідальність за результати проекту.

Як правило, тестування безпеки є обов’язком команди тестувальників і проводиться окремо від діяльності з розробки програмного забезпечення та планування. Функції безпеки не вписуються в жодний процес розробки програмного забезпечення. Однак у більшості випадків безпека є невід’ємною частиною програмного забезпечення, тому постійну інтеграцію та розгортання (CI/CD) не слід припиняти для тестування безпеки до наступної ітерації розробки.

Подібно до продуктивності або взаємодії з користувачем (UX), фахівці з безпеки прагнуть досягти цілей безпеки. Вони є експертами в бездоганній інтеграції інструментів безпеки в інструментарій розробки та середовище CI/CD.

Безпека у процесі розробки

Завдяки постійним доповненням безпека будь–якого програмного забезпечення розвивається з часом. Замість того, щоб концентруватися на комерційній цінності програмного забезпечення та досягненні його місії, команда розробників в кінцевому підсумку змушена додавати функції безпеки, який вимагає уваги для фахівців з безпеки, до обов'язків яких входить обробка аспектів безпеки.

Наприклад, коли програмне забезпечення запитує користувача збереження паролів, або коли воно включає шифрування і т. д., користувач бачить внутрішню роботу системи безпеки. Необхідно змінити код, щоб функції безпеки діяли потай від користувача, а не відкрито, дозволяючи користувачам бачити, як працює програмне забезпечення.

Рання імплементація.

Функції безпеки стосуються елементів програмного забезпечення, які забезпечують його внутрішню роботу, наприклад шифрування, зберігання паролів і параметри керування збоями. Коли визначено потреби в безпеці, команда розробників має визначити пріоритетність модифікації коду, щоб зробити функцію безпечною, а не додавати додатковий механізм безпеки для користувача.

Вносячи невеликі поступові зміни з часом, можна покращити безпеку програмного забезпечення. Важливо зосередитися на загальній цінності програмного забезпечення для кінцевого користувача, а не лише на додаванні функцій безпеки. По можливості слід використовувати безпечні за замовчуванням фреймворки. Розробкою та впровадженням функцій безпеки, таких як автентифікація, схеми зберігання паролів і криптографічні алгоритми, повинні займатися досвідчені фахівці, які спеціалізуються в цих областях. Бажано використовувати перевірені та надійні реалізації, а не розробляти їх з нуля.

Робота з ризиками замість помилок

Більшість заходів із розробки програмного забезпечення передбачає виявлення та усунення вразливостей безпеки, щоб гарантувати, що програмне забезпечення вважається «безпечним». Розробляючи програмне забезпечення, фахівці з безпеки та розробники повинні розглядати способи мінімізації ризиків для бізнесу, користувачів і даних. Однак безпека рідко є простим питанням виправлення помилки. Натомість спеціалісти з програмного забезпечення та безпеки повинні віддавати пріоритет управлінню ризиками під час розробки рішень для боротьби з потенційними проблемами безпеки.

Цілісне уявлення про ризики є найефективнішим підходом до управління ризиками, незалежно від того, чи це пов’язано з написанням коду, моніторингом підозрілої діяльності або використанням юридичних і договірних засобів контролю замість технічних. Бажано уникати створення вичерпного списку окремих помилок, які необхідно усунути.

Основні проблеми

Принципи Agile-методології не можуть бути повністю застосовані до управління кібербезпекою, оскільки лише стадії «Передбачити» та «Запобігти» чотирьохетапного життєвого циклу, які включають стратегічне планування, оцінку ризиків і розробку рішень, сумісні з Agile.

Крім того, на якість програмного забезпечення, розробленого для онлайн-бізнес-рішень, значною мірою впливає досвід і знання команди розробників і спеціалістів з кібербезпеки. Хоча більшість вразливостей можна легко усунути за допомогою існуючих рішень, кожен новий проект потребує часу та навичок команди для впровадження цих інструментів.

1.3. Вимоги до безпеки веб–програми

Дослідження вразливостей веб–застосунків, що проводяться компанією Positive Technologies в 2016 році показало, що загальний рівень захищеності веб–додатку продовжує знижуватися. Розробники прагнуть забезпечити максимальну функціональність систем та не завжди приділяють належну увагу безпеці коду. При цьому відзначається широка поширеність недоліків, пов'язаних з помилками адміністрування. Їхня небезпека, як правило, невисока, однак у разі експлуатації деяких таких уразливостей порушник може не тільки отримати чутливу інформацію (наприклад, внаслідок її розголошення на сторінках додатків), а й отримати несанкціонований доступ до системи (у випадках підбору та перехоплення облікових даних або успішної атаки на сесію).

Використання коду

Якщо програма може отримувати введення користувача, яке входить до базової бази даних, команди або виклику, програма може бути вразливою для атаки на основі ін'єкції коду. Ін'єкційні недоліки є набір вразливостей безпеки, які виникають, коли підозрілі дані вставляються у додаток у вигляді команди або запиту. Відомі ін'єкційні атаки включають SQL, OS, XXE та LDAP.

Найбільш поширеними атаками для ін'єкцій коду є SQL Injections, також відомі як SQLi. Атака SQLi виконується, коли невірний код відправляється на сервер бази даних, що призводить до псування даних. І цей стиль атаки настільки простий і простий, будь–хто, хто має доступ до Інтернету, може це зробити – SQLi–скрипти доступні для скачування і можуть бути легко придбані.

Некоректна автентифікація та управління сесією

Коли функції програми не реалізовані правильно, злочинець може зламати або скомпрометувати компрометують паролі, ідентифікатори сеансів та використовують інші недоліки, використовуючи вкрадені облікові дані. Сеанси повинні бути унікальними для кожного окремого користувача, і без будь–якого необхідного керування сеансом зловмисник може проникнути всередину, замаскований під користувачем, щоб вкрасти токени та паролі, щоб отримати доступ до нього.

Міжсайтовий скриптинг

Міжсайтовий скриптинг широко відомий як XSS є вразливістю, яка часто зустрічається у веб-додатках. XSS дозволяє зловмисникам впроваджувати клієнтські скрипти у загальнодоступні веб–сторінки і, у багатьох випадках, може використовуватися зловмисниками для проходження через засоби контролю доступу.

Це робиться шляхом обману браузера, щоб він приймав дані з ненадійного джерела, і зазвичай відбувається, коли зловмисники використовують знайомий код (наприклад, JavaScript), оскільки розробники не очищають ці символи.

Програми, які дозволяють вводити користувача без повного контролю над виходом, можуть постраждати від атак XSS. Коли атака XSS успішна, зловмисники можуть завдати серйозної шкоди веб–додатку.

Недостатній моніторинг

Недостатня реєстрація та моніторинг у поєднанні з відсутньою чи неефективною інтеграцією з реагуванням на інциденти дозволяють зловмисникам продовжувати атакувати системи, підтримувати сталість, повертатися до більшої кількості систем та підробляти, вилучати чи знищувати дані. Більшість досліджень з порушенням показують, що час виявлення порушення становить понад 200 днів, які зазвичай виявляються зовнішніми сторонами, а не внутрішніми процесами або моніторингом.

Міжсайтові запити

Атака CSRF змушує браузер з жертви відправляти надісланий HTTP–запит, у тому числі кукі–файл сеансу жертви та будь–яку іншу автоматично включну інформацію автентифікації в вразливу веб–додаток. Це дозволяє зловмиснику змусити браузер жертви генерувати запити, які вважаються валідними запитами жертви вразливим додатком.

1.4. Висновок

Хоча існуючі фреймворки задовольняють усі перераховані вимоги безпеки, частота появи цих вразливостей не змінюється протягом багатьох років.

Як видно з рис. 1.1., ін'єкції, міжсайтовий скриптинг, ризики пов'язані з авторизацією та сесіями залишаються на лідируючих позиціях.

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Рис. 1.1. OWASP. 10 найпоширеніших уразливостей.

Дане спостереження дозволяє зробити висновок про неефективність існуючих інструментів розробки.

Дослідження методологій розробки показує суперечність між популярним сімейством методологій розробки Agile та вимогами безпеки. Цей конфлікт призводить до ігнорування вимог безпеки на користь функціональних вимог та прискорення виходу продукту на ринок.

Іншою важливою проблемою процесу розробки є висока залежність якості від досвіду команди розробників, пов'язана з необхідністю налаштування та використання інструментів розробки.

Дані проблеми дозволяють сформулювати такі критерії необхідного розв'язання:

- низька залежність від досвіду команди;
- низька трудомісткість;
- закриття найчастіших вразливостей.

Виходячи з цих критеріїв пропонується наступне рішення:

Наведену нижче структуру можна використовувати для запуску веб-програми та використання Spring Security для механізмів керування сеансами, автентифікації та авторизації. Таким чином можна реалізувати перевірку вхідних даних і механізми зберігання даних користувача, що зменшує складність використання. Крім того, веб-додаток можна розробити з уже обмеженого стану зі стандартними налаштуваннями відповідно до методології Agile.

Такий підхід дозволяє розробникам обмежити свої можливості або «зменшити ризики та випустити продукт пізніше», або «не зменшувати ризики та випустити продукт раніше», обидва з яких є прийнятними з точки зору ринку. Натомість вони можуть зосередитися на підключенні фреймворку та реалізації необхідної функціональності, що дає змогу створювати продукт без найпоширеніших загроз навіть для тих, хто має невеликий досвід розробки веб-додатків.

Крім того, це рішення надає можливість змінювати параметри безпеки відповідно до бізнес-логіки поточного продукту, дозволяючи розробникам легко змінювати всі параметри безпеки в наступних ітераціях. Завдяки цьому процес розробки стає більш гнучким, ефективним і чутливим до мінливих вимог.

2 РЕАЛІЗАЦІЯ ФУНКЦІЙ

2.1 Обґрунтування вибору технологій

На даний момент Java – найбільш популярний інструмент для веб–розробки.

Мова Java міцно зміцнила свої позиції на першому місці рейтингу мов програмування TIOBE. За 2023 рік у загальному рейтингу він зменшився на 6%, але займає майже 15% усього ринку. Поріг входження в середу розробників цією мовою поступово знижується.

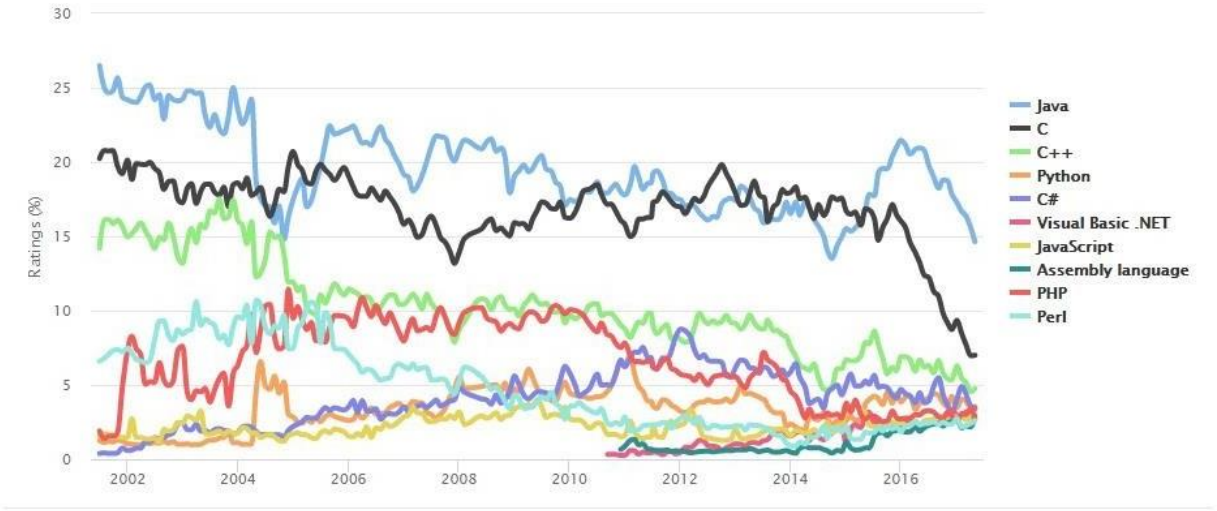


Рисунок 2.1. Діаграма використання мов розробки

Java також є найбільш популярною мовою для веб–розробки, що показано на малюнку 2.2.

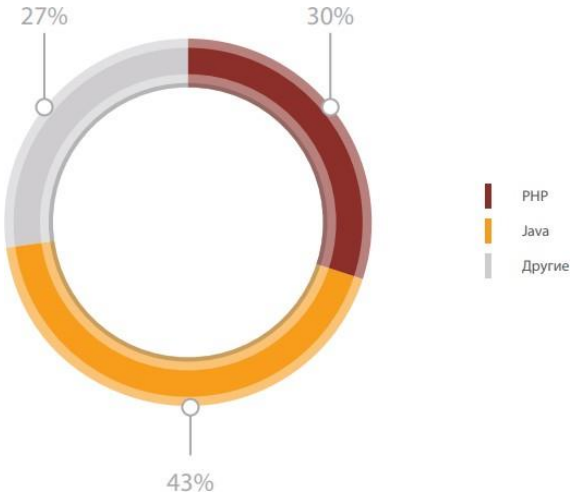


Рисунок 2.2. Діаграма використання мов програмування для розробки веб–застосунків за даними компанії Positiv Technologies

Як система контролю версій було обрано git.

Git – це розподілена система управління версіями з відкритим вихідним кодом і вільна для поширення, призначена для обробки від невеликих до великих проектів з високою швидкістю і ефективністю.

Як репозиторій був обраний GitHub як найбільший веб–сервіс для хостингу IT–проектів з відкритим вихідним кодом.

2.2Криптографія

Вибір та обґрунтування засобу забезпечення криптографічного захисту, хешування пароля та генерація солі

Як хешируючу функцію був обраний алгоритм bCrypt, заснований на алгоритмі Blowfish. З–за присутності стандартної реалізації, популярності та генерації випадкової солі.

Blowfish зарекомендував себе як надійний алгоритм, тому реалізований у багатьох програмах, де не потрібна часта зміна ключа і необхідна висока швидкість шифрування/розшифрування:

- хешування паролів;
- захист електронної пошти та файлів;
- GnuPG (безпечне зберігання та передача);
- у лініях зв'язку: зв'язка ElGamal (не запатентований) або RSA (дія патенту закінчилася у 2000 році) та Blowfish замість IDEA;
- у маршрутизаторі Intel Express 8100 із ключем довжиною 144 біти;
- забезпечення безпеки у протоколах мережного та транспортного рівня;
- SSH (транспортний рівень);
- OpenVPN (створення зашифрованих каналів);

Швидкість алгоритму шифрування багато в чому залежить від використовуваної техніки та системи команд. На різних архітектурах один алгоритм може значно випереджати за швидкістю своїх конкурентів, але в іншому ситуація може зрівнятися і навіть змінитись у протилежний бік. Більш того, програмна реалізація значно залежить від компілятора, що використовується. Використання асемблерного коду може підвищити швидкість шифрування. На швидкість шифрування впливає час виконання операцій mov, add, xor, причому час виконання операцій збільшується при зверненні до оперативної пам'яті (для процесорів Pentium серії приблизно в 5 разів). Blowfish показує вищі результати під час використання кешу для зберігання всіх з'єднань. І тут він випереджає алгоритми DES, IDEA. На відставання IDEA впливає операція додавання за модулем 232+1.

2.3 Аутентифікація

Авторизація побудована на принципах фреймворку Spring Security з додатковим регулюванням бази даних, перевірки запитів за чорним списком.

Основа авторизації побудована на фреймворку Spring Security, що включає реалізацію AuthenticationProvider, звану DaoAuthenticationProvider. Цей постачальник автентифікації сумісний з усіма механізмами автентифікації, які генерують кілька значень:

Ім'я користувача та Пароль і, ймовірно, є найчастіше використовуваним постачальником у рамках цього фреймворку. Як і більшість іншихпостачальників утентифікації, DaoAuthenticationProvider використовує UserDetailsService для пошуку імені користувача, пароля. На відміну від більшості інших постачальників автентифікації, які використовують UserDetailsService, цей постачальник автентичності фактично вимагає, щоб пароль був представлений у запиті на автентифікацію.

PasswordEncoder та SaltSource є необов'язковими. PasswordEncoder забезпечує кодування та декодування паролів, представлених в об'єкті UserDetails, який повертається з налаштованого UserDetailsService. SaltSource дозволяє заповнювати паролі, що підвищує безпеку паролів репозиторії аутентифікації. Реалізація PasswordEncoder має Spring Security, що охоплює кодування MD5, SHA і cleartext. Також реалізовані дві реалізації SaltSource: SystemWideSaltSource, який кодує всі паролі з тією ж сіллю та ReflectionSaltSource, який перевіряє задану властивість об'єкта UserDetails, що повертається, для отримання солі.

На додаток до властивостей вище, DaoAuthenticationProvider підтримує додаткове кешування об'єктів UserDetails. Інтерфейс UserCache дозволяє DaoAuthenticationProvider поміщати об'єкт UserDetails в кеш і витягувати його з кеша при наступних спробах автентифікації для одного і того ж імені користувача. За промовчанням DaoAuthenticationProvider використовує NullUserCache, який не виконує кешування.

У більшості випадків, коли програма є веб–додатком зі збереженням стану, немає необхідності у використанні

кеша, оскільки інформація про автентифікацію користувача зберігатиметься в HttpSession.

Було прийнято конструктивне рішення не підтримувати блокування облікового запису в DaoAuthenticationProvider, оскільки це збільшило б складність інтерфейсу UserDetailsService. Наприклад, збільшення кількості невдалих спроб аутентифікації знадобиться метод. Така функціональність може бути легко забезпечена за рахунок використання функцій публікації подій програми, що обговорюються нижче на рисунку 2.3.

DaoAuthenticationProvider повертає об'єкт аутентифікації, який, у свою чергу, має свій основний набір властивостей. Основним буде або String (по суті ім'я користувача), або об'єкт UserDetails (який був переглянутий за допомогою UserDetailsService). За замовчуванням повертається UserDetails, оскільки це дозволяє додаткам додавати додаткові властивості, які потенційно використовуються в додатках, такі як повне ім'я користувача, адресу електронної пошти і т.д.

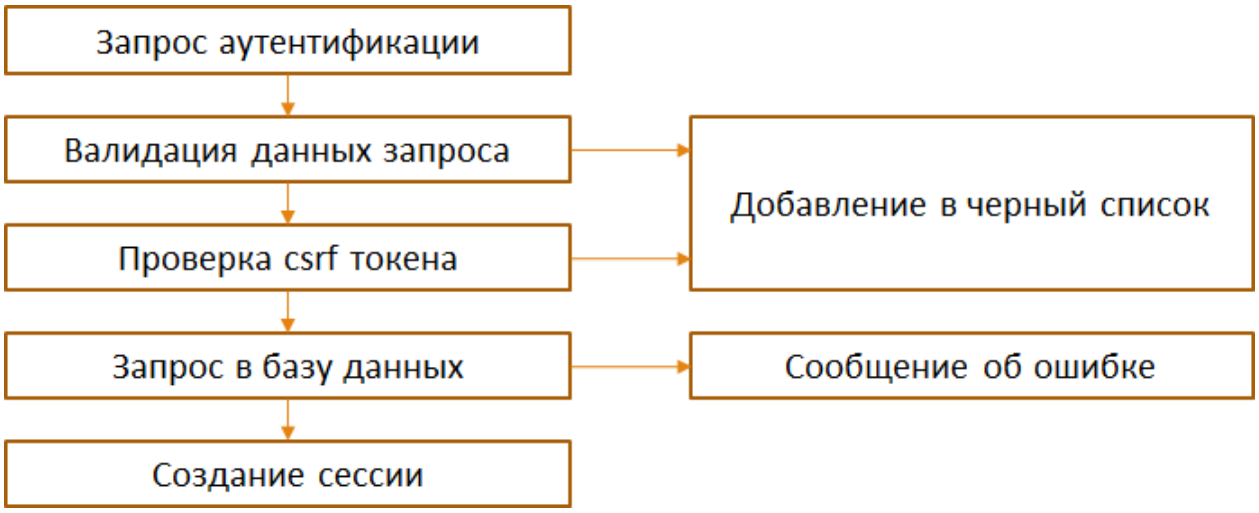


Рисунок 2.3. Механізм автентифікації.

2.4 База даних

Робота з об'єктно–орієнтованим програмним забезпеченням та реляційними базами даних може бути громіздкою та трудомісткою. Витрати на розробку значно вищі через невідповідність парадигми тим часом, як дані представлені в об'єктах порівняно з реляційними базами даних. У цьому розробці на вирішення завдань

Об'єктно–реляційне відображення використовується бібліотека Hibernate. Hibernate – це рішення Object / Relational Mapping (ORM) для середовищ Java. Термін Object/Relational Mapping відноситься до методу відображення даних між поданням об'єктної моделі в подання реляційної моделі даних. З іншого боку, стаття OrmHate Мартіна Фаулера розглядає багато проблем невідповідності цих методів.

Щоб повністю зрозуміти Hibernate, потрібно мати базове розуміння концепцій SQL і принципів моделювання даних, хоча це не вимагає глибокого розуміння SQL. Hibernate обробляє відображення даних з класів Java на таблиці бази даних і типів даних Java на типи даних SQL, а також надає служби запитів і пошуку даних. Усунувши потребу в ручній обробці даних SQL і JDBC, Hibernate може значно скоротити час розробки. Його мета полягає в тому, щоб звільнити розробників від 95% звичайних завдань із збереження даних, дозволяючи при цьому використовувати в проекті чистий SQL.

Гібернація не є найкращим рішенням для програм, орієнтованих на дані, які використовують лише збережені процедури для реалізації бізнес-логіки в базі даних. Однак він дуже ефективний з об’єктно-орієнтованими моделями домену та бізнес-логікою Java середнього рівня. Крім того, Hibernate може допомогти спростити загальне завдання перекладу наборів результатів із табличного представлення в об’єктний граф шляхом видалення або інкапсуляції специфічного постачальника SQL.

Виходячи з поставлених завдань, як засіб об'єктно–реляційного відображення був обраний Hibernate. Створення імплементації UserDetails, яка забезпечує реалізацію:

1. Хешування пароля з використанням випадкової генерації;
2. Можливість блокування користувача;
3. Реалізовано білдер;

Для реалізації аутентифікації та авторизації може використовуватися реалізований AuthorizationService, що використовує клас UserEntity, код якого представлений нижче.

```
@Data @Builder @Entity
@Table(name ="user")@AllArgsConstructor
public classUserEntityimplementsUserDetails {@Id
@Column(name ="id")
@GeneratedValue(strategy = GenerationType.AUTO)
private longid;


@Basic @NonNull
@Column(name ="username")privateStringusername;

@Basic @NonNull
@Column(name ="password")privateStringpassword;

@Basic @NonNull
@Column(name ="enabled")private booleanenabled=true;

@Basic @NonNull
@Column(name="accountNonExpired")private
booleanaccountNonExpired=true;

@Basic
@NonNull
@Column(name="accountNonExpired")private
booleanaccountNonLocked=true;

@Basic @NonNull
@Column(name="кредитентівNonExpired")private
booleancredentialsNonExpired=true;
}
```

Інтерфейс централізованого сховища. Захоплює тип сутності керування, і навіть тип ідентифікатора типу домену. Загальна мета полягає в тому, щоб зберігати інформацію про тип, а також виявляти інтерфейси, які розширюють процес під час сканування для легкого створення Spring bean.

Доменні репозиторії, які розширюють цей інтерфейс, можуть вибірково виявляти методи CRUD

Репозиторій для об'єктів UserEntity:

```
public interfaceUserRepositoryextendsJpaRepository<UserEntity, Long>
```

Налаштування бази даних за замовчуванням:

```
@Bean
@ConditionalOnBean(name ="dataSource")@ConditionalOnMissingBean
publicLocalContainerEntityManagerFactoryBean entityManagerFactory() {
finalLocalContainerEntityManagerFactoryBean em =new
LocalContainerEntityManagerFactoryBean(); em.setDataSource(dataSource());
em.setPackagesToScan("com.astel.security.model");
em.setJpaVendorAdapter(newHibernateJpaVendorAdapter());if(additionalProperties(
) !=null) {
em.setJpaProperties(additionalProperties());
}
returnem;
}
@Bean
@ConditionalOnMissingBean(type ="JpaTransactionManager")
JpaTransactionManager transactionManager(finalEntityManagerFactory
entityManagerFactory) {
finalJpaTransactionManager transactionManager
=newJpaTransactionManager();
transactionManager.setEntityManagerFactory(entityManagerFactory);
returntransactionManager;
}
```

@ConditionalOnResource(Resources
="classpath:mysql.properties")@Conditional(HibernateCondition.class)

finalProperties additionalProperties() {

finalProperties hibernateProperties =newProperties();
hibernateProperties.setProperty("hibernate.hbm2ddl.auto",

env.getProperty("mysql–hibernate.hbm2ddl.auto"));
hibernateProperties.setProperty("hibernate.dialect",env.getProperty("mysql–

hibernate.dialect"));

hibernateProperties.setProperty("hibernate.show_sql",env.getProperty("mysql–hibernate.show_sql") !=null?env.getProperty("mysql–hibernate.show_sql") :"false");

returnhibernateProperties;

}

2.5 Валідація даних

Реалізовано захист від Міжсайтової підробки запитів (CSRF) шляхом внесення стандартних налаштувань Spring Security.

Код вікна логіна включає поле

<inputname="_csrf"type="hidden"value="67b99669–74d0–4572–9cfb–a859da5d3313" />

Значення ключа є випадковим і унікальним для кожної сесії.

Для захисту від Міжсайтового скриптингу використовується проект з відкритим вихідним кодом Harbinger, що реалізує функції міжмережевого екрана та розповсюджується під ліцензією Apache 2.

Для підключення залежності від цього проекту до gradle потрібно додати наступні рядки:

allprojects { repositories {

maven {url'https://jitpack.io'}

}}

compile'com.github.ctrl–alt–dev:harbinger:v1.0.0'

Для реалізації фільтрів використовуються правила, які зберігаються в налаштуваннях проекту, у текстовому файлі, по одному на рядок. Параметри розділені комами.

Правила конфігуруються за такими принципами показаних у Таблиці 2.1.

Таблиця 2.1. Конфігурація правил для валідації даних:

Назва	Значення	приклад
Ім'я	Рядок	XSS
Пріоритет	LOW, MID, HIGH	MID
Паттерн	Паттерн заснований на регулярних виразах	<script[^>]*>
Коментар	#	# коментар

Стандартний список правил:

#

XSS

XSS,MID,<script[^>]*> XSS,MID,([\s\'"`;\/0–9\=]+on\w+\s*=)

XSS,MID,(?:[\s]style=[\s\S]|<style[^>]*>[\s\S]*?|<object[^>]*>[\s\S]*?|<meta[^>]

>[\s\S]?|<applet[^>]*>[\s\S]*?)#

File Inclusion

LFI,MID,(?:\/etc\/|\/..|\\..|web\.xml|boot\.ini\b)#

SQLi

SQL,LOW,^'$ SQL,MID,;[ ]*—

SQL,MID,['"`] *(or|and) *['"`]

При вхідному запиті HTTP ініціалізуються дані запиту Дані можуть бути доповнені поточним користувачем, ідентифікатором сеансу або віддаленою IP–адресою до його передачі до збирача даних запитів. Складальник даних реєструє та групує дані по сеансу та IP та підсумовує результат, створюючи агрегування доказів.

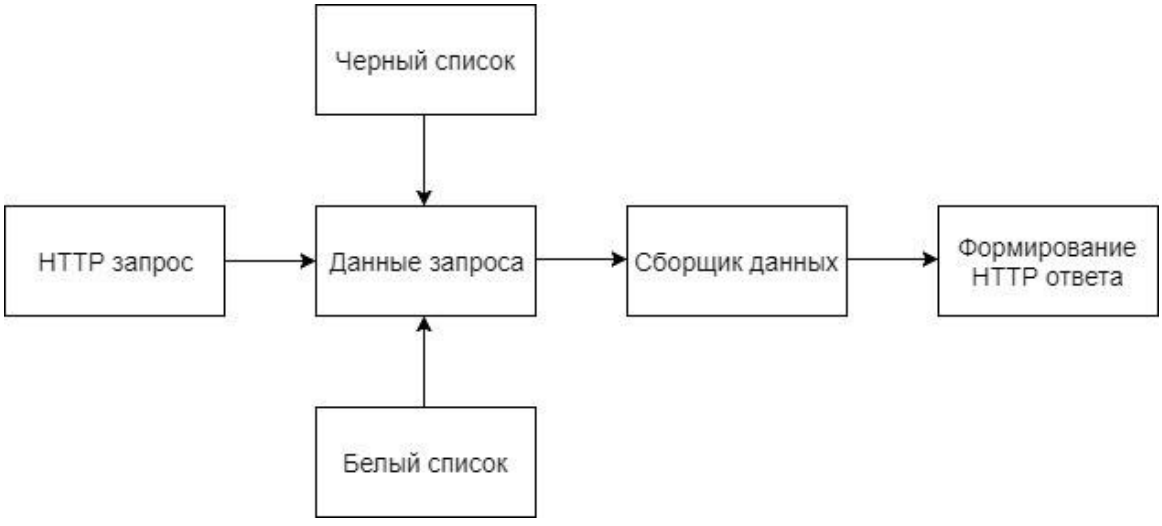


Рисунок 2.4. Фільтрування запитів

Потім це агрегування подається до Реагування відповіді, щоб визначити дію відповіді, яка може бути відхиленням введення, анулюванням сеансу, тимчасовим занесенням до чорного списку IP або нічим.

Є можливість використовувати білий список, щоб дозволити збирачеві денних ігнорувати будь–які дані, що відповідають певним характеристикам. Білий список підтримує придушення даних на основі IP–адреси, URL–адреси, імені параметра та користувача. Також він підтримує логічні оператори OR та AND.

2.6Логування

Багато систем дозволяють керувати мережевими пристроями, операційною системою, веб–сервером, поштовим сервером та сервером бази даних, але часто не реєструється журнал подій програм, вимкнений або погано налаштований. Він забезпечує набагато глибше розуміння, ніж керування інфраструктурою. Веде веб–додатки (наприклад, веб–сайт або веб–сервіс) набагато більше, ніж включення журналів веб–сервера.

Журнали додатків мають бути узгодженими у додатку, узгодженими у портфелі додатків організації та, якщо це

необхідно, використовувати галузеві стандарти, тому зареєстровані дані подій можуть використовуватися, корелюватися, аналізуватися та керуватися за допомогою різних систем.

Журнали програм завжди повинні бути включені для подій безпеки. Журнали додатків є безцінними даними для:

1. Ідентифікація інцидентів безпеки;
2. Моніторинг порушень політики;
3. встановлення вихідних умов;
4. Допомога усунення невідповідностей;
5. Надання інформації про проблеми та незвичайні умови;
6. Внесення додаткових даних щодо конкретних випадків для розслідування інцидентів, які відсутні в інших журнальних джерелах;

Як правило, програми зберігають дані журналу подій у файловій системі або базі даних, наприклад SQL або NoSQL. Програми, встановлені на мобільних пристроях або настільних комп'ютерах, також можуть використовувати локальне сховище та бази даних під час надсилання даних у віддалене сховище. Вибрана структура може вплинути на доступні параметри журналювання. Усі програми можуть надсилати дані подій у віддалені системи, такі як система централізованого збору журналів або система керування, як-от SIEM або SEM, або інша програма, розташована в іншому місці.

При використанні файлової системи для журналювання рекомендується використовувати окремий розділ із операційною системою, файлами інших програм і вмістом користувача. Файлові журнали повинні мати суворі дозволи щодо доступу користувачів до каталогів і дозволів на файли в каталогах.

У веб-додатках бажано не відображати журнали в місцях, доступних через Інтернет. Якщо вони відображаються, вони повинні мати обмежений доступ і бути налаштованими за допомогою типу MIME звичайного тексту (а не HTML).

Для реєстрації в базі даних найкраще використовувати окремий обліковий запис бази даних, призначений виключно для реєстрації даних із дуже обмеженими дозволами для доступу до бази даних, таблиць, функцій і команд.

Стандартні формати та захищені протоколи слід використовувати для запису та надсилання даних подій або файлів журналу до інших систем, наприклад Common Log File System (CLFS), Common Event Format (CEF) через syslog і, можливо, Common Expression (CEE) у майбутньому . Стандартні формати дозволяють спростити інтеграцію з централізованими службами реєстрації.

Кожен запис у журналі має містити достатню інформацію для подальшого моніторингу та аналізу. Це можуть бути повні дані контенту, але швидше за все це витримка або просто зведені властивості.

Журнали програм повинні записувати «коли, де, хто і що» для кожної події. Властивості для них будуть різними залежно від архітектури, класу програми та хост–системи/пристрою, але часто включають таке:

Коли:

1. Дата та час реєстрації (міжнародний формат);
2. Дата та час події – позначка часу події може відрізнятися від часу реєстрації, наприклад. сервер, де клієнтська програма розміщена на віддаленому пристрої, який виконується лише періодично або періодично;
3. Ідентифікатор взаємодії;

Де:

1. Ідентифікатор програми, наприклад. ім'я та версія;
2. Адреса програми, наприклад. ім'я кластера/хоста або сервер IPv4 або IPv6 адресу та номер порту, ідентифікатор робочої станції, ідентифікатор локального пристрою;
3. Обслуговування, наприклад. ім'я та протокол геолокації;
4. Вікно/форма/сторінка, наприклад. URL точки входу та HTTP–метод для веб–програми, ім'я діалогового вікна;
5. Розташування коду, наприклад. ім'я сценарію, ім'я модуля; Хто
6. Адреса джерела, наприклад. ідентифікатор пристрою/машини користувача, IP–адреса користувача, ідентифікатор осередку/RF–вежі, номер мобільного телефону;
7. Ідентифікація користувача (якщо вона автентифікована або відома іншим чином), наприклад. значення первинного ключа таблиці бази даних користувача; ім'я користувача; номер ліцензії;

Які:

1. Тип події;

2. Істотність події, наприклад. {0 = аварійний, 1=попередження, ..., 7 = debug}, {fatal, error, warning, info, debug, trace};

3. Прапор події безпеки (якщо журнали також містяться дані подій безпеки);

4. Опис;

При реєстрації даних необхідно враховувати чинне законодавство. Наприклад, перехоплення деяких повідомлень, моніторинг працівників та збирання деяких даних без згоди можуть бути незаконними.

Не можна виключати будь–які події від «відомих» користувачів, таких як інші внутрішні системи, «довірені» сторонні компанії, роботи пошукових систем, системи безперебійного/технологічного та інших віддалених систем моніторингу, тестери пера, аудитори. Однак, ви можете включити прапор класифікації для кожного з них у записані дані.

Зазвичай не слід записувати такі дані в журнали, але їх слід видалити, замаскувати, дезінфікувати, хешувати або зашифрувати:

1. Вихідний код програми;
2. Значення ідентифікації сеансу (подумайте про заміну хешованим значенням, якщо необхідно для відстеження подій, специфічних для сеансу);
3. Доступ до токенів;
4. Чутливі персональні дані та деякі форми особистої інформації, що ідентифікується (PII), наприклад. здоров'я, урядові ідентифікатори, уразливі люди;
5. Аутентифікаційні паролі;
6. Рядки підключення бази даних;
7. Ключі шифрування та інші основні секрети;
8. Дані власника банківського рахунку або платіжної картки;
9. Дані вищої класифікації безпеки, ніж система реєстрації дозволяють зберігати;
10. Комерційна інформація;
11. Інформація незаконно збиратиме у відповідних юрисдикціях;
12. Інформація, яку користувач відмовився від збору або не погодився, наприклад, використання не відстежується, або коли згода на збір закінчилася;

Іноді можуть існувати такі дані і, водночас, корисні для подальшого дослідження, також може бути необхідно обробити будь–яким особливим чином до того, як подія буде записана:

1. Шляхи файлів
2. Рядки підключення бази даних
3. Внутрішні мережеві імена та адреси
4. Нечутливі особисті дані (наприклад, особисті імена, телефонні номери, адреси електронної пошти)

Необхідно розробити методи деідентифікації особистих даних, такі як видалення, скремблювання або псевдонімізація прямих і непрямих ідентифікаторів, коли особистість особи не потрібна, або ризик вважається занадто великим.

Реалізується створенням інтерфейсу SecuredService та реєстрацією всіх методів імплементації даного інтерфейсу за допомогою log4j.

```
@Aspect @Component @Log4j
public classLogging {@After("target(com.astel.security.SecuredService)")
publicObject securedMethodsLogging(ProceedingJoinPoint pjp)throwsThrowable
{
Object retVal = pjp.proceed();
log.info(pjp.getTarget().getClass().getName() +" "+
pjp.getSignature().getName());
returnretVal;
}
}
```

Стандартні налаштування логування представлені в наступною таблиці 2.2:

Параметр	Значення
File	${catalina.home}/logs/log_file.log
rootLogger	INFO, file
MaxFileSize	10MB
MaxBackupIndex	10
ConversionPattern	%d{yyyy–MM–dd HH:mm:ss} %–5p %c{1}:%L – %m%n

Таблиця 2.2. Базові налаштування логування

2.7Перевірка безпеки програми

Для перевірки безпеки програми створеної на основі даного фреймворку використовується сканер уразливостей Wapiti.

Wapiti побудований на модулях, які підключаються під час сканування

— backup, blindsql, crlf, exec, file, htaccess, ніхто, permanentxss, sql, xss, buster, shellshock.

Wapiti є консольним сканером веб–додатків, який у своїй основі несе принцип BlackBox тестування (аналізуються не вихідні програми, а відповіді сервера на запити зі зміненими параметрами). Утиліта спочатку аналізує структуру сайту, шукає доступні сценарії, аналізує параметри, та був включає свій фаззер.

Фазинг – це технологія автоматизованого тестування програмного забезпечення з метою виявлення потенційних уразливостей, що охоплює велику кількість граничних випадків шляхом породження некоректних вхідних даних. В якості вхідних даних при цьому можуть виступати файли, що обробляються додатком, інформація, що передається по мережевим протоколам, функції прикладного інтерфейсу і т.д. Термін виник у епоху аналогових телефонних мереж, коли виявилося, що випадковий шум (fuzz) лінії може викликати збій у керуючим модемом програмному забезпеченні.

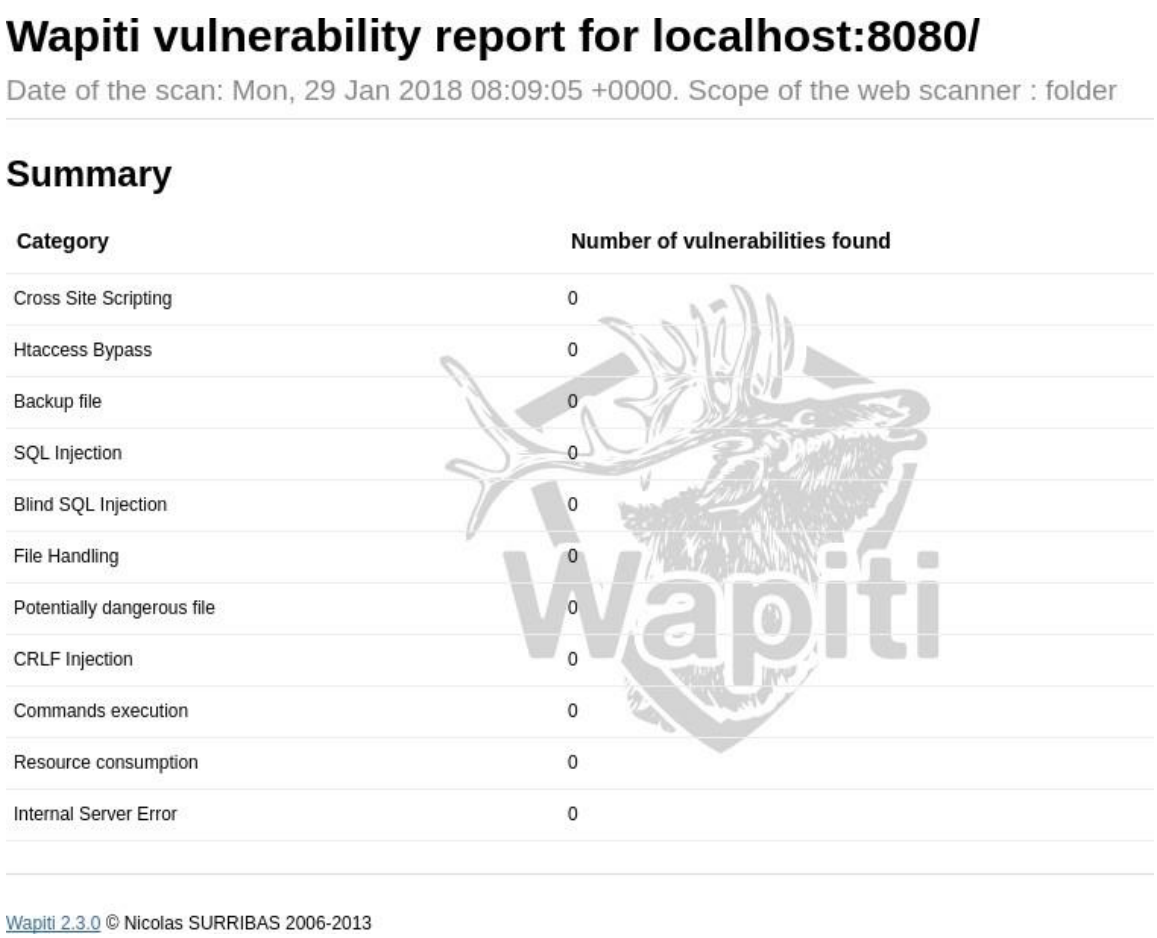
Вперше роботу на цю тему було опубліковано в 1988 році професором Університету штату Вісконсін Бартоном Міллером. Він

створив найпростіший фазер параметрів командного рядка програм ОС UNIX. Тим не менш, довгий час технологія була невідомою для більшості фахівців і лише в останні роки набула значного поширення. В даний момент, фазинг впроваджується як обов'язковий етап у процес розробки програмного забезпечення.

Щоб протестувати безпеку, що реалізується фреймворком, створимо базовий додаток, запустимо його на локальному сервері на основі фреймворку та запустимо

wapiti http://localhost:8080

Звіт перевірки на вразливості показаний на Рисунку 2.5.



Wapiti vulnerability report for localhost:8080/

Date of the scan: Mon, 29 Jan 2018 08:09:05 +0000. Scope of the web scanner : folder

Summary

Category	Number of vulnerabilities found
Cross Site Scripting	0
Htaccess Bypass	0
Backup file	0
SQL Injection	0
Blind SQL Injection	0
File Handling	0
Potentially dangerous file	0
CRLF Injection	0
Commands execution	0
Resource consumption	0
Internal Server Error	0

Wapiti 2.3.0 © Nicolas SURRIBAS 2006-2013

Рисунок 2.5. Звіт про автоматизовану перевірку веб–додатку.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ РОЗРОБЛЕНОГО ФРЕЙМВОРКА

3.1 Галузь застосування

Цей фреймворк необхідний для створення нових веб–додатків за умови максимально швидкого старту з імплементованими налаштуваннями безпеки. Відповідно до рис. 3.1. Для конфігурування безпеки на пізніх стадіях необхідне додаткове налаштування, зумовлене бізнес–логікою самого додатка.

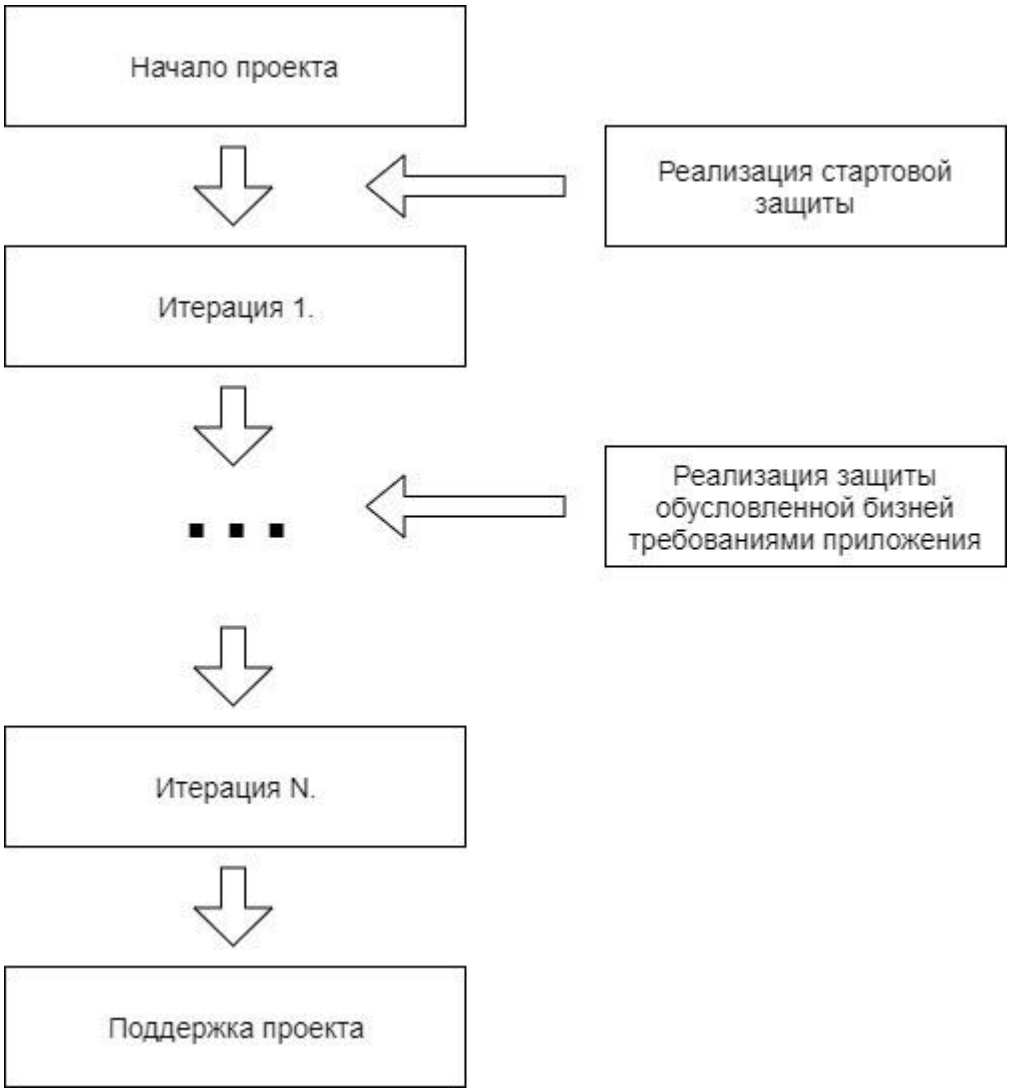


Рисунок 3.1. Реалізація захисту під час розробки програми.

Для складання даного проекту необхідно:

- Завантажити та встановити систему контролю версій git;
- Завантажити та встановити систему автоматизованого складання Gradle;
- Налаштувати будь–яке середовище розробки для java;
- Клонувати репозиторій із веб–сервісу GitHub

https://github.com/Astel/SecurityFramework.git

- Імпортувати цей репозиторій як gradle проект;
- Запустити командуgradle buildз консолі або з плагіна у вибраному середовищі розробки;

3.2Використання

Для використання потрібно:

- Підключити залежність від гітрепозиторію. Рекомендується використовувати плагін Jitpack.

JitPack – це репозиторій пакетів для проектів JVM та Android. Він будує проекти Git на вимогу та надає готові до використання артефакти (jar, aar). При використанні Jitpack немає необхідності проходити стадії збирання та завантаження проекту. Все, що потрібно зробити, це завантажити проект на GitHub, і JitPack подбає про все інше.

Якщо проект вже включений в GitHub, JitPack гарантує можливість складання.

- Додайте репозиторій JitPack у файл збирання:
- Додати залежність
- Публікація Javadoc. Якщо проект створює javadoc.jar, можна переглядати файли javadoc безпосередньо:

https://jitpack.io/com/github/USER/REPO/VERSION/javadoc/

Можливості JitPack:

- Робота з приватними репозиторіями
- Динамічні версії Є можливість використовувати динамічну версію Gradle '1. +' та діапазони версій Maven для релізів. Вони дозволяють випуски, які вже були збудовані.

JitPack періодично перевіряє наявність нових випусків та будує їх з випередженням.

- Побудувати за тегом, commit id або anyBranch–SNAPSHOT.
- Можливість використовувати власне доменне ім'я для groupId Тоді код підключення залежності для різних систем збірок буде

виглядати так:

Система автоматичного збирання gradle

Файл build.gradle:

```
allprojects { repositories {
maven {url'https://jitpack.io'}
}
}
compile'com.github.astel:security:v1.0.0'
```

Система автоматичного збирання maven

Файл pom.xml:

```
<repositories>
<repository>
<id>jitpack.io</id>
<url>https://jitpack.io</url>
</repository>
</repositories>
<dependency>
<groupId>com.github.astel</groupId>
<artifactId>security</artifactId>
<version>v1.0.0</version>
```

Система автоматичного збирання sbt

Файл build.sbt:

```
resolvers+="jitpack"at"https://jitpack.io"

libraryDependencies +="com.github.astel"%"security"%"v1.0.0"
```

Створити публічний клас з анотацією @SpringBootApplication та точкою входу до програми.

Далі наведено повний приклад коду:

```
@SpringBootApplication
public classApplication {
public static voidmain(String[] args) { SpringApplication.run(Application.class, args);
}
}
```

На малюнку 3.1. показано стандартне вікно введення даних у вікні веб–програми

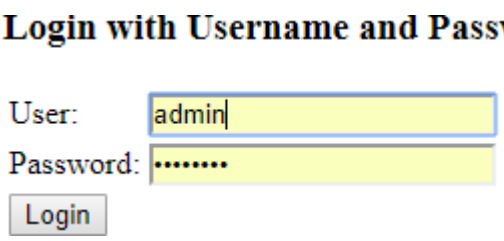


Рисунок 3.1. Вікно аутентифікації.

При введенні неправильних вхідних даних з'являється повідомлення з помилкою про помилкові дані

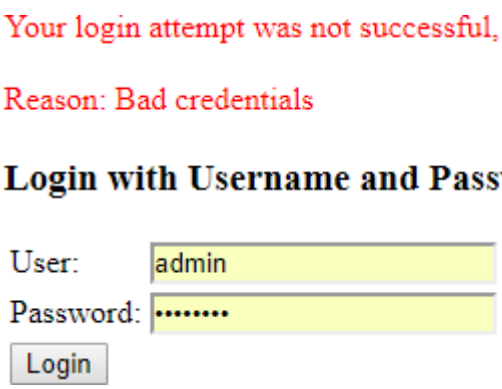


Рисунок 3.2 Помилка. Неправильні вхідні дані.

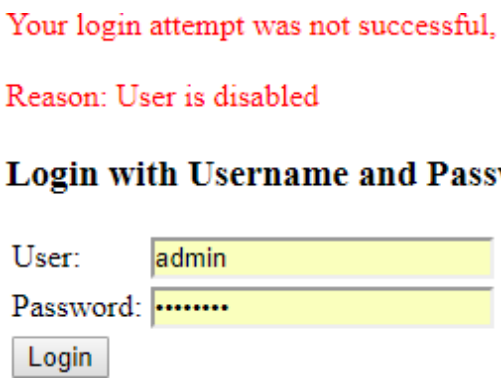


Рисунок 3.3. Помилка. Користувач заблоковано.

3.3 Налаштування контексту безпеки

Фреймворк намагається автоматично налаштувати програму, що розробляється. Наприклад, якщо компоненти з'єднання з базою даних не налаштовані на момент компіляції програми, фреймворк автоматично налаштує базу даних у пам'яті.

Щоб вибрати автоматичне налаштування, потрібно додати анотацію @EnableAutoConfiguration або @SpringBootApplication до одного з класів @Configuration, які використовуються в програмі. Рекомендується додати її до основного класу @Configuration.

Автоконфігурація не постійна, будь–якої миті розробник можете почати визначати свою власну конфігурацію для заміни певних частин автоматичної конфігурації. Наприклад, розробник може додати свій власний компонент DataSource для додавання взаємодії з базою даних у проект. До цієї бази даних також будуть додані таблиці для сутностей: UserEntity, CredentialsEntity, RoleEntity.

Додавання контексту за замовчуванням у проект реалізується механізмів автоконфігурації. Щоб змінити контекст безпеки, достатньо реалізувати свого спадкоємця класу WebSecurityConfigurerAdapter.

Сервіс авторизації

Імплементація сервісу авторизації за умовчанням додається до проекту за допомогою механізму автоконфігурації. Щоб змінити контекст безпеки, достатньо реалізувати свого спадкоємця класу UserDetailsService.

Підключається разом усім фреймворком. Можна підключити окремо, додавши в конфігурацію

```
@Bean
public UserDetailsServiceuserDetailsService() {return new UserDetailsService();
}
```

Функціонал роботи захищеного сервісу показано на малюнку 3.2.

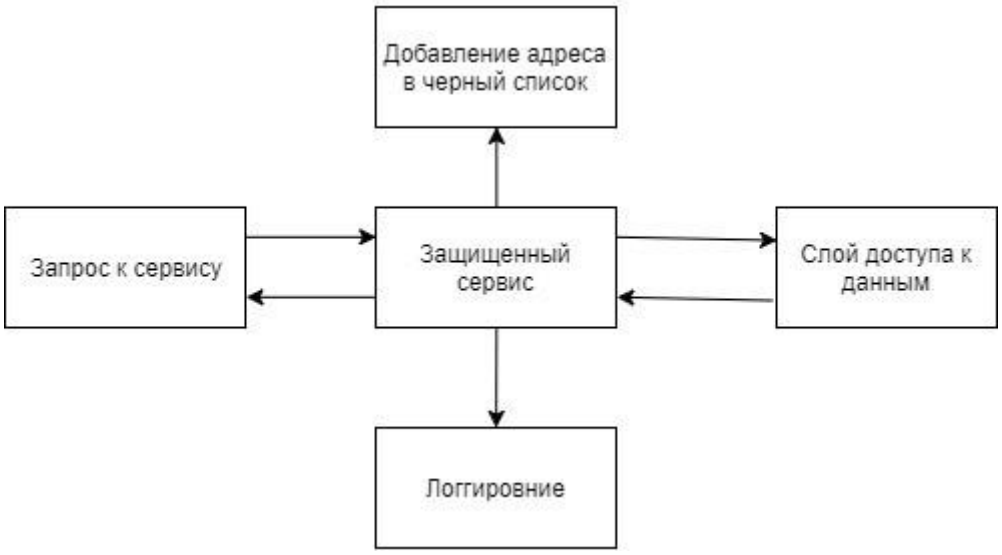


Рисунок 3.4. Робота SecurityService

При порушенні правил, IP–адреса користувача заноситься до чорного списку.

3.4 База даних

За замовчуванням у SecuredService класу цього фреймворку використовується база даних H2 та сутності UserEntity та CredentialsEntity.

H2– Відкрита кросплатформова Система Управління Базами Даних (СУБД), повністю написана мовою Java.

Незважаючи на малий розмір (трохи більше 1 МБ), H2 підтримує наступні можливості «з коробки»:

- Два режими роботи (клієнт–сервер, вбудований);
- Два режими зберігання даних (файлова система, пам'ять);
- Підтримка планів виконання запитів;
- Підтримка кластеризації та реплікації;
- Шифрування даних;
- Зовнішні таблиці;
- Драйвер ODBC;
- Повнотекстовий пошук;
- Визначення доменів;

- Мультиверсійний конкурентний доступ;
- Підтримка послідовностей;
- Підтримка ключових слів LIMIT та OFFSET у запитах;
- Тимчасові таблиці;
- Обчислювані стовпці;
- Користувальницькі агрегатні функції;
- Користувальницькі процедури, що зберігаються;
- Стиснення CLOB/BLOB об'єктів;
- Робота з CSV файлами на читання та запис;
- Браузерна консоль керування;
- Запуск як сервіс Windows;

При реалізації бази даних рекомендується використовувати ці об'єкти для зберігання даних користувача.

Ім'я параметра	Значення
Ім'я класу драйвера	com.mysql.cj.jdbc.Driver
Url бази даних	jdbc:h2:file:~/db
Ім'я користувача	dbuser
Пароль	Випадково виводиться в логCredentials.log під час старту.

Таблиця 3.1. Налаштування бази даних за замовчуванням:

Для вирішення завдань об'єктно–реляційного відображення використовується бібліотека Hibernate. Її налаштування за умовчанням представлені у таблиці 3.2

Ім'я параметра	Значення
Політика оновлення DDL схем	h2–hibernate.hbm2ddl.auto
Діалект SQL	h2–hibernate.dialect
Виведення SQL у лог	false

Таблиця 3.2. Налаштування Hibernate

Валідація даних

Валідація даних відбувається для всіх даних SecuredService. Також рекомендується використовувати анотацію @Tripwired для виявлення потенційно шкідливого введення даних форм або об'єктів транспортного шару (DTO).

Java–анотація – в мові Java спеціальна форма синтаксичних метаданих, яка може бути додана у вихідний код.

Анотації використовуються для аналізу коду, компіляції чи виконання. Анотовані пакети, класи, методи, змінні та параметри.

Виглядає як @ИмяАннотации, що передує визначення змінної, параметра, методу, класу, пакета.

Цю інструкцію можна використовувати для: полів, способів, властивостей.

Логування

За замовчуванням увімкнено для всіх методів SecuredService звикористанням бібліотеки log4j.

Log4j– бібліотека журналування Java програм, частина спільного проекту Apache Logging Project.

Log4j спочатку розвивався в рамках парасолькового Apache Jakarta Project, відповідального за всі Java–проекти Apache, але згодом виділився в окремий, дуже популярний проект журналування.

Налаштування логера можна змінити, додаючи їх у application.properties

Криптографія

У цьому фреймворку для хешування паролів за промовчанням використовується алгоритм bcrypt.

Хешування – це перетворення рядка символів зазвичай більш коротке фіксоване значення або ключ, який представляє вихідний рядок. Хешування використовується для індексації та вилучення елементів у базі даних, тому що швидше знайти елемент, використовуючи коротший хешований ключ, ніж знайти його з використанням вихідного значення. Він також використовується у багатьох алгоритмах шифрування.

Для зміни алгоритму хкшування можна скористатися методом:

public ucserDetailsServiceImpl.setPasswordEncoder(password PasswordEncoder);

Також можна реалізувати свій захищений сервіс з своєю реалізацією криптографії.

4 ТЕХНІКО–ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ВКР

4.1 Концепція

Ця робота присвячена розробці фреймворку безпеки для веб–додатків.

Короткий технічний опис ВКР

Фреймворк призначений для реалізації захисту веб–додатку на перших етапах розробки. Цей фреймворк дозволяє розробнику зосередитися на реалізації функцій, що продають, і змінювати конфігурацію захисту пізніше під унікальні вимоги веб–додатку.

Ринок та план маркетингу

Цей програмний модуль є готовим об'єктом для продажу. Одне з можливих застосувань даного модуля – розробка бізнес–рішень у всіх сферах. Тоді, як цільову аудиторію для продажу даного модуля слід розглядати великі та середні підприємства, які займаються бізнес яких переходить у веб і підприємства, що займаються переведенням бізнес процесів підприємств у веб. Просування даного програмного модуля на ринку передбачається за рахунок статей на ресурсах про веб–розробку, надання зацікавленим у використанні даного модуля, можливість оцінити модуль та прийняти рішення про подальшу купівлю платної підтримки для підприємств та приватних пожертвувань.

Визначення трудомісткості ВКР

В основі визначення вартості ЗКР лежить перелік виконаних робіт та їх трудомісткість. Тривалість кожного етапу визначається з календарного плану виконання ВКР та подана у табл. 4.1.

п/п	Найменування робіт	Трудомісткість (чол./дні)	
		Керівник	Виконавець
	Розробка завдання на дипломну роботу	2	2
	Вивчення літератури	4	22
	Розробка та реалізація програмного рішення	2	40
	Тестування програмного рішення, порівняння з існуючими програмними рішеннями	2	10
	Виконання техніко–економічного обґрунтування випускної кваліфікаційної роботи	–	7
	Оформлення пояснювальної записки та презентації	1	16
	Разом	11	80

Таблиця 4.1 – Трудомісткість виконання робіт

На основі трудомісткості виконання робіт із розробки програмного модуля розраховуються витрати на оплату праці виконавців, які є однією з основних статей калькуляції собівартості розробки.

4.2 Розрахунок собівартості розробки ВКР

Калькуляція собівартості розробки системи здійснюється за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні потреби;
- матеріали;
- витрати на роботи, що виконуються сторонніми організаціями;
- витрати на утримання та експлуатацію обладнання;
- амортизаційні відрахування;
- накладні витрати;
- спецобладнання;

Витрати за статтями «Витрати з робіт, виконуваними сторонніми організаціями», «Витрати утримання і експлуатацію устаткування» і «Спецоборудование» передбачені, т.к. під час виконання цієї ВКР послуги сторонніх організацій не знадобилися, витрати на утримання та експлуатацію обладнання в рамках виконання даної ВКР зневажливо малі, а спеціальне обладнання для наукових та експериментальних робіт не використовувалося.

Калькуляція витрат за статтею «Витрати на оплату праці»

Витрати на додаткову заробітну плату виконавців визначаються за формулою

$$З_{(дод.з/пл)}=З_{(осн.з/пл)}*Н_{доп}/100,$$

де З (доп.з/пл) – Витрати додаткову заробітну плату виконавців в грнлях; НДОП – норматив додаткової заробітної плати у відсотках. За виконання розрахунків у цій ВКР норматив додаткової зарплати приймається рівним 14%.

У цьому дослідженні наводиться значення середньої нарахованої заробітної плати працівників за професійними групами. Останнє дослідження включає результати, отримані за підсумками 2015 року, які будуть використані в даній ВКР. Середня заробітна плата для групи

«Фахівці вищого рівня кваліфікації» за підсумками 2015 року становили 37 023 грн, а для групи «Фахівці середнього рівня кваліфікації» – 29 492 грн.

п/п	Виконавець	Передбачувана посада	Ставка заробітної плати (грн./місяць)
1	Керівник	Менеджер проекту	37 023
2	Виконавець	Інженер–програміст	29 492

Таблиця 4.2 – Ставка заробітної плати виконавців

Денна ставка заробітної плати визначається як відношення ставки заробітної плати виконавця за місяць до середньої кількості робочих днів на місяць. У цьому проекті 21 робочий день прийнято за середню кількість робочих днів на місяць.

Денна ставка становитиме

Ср = (37023) / 21 = 1763 грн. / День для керівника та

Сі = (29492) / 21 = 1404,38 грн. / День

для виконавця. Час, витрачений кожним виконавцем попри всі роботи з ВКР складає

Тр = 11 днів

для керівника та Ті = 80 днів для виконавця.

Витрати на основну та додаткову заробітну плату виконавців складуть

З (осн.з / пл) = Тр * Ср + Сі * Ти = 8 * 1763 + 102 * 1404,38 = 131743,4 грн.

З(доп.з/пл) = 131743,4 * 14/100 = 18444,07 грн.

Витрати по статті «Витрати на оплату праці» складаються з витрат на основну та додаткову заробітну плату та становитимуть

З(осн.з/пл)+ З(дод.з/пл)= 131743,4 + 18444,07 = 150187.47

Калькуляція видатків за статтею «Відрахування на соціальні потреби»

Відрахування на страхові внески, обов'язкове соціальне, пенсійне та медичне страхування з основної та додаткової заробітної плати виконавців визначаються за формулою Зсоц=(З(осн.з/пл)+З(доп.з/пл))*Нсоц/100

де НСОЦ – норматив відрахувань на страхові внески на обов'язкове соціальне, пенсійне та медичне страхування (%). Норматив відрахувань на страхові внески у ВКР приймається рівним 30%. Витрати за статтею «Відрахування на соціальні потреби» становитимуть

З соц = (131743,4 + 18444,07) * 30/100 = 45056.24 грн.

Калькуляція витрат за статтею «Матеріали»

Розрахунок витрат на покупні комплектуючі вироби визначається за формулою

$$ЗМ = \sum_{l} G_l * Ц_l * (1 + Н_{т.зв.} / 100)$$

де ЗМ – витрати на сировину та матеріали (грн.); l – індекс виду сировини чи матеріалу; G l – норма витрати l–того матеріалу на одиницю продукції (в одиницях); Ц l – вартість придбання одиниці l–го матеріалу (грн./шт.); Нт.з – норма транспортно–заготівельних витрат (в даному випадку –10%).

Під час розробки даного проекту потрібно було роздрукувати деякі електронні джерела та підсумковий звіт, а також знадобився флеш–накопичувач для зберігання резервної копії даних. Калькуляція витрат за цією статтею наведена у табл.4.3.

/п	Матеріали	Од. вим.	Кількість	Ціна
	Папір для офісної техніки (Svetocopy, A4, 500 аркушів)	упаковка	1	217
	Картридж для лазерного принтера Xerox 106R02181	шт	1	2 199
	USB–флеш–накопичувач Kingston 16Gb	шт	1	960

Таблиця 4.3 – Калькуляція витрат за статтею "Матеріали"

Витрати за статтею «Матеріали» становитимуть ЗМ=(217*1+2199*1+960*1)*(1+10/100)=3713,6 крб.

Витрати на роботи, що виконуються сторонніми організаціями

Для розробки використовували одну сторонню організацію: інтернет від провайдера «InterZet»

Розмір тарифу інтернету від компанії "ДОМ.UA" З інтер. = 350 грн./місяць

ставка ПДВ – 18%

Для розрахунку витрат на інтернет Зінтер. скористаємося формулою: З интер.= Т интер * З интер. / 1,18

де Тинтер – це кількість місяців використання інтернету. З інтер. = 2,5 * 297 = 742,5 грн.

У результаті, витрати на роботи, виконуваними сторонніми організаціями З стор.орг. є рівними витратами на оплату інтернету

Зстор.орг.= Зінтер.

З стор.. = 742,5 = 742,5грн.

Витрати на утримання, експлуатацію та амортизацію обладнання

При створенні методики використовувалося устаткування як комп'ютера з початковою вартістю 21 999 крб. Для нього необхідно прорахувати витрати на електроенергію та амортизаційні відрахування. Також враховуватиметься освітлення необхідне роботи.

- обсяг тарифу на споживання електроенергія З ел.= 4,32 грн./кВт*ч;
- обсяг споживання комп'ютером за 8 годин роботи М ел. = 1,7 кВт;
- розмір споживання освітленням за 8 годин роботи М ел. = 1,7 кВт;
- час використання основного засобу інженерного Т инж = 2,5
- час використання основного засобу інженерного Т м.п. = 0.5
- річна норма амортизації комп'ютера Н а. = 20%

Для розрахунку витрат на електроенергію на комп'ютери З ел.копм., скористаємося формулою:

Зел.копм.= Тм.п.+ Тм.п.* Мел.* Зел./1,18*8

З ел.копм. = 11 + 80 * 1,76 * 3,66 * 8 = 4529.61 грн.

Для розрахунку витрат на електроенергію освітлення З ел. скористаємося формулою:

Зел.світло.= (Тм.п.+ Тінж) * Мел.* Зел./1,18*8

З ел. = (11 + 80) * 1,76 * 3,66 * 8 = 4529.61 грн.

Загальні витрати на електроенергію З ел. є сумою витрат електроенергії, що пішла на комп'ютери З ел. та освітлення З ел.світл.: З ел. = З ел.копм. + З ел.

З ел. = 4529.61 + 4529.61 = 9059.22 грн.

Калькуляція видатків за статтею «Амортизаційні відрахування»

Амортизаційні відрахування за основним засобом за рік визначаються як

Аі= Цп.н.і* Наі/ 100

де Аі – амортизаційні відрахування за рік за основним засобом (грн.); Ц п.н.і – первісна вартість і–го основного засобу (грн.);

Н аі – Річна норма амортизації і–го основного засобу (%). Розмір амортизаційних відрахувань по і–му основному засобу визначається за формулою: А і ВКР = А і * Т і ВКР / 12,

де Аі ВКР – амортизаційні відрахування за і–м основним засобом, що використовується студентом у роботі над ВКР (грн.); Аі – амортизаційні відрахування протягом року по і–му основному засобу (крб.); Ті ВКР – час, протягом якого студент використовує перший основний засіб (міс.).

Під час розробки проекту використовувався ноутбук Acer Aspire V3–551 вартістю 21999 грн. Відповідно до постанови від 01.01.2002 №1 (ред. від 07.07.2016) «Про класифікацію основних засобів, що включаються до амортизаційних груп» персональні комп'ютери та друкуючі пристрої до них відносяться до другої групи основних засобів, що включаються до амортизаційних груп. Друга група призначена для майна з терміном корисного використання понад 2 до 3 років включно. Так як до обчислювальної техніки, поряд з персональними комп'ютерами, належать і ноутбуки, то в цій роботі прийнято нормативний термін служби ноутбука, що використовується, рівний 3–м років.

У цьому ВКР річна норма амортизації буде розрахована лінійним шляхом, тобто. якщо термін служби ноутбука становить 3 роки, то річна норма амортизації складе

Н аі = 100/3 = 33,33%

Амортизаційні відрахування за цим основним засобом за рік складуть

Ан = 21999 * 33,33/100 = 7332,26 грн.

тоді амортизаційні відрахування за основними засобами за час виконання ВКР виконавцем складуть

ТіВКР = Ті/21 = 80/21 = 3,80 міс.

АіВКР = 7332,26 * 3,80 / 12 = 2321,88 грн.

Калькуляція витрат за статтею "Накладні витрати"

Порядок розрахунку норматив накладних витрат визначається за методикою організації, з урахуванням якої виконується проект. У проекті норматив накладних витрат розраховується як 20% від суми всіх витрат. Витрати за статтею «Накладні витрати» становитимуть

$З_{нр}$=($З_{осн.з/пл}$ +$З_{дод.з/пл}$ + $З_{п}$+ $З_{соц}$+ АіВКР) * 0,2 =

= (131743,4 + 18444,07 + 45056.24 + 3713,6 + 2321,88) * 0,2 =

= 40255,83 грн.

Розрахунок сумарної вартості виконання проекту

Сумарна вартість виконання цього проекту склала З підсумком = 241 535,02 грн.

Детальний кошторис витрат за статтями калькуляції подано у таблиці 4.4.

№	Найменування статті витрат	Сума, грн
1	Витрати на оплату праці	150187,47
2	Відрахування на соціальні потреби	45056,24
3	Матеріали	3 713,6
4	Витрати на роботи, що виконуються сторонніми організаціями	742,5
5	Витрати на утримання та експлуатацію обладнання	9059,22
6	Амортизаційні відрахування	2321,88

Таблиця 4.4 – Кошторис витрат за проект

7	Накладні витрати	40255,83
8	Спецобладнання	0
РАЗОМ		251336,74

Таблиця 4.4 – Кошторис витрат за проект

Економічний ефект від використання ВКР

Економічний ефект від використання цього фреймворку оцінено на основі вартості аудиту захисту програмного забезпечення

Компанія	Вартість використання аудиту
PENTESTIT	від 30 000 грн
REG.UA	15 000 грн

Таблиця 4.5 – Вартість аналогічного ПЗ

Необхідно оцінити вартість використання програмного модуля одним користувачем протягом року. Вартість ВКР становила 241 535,02 грн. У роботі розмір прибутку розраховується як 30% від собівартості і становитиме

П = 251336,74 * 30/100 = 75401,02 грн.

Підсумкова ціна ВКР складе

Ц вкр = З результат + П = 251336,74 + 75401,02 = 326737,76 грн.

Підсумкова вартість ВКР з урахуванням ПДВ, рівним 18% (αндс = 0,18) складе Ц вкр.ндс = Ц вкр * (1 + αндс) = 326737,76 * (1 + 0,18) = 385550,55 грн.

Фреймворк, розроблений у рамках цього проекту, не накладає жодних технічних обмежень на кількість можливих користувачів. Вартість супроводу на місяць даного програмного модуля одним користувачем буде визначена як середнє значення мінімально можливих цін аудиту безпеки та становитиме

Ц місяць = (30 000 + 15 000) / 2 = 22 500 грн.

Кількість місяців підтримки даного програмного модуля, яке необхідно продати за даної ціни за місяць для досягнення бажаного прибутку та окупності вкладених інвестицій, складе

N = Ц вкр.ндс / Ц місяць = 385550,55 / 22500 = 18 міс. Ця кількість може бути продана за 2 роки.

4.3 Висновки

Трудомісткість розробки цієї ВКР становить 91 людино–днів, а собівартість 251336,74 грн. Необхідний термін для окупності вкладених інвестицій та отримання бажаного прибутку становить 2 роки. Розроблене рішення передбачає послуги з впровадження та підтримки, що може дати стабільний дохід. Все перераховане вище робить розробку економічно доцільною.

ВИСНОВОК

Критерії розробки фреймворку були встановлені шляхом дослідження існуючих рішень і процесу розробки веб-додатків. Було визначено, що існує потреба у комплексній структурі, яка мінімізує ризики під час запуску веб-додатку. Крім того, методологія розробки Agile вимагає, щоб програма розроблялася на основі базових стандартних параметрів.

В рамках цього проекту було створено структуру для розробки безпечних веб-додатків із захистом бази даних, журналюванням, хешуванням паролів і перевіркою даних запитів. Ця структура призначена для забезпечення безпеки веб-додатків із самого початку з мінімальною функціональністю, з розумінням того, що конфігурація безпеки буде розвиватися протягом розробки. Фреймворк продовжить підтримувати існуючі вимоги, коли буде реалізовано бізнес-функціональність.

Майбутні перспективи проекту включають інтеграцію з автоматичними системами сканування безпеки та запобігання поширеним проблемам, пов’язаним із проектуванням служб REST. Сканер уразливостей Wapiti використовувався для тестування програми, розробленої з використанням цього фреймворку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Software Framework [Електронний ресурс] – режим доступу до ресурсу: https://en.wikipedia.org/wiki/Software_framework
2. Positive Technologies – Вразливості веб–застосунків 2016 [Електронний ресурс] – режим доступу до ресурсу: https://www.ptsecurity.com/upload/corporate/ua–ua/analytics/Web– Vulnerability– 2016–ukr.pdf
3. Spring Boot Docs [Електронний ресурс] – режим доступу до ресурсу: https://docs.spring.io/spring–boot/docs/current/reference/html/boot–features–developing–auto–configuration.html
4. Inversion of Control Containers and Dependency Injection pattern [Електронний ресурс] – режим доступу до ресурсу: https://martinfowler.com/articles/injection.html
5. Spring Security Reference [Електронний ресурс] – режим доступу до ресурсу: https://docs.spring.io/spring–security/site/docs/5.0.0.RELEASE/reference/htmlsingle/
6. Agile. Secure software development [Електронний ресурс] – режим доступу до ресурсу: https://www.checkmarx.com/2023/04/20/six–steps–secure–software–development–agile–era/
7. Управління вимогами безпеки в Agile проектах – Rohit Sethi [Електронний ресурс] – режим доступу до ресурсу: http://agilerussia.ua/practices/security–requirements/
8. Журнал школи IT–менеджменту [Електронний ресурс] – режим доступу до ресурсу: http://journal.itmane.ua/node/1572/
9. OWASP – Top Ten Proactive Controls [Електронний ресурс] – режим доступу до ресурсу: http://master.cmc.msu.ua/files/OWASP_Top_Ten_Proactive_Controls_ v2_ukr.pdf
10. Technical University of Crete. Password–Hashing. : Chania, 27 June 2023

11. Wikipedia – Blowfish [Електронний ресурс] – режим доступу до ресурсу: https://ua.wikipedia.org/wiki/Blowfish

11. Habrahabr –SQL vs ORM[Електронний ресурс] – режим доступу до ресурсу: https://habrahabr.ua/company/pgdayrussia/blog/328690/

12. Мартін Фаулер. OrmHate [Електронний ресурс] – режим доступу до ресурсу: https://martinfowler.com/bliki/OrmHate.html

13. Тест та порівняння ефективності WAF [Електронний ресурс] – режим доступу до ресурсу: https://www.anti–malware.ua/compare/web–application–firewall

14. OWASP – Fuzzing [Електронний ресурс] – режим доступу до ресурсу: https://www.owasp.org/index.php/Fuzzing

15. Fuzzing: the Past,the Present and the Future [Електронний ресурс] – режим доступу до ресурсу: http://actes.sstic.org/SSTIC09/Fuzzing–the_Past–the_Present_and_the_Future/SSTIC09–article–A–Takanen–Fuzzing– the_Past–the_Present_and_the_Future.pdf

16. Крейг Воллс. Spring in Action. ДМК Прес. 2015 17.OWASP Guide Project [Електронний ресурс] – режим доступу до ресурсу: https://www.owasp.org/index.php/OWASP_Guide_Project

18. Алексєєва О.Г. Методичні вказівки щодо економічного обґрунтування випускних кваліфікаційних робіт бакалаврів: Метод. вказівки, СПб.: Вид–во СПбГЕТУ "ЛЕТИ", 2013.

19. OWASP – Logging Cheat Sheet [Електронний ресурс] – режим доступу до ресурсу: https:///www.owasp.org/index.php/Logging_Cheat_Sheet

20. Business logic in a web application[Електронний ресурс] – режим доступу до ресурсу: https://www.quora.com/Where–should–we–add–business–logic–in–a– web–application–on–client–side–or–server–side