Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)

(назва факультету)

(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: **Побудова середовища розробки веб-додатків з оптимальним рівнем безпеки**

Виконав: студент (ка) ____ курсу, групи ____

спеціальності (напряму підготовки) 125-кібербезпека
(шифр і назва спеціальності (напряму підготовки))

(підпис) Лавринець О.О. (прізвище та ініціали)

Керівник (підпис) Грод І. М. (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Пояснювальна записка на 87 сторінках, вона містить 7 розділів, 16 ілюстрацій, 5 таблиць, 36 джерел в переліку посилань.

Ключові слова: FRAMEWORK, AGILE, SPRING, ВЕБ-ДОДАТОК, ВРАЗЛИВОСТІ ВЕБ-ДОДАТКІВ, БЕЗПЕКА ВЕБ-ДОДАТКІВ.

Об'єкт дослідження: фреймворки для розробки веб-додатків

Мета роботи (проекту): Створити максимально повний фреймворк для старту веб-додатку, використовуючи Spring Security для управління сесіями, механізмами аутентифікації і авторизації. Це дозволить реалізувати валідацію вхідних даних, механізм зберігання даних користувачів і знизити трудомісткість використання. Далі веб-додаток буде розвиватися з уже обмеженого стандартними налаштуваннями стану відповідно до гнучкої моделі розробки додатків.

У першому розділі розглянуто наявні на даний момент часу фреймворки, методології та вимоги до безпеки веб-додатків, виявлені вимоги до подальшої розробки.

Другий розділ спрямовано на обґрунтуванню вибору технологій.

Третій розділ присвячений збиранню та використанню побудованого середовища.

Четвертий розділ присвячений проблемі оптимізації тестування веб-додатків.

П’ятий розділ містить розрахунки техніко-економічного обґрунтування використання побудованого середовища.

ABSTRACT

The thesis consists of 87 pages, has 7 sections, 16 form, 5 tables, 36 points in the link list.

Keywords: FRAME, AGIL, SPRING, WEB ADDITION, WEB ADDITIONAL VULNERABILITIES, WEB ADDITIONAL SECURITY.

Object of study: frameworks for web application development

Purpose (project): To create the most complete framework for launching a web application, using Spring Security to manage sessions, authentication and authorization mechanisms. This will allow the validation of input data, the mechanism of storage of user data and reduce the complexity of use. In the future, the web application will evolve from the already limited standard status settings according to the flexible application development model.

The first section looks at currently available frameworks, methodologies, and requirements for web application security, and identifies requirements for further development.

The second section focuses on the justification for technology choices.

The third section is about building and using the built environment.

The fourth section deals with the problem of optimizing web application testing.

The fifth section contains the feasibility studies on the use of the built environment.

Table of Contents

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

У цій пояснювальній записці застосовують такі терміни з відповідними визначеннями:

CSRF - Cross Site Request Forgery - міжсайтовий підробка запиту

XSS - Міжсайтовий скриптинг

Framework - програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. [1]

MVC - Модель-Представлення-Контролер

OWASP - Open Web Application Security Project

REST - Representational State Transfer

ВСТУП

Досягнення в області веб-технологій в поєднанні зі змінною бізнес-середовищем означають, що веб-додатки стають все більш поширеними в корпоративній, державної та урядової сферах послуг.

Хоча веб-додатки можуть забезпечити зручність і ефективність в сфері послуг, також вони представляють ряд нових загроз безпеці, які потенційно можуть становити значні ризики для інфраструктури інформаційних технологій організації, якщо її не обробляти належним чином.

Більш десятиліття організації залежать від заходів безпеки для захисту своєї IT-інфраструктури. Однак традиційні мережеві заходи безпеки і технології можуть бути недостатніми для захисту мережі від нових загроз, оскільки атаки тепер спеціально націлені на недоліки безпеки в дизайні веб-додатків. Нові заходи безпеки, як технічні, так і адміністративні, повинні бути реалізовані поряд з розробкою веб-додатків.

Щоб усунути загрози, пов'язані з цими новими службами додатків, важливо розуміти уразливості, що зазвичай зустрічаються в веб-додатках і процес розробки веб-додатків.

При використанні водоспадної моделі розробки вимоги безпеки вбудовуються на етапі розробки документації. У цього підходу є мінуси як стандартні для цієї моделі розробки так і особливі для вимог безпеки. Наприклад, виявлення нової уразливості в використовуваної технології на останніх стадіях розробки може коштувати досить дорого.

1 АНАЛІЗ ПРИДМЕТНОЇ ОБЛАСТІ

В даний час на ринку переважають гнучкі методології розробки - Agile. І рівень виконання вимог безпеки вкрай залежний від декількох факторів:

- досвідченість команди. Досвід в розробці захищених систем.
- наявність Project Backlog.
- присутність в проекті фахівця з комп'ютерної безпеки.
- рівень зрілості компанії. Наявність корпоративних правил забезпечення безпеки в проектах.
- досвід підтримки веб-додатків

Інша проблема полягає в малій цінності вимог безпеки для замовника в порівнянні з продає функціональністю. Таким чином вимоги безпеки можуть бути відсунуті реалізацією продає функціональності. Це є однією з основних причин низького ступеня захищеності більшості сучасних 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 призначеного для користувача коду, таким чином надаючи програмне забезпечення для конкретних додатків. Програмне середовище забезпечує стандартний спосіб створення і розгортання додатків. Програмне середовище - універсальна, багаторазова програмне середовище, що забезпечує певну функціональність як частина більш великої програмної платформи для полегшення розробки програмних додатків, продуктів і рішень. Така програмне середовище може включати в себе програми підтримки, компілятори, бібліотеки коду, набори інструментів і інтерфейси

прикладного програмування (API), які об'єднують всі різні компоненти, щоб забезпечити розробку проекту або системи[2].

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

● інверсія управління: в фреймворками, на відміну від бібліотек або стандартних користувальницьких додатків, управління потоком всієї програми не продиктовано викликає, а структурою [4].

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

● незмінний код структури: фреймворк, в загальному, не повинен змінюватися, беручи при цьому призначені для користувача розширення. Іншими словами, користувачі можуть розширювати структуру, але не повинні змінювати свій код.

Розробники фреймворків прагнуть полегшити розробку програмного забезпечення, дозволяючи розробникам і програмістам приділяти час виконання вимог до програмного забезпечення, а не працювати з 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 і управління сесіями.

Фреймворки часто значно збільшують обсяг програмних продуктів викликаючи явище, зване «роздування коду». Через потреб додатків,

орієнтованих на потреби клієнтів, як паралельні, так і додаткові структури іноді виявляються в продукті. Крім того, через складність їх API, передбачуване скорочення загального часу розробки може бути не досягнуто через необхідність витрачати додатковий час на вивчення структури; ця критика явно дійсна, коли особливий або новий фреймворк вперше зустрічається співробітникам відділу розробки. Якщо такий фреймворк не використовується в наступних проектах, час, що витрачається на вивчення структури, може коштувати дорожче, ніж написаний без використання фреймворка код; багато програмісти зберігають копії корисного шаблону для загальних потреб.

Однак, як тільки фреймворк буде вивчений, майбутні проекти можуть бути завершені з використанням менших ресурсів; концепція фреймворка полягає в тому, щоб скласти набір стандартних рішень, що підходить до більшості типових завдань, щоб при використанні фреймворка рівень продуктивності розробника збільшувався. Це твердження не відноситься до кількості коду, в кінцевому підсумку пов'язаного з вихідним продуктом, а також його відносної ефективності і стислості.

Використання будь-якого рішення у вигляді бібліотеки обов'язково включає в себе додаткові функції і невикористовувані сторонні компоненти, якщо програмне забезпечення не є компоновщиком-компілятором, що створює жорсткий (невеликий, повністю контрольований і вказаний) виконуваний модуль.

Ця тенденція в суперечці піднімає важливе питання про структуру програмного забезпечення. Використання каркаса, який є елегантним, в порівнянні з кодом, який просто вирішує проблему, як і раніше є мистецтвом, а не наукою. А також відповідає принципу DRY1 «Програмна елегантність»

має на увазі ясність, стислість і невеликі відходи (додаткові або сторонні функції, велика частина 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 який тому легко модифікується), в порівнянні з тим, який просто генерує правильний код. Через елегантності, проблема полягає в тому, що відносно невелика кількість програмних фреймворків витримало випробування часом: кращі фреймворки змогли витончено розвиватися, оскільки основна технологія, на якій вони були побудовані, була вдосконалена. Навіть там, розвинувшись, багато такі пакети збережуть успадковані можливості, що роздувають остаточне програмне забезпечення, оскільки в іншому випадку замінні методи були збережені паралельно з більш новими методами.

Далі представлені основні існуючі фреймворки безпеки для веб додатків.

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

Переваги: 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 настройки, який широко використовується для вирішення питань аутентифікації і авторизації в веб-додатках, побудованих на Java. [5]

Переваги: реалізація веб-запитів і викликів методів з використанням технології аспектно-орієнтованого програмування.

Недоліки: сильна залежність від великовагового контексту Spring

Apache Shiro вважається високоефективним фреймворком безпеки для Java, який виконує криптографію, авторизацію і управління сеансом на всіх типах додатків Java незалежно від їх розміру. Був розроблений як інтуїтивно зрозумілий і простий у використанні, але при цьому забезпечує надійні функції безпеки.

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

Висновки

Дані рішення надають інструменти для забезпечення безпеки на всіх рівнях. Слабким місцем даних рішень є валідація даних. Також, вони не пропонують реалізації брандмауера і механізму репутації для запитів до ресурсів. Дані фреймворки не пропонують можливість швидкого старту для додатка тому що зачіпають тільки аспекти безпеки.

1.2. Методології.

Замкова модель. Ця модель має на увазі, що всі ризики повинні бути визначені, можливий збиток - оцінений, а на кожне несподівана подія має бути заготовлено план. Відповідно до погрозами і ризиками повинні бути сформовані організаційні і технічні заходи щодо захисту компанії від загроз

ІБ. Цей підхід в деяких джерелах іменується «Castle model of cyber security» або замкова модель.

Його основні принципи - поділ всього інформаційного простору на внутрішнє, «безпечне», і зовнішнє, повне загроз і зловмисників. Ці два сегменти розділені «стінами», можливо, збудованими пошарово, а зв'язок між зовнішнім і внутрішнім світом забезпечують шлюзи - «gateways».

Agile Cybersecurity Action Plan - гнучкий спосіб управління кібербезпекою. [6] Він протиставляється «замкової моделі» побудови кібербезпеки і покликаний допомогти CISO максимально ефективно «відпрацьовувати» зміни ландшафту загроз і ризиків. Процес має на увазі активну спільну роботу крос-функціональних і крос-організаційних команд для:

- створення та оновлення профілів ризиків;
- оцінки ефективності роботи ІБ-інфраструктури компанії і відповідності її існуючим профілів ризиків;
- виявлення потенційних проблем і недоліків до того, як вони перетворяться в інциденти;
- створення плану усунення виявлених проблем і недоліків; Agile. Призначені для користувача історії. Концепція пропонує різні методи для визначення функціональних вимог в рухомих призначеними для користувача історіями процесах розробки. Нефункціональні вимоги діляться на два великих типи:
- Нефункціональні призначені для користувача історії: Блоки тестируемой функціональності, написані в форматі користувальницьких історій. Акторами в цих історіях може бути внутрішній IT персонал.

Наприклад: "Як аналітик безпеки, я хочу щоб система припиняла невдалі спроби аутентифікації, так щоб додаток не було схильне грубим атакам";

● Обмеження: Це загальні питання, які можуть відображатися на декількох призначених для користувача історіях. Це свого роду "податок" на відповідні роботи. Наприклад, вимога щоб всі розробники валідували дані з полів HTTP форм в web додатках - це обмеження;

Визначивши дані нефункціональні вимоги існує можливість зробити безпеку видимої і виправдати час, виділений на ті поліпшення, які зазвичай не покращують взаємодію користувача з системою. [7]

Впровадження процесів інформаційної безпеки в Agile

Сьогодні в умовах жорсткої конкуренції на перший план виходить критерій часу виведення програмного забезпечення на ринок. Зараз передові IT компанії використовують підходи безперервного постачання змін. Протягом одного дня може проводиться кілька оновлень продукту. У той час як час поставки першої версії продукту зменшується при використанні agile, то ефективність команд розробки - яку цінність приносить команда в одиницю часу, падає. Якщо в Водоспадної моделі, де використовується детальна документація, команда розробників зайнята написанням коду більшу частину робочого часу, то при використанні agile необхідно чимало часу для спілкування з людьми. Відповідальність за прийняття рішень також зсувається в бік команди розробників і вся команда в повному складі відповідає за результати.

Безпека - це відповідальність тестирующей команди і розглядається окремо від дій і планування, які входять в процес розробки програмного

забезпечення. Природно, функції безпеки не органічно вписуються ні в одну з програмних розробок.

Можна сформулювати основні принципи, які ляжуть в основу процесу розробки:

Безпека - частина процесу розробки

У більшості випадків безпека є частина програмного забезпечення. Безперервна інтеграція и Безперервна розгортання (CI / CD) НЕ повинні Припиняти для тестування безпеки перед наступний ітерацією розробки.

Фахівці з безпеки - так само як и фахівці будь-який інший дисциплін, Такі як продуктивність або досвід взаємодії (UX), що працюють над досягнені цілей в Галузі безпеки. Смороду є експертами в створенні засобів безпеки без проблем в інструментальній ланцюжки розробки та середовіщі CI / CD. [8]

Безпека в процесі розробки

Завдяки постійним доповненням безпеку будь-якого програмного забезпечення розвивається з плином часу. Замість того, щоб концентруватися на комерційній цінності програмного забезпечення та досягненні його місії, команда розробників в кінцевому підсумку була змушена додаємо функції безпеки, який вимагає уваги для фахівців з безпеки, в чиї обов'язки входить обробка аспектів безпеки.

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

рання імплементація

Функції безпеки - це аспекти програмного забезпечення, які забезпечують внутрішню роботу додатка. Вони включають в себе використання шифрування, зберігання паролів, вибір способу управління збоями і т. Д. Коли потреби в безпеці виявлені, команда розробників повинна приділити пріоритетну увагу зміни коду таким чином, щоб зробити цю функцію безпечної, а не надати користувачеві додатковий механізм забезпечення безпеки .

Невеликі ітеративні зміни змінюють безпеку нашого програмного забезпечення з плином часу. Важливо зосередитися на цінності розроблюваного програмного забезпечення для кінцевого користувача в цілому, а не на додаванні або впровадженні функцій безпеки. Наскільки це можливо, необхідного використовувати фреймворки, безпечні за замовчуванням. Проектування і впровадження таких функцій безпеки, як аутентифікація, схеми зберігання паролів, криптографічні алгоритми краще надати досвідченим професіоналам, чия повсякденна діяльність пов'язана з цими специфічними темами. Переважно використовувати перевірені і надійні реалізації, замість того, щоб займатися їх розробкою самостійно.

Робота з ризиками, замість помилок

Більшість заходів з розробки програмного забезпечення включають в себе пошук вразливостей безпеки, їх виправлення і визначення програмного забезпечення як «безпечного». При створенні програмного забезпечення фахівцям і розробникам служб безпеки необхідно враховувати, як мінімізувати ризики для бізнесу, користувачів або даних. Дуже рідко забезпечення безпеки настільки ж просто, як виправлення помилки. Але часто це складніше, оскільки фахівцям з програмного забезпечення і безпеки

доводиться розробляти способи боротьби з огляду на управлінням ризиками і пріоритетами.

Цілісний погляд на речі, які можуть піти не так, - кращий спосіб впоратися з управлінням ризиками, будь то управління ризиками шляхом написання коду; моніторинг і реагування на підозрілу активність, використовуючи юридичний і контрактний контроль замість технічного контролю, замість того, щоб складати нескінченний список окремих помилок, які необхідно придушити.

Основні проблеми

Цінності Agile неспроможні для того, щоб тотально змінити підхід до управління кібербезпекою. З чотирьох основних елементів життєвого циклу - Передбачати, Запобігати, Виявляти, Відповідати - Agile може працювати тільки на стадіях Передбачати і Запобігати, куди потрапляють завдання, пов'язані зі стратегічним плануванням, оцінкою ризиків і виробленням способу їх вирішення.

Іншою важливою проблемою розробки бізнес рішень в середовищі середовищі Інтернет є істотний вплив досвіду команди розробки і фахівців з комп'ютерної безпеки на якість програмного забезпечення. Велика частина вразливостей веб додатків легко виправляється за допомогою існуючих рішень, але імплементація цих інструментів в кожному новому проекті вимагає витрат за часом і досвіду команди.

1.3. Вимоги до безпеки веб-додатки

Дослідження вразливостей веб-додатків, що проводяться компанією Positive Technologies в 2016 році показало, що загальний рівень захищеності веб-додатки продовжує знижуватися. Розробники прагнуть забезпечити

максимальну функціональність систем і не завжди приділяють належну увагу безпеці коду. При цьому, наголошується широку поширеність недоліків, пов'язаних з помилками адміністрування. Їх небезпека, як правило, невисока, проте в разі експлуатації деяких таких вразливостей порушник може не тільки отримати чутливу інформацію (наприклад, в результаті її розголошення на сторінках додатків), але і отримати несанкціонований доступ до системи (у випадках підбору і перехоплення облікових даних або успішної атаки на сесію).

Найбільш поширеними уразливими веб додатки є наступні уразливості: Впровадження коду

Якщо додаток може отримувати призначений для користувача введення, який входить в базову базу даних, команду або виклик, додаток може бути вразливе для атаки на основі ін'єкції коду. Ін'єкційні недоліки є набором вразливостей безпеки, які виникають, коли підозрілі дані вставляються в додаток у вигляді команди або запиту. Відомі ін'єкційні атаки включають SQL, OS, XXE і 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 в Інтернет, може це зробити - SQLi-скрипти доступні для скачування і можуть бути легко придбані.

Некоректна аутентифікація і управління сесією

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

Міжсайтовий скриптинг

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

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

Додатки, які дозволяють вводити користувача без повного контролю над виходом, можуть сильно постраждати від атак XSS. Коли XSS-атака успішна, зловмисники можуть завдати серйозної шкоди веб-додатком.

Недостатнє журнал і моніторинг

Недостатня реєстрація та моніторинг в поєднанні з відсутньою або неефективною інтеграцією з реагуванням на інциденти дозволяють зловмисникам продовжувати атакувати системи, підтримувати сталість, повертатися до більшої кількості систем і підробляти, витягувати або

знищувати дані. Більшість досліджень з порушенням показують, що час виявлення порушення становить понад 200 днів, зазвичай виявляються зовнішніми сторонами, а не внутрішніми процесами або моніторингом. міжсайтовий запити

Атака CSRF змушує браузер з жертви відправляти відправлений HTTP-запит, в тому числі куки-файл сеансу жертви і будь-яку іншу автоматично включається інформацію аутентифікації в уразливе веб-додаток. Це дозволяє зловмисникові змусити браузер жертви генерувати запити, які вважаються валідними запитами жертви вразливим додатком.

1.4. Висновок

Хоча існуючі фреймворки задовольняють всім перерахованим вимогам безпеки, частота появи даних вразливостей не змінюється протягом багатьох років.

Як видно з рисунка 1.1, Ін'єкції, міжсайтовий скриптинг, ризики пов'язані з авторизацією і 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.]

Рисунок 1.1 OWASP. 10 найбільш поширених вразливостей. 2013 - 2017 роки.

Дане спостереження дозволяє зробити висновок про неефективність існуючих інструментах розробки.

Дослідження методологій розробки показує протиріччя між популярним сімейством методологій розробки Agile і вимогами безпеки. Цей конфлікт призводить до ігнорування вимог безпеки на користь функціональним вимогам і прискоренню виходу продукту на ринок.

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

Дані проблеми дозволяють сформулювати наступні критерії необхідного рішення:

- низька залежність від досвіду команди
- низька трудомісткість
- закриття найбільш розповсюджених вразливостей

Виходячи з цих критеріїв пропонується наступне рішення:

Створити максимально повний фреймворк для старту веб-додатку, використовуючи Spring Security для управління сесіями, механізмами аутентифікації і авторизації. Це дозволить реалізувати валідацію вхідних даних, механізм зберігання даних користувачів і знизити трудомісткість використання. Далі веб-додаток буде розвиватися з уже обмеженого стандартними настройками стану відповідно до Agile.

Це рішення дозволяє обмежити вибір розробника з варіантів: "знизити ризики і вивести продукт пізніше" і "не зменшувати ризики і вивести продукт раніше", прийнятних з точки зору ринку. До одного: "підключити фреймворк і реалізувати функціонал".

Дане рішення дозволяє розробнику з малим досвідом розробки веб-додатків, досвіду використання інструментів для розробки і стислими термінами виконання завдання створити продукт без найпоширеніших загроз.

Інша позитивна сторона даного рішення - можливість зміни параметрів безпеки відповідно до бізнес логікою поточного продукту. Таким чином, при наступних ітераціях, розробник може з легкістю змінити всі налаштування безпеки в своєму додатку.

2. РЕАЛІЗАЦІЯ ФУНКЦІЙ

2.1. Обґрунтування вибору технологій

На даний момент, Java - найбільш популярний інструмент для веб-розробки.

Мова Java міцно зміцнив свої позиції на першому місці рейтингу мов програмування TIOBE. За 2019 рік у загальному рейтингу він зміцнив свою позицію та займає більше 17% всього ринку. Мова має Поріг входження в середу розробників на цій мові поступово знижується.

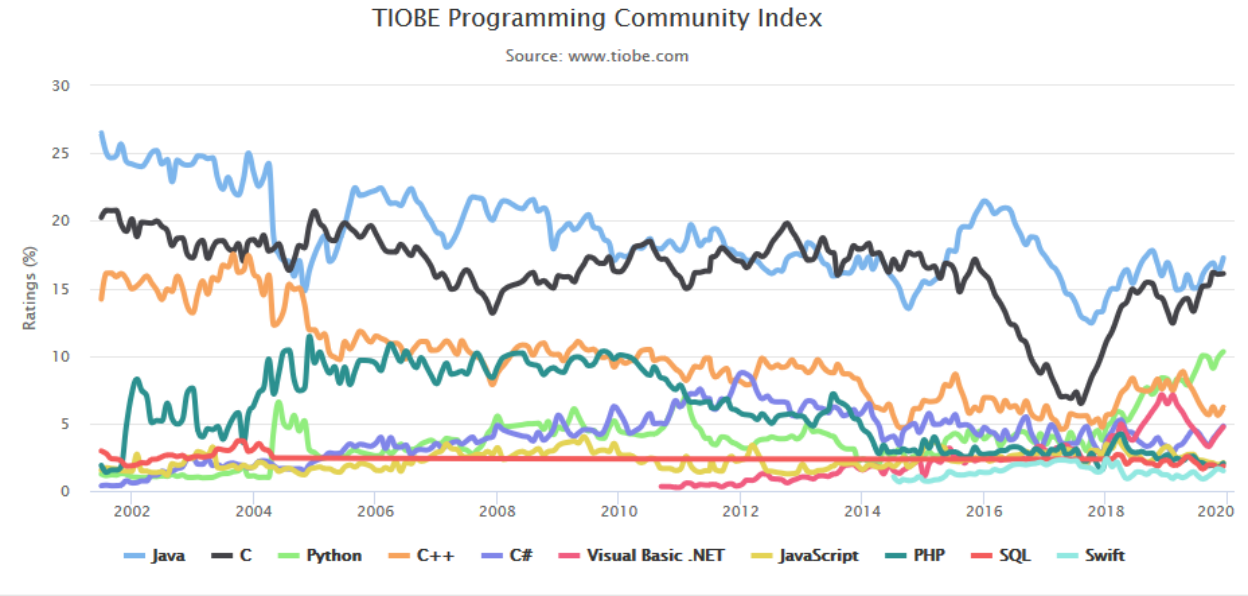


Рисунок 2.1. Діаграма використання мов розробки

В якості системи контролю версій був обраний git.

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

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

2.2. Хешування

Як хеш-функцію було обрано алгоритм bCrypt, заснований на алгоритмі Blowfish. Через присутність стандартної реалізації, популярність та генерації випадкової солі.

Bcrypt, був розроблений на відміну від звичайних методів хешування з метою зробити хешування максимально складним. Для звичайних цілей застосування це зусилля не суттєве порівняно з іншими факторами, лише якщо розрахунок слід проводити часто поспіль (наприклад, у випадку атаки перебором (анг. brute force)), відбувається значне уповільнення.

Основні переваги алгоритму:

- Bcrypt має регульований коефіцієнт витрат, щоб протистояти атакам перебором. З цим фактором зусилля можна також збільшити, якщо продуктивність комп’ютерів продовжить розвиватися в майбутньому.
- Переваги продуктивності для різних мов програмування є незначними.
- Помірна потреба в пам'яті обмежує перевагу апаратних оптимізацій. Bcrypt вимагає 4 КБ оперативної пам’яті.

Реалізація програмного забезпечення базується на операціях, оптимізованих для процесорів, таких як ексклюзивні операції, або додавання, або зсув.

2.3. Аутентифікація

Авторизація базується на принципах середовища Spring Security з додатковим регулюванням бази даних шляхом контролю запитів в чорний список.

Spring Security досить простий. Ви визначаєте деякі ролі і дозволу, які ви хочете у вашому додатку. Потім ви можете визначити статичних користувачів або завантажити їх із зовнішнього джерела. Потім ви в основному анотуєте точки входу в свій додаток і вказуєте правила, що визначають, чи є у кого-то доступ. Наприклад: X має дозвіл Y або X має роль Z, також є можливість комбінувати вирази з і / або і отримувати дійсно складні дозволи. Можна визначити свої власні правила, якщо вам потрібно щось більш складне або ви хочете, щоб умови ґрунтувалися на інших речах, а не на ролях або дозволах - наприклад, вам може знадобитися надати доступ тільки в певний час або тільки якщо певний запис даних існує в базі даних.

Інфраструктура авторизації була побудована на платформі Spring Security, яка включає додаток AuthenticationProvider під назвою DaoAuthenticationProvider. Цей постачальник аутентифікації сумісний з усіма механізмами аутентифікації, які генерують пару значень: ім'я користувача і пароль. Як і більшість інших постачальників аутентифікації, DaoAuthenticationProvider використовує UserDetailsService для пошуку імені користувача та пароля.

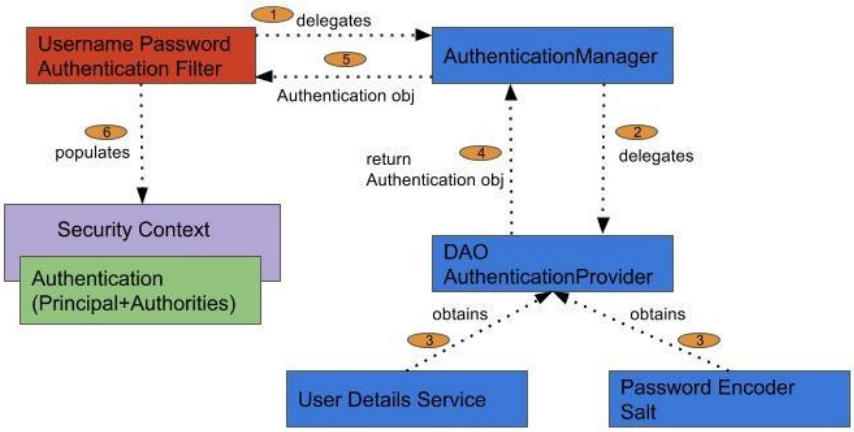


Рисунок 2.2 Механізм аутентифікації

Та на відміну від інших постачальників аутентифікації, які використовують UserDetailsService, цей постачальник аутентифікації насправді вимагає наявності пароля в запиті аутентифікації. PasswordEncoder і SaltSource є необов'язковими. PasswordEncoder забезпечує кодування і декодування паролів, представлених в об'єкті UserDetails, який повертається з налаштованого UserDetailsService. SaltSource дозволяє заповнювати паролі сіллю, що підвищує безпеку паролів в репозиторії аутентифікації. Реалізації PasswordEncoder забезпечені Spring Security, що охоплюють кодування MD5, SHA і cleartext. Також реалізовані дві реалізації SaltSource: SystemWideSaltSource, який кодує всі паролі з тієї ж сіллю і ReflectionSaltSource, який перевіряє заданий властивість об'єкта, що повертається UserDetails для отримання солі.

За замовчуванням DaoAuthenticationProvider використовує NullUserCache, який не виконує кешування та він підтримує додаткове кешування об'єктів UserDetails. Інтерфейс UserCache дозволяє DaoAuthenticationProvider поміщати об'єкт UserDetails в кеш і витягувати його з кеша при наступних спробах аутентифікації для одного і того ж імені користувача.

У більшості випадків, коли веб-додаток має збереження стану, немає необхідності у використанні кеша, оскільки інформація про аутентифікації користувача буде зберігатися в поточній сесії клієнта.

Було прийнято конструктивне рішення не підтримувати блокування облікового запису в DaoAuthenticationProvider, так як це збільшило б складність інтерфейсу UserDetailsService. Наприклад, для збільшення кількості невдалих спроб аутентифікації потрібно метод. Така

функціональність може бути легко забезпечена за рахунок використання функцій публікації подій додатки, обговорюваних нижче.

DaoAuthenticationProvider повертає об'єкт аутентифікації, який, в 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, так як це дозволяє додаткам додаткові властивості, потенційно використовувані в додатках, такі як повне ім'я користувача, адресу електронної пошти і т. д.

2.4. База даних

Робота з об'єктно-орієнтованим програмним забезпеченням і реляційними базами даних може бути громіздкою і трудомісткою. Витрати на розробку значно вище через невідповідність парадигми між тим, як дані представлені в об'єктах в порівнянні з реляційними базами даних. У даній розробці для вирішення завдань об'єктно-реляційного відображення використовується бібліотека Hibernate. Hibernate - це рішення Object / Relational Mapping (ORM) [11] для середовищ Java. Термін Object / Relational Mapping відноситься до методу відображення даних між поданням об'єктної моделі в уявлення реляційної моделі даних. Крім того, стаття OrmHate Мартіна Фаулера [15] розглядає багато проблем невідповідності цих методів.

Хоча ви можете використовувати Hibernate не потрібно глибокого розуміння SQL, потрібно базове розуміння концепцій, яке може допомогти зрозуміти Hibernate більш швидко і повністю. Особливе значення має розуміння принципів моделювання даних.

Hibernate піклується про відображенні даних з класів Java в таблиці бази даних і з типів даних Java в типи даних SQL. Крім того, він надає служби запитів і пошуку даних. Він може значно скоротити час розробки, витрачений на ручну обробку даних в SQL і JDBC. Мета Hibernate полягає в тому, щоб позбавити розробника від 95% загальних завдань, пов'язаних зі збереженням даних, за рахунок усунення необхідності ручної обробки даних вручну з використанням SQL і JDBC. Однак, на відміну від багатьох інших рішень, Hibernate не приховує від розробника можливості використання чистого SQL в своєму проекті і гарантує, що інвестиції в реляційні технології і знання залишаються актуальними, як завжди.

Hibernate може бути не кращим рішенням для додатків, орієнтованих на дані, які використовують тільки збережені процедури для реалізації бізнес-логіки в базі даних, це найбільш корисно з об'єктно-орієнтованими моделями домену та бізнес-логікою на рівні середнього рівня Java. Однак Hibernate, безумовно, допоможе видалити або инкапсулировать специфічний для постачальника код SQL і спростити загальну задачу перекладу наборів результатів з табличного представлення в граф об'єктів.

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

- хешуванні пароля з використанням випадкової солі;
- Можливість блокування користувача;
- Реалізовано білдер;

Для реалізації аутентифікації і авторизації може використовувати реалізований AuthorizationService, який використовує клас UserEntity, код якого представлений нижче.

```
@Data
@Builder @Entity
@Table(name = "user")
@AllArgsConstructor @NoArgsConstructor
public class UserEntity implements UserDetails {
   @Id
   @Column(name = "id")
   @GeneratedValue(strategy = GenerationType.AUTO)
private long id;


   @Basic
   @NonNull
   @Column(name = "username")
private String username;


   @Basic
   @NonNull
   @Column(name = "password")
private String password;


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


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


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


   @Basic
   @NonNull
   @Column(name = "credentialsNonExpired")
 private boolean credentialsNonExpired = true;
}
```

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

Доменні репозиторії, що розширюють цей інтерфейс, можуть вибірково виявляти методи CRUD Репозиторий для об'єктів UserEntity: `public interface UserRepository extends JpaRepository<UserEntity, Long>` Налаштування бази даних за замовчуванням:

```
@Bean
@ConditionalOnBean(name = "dataSource")
@ConditionalOnMissingBean
public LocalContainerEntityManagerFactoryBean entityManagerFactory() {    final
LocalContainerEntityManagerFactoryBean em = new
LocalContainerEntityManagerFactoryBean();    em.setDataSource(dataSource());
   em.setPackagesToScan("com.astel.security.model");
em.setJpaVendorAdapter(new HibernateJpaVendorAdapter());    if
(additionalProperties() != null) {
em.setJpaProperties(additionalProperties());
   }    return em;
}

@Bean
@ConditionalOnMissingBean(type = "JpaTransactionManager")
JpaTransactionManager transactionManager(final EntityManagerFactory
entityManagerFactory) {    final JpaTransactionManager transactionManager = new
JpaTransactionManager();
transactionManager.setEntityManagerFactory(entityManagerFactory);    return
transactionManager;
}

@ConditionalOnResource(resources = "classpath:mysql.properties")
@Conditional(HibernateCondition.class) final Properties additionalProperties()
{
   final Properties hibernateProperties = new Properties();
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");    return hibernateProperties;
}
```

2.5. Валідація даних

Реалізований захист від міжсайтової підробки запитів (CSRF) шляхом внесення стандартних налаштувань Spring Security.

Код вікна логіна включає в себе поле

```
<input name="_csrf" type="hidden" value="67b99669-74d0-4572-
9cfba859da5d3313" />
```

Значення ключа випадкове і унікальне для кожної сесії.

Для захисту від міжсайтового скриптинга використовується проект з відкритим вихідним кодом Harbinger, який реалізує функції брандмауера [14] і поширюваний під ліцензією Apache 2.

Для підключення залежно від цього проекту в 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, в текстовому файлі, по одному на рядок. Параметри розділені комами.

Правила конфігуруються за наступним принципам показаних в Таблиці 2.1

Таблиця 2.1. Конфігурація правил для валідації даних

Назва	Значення	Приклад
Ім’я	Стрічка	XSS
Пріоритет	LOW, MID, HIGH	MID
Шаблон	Патерн заснований на регулярних виразах	<script[^>]*>
Коментарі	#	# Коментар

Стандартний список правил для валідації даних наведено нижче:

```
#
# 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) *["'`]
```

Рисунок 2.3 Стандартний список правил

При вхідному HTTP-запиті инициализируются дані запиту Дані можуть бути доповнені поточним користувачем, ідентифікатором сеансу або віддаленим IP-адресою до його передачі в збирач даних запитів. Складальник даних реєструє і групує дані по сеансу і IP і підсумовує результат, створюючи агрегування доказів.

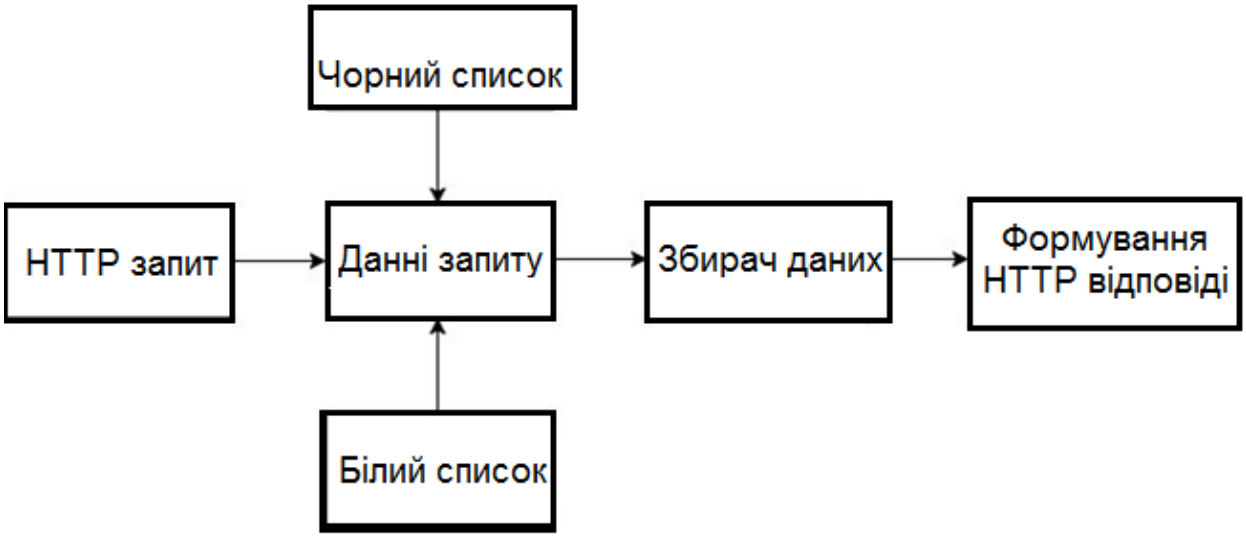


Рисунок 2.4 фільтрація запитів

Потім це агрегування подається в Реагування відповіді, щоб визначити дію відповіді, яке може бути відхиленням введення, анулюванням сеансу, тимчасовим занесенням в чорний список IP або нічим.

Є можливість використовувати білий список, щоб дозволити збирачеві денних ігнорувати будь-які дані, які відповідають певним характеристикам. Білий список підтримує придушення даних на основі IP-адреси, URL-адреси, імені параметра і користувача.

Також він підтримує логічні оператори OR і AND.

2.6. Логування

Багато системи дозволяють управляти мережевими пристроями, операційною системою, веб-сервером, поштовим сервером і сервером бази даних, але часто не реєструється журнал подій додатків, відключений або погано налаштований. Він забезпечує набагато більш глибоке розуміння, ніж управління інфраструктурою. Ведення веб-додатків (наприклад, веб-сайт або веб-сервіс) набагато більше, ніж включення журналів веб-сервера. [13]

Журнали додатків повинні бути узгодженими в додатку, узгодженими в портфелі додатків організації і, якщо це необхідно, використовувати галузеві стандарти, тому зареєстровані дані подій можуть використовуватися, королювання, аналізуватися і управлятися за допомогою самих різних систем.

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

- Ідентифікація інцидентів безпеки;
- Моніторинг порушень політики;
- Встановлення вихідних умов;
- Допомога в усуненні невідповідностей;
- Надання інформації про проблеми і незвичайних умовах;
- Внесення додаткових даних для конкретних випадків для розслідування інцидентів, які відсутні в інших журнальних джерелах;
- Допомога в захисті від виявлення і використання вразливостей за допомогою виявлення атак;

Додатки зазвичай записують дані журналу подій в файлову систему або базу даних (SQL або NoSQL). Додатки, встановлені на настільних комп'ютерах і на мобільних пристроях, можуть використовувати локальні сховища і локальні бази даних, а також відправляти дані на віддалене сховище. Обрана структура може обмежити доступні варіанти. Всі типи додатків можуть відправляти дані подій на віддалені системи (а не на локальне сховище). Це може бути централізована система збору та управління журналом (наприклад, SIEM або SEM) або іншу програму в іншому місці.

При використанні файлової системи краще використовувати окремий розділ, ніж той, який використовується операційною системою, інші файли додатків і призначений для користувача контент

Для файлових журналів застосовуються строгі дозволи, що стосуються доступу користувачів до каталогів, а також дозволів файлів в каталогах

У веб-додатках журнали не повинні відображатися в місцях, доступних в Інтернеті, і, якщо це зроблено, повинні мати обмежений доступ і бути налаштовані з використанням простого текстового типу MIME (А не HTML)

При використанні бази даних краще використовувати окремий обліковий запис бази даних, яка використовується тільки для запису даних журналу, і яка має дуже обмежувальні бази даних, таблиці, функції та дозволу команд

Використовуються стандартні формати над захищеними протоколами для запису і відправки даних подій або файлів журналів в інші системи, наприклад. Загальна файлова система журналу (CLFS), Common Event Format (CEF) через syslog, можливо, Common Expression (CEE) в майбутньому;

стандартні формати полегшують інтеграцію з централізованими службами реєстрації.

Кожен запис в журналі повинна 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 додатків повинні записувати «коли, де, хто і що» для кожної події. Властивості для них будуть різними в залежності від архітектури, класу програми та хост-системи / пристрої, але часто включають таке: Коли

● Дата і час реєстрації (міжнародний формат);

● Дата і час події - відмітка часу події може відрізнятися від часу реєстрації, наприклад. сервер, де клієнтську програму розміщено на віддаленому пристрої, яке виконується лише періодично або час від часу;

● Ідентифікатор взаємодії; де

● Ідентифікатор додатка, наприклад. ім'я та версія;

● Адреса додатки, наприклад. ім'я кластера / хоста або сервер IPv4 або IPv6 адресу і номер порту, ідентифікатор робочої станції, ідентифікатор локального пристрою;

● Обслуговування, наприклад. ім'я та протокол геолокації;

● Вікно / форма / сторінка, наприклад. URL точки входу і HTTP-метод для веб-додатки, ім'я діалогового вікна;

● Розташування коду, наприклад. ім'я сценарію, ім'я модуля; хто

● Адреса джерела, наприклад. ідентифікатор пристрою / машини користувача, IP-адреса користувача, ідентифікатор осередки / RFбашні, номер мобільного телефону;

● Ідентифікація користувача (якщо вона аутентифицироваться або відома іншим чином), наприклад. значення первинного ключа таблиці бази даних користувача, ім'я користувача, номер ліцензії;

які

● Тип події;

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

● Прапор події безпеки (якщо в журналах також містяться дані подій безпеки);

● Опис;

При реєстрації даних необхідно враховувати чинне законодавство. Наприклад, перехоплення деяких повідомлень, моніторинг співробітників і збір 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 в записані дані.

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

● Вихідний код програми;

● Значення ідентифікації сеансу (подумайте про заміну хешірованного значенням, якщо необхідно для відстеження подій, специфічних для сеансу);

● Доступ до токені;

● Чутливі персональні дані і деякі форми особистої ідентифікованої інформації (PII), наприклад. здоров'я, урядові ідентифікатори, вразливі люди;

● Аутентифікаційні паролі;

● Рядки підключення бази даних;

● Ключі шифрування і інші основні секрети;

● Дані власника банківського рахунку або платіжної картки;

● Дані вищою класифікації безпеки, ніж система реєстрації, дозволяють зберігати;

● Комерційна інформація;

● Інформація незаконно збирати у відповідних юрисдикціях;

● Інформація, яку користувач відмовився від збору або не погодився, наприклад, використання не відстежується, або коли згоду на збір минув;

Іноді також можуть існувати такі дані і, в той же час, корисні для подальшого дослідження, також може бути необхідно обробити яким-небудь особливим чином до того, як подія буде записано:

● Шляхи файлів

● Рядки підключення бази даних

● Внутрішні мережеві імена та адреси

● Нечутливі особисті дані (наприклад, особисті імена, номери телефонів, адреси електронної пошти)

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

Реалізується створенням інтерфейсу SecuredService і реєстрацією всіх методів виконання зобов'язань даного інтерфейсу за допомогою log4j

```
@Aspect
@Compon
ent
@Log4j
public class Logging {
   @After("target(com.astel.security.SecuredService)")
   public Object securedMethodsLogging(ProceedingJoinPoint pjp) throws
Throwable
{
       Object retVal = pjp.proceed();
       log.info(pjp.getTarget().getClass().getName() + " " +
pjp.getSignature().ge
tName());
return retVal;
   }
}
```

Стандартні настройки логування представлені в наступній таблиці:

Таблиця 2.2 Стандартні настройки логування

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

2.7. Перевірка безпеки додатку

Для перевірки безпеки додатку створеного на основі даного фреймворка використовується сканер вразливостей Wapiti.

Wapiti побудований на модулях, які підключаються при сканування - backup, blindsql, crlf, exec, file, htaccess, nikto, permanentxss, sql, xss, buster, shellshock.

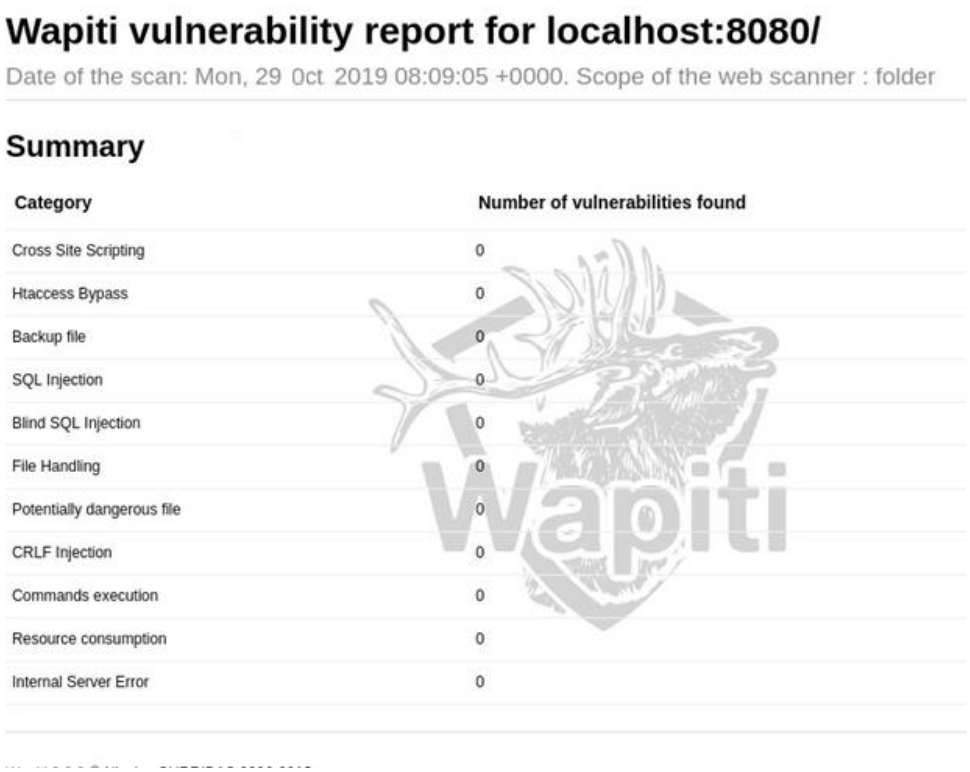
Wapiti є консольним сканером веб-додатків, який в своїй основі несе принцип BlackBox тестування (аналізує не програмний код додатка, а відповіді сервера на запити зі зміненими параметрами). Утиліта спочатку аналізує структуру сайту, шукає доступні сценарії, аналізує параметри, а потім включає свій фаззер. [17]

“Фаззинг (анг. Fuzzing) - це автоматизована методика тестування програмного забезпечення, в якій програма, що підлягає тестуванню, багаторазово завантажується випадковими даними на одному або декількох вхідних інтерфейсах. Випадкові дані зазвичай можуть створювати ситуації в роботі програми, які неможливо досягти за допомогою інших методів тестування. Програми часто не розроблені для будь-яких вхідних даних, а потім можуть випадково вийти з ладу, якщо дані не є правдоподібними, і,

таким чином, також виявляють вразливості. Отже, метушня - одна з найважливіших методик, яку використовують фахівці з безпеки. ” [16]

Щоб протестувати безпеку, реалізовану фреймворком, створимо базове додаток, запустимо його на локальному сервері на основі фреймворку і запустимо wapiti.

Звіт по перевірки на уразливості показаний на Рисунку 2.5.



Wapiti vulnerability report for localhost:8080/

Date of the scan: Mon, 29 Oct 2019 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

Рисунку 2.5 Звіт про автоматизовану перевірку веб додатку

Висновок

Даний відповідає поставленим вимогам та реалізує захист розробленого додатку від найпоширеніших вразливосте. Далі веб-додаток може розвиватися з уже обмеженого стандартними настройками стану відповідно до гнучкої методології розробки.

3 ЗАСТОСУВАННЯ РОЗРОБЛЕНОГО ФРЕЙМВОРКА

3.1. Галузь застосування

Даних фреймворк необхідний для створення нових веб додатків за умови максимально швидкого старту з імплементованими настройками безпеки. Згідно Рисунку 3.1 для конфігурації безпеки на пізніх стадіях необхідна додаткова настройка обумовлена бізнес-логікою самого додатка

Рисунку 3.1 Реалізація захисту під час розробки програми

3.2. Складання

Для складання даного проекту необхідно

● Завантажити і встановити систему контролю версій git;

● Завантажити і встановити систему автоматизованого складання Gradle;

● Встановити будь-яке середовище розробки для java;

● Клонувати репозиторій з веб-сервісу GitHub

https://github.com/hensey/adframework.git

● Імпортувати цей репозиторій як gradle проект;

● Запустити команду gradle build з консолі або з плагіна в вибраному середовищі розробки;

3.3. Використання

Для використання потрібно:

● Підключити залежність від гіт сховища. Рекомендується використовувати плагін Jitpack.

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

Якщо проект вже включено до GitHub, JitPack гарантує, можливість складання.

● Додайте репозиторій JitPack в файл збірки:

● Додати залежність

● Публікація Javadoc. Якщо проект створює javadoc.jar, є можливість переглядати файли javadoc безпосередньо:

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

Можливості JitPack:

- Робота з приватними репозиторіями
- Динамічні версії. Існує можливість використовувати динамічну версію Gradle '1. + 'I діапазони версій Maven для релізів. Вони дозволяють випуски, які вже були побудовані.
- JitPack періодично перевіряє наявність нових випусків і будує їх з випередженням.
- Побудувати по тегу, commit id або anyBranch-SNAPSHOT.
- Можливість використовувати власне доменне ім'я для groupId

Тоді код підключення залежності для різних систем збірок буде виглядати наступним чином:

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

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>
</dependency>
```

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 і точкою входу в додаток

Нижче наведено повний приклад коду:

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

На рисунку 3.1. показано стандартне вікно введення призначених для користувача даних у вікні веб-додатки

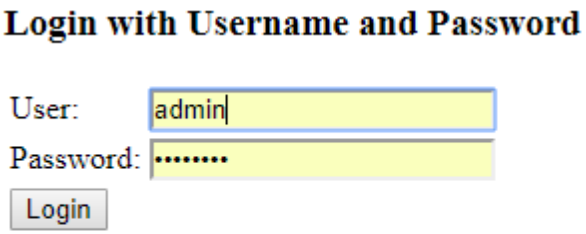


Рисунок 3.1. вікно аутентифікації

При введенні неправильних вхідних даних з'являється повідомлення з помилкою про помилкових даних

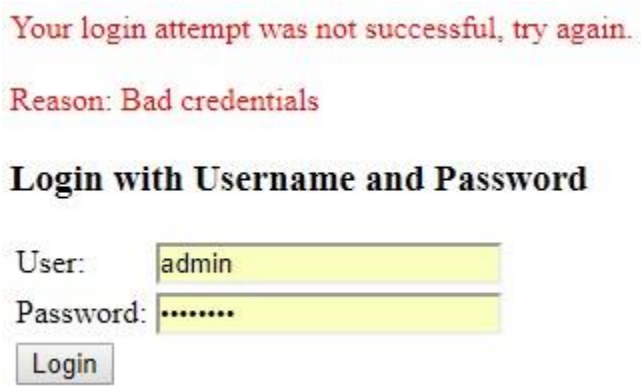


Рисунок 3.2 Помилка. Неправильні вхідні дані.

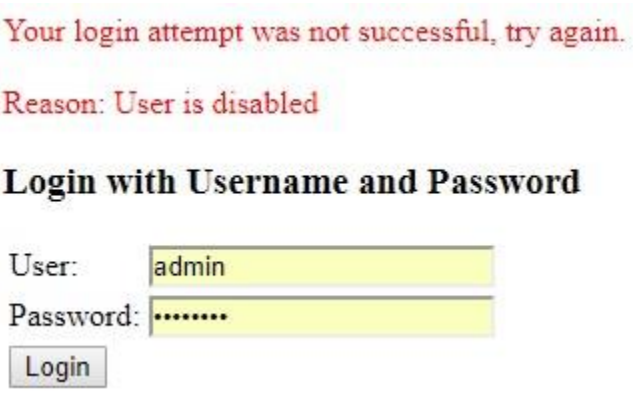


Рисунок 3.3. Помилка. Користувач заблокований

3.3.1 Налаштування контексту безпеки

Фреймворк намагається автоматично налаштувати що розробляється. Наприклад, якщо будь-які компоненти з'єднання з базою даних не налаштовані на момент компіляції додатка, фреймворк автоматично налаштує базу даних в пам'яті. [18]

Щоб вибрати автоматичне налаштування потрібно додати анотацію @EnableAutoConfiguration або @SpringBootApplication до одного з класів

@Configuration, використовуваних в додатку. Рекомендується додати її до основного класу @Configuration.

Автом не постійна, в будь-який момент розробник можете почати визначати свою власну конфігурацію для заміни певних частин автоматичної конфігурації. Наприклад, розробник може додасте свій власний компонент DataSource для додавання взаємодії з базою даних в проект. У цю базу даних також будуть додані таблиці для сутностей: UserEntity, CredentialsEntity, RoleEntity.

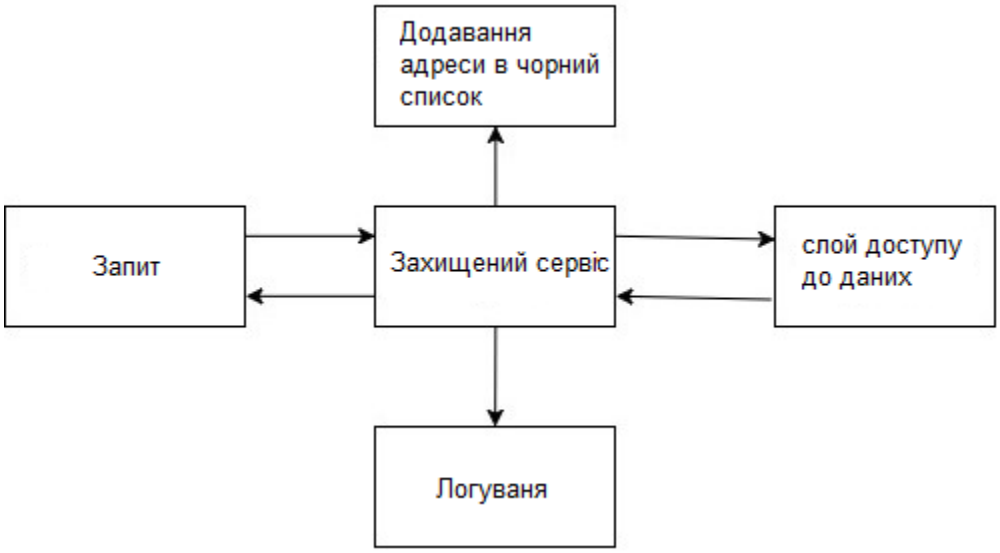
Додавання контексту за замовчуванням в проект реалізується механізмів автоконфігурації. [19] Щоб змінити контекст безпеки досить реалізувати свого спадкоємця класу WebSecurityConfigurerAdapter

3.3.2 Сервіс авторизації

Імплементація сервісу авторизації за замовчуванням додається в проект за допомогою механізму автоконфігурації. Щоб змінити контекст безпеки досить реалізувати свій клас що наслідуватиме UserDetailsService.

Підключається разом всім фреймворком. Можна підключити окремо додавши в конфігурацію

Функціонал роботи захищеного сервісу показаний на рисунку 3.2



Рисунку 3.2 Робота SecurityService

При порушенні правил, IP-адреса користувача заноситься в чорний список.

3.3.3 База даних

За замовчуванням в SecuredService класі даного фреймворка використовується база даних H2 і сутності UserEntity і CredentialsEntity.

H2 - відкрита кроссплатформенная Система Управління Базами Даних (СУБД), повністю написана на мові Java.

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

- Два режиму роботи (клієнт-сервер, вбудований)
- Два режиму зберігання даних (файлова система, пам'ять)
- Підтримка планів виконання запитів
- Підтримка кластеризації і реплікації

- Шифрування даних
- Зовнішні (пов'язані) таблиці
- Драйвер ODBC
- Повнотекстовий пошук
- Визначення доменів
- Мультіверсіонний конкурентний доступ
- Підтримка послідовностей
- Підтримка ключових слів LIMIT і OFFSET в запитах
- Користувальницькі агрегатні функції
- Призначені для користувача процедури, що зберігаються
- Стиснення CLOB / BLOB об'єктів
- Робота з CSV файлами на читання і запис
- Браузерна консоль управління
- Запуск як сервіс Windows

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

Таблиця 3.1. Налаштування бази даних за замовчуванням:

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 і виводиться Credentials.log при старті

Для вирішення завдань об'єктно-реляційного відображення використовується бібліотека Hibernate. Її настройки за замовчуванням представлені в таблиці 3.2

Таблиця 3.2. налаштування Hibernate

Ім’я параметра	Значення
Політика обновлення DDL схем	h2-hibernate.hbm2ddl.auto
Діалект SQL	h2-hibernate.dialect
Вивід SQL в лог	false

3.3.4 Валідація даних

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

Java-анотація - це мовний елемент, який дозволяє інтегрувати метадані у вихідний текст.

«Анотації використовуються для аналізу коду, компіляції або виконання. Аннотируемой пакети, класи, методи, змінні і параметри.

Виглядає як @ ІмяАннотації, що передують визначення змінної, параметра, методу, класу, пакета»[34].

Дану анотацію можна застосовувати для: полів, методів, параметрів.

3.3.5 Логування

За замовчуванням включено для всіх методів SecuredService з використанням бібліотеки log4j.

Log4j - бібліотека журналирования Java програм, частина спільного проекту «Apache Logging Project».

Log4j спочатку розвивався в рамках зонтичного «Apache Jakarta Project», відповідального за всі Java-проекти Apache, але згодом виділився в окремий, дуже популярний проект журналирования.

Налаштування Логгер можна змінити додаючи їх в application.properties

3.3.6 криптографія

В даному фреймворку для хешування паролів за замовчуванням використовується алгоритм bcrypt.

Хешування - це перетворення рядка символів в зазвичай коротший фіксоване значення або ключ, який представляє вихідну рядок. Хешування використовується для індексації і вилучення елементів в базі даних, тому що швидше знайти елемент, використовуючи більш короткий хешировать ключ, ніж знайти його за допомогою оригінального значення. Він також використовується в багатьох алгоритмах шифрування.

Для зміни алгоритму хешування можна скористатися методом:

```
public ucserDetailsServiceImpl.setPasswordEncoder(password PasswordEncoder);
```

Також можна реалізувати свій захищений сервіс з своєї реалізацією криптографії.

4 ОПТИМІЗАЦІЯ ТА АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

Тестування грає життєво важливу роль в процесі розробки і створення якісного програмного забезпечення. Необхідно серйозно ставитися до аналізу і проектування структурованого процесу, який забезпечить своєчасний і успішний випуск проекту.

Важливо пам'ятати, що довіра користувачів дуже просто втратити, і виправити допущені помилки може коштувати дорожче, ніж спочатку провести повну підготовку і тестування. як оптимізувати і автоматизувати процеси тестування на проникнення за допомогою спеціалізованих утиліт і їх розширень. Автоматизоване тестування має ряд переваг та недоліків:

- При автоматизованому тестуванні, як правило, значно економиться час тестування, можна покрити більшу площу веб-додатки за менший час.
- Велика кількість перевірок. Автоматизовані системи містять величезну кількість патернів атак, ознак вразливостей, і, як правило, розширюваність.
- Перебір файлів і папок, підбір паролів - тут, я думаю все зрозуміло і так.
- Регламентне сканування і процедури інвентаризації - для цих цілей автоматизовані системи підходять найкраще.

Недоліки:

- False positive спрацьовування. Дуже часто сканери керуючись формальними ознаками виявляють уразливості, яких немає. Класика жанру - при скануванні Single Page Application сканер

отримує код відповіді 200 на всі свої запити і виводить довгий список вразливостей, яких насправді немає.

- Вони "дуже галасливі". При скануванні сайту створюється дуже багато подій в журналах веб-сервера, за якими легко визначити атаку.
- Навантаження на веб-сервер. Іноді автоматизоване сканування може дати відчутну навантаження на веб-сервер, що може привести до нестабільної роботи веб додатки (хоча цей мінус відноситься до конфігурації веб-сервера).
- Блокування засобами захисту. Як правило, ознаки автоматизованих систем добре знайомі розробникам і вони враховують їх при проектуванні. Як підсумок - відбувається блокування (по User Agent, маркерами сканера або частоті запитів).
- Не враховують помилки логіки.

4.1 Побудова автоматизованих систем, оптимізованих під веб-додаток

Для того, щоб автоматизована система була максимально ефективною, вона повинна мати такі можливості:

- розширення функціоналу, в тому числі за допомогою сторонніх модулів;
- мультиформатність результатів тестування;
- імпорт / експорт результатів тестування;
- стандартизовані результати тестування;
- можливість порівняння результатів;
- можливість інтеграції системи в більш складну.

Ці фактори дозволять побудувати систему, що відповідає вашим вимогам і цілям.

Як приклад "готової системи" можу привести описану раніше Sparta. Для того, щоб тестування було максимально ефективним, контрольованим, а також для комфортної валідації вразливостей необхідно враховувати всі компоненти системи, архітектуру тестованої програми і зв'язність рішень.

4.2 Утиліти

В якості оптимальної основи було вирішено зупинитись на двох кроссплатформенних системах для тестування веб-додатків (як в ручному, так і в автоматизованому режимі): OWASP ZAP (free версія) і BurpSuite (free + платна версії).

Найважливіша відмінність цих систем від класичних сканерів - це принцип роботи: сканер "довбає по сайту" безпосередньо, виявляючи ті чи інші ознаки вразливостей, часто пропускаючи величезні ділянки веб-додатки. А Zap і Burp працюють в якості проксірующего механізму, що дозволяє додати всі області сайту (як вбудованим "павуками", так і при ручному серфінгу додатки). Також, важливою особливістю є можливість "на льоту" розбирати кожен запит.

Величезним плюсом цих додатків є можливість розширення за допомогою плагінів / компонентів:

- магазин додатків (готові рішення);
- інтеграція сторонніх модулів (поза магазином додатків);
- написання власних (кастомізованих під конкретний проект).

Як приклад застосування пропоную розглянути вразливе веб-додаток з наступними характеристиками:

- відома CMS;
- підтримка плагінів / компонентів;
- містить уразливості (в тому числі OWASP A1 - sql injection);
- застосовує заходи безпеки.

Виходячи з цих даних нам необхідно вибрати і використовувати такі компоненти (мінімальний набір).

4.3.1 Owasp ZAP

Як можна здогадатися з назви, за випуск OWASP ZAP відповідає та сама організація OWASP, що ми згадали у вступі. Це безкоштовний інструмент для тестування на проникнення і для пошуку вразливостей в веб-додатках.

Робоча область OWASP ZAP складається з декількох вікон. Внизу - вкладки з поточними завданнями і процес їх виконання, зліва - дерево сайтів, додатково можна вивести в праву частину вікна запитів і відповідей.

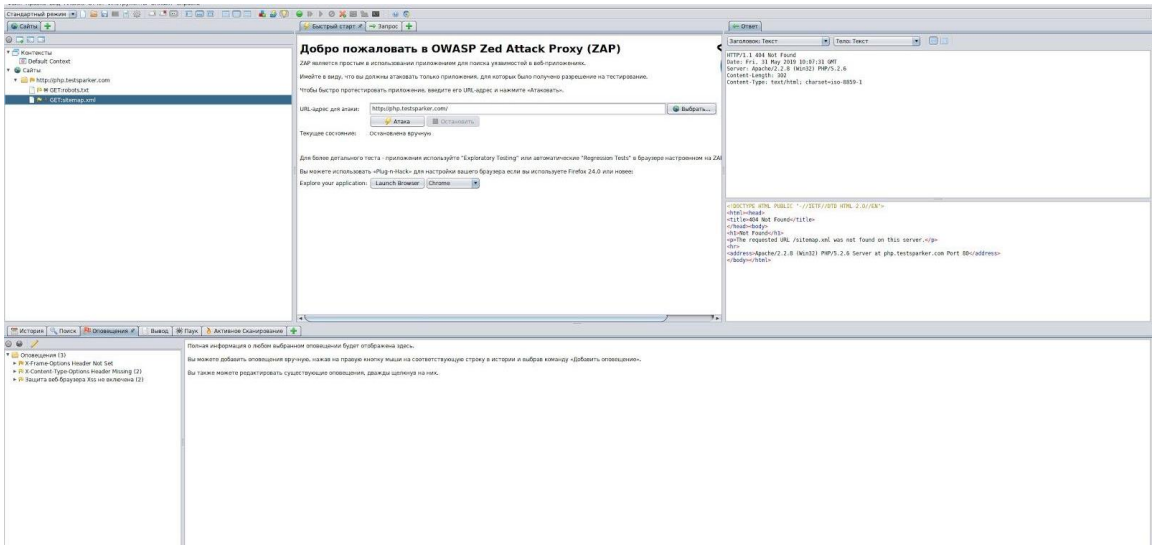


Рисунок 4.1 Інтерфейс програми Owasp ZAP

Тут необхідно вибрати ті інструменти, які допоможуть виявляти і експлуатувати уразливості, позначені в списку на рисунку 4.2 нижче:

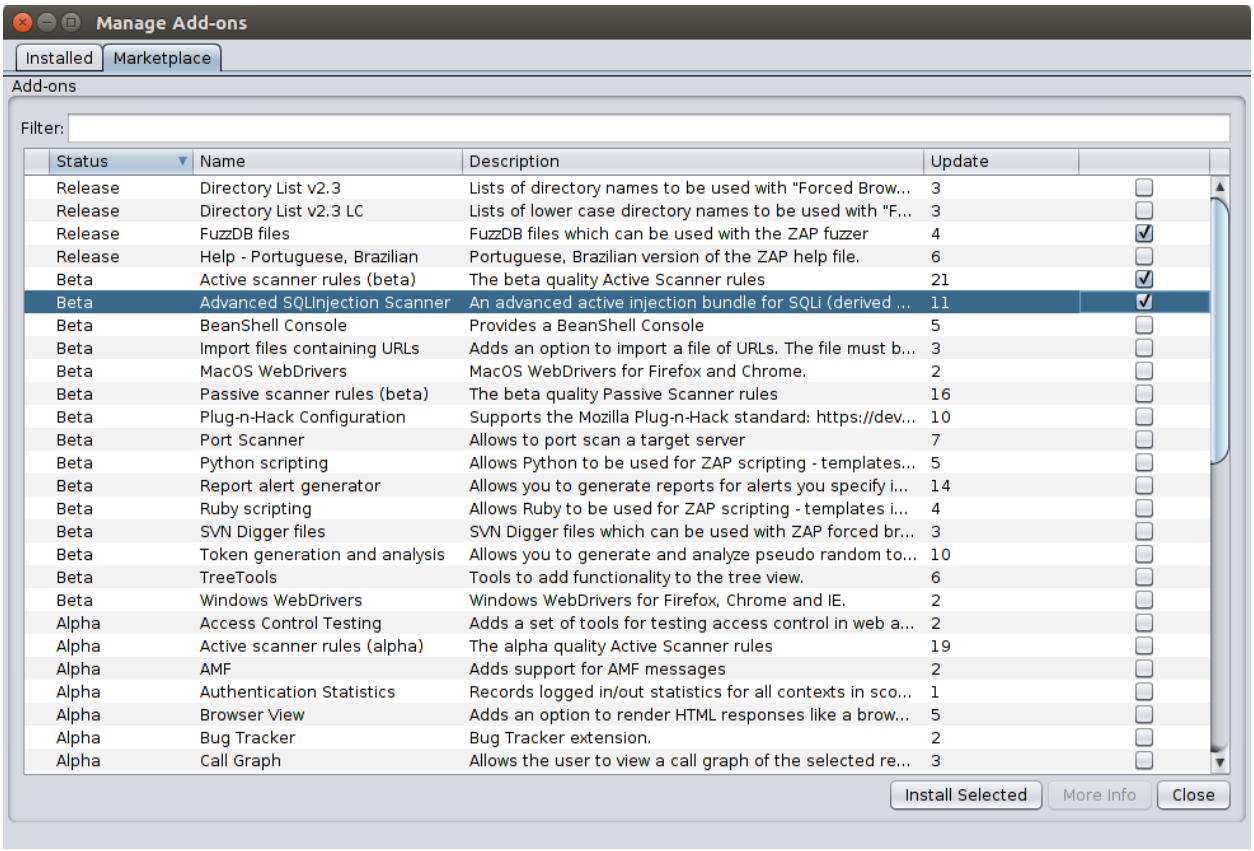


Рисунок 4.2 список інструментів що необхідно встановити

Є всі необхідні інструменти для пентеста веб-додатку, простий і зрозумілий інтерфейс, швидке сканування. І при цьому гнучкі, глибокі настройки для більш детального сканування, що може послужити відправною точкою для подальшого ручного пошуку вразливостей. Нижче ми ще розповімо про сканер Burp Suite, який має з OWASP ZAP багато спільного. За кількістю і якістю знайдених вразливостей перший розглянутий нами сканер показав дуже непоганий результат. Рекомендований до використання в роботі.

4.3.2 Burp Suite

Burp Suite - це платформа для проведення аудиту безпеки веб-додатків. Містить інструменти для складання карти веб-додатки, пошуку файлів і папок, модифікації запитів, фаззинга, підбору паролів і багато іншого. Також існує магазин додатків BApp store, що містить додаткові розширення, що збільшують функціонал додатка. Варто відзначити і появу в останньому релізі мобільного помічника для дослідження безпеки мобільних додатків - MobileAssistant для платформи iOS. Інтуїтивно зрозумілий інтерфейс із спеціально спроектованими табами, що дозволяють поліпшити і прискорити процес атаки. Сам інструмент вдає із себе проксірующий механізм, що перехоплює і обробляє всі вхідні від браузера запити. Є можливість встановлення сертифіката burp для аналізу https з'єднань.

Якщо подивитися статистику і репорти bug-bounty програм - практично скрізь на скріншотах можна зустріти використання цього інструменту. На ряду з OWASP ZAP це найпопулярніший набір утиліт для тестування веб-додатків. Нам необхідно виконати завдання по ідентифікації CMS і встановлених компонентів, виявити застарілі версії, спробувати обійти WAF і проексплуатувати SQL-ін'єкцію.

В першу чергу необхідно дотримуватися методології тестування веб-додатки. У цьому нам допоможе представлений на останньому Def Con HUNT Burp Suite Extension:

- Визначаємо загальні параметри для певних класів вразливостей.
- Організовуємо методологію тестування всередині Burp Suite.

Рисунок 4.3 параметри класів вразливостей

Далі нам знадобиться плагін (в BApp store) від Vulners.com

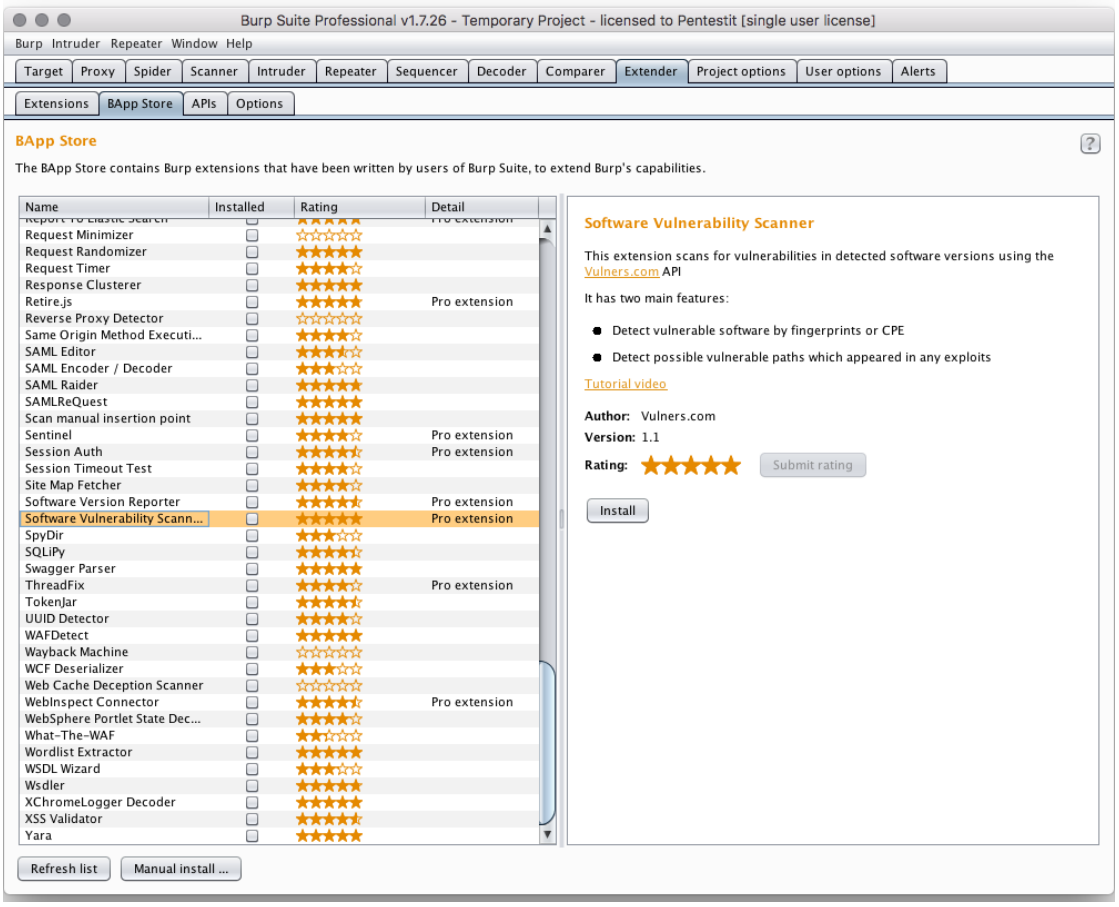


Рисунок 4.4 плагін від Vulners.com

Також, може бути корисний розширений набір фаззинга для sql-ін'єкцій (якого немає в безкоштовній версії - sql).

Далі, прискорити "розкручування" ін'єкції нам допоможе sql map - для інтеграції якого необхідно скористатися плагіном SQLiPy:

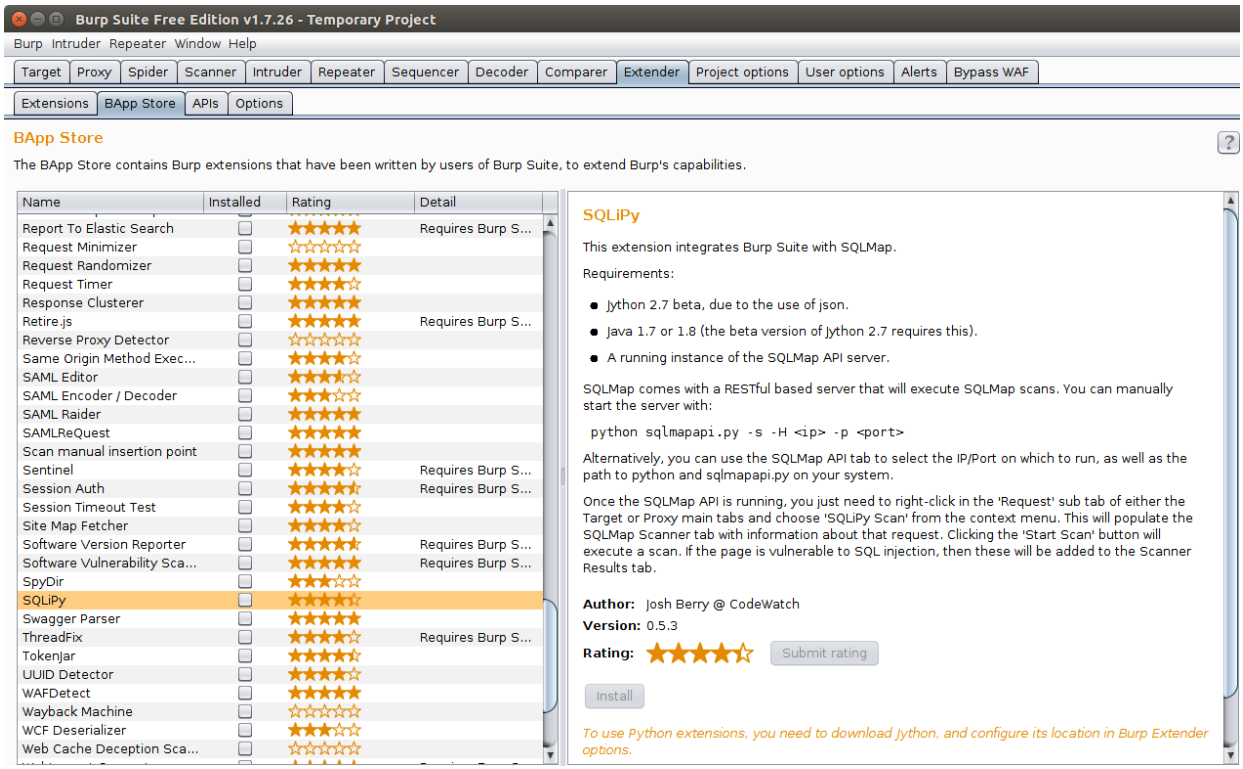


Рисунок 4.5 плагін SQLiPy

Допустимо, що веб-додаток захищено WAF - можуть стати в нагоді розширення What the WAF і Waf bypass.

Як засіб автоматизації можна використовувати вбудований сканер, або скористатися інструментом Burp Automator:

Вимоги:

- burp-rest-api
- Burp Suite Professional

- slackclient

Цей інструмент дозволить автоматизувати перевірки, використовую в якості основи Burp Suite. Як результат ви отримаєте звіт про проведене скануванні:

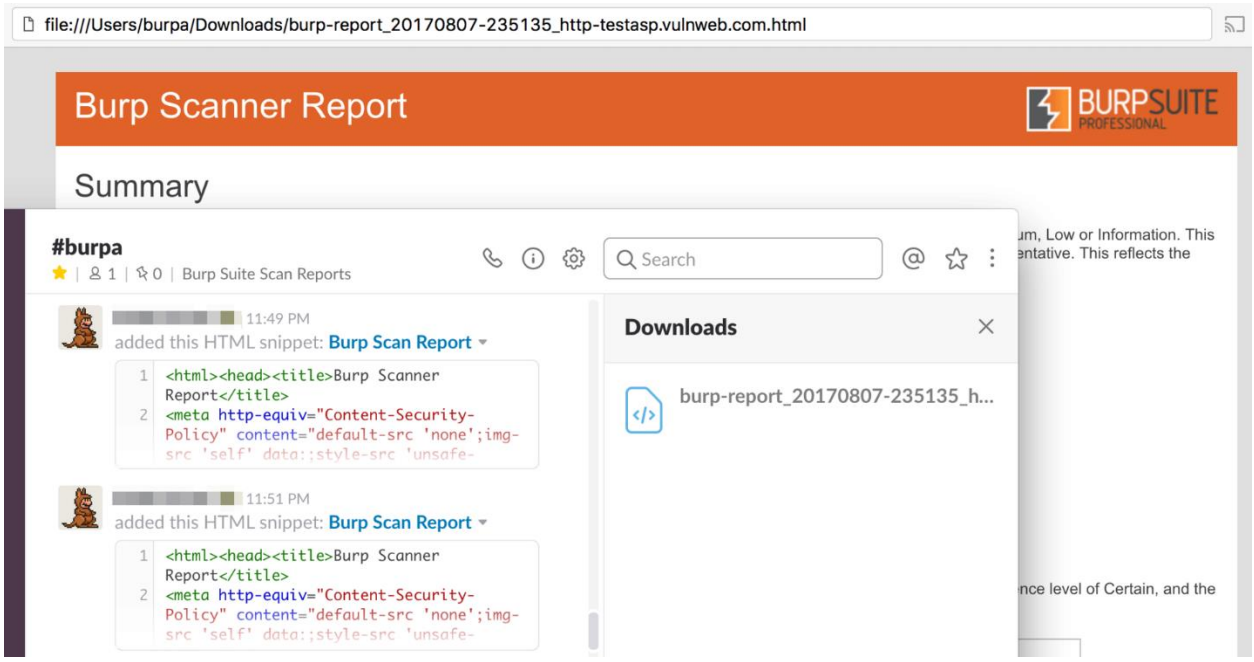


Рисунок 4.6 звіт про проведене сканування

4.4 Висновок

Burp Suite заслужено є одним з найпопулярніших інструментів пентестеров по всьому світу завдяки гнучким можливостям, способам комбінувати ручні і автоматизовані методи аналізу при проведенні тестування безпеки веб-додатків.

Для того щоб максимально ефективно застосовувати інструменти автоматизованого тестування необхідно мати базис ручної перевірки, для більш точного налаштування системи

5 ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Концепція

Дана робота присвячена розробці фреймворка безпеки для веб-додатків.

Короткий технічний опис

Фреймворк призначений для реалізації захисту веб-додатки на перших етапах розробки. Даний фреймворк дозволяє розробнику зосередитися на реалізації функцій і змінювати конфігурацію захисту пізніше під унікальні вимоги веб додатку.

Ринок і план маркетингу

Даний програмний модуль являє собою готовий об'єкт для продажу. Одне з можливих застосувань даного модуля - розробка бізнес рішень у всіх сферах. Тоді, в якості цільової аудиторії для продажу даного модуля слід розглядати середні підприємства, які займаються бізнесом яких переходить в веб та підприємства, що займаються переведенням бізнес процесів підприємств в інтернет простір. Просування даного програмного модуля на ринку передбачається за рахунок статей на ресурсах по веб-розробці, надання зацікавленим у використанні даного модуля, можливість його опробування і прийняти рішення про подальшу покупку платної підтримки для підприємств і приватних пожертвувань.

Визначення трудомісткості ДП

В основі визначення вартості ДП лежить перелік виконаних робіт та їх трудомісткість. Тривалість кожного етапу визначається з календарного плану виконання ДП і представлена в табл.5.1.

Таблиця 5.1 - Трудомісткість виконання робіт

№	Назва роботи	Трудомісткість (люд./дні)	
		Керівник	Виконавиць
1	Розробка завдання	2	2
2	Вивчення літератури	4	22
3	Розробка і реалізація програмного рішення	2	40
4	Тестування, порівняння з існуючими програмними рішеннями	2	10
6	Оформлення пояснювальної записки и презентації	1	16
	Всього	11	73

Розрахунок вартості проекту

Калькуляція собівартості розробки системи здійснюється за такими статтями:

- витрати на оплату праці;
- матеріали;

● витрати по роботах, що виконуються сторонніми організаціями;

● витрати на утримання та експлуатацію обладнання;

● спецобладнання;

Витрати за статтями «Витрати по роботах, що виконуються сторонніми організаціями», «« Витрати на утримання та експлуатацію обладнання »» і «Спецобладнання» не передбачені, тому що в ході виконання даної ВКР послуги сторонніх організацій не потрібні були, витрати на утримання та експлуатацію обладнання в рамках виконання даної ВКР зневажливо малі, а спеціальне обладнання для наукових і експериментальних робіт не використовувалося.

Витрати на оплату праці

Ставка заробітної плати для виконавців визначена виходячи з значення середньої зарплати по Україні за даними пошукової системи trud.com. Згідно поділу персоналу за категоріями, використаному в даному статистичному дослідженні, працівники, зайняті інженерно-технічними, економічними та іншими роботами, а зокрема інженери і програмісти, відносяться до категорії «Спеціаліст». В даному дослідженні наводиться значення середньої нарахованої заробітної плата працівників за професійними групами. Останнє дослідження включає результати, отримані за 2019 рок, які і будуть використані в даній роботі. Середня заробітна плата для групи «Фахівці вищого рівня кваліфікації» склала 17 023 гривень, а для групи «Фахівці середнього рівня кваліфікації» - 10 492 гривень.

В даному проекті 21 робочий день прийнято за середню кількість робочих днів у місяці.

Денна ставка складе:

- 17023/21 = 810 грн/день – для керівника проекту
- 10492/21= 499 грн/день – для виконавця проекту

Витрати на оплату праці становитимуть

$З_{прац}$=810*11 + 499* 73 = 45337 грн

Калькуляція витрат на матеріали

В ході розробки даного проекту було потрібно роздрукувати деякі електронні джерела та підсумковий звіт, а також потрібен флеш-накопичувач для зберігання резервної копії даних. Калькуляція витрат по даній статті наведена в табл.5.2

Таблиця 5.2 витрати на матеріали

п/п	Матеріали	Одиниця виміру	Кількість	ціна (грн.)
1	Бумага для офісної техніки (Zoom, A4, 500 листів)	упаковка	1	77
2	Картридж для лазерного принтера Xerox 106R02181	шт	1	700
3	USB-флеш-накопичувач Kingston 16Gb	шт	1	120

Всього витрат на матеріали 897 грн.

Сумарні витрати на проект становлять

Витрати на оплату праці + витрати на матеріали = 45337 грн + 897 грн = 46234 грн.

Розрахунок окупності (економічної ефективності)

Фреймворк, розроблений в рамках даного проекту, що не накладає жодних технічних обмежень на кількість можливих користувачів. Вартість супроводу в місяць даного програмного модуля одним користувачем буде визначена як середнє значення мінімально можливих цін аудиту безпеки і складе 5500 грн/місяць[24].

Кількість місяців підтримки даного програмного модуля, яке необхідно продати при даній ціні за місяць для досягнення бажаного прибутку і окупності вкладених інвестицій, складе:

Сумарні витрати на проект / ціна на місяць = 46234/6700 = 9 місяців

Дана кількість може бути продано за рік.

Висновок

Трудомісткість розробки становить 84 людино-днів, а собівартість 46234 гривень. Необхідний термін для окупності вкладених інвестицій і отримання бажаного прибутку становить 1 рік. Розроблене рішення передбачає послуги з впровадження та підтримки, що може принести стабільний дохід. Все це робить розробку економічно доцільною.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Охорона праці

Робоче приміщення, у якому проходило дослідження та побудова середовища розробки веб-додатків з оптимальним рівнем безпеки, має відповідати вимогам щодо охорони праці при організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ВДТ).

Робоче приміщення та місце відповідає вимогам щодо охорони праці при організації роботи з ВДТ електронно-обчислювальних машин.

Відповідно до ДСН 3.3.6.042-99 роботи, що виконуються користувачами ЕОМ, відносяться до легких фізичних робіт – категорії Іа. У виробничих приміщеннях на робочих місцях з ВДТ мають забезпечуватись оптимальні значення параметрів мікроклімату.

Згідно ДБН В.2.5-28:2018 приміщення, що розглядається, повинне мати природне і штучне освітлення.

Денне (природне) освітлення приміщення відбувається за системою однобічного бічного освітлення. Природне світло проникає у приміщення через три світлові прорізи (віконні отвори), які мають регулювальні пристрої для відкривання. Також наявні штори (жалюзі) з можливістю захисту працюючих від прямого попадання сонячних променів і регулювання рівня освітленості в приміщенні. Вікна приміщення орієнтовані на північний захід. Оскільки будинок розташований у відносній віддаленості від прилеглих будівель, то які небудь перешкоди природному освітленню розглянутого приміщення відсутні.

Всередині приміщення стіни обклеєні світлими шпалерами, стеля побілена (переважає білий колір), у якості підлогового покриття використаний лінолеум світло-жовтого кольору.

Наявність постійного шуму в робочій зоні призводить до розладу центральної нервової системи і до таких захворювань як неврози, однак фактичний обмірюваний рівень шуму в робочій зоні склав 43 дБА, що задовольняє нормативному рівню шуму (не повинен перевищувати 50 дБА), тому додаткових заходів по поліпшенню цього фактору не потрібно.

Проаналізуємо стан електробезпеки в робочому приміщенні:

- всі прилади в кабінеті використовують напругу 220 В;
- електропроводка захована і ізольована від працівників спеціальним коробом;
- всі робочі місця з ПЕОМ використовують спільні розетки по 220 В;
- споживачі електроенергії — 2 ПЕОМ у вигляді ноутбуків;
- відносна вологість повітря – 60%, температура повітря +20 °С — +23 °С, струмопровідний пил і хімічно активні речовини в повітрі відсутні;
- підлога: ізолююча – лінолеум.

Проаналізувавши наведене вище, можемо сказати, що кабінет відноситься до приміщень без підвищеної електронебезпеки.

ПЕОМ, що використовуються в даному кабінеті підключаються до трифазної мережі і мають захисне занулення (за допомогою окремого захисного нульового провідника). Корпуси ВДТ та принтера виготовлені з пластику і не являються струмопровідними. Щодо корпусів самих ПЕОМ,

вони виготовлені зі струмопровідного матеріалу, крім передньої панелі, що виготовлена з пластику.

При виконанні робіт по ремонту і обслуговуванню ПЕОМ обслуговуючий персонал зобов'язаний керуватися “Правилами техніки безпеки при експлуатації електроустановок споживачами". До роботи не допускаються особи, які не пройшли навчання з техніки безпеки.

Джерелом електромагнітного випромінювання в сучасному офісі є візуальні дисплейні термінали. Нормування електромагнітного випромінювання ВДТ а здійснюється згідно НПАОП 0.00-7.15-18 “Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями”.

6.2 Використання комп’ютерної техніки по оцінці обстановки

XXI століття, в яке ми входимо сьогодні, недаремно називають ерою інформації. Електронні інформаційні технології дедалі ближче підступають до людини не лише на виробництві, й у побуті як надійний помічник та універсальний інструмент для вирішення багатьох найскладніших завдань.

Електронній “розум” з властивими йому технічними функціональними перевагами давно ефективно працює у сферах, де потрібні точність, оперативність, багатоплановий аналіз і прогноз. До таких галузей передусім можна віднести і діяльність Міністерства з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи.

До завдань Інформаційно-аналітичного управління МНС, яке має знати все про все, належить:

- оперативне забезпечення керівництва Міністерства інформа-цією щодо потенційних передумов виникнення НС та хід ліквідації їх

наслідків, здійснення інформаційної взаємодії з міністерствами, відомствами, іноземними та міжнародними структурами з метою забезпечення керівництва оперативною інформацією з питань НС;

- розроблення рекомендацій щодо планування та ор-ганізації заходів з попередження НС для територіальних структур МНС, інших центральних органів виконавчої влади;
- організація заходів щодо аналізу, впровадження, розвитку та забезпечення урядової інформаційно-аналітичної системи з питань надзвичайних ситуацій (УІАС НС), виконання робіт, пов'язаних з формуванням і реалізацією єдиної науково-технічної політики в галузі створення та впровадження сучасних інформаційних технологій тощо.

Управління забезпечує роботу Кризового центру, який є керівною ланкою в ланцюзі сегментів інформаційно-аналітичної системи з питань надзвичайних ситуацій.

У випадку НС у Кризовому центрі діють оперативні групи Міністерства; міжгалузеві оперативні групи; Урядова комісія з ліквідації НС, включаючи фахівців та експертів.

Кризовий центр, у технічному плані, у завершеному вигляді має становити територіально-розподільчий комплекс технічних засобів та систем програмного забезпечення. Для виконання функцій інформаційного аналізу в систему вводяться найрізноманітніші бази даних. Серед яких, зокрема: загальнодержавний класифікатор НС, загальнодержавний паспортний реєстр техногенне та екологічно небезпечних об'єктів, оперативної інформації про надзвичайні ситуації, що сталися, дані нормативно-правового забезпечення,

інформація щодо методик аналізу і прогнозування НС та їх наслідків, дані про забруднення довкілля внаслідок НС і т. ін.

Для інформаційного аналізу також передбачаються електронні картографічні дані з тематичними планами розміщення техногенне небезпечних об'єктів енергетики, зв'язку, хімічних підприємств, транспортних магістралей, нафто- і продуктопроводів, гідроспоруд, АЕС, великих промислових підприємств і т. ін. Крім того - характеристики стану техногенне та екологічно небезпечних об'єктів інженерних споруд, магістралей, а також інформація про надзвичайні ситуації, пов'язані з пожежами та екологією, метеоумовами, прогнозним станом водних ресурсів, газовими родовищами, зсувними зонами тощо.

В Інформаційно-аналітичному управлінні відкрито інформацій-ний сервер МНС. В Інтернеті доступ до нього мають усі бажаючі. Таким чином, журнали і газети, які видаються Міністерством, а також інформацію про надзвичайні ситуації за місяць, квартал, півроку, рік - сьогодні вже сканують 400 браузерів з п'ятдесяти країн світу. Наприклад, США більше користується нашою інформацією, аніж Росія. Вважаю, що сьогодні - це один з ефективних методів інформаційного прориву.

Нині проводиться велика робота з паспортизації потенційно небезпечних об'єктів та створення Державного реєстру аварійне небезпечних об'єктів.

Створені електронні карти для інформаційного аналізу і прогнозування ситу-ацій. Проте не таких масштабів, як це необхідно сьогодні для ефективної роботи. У Кризовому центрі є три масштаби таких карт (1:1 000 000, 1:4 000 000, 1:5 000 000).

На електронній карті можливо простежити всі дороги, залізниці, нафтопроводи, аміакопроводи, енергомережі, річки.

Останнім часом розпочата активна робота над створенням атласу карт надзвичайних ситуацій, що стались в областях. У ньому буде уміщено інформацію про потенційно небезпечні техногенні і природні об’єкти. Карти забезпечуватимуться описами і діаграмами. Картографічна основа залишатиметься стабільною, а інформаційна складова - заходи і т. ін. - буде змінюватись. На картах, зокрема, будуть позначені техногенне небезпечні об’єкти паливно-енергетичного комплексу, пожежонебезпечні об’єкти, хімічно- і вибухонебезпечні об’єкти та ін.

Потрібно класифікувати всю Україну - окремі карти відображатимуть стан екологічної безпеки в Україні, кількість НС техногенного та природного характеру на тій чи тій території, аварії на шахтах і водних об’єктах.

Завершивши роботу над картами із загальним відображенням техногенне небезпечних об’єктів і НС техногенного характеру, потрібно готувати карти за галузевим принципом. Транспорт, енергетика, хімічна, видобувна промисловість і т. ін.

Наступними будуть карти розташування потенційно небезпечних об’єктів природного характеру. Знову ж узагальнені дані про природні об’єкти підвищеної небезпеки. Потім - окремо про стихійно-метеорологічні явища (сніг, ожеледь, хуртовини, паводки, землетруси, селі, зсуви).

Над картою підтоплень уже працюють. Дещо використовується з того, що вже створено іншими відомствами, зокрема Екологічною лігою України та іншими державними і громадськими організаціями.

Загалом функції Управління значно ширші, пов’язані, в першу чергу, з оперативним інформаційним забезпеченням заходів, спрямованих на запобігання і ліквідацію НС, які, на жаль, ще часто виникають в Україні.

7 ЕКОЛОГІЯ

1. Програмне забезпечення еколого - статистичних досліджень.

Оперативна, якісна і точна обробка великих масивів статистичної інформації може бути виконана лише з використанням сучасних засобів обчислювальної техніки. Наявність потужних, надійних і разом з тим простих в експлуатації програмних продуктів статистичного аналізу звільняє дослідника від рутинних операцій, розширює сферу застосування статистичних методів в різних галузях людської діяльності, сприяє появі якісно нових можливостей статистичного аналізу і моделювання даних. Використання пакетів прикладних програм це єдиний реальний практичний інструмент розв'язування задач багатофакторного кореляційно-регресійного та аналізу в багатовимірному просторі.

Програмне забезпечення статистичних досліджень досить розвинуте. Сучасний ринок програмних продуктів пропонує різноманітні пакети програм для статистичної обробки даних. Всесвітньо відомі статистичні пакети для комплексної обробки даних: BMDP, SPSS, SAS, Systat, Minitab, S-Plus, Statgraphics Statistica та інші.

Використання згаданих пакетів програм дає змогу автоматизувати процес статистичного дослідження в таких напрямах:

- створення файлів даних і таблиць;
- групування даних;
- графічний аналіз даних;
- розрахунок варіаційних характеристик вибіркових сукупностей;
- побудова рядів розподілу;
- аналіз рядів динаміки і прогнозування їх майбутніх рівнів;

- кореляційно-регресійний аналіз;
- багатомірний аналіз.

З 1995 р. Світовим лідером на ринку статистичного програмного забезпечення визнається інтегрована система Statistical для Windows (версія 5.0), розроблена фірмою Stat Soft. Перша версія програми з'явилася у 1991р. для операційної системи MS-DOS і була новим напрямом розвитку статистичного програмного забезпечення. В ній реалізовано графічно-орієнтований підхід до статистичного аналізу даних, суть якого полягає в отриманні всебічного візуального представлення інформації на всіх етапах статистичної обробки даних.

Багатофункціональна, графічно орієнтована на обробку масових даних система Statistica відповідає основним стандартам Windows (динамічний обмін даними з іншими додатками, підтримка основних операцій з буфером обміну, робота в мережевому середовищі та інші).

Передусім це стандарти користувацького інтерфейсу — MDI, використання буфера-обміну, механізму динамічного зв'язку (DDE) з іншими додатками; система підтримує всі операції, реалізовані за допомогою методу Drag-and-Drop — «Перетягти та опустити», включаючи автозаповнення, інші.

Складніші процедури обробки даних у системі Stratgraphics виконує спеціалізований модуль Data Management — «Управління даними», а для обробки великих масивів даних або даних з довгими текстовими значеннями застосовують процедури Megafile Manager Data — «Менеджера метафайлів».

Система Stratgraphics працює з чотирма типами документів. Це:

- електронна таблиця Spreadsheet, призначена для введення і перетворення первинних даних;

- електронна таблиця Scrollsheet — для виведення результатів аналізу;
- графік — для візуалізації результатів обробки та аналізу даних;
- звіт — файл у формі RTF (розширений текстовий формат), в якому зберігається текстова, числова і графічна інформація.

Усі статистичні процедури системи розбито на окремі модулі, кожен з яких об'єднує групу логічно зв'язаних між собою статистичних методів і в рамках конкретної моделі забезпечує повний і всебічний аналіз закономірностей.

Наприклад, у модулі Basic Statistics/Table — «Основні статистики і таблиці» пропонується широкий вибір методів розвідувального статистичного аналізу:

- характеристики варіації і форми розподілу,
- групування та класифікації,
- таблиці дисперсійного аналізу Anova,
- всі види коефіцієнтів щільності зв'язку,
- критерії для тестування нормальності розподілу,
- критерії істотності зв'язку тощо.

Модуль Multiple Regression — «Множинна регресія» включає:

- вичерпний набір засобів множинної лінійної і нелінійної регресії,
- багатофакторного прогнозування,
- аналіз залишків і викидів,
- тестування гіпотез регресійного аналізу.

Модуль Time Series/Forecasting — «Часові ряди і прогнозування» об'єднує процедури аналізу закономірностей динаміки: тенденцій розвитку і коливань,

- різні методи згладжування рядів,
- описування трендів,
- описування сезонної декомпозиції,
- методи авторегрсійного аналізу, методи прогнозної екстраполяції.

Система Statistica включає модуль Anova/Manova — «Дисперсійний аналіз», увесь арсенал методів багатовимірного аналізу (кластерний, дискримінантний, факторний аналіз, факторне шкалювання, канонічні кореляції).

2. Комплексна оцінка екологічності виробництва.

Під комплексною оцінкою екологічності виробництва, далі ЕВ, розуміють висновок про рівень екологічності господарської діяльності з урахуванням чинника техногенної безпеки у взаємозв'язку з виробничими ресурсами, умовами і фінансово-економічними результатами господарської діяльності. Можна також сказати, що комплексна оцінка ЕВ підприємства являє собою його характеристику, отриману в результаті дослідження, і містить висновки про результати екологічної діяльності підприємства, галузі, регіону.

Функції комплексної оцінки. Комплексна оцінка ЕВ підприємства може бути:

- інструментом обліку, аналізу, планування і регулювання;
- показником еколого-економічного стану господарського об'єкта;

- критерієм порівняльної оцінки екологічності виробництва різних об'єктів;
- показником ефективності прийнятих управлінських рішень у сфері природокористування й охорони навколишнього середовища, а також повноти їх реалізації;
- основою вибору можливих варіантів розвитку екологізації виробництва.

Таким чином, оцінювання екологічної діяльності суб'єктів господарювання, ЕВ і стану соціально-еколого-економічної системи проводиться за одним показником (критерієм), який характеризує всі сторони функціонування об'єкта. Отримання комплексної оцінки екологічної діяльності підприємства та ЕВ на основі системи показників має елемент порівняння (як і комплексна оцінка господарської діяльності). Тобто вона (комплексна оцінка) по-суті виступає як порівняльна комплексна або рейтингова оцінка.

Вимоги до комплексної оцінки ЕВ. Комплексна оцінка ЕВ має задовольняти такі вимоги:

- виражати сутність виробничих та еколого-економічних відносин;
- охоплювати головні сторони виробничо-господарської та екологічної діяльності підприємства;
- використовувати обмежену кількості узагальнених еколого-економічних показників;
- бути еластичною - побічно визначати динаміку суспільне необхідних (повних) витрат у сфері природокористування і охорони навколишнього середовища;

- забезпечувати порівнянність показників у часі та просторі;
- вибір показників має визначатися метою регулювання природокористування.

Методологічна основа комплексної оцінки ЕВ. Методологічною основою оцінки складових ЕВ виступає індексний метод. За допомогою індексів (у межах від 0 до 1) характеризується наближення того чи іншого показника до необхідного (оптимального).

Етапи комплексної оцінки ЕВ. Процедура комплексної порівняльної оцінки ЕВ виконується у вигляді таких відносно самостійних етапів:

- поставлення цілей і завдань комплексної оцінки ЕВ, включаючи вибір підприємств і видів їх виробничо-економічної діяльності;
- обґрунтування та вибір системи еколого-економічних і фінансово-економічних показників;
- організація збирання вихідної інформації, розрахунку і оцінки окремих показників і вагових коефіцієнтів;
- вибір об'єкта як бази для порівняння;
- розроблення алгоритму і розрахунку комплексних показників ЕВ;
- перевірка адекватності комплексних узагальнених оцінок еколого-економічної ситуації;
- аналіз і використання порівняльних комплексних рейтингових оцінок у процесі прийняття управлінських рішень щодо екологізації промислового виробництва.

База порівняння при комплексній оцінці ЕВ. Реалізація методів порівняльної комплексної рейтингової оцінки передбачає наявність бази порівняння. В економічному аналізі використовуються поняття підрозділу-

еталона, підприємства-еталона або об'єкта-еталона. Ряд авторів пропонує використовувати як підприємство-еталон так зване абсолютне підприємство, у якому всі розглянуті показники мають найкраще значення серед даної сукупності підприємств галузі. У ряді випадків типовим об'єктом порівняння вважається об'єкт, значення показників якого дорівнюють середнім арифметичним або нормативним величинам досліджуваної сукупності підприємств.

ВИСНОВКИ

Виходячи з проведеного дослідження існуючих рішень і процесу розробки веб-додатків, були розроблені критерії для розробки фреймворка. Була виявлена необхідність створення максимально повного фреймворка для старту веб-додатки, всі ризики якого будуть спочатку знижені. Далі веб-додаток має розвиватися з уже обмеженого стандартними настройками стану відповідно до гнучкої методологією розробки Agile.

В рамках даної роботи був розроблений фреймворк для розробки захищених веб додатків реалізує захист бази даних, логування, хешування паролів, валідація даних запитів.

Рекомендації до використання:

Даний фреймворк є інструментом забезпечення безпеки веб додатки з самого старту розробки при мінімальній функціональності. Передбачається, що при подальшому розвитку додатки конфігурація безпеки буде змінюватися. А при реалізації бізнес функціональності будуть підтримуватися існуючі вимоги.

Перспективи подальшого розвитку проекту включають:

- інтеграцію з системами автоматичного сканування безпеки
- запобігання виникненню найбільш частих кейсів пов'язаних з проектуванням REST-сервісів

Також була проведена перевірка додатка створеного з використанням даного фреймворка за допомогою сканера вразливостей Wapiti.

БІБЛІОГРАФІЯ

1. Wikipedia - Software Framework [Електронний ресурс] // URL: https://en.wikipedia.org/wiki/Software_framework (Дата звернення: 23.11.19)
2. Positive Technologies - Уразливості веб-додатків 2018 [Електронний ресурс] // URL: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/Web-
3. Vulnerability-2018.pdf (Дата звернення: 23.11.19)
4. Spring Boot Docs [Електронний ресурс] // URL: https://docs.spring.io/spring-boot/docs/current/reference/html/bootfeatures-developing-auto-configuration.html (Дата звернення:23.11.19)
5. Inversion of Control Containers and Dependency Injection pattern [Електронний ресурс] // URL: https://martinfowler.com/articles/injection.html (Дата звернення: 23.11.19)
6. Spring Security Reference [Електронний ресурс] // URL: https://docs.spring.io/spring-security / site / docs / 5.0.0.RELEASE / reference / htmlsingle / (Дата звернення: 23.11.19)
7. Agile. Secure software development [Електронний ресурс] // URL: https://www.checkmarx.com/2017/04/20/six-steps-secure-softwaredevelopment-agile-era/ (Дата звернення: 23.11.19)
8. Управління вимогами безпеки в Agile проектах - Rohit Sethi [Електронний ресурс] // URL:http://agilerussia.ru/practices/security-requirements/ (Дата звернення:20.11.19)
9. Журнал школи IT-Менеджмент [Електронний ресурс] // URL:http://journal.itmane.ru/node/1572/ (Дата звернення: 20.11.19)

10. OWASP - Top Ten Proactive Controls [Електронний ресурс] // URL:http://master.cmc.msu.ru/files/OWASP_Top_Ten_Proactive_Controls_ v2_rus.pdf (Дата звернення: 23.11.19)
11. Technical University of Crete. Password-Hashing. : Chania, 27 June 2017
12. Wikipedia - Blowfish [Електронний ресурс] // URL:https://ru.wikipedia.org/wiki/Blowfish (Дата звернення: 23.11.19)
13. Habrahabr - SQL vs ORM [Електронний ресурс] // URL: https://habrahabr.ru/company/pgdayrussia/blog/328690/ (Дата звернення: 23.11.19)
14. Мартін Фаулер. OrmHate [Електронний ресурс] // URL: https://martinfowler.com/bliki/OrmHate.html (Дата звернення:23.11.19)
15. Тест і порівняння ефективності WAF [Електронний ресурс] //URL: https://www.anti-malware.ru/compare/web-application-firewall(Дата звернення: 23.11.19)
16. OWASP - Fuzzing [Електронний ресурс] // URL: https://www.owasp.org/index.php/Fuzzing (Дата звернення:23.11.19)
17. Fuzzing: the Past, the Present and the Future [Електронний ресурс] // URL: http://actes.sstic.org/SSTIC09/Fuzzing-the_Pastthe_Present_and_the_Future/SSTIC09-article-A-Takanen-Fuzzingthe_Past-the_Present_and_the_Future.pdf (Дата звернення: 23.01.18)
18. Крейг Воллс. Spring in Action. ДМК Прес. 2015 17. OWASP Guide Project [Електронний ресурс] //

URL:https://www.owasp.org/index.php/OWASP_Guide_Project (Дата звернення: 23.11.19)

19. OWASP - Logging Cheat Sheet [Електронний ресурс] // https://www.owasp.org/index.php/Logging_Cheat_Sheet (Дата звернення: 23.11.19)
20. Business logic in a web application [Електронний ресурс] // URL: https://www.quora.com/Where-should-we-add-business-logic-in-aweb-application-on-client-side- or-server-side (Дата звернення:23.11.19)
21. Scrumи kanban:выжимаем максимум [Електронний ресурс] http://scrum.org.ua / (Дата звернення:23.11.19)
22. Стів Макконелл. Досконалий код, 2-е видання. СанктПетербург: видавництво "Пітер", 2012
23. Мартін Фаулер. UML Основи, 3-е видання. Санкт-Петербург: видавництво "СімволПлюс", 2005.
24. [Електронний ресурс] https://ua.trud.com/ua/salary/2/67664.html side (Дата звернення:23.11.19)
25. [Електронний ресурс] https://www.pentestit.ru/site-security-audit/ side (Дата звернення:23.11.19)
26. Петренко А.І. ОСНОВИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ ОБ"ЄКТІВ ТА СИСТЕМ (Конспект лекцій) / Петренко А.І. – К. : Видавництво ВМУРОЛ «Україна», 2002. – 211 с.
27. Larisa Globa. Modified model driven software development / Larisa Globa.-"Polish J. of Environ. Stud ", Vol. 18, No. 4A (2009), pp.39-43.

28. Лаврищева Е.М. Методы и средства инженерии программного обеспечения [учебник, Московский физико-технический институт (государственный университет)] / Лаврищева Е.М. , Петрухин В.А.. – Москва, 2006. – 304 с.
29. Yordon E. Structured Design [2nd edition] / Yordon E., Larry L. Constantine. - New York: Yordon Press, 1984. - 700 p.
30. Yourdon E. Modern Structured Analysis / Yourdon E. - Prentice-Hall, 1989. - 672 p.
31. Майерс Г. Надежность программного обеспечения / Пер. с англ. Ю.Ю.Галимова; Под ред. В.Ш.Кауфман. –М.: Мир, 1980. – 360 с.
32. Jackson M. Software requirement & specifications / Jackson M. –Wokingham, England: Addison–Wesley, ACM Press Books, 1995. –228 p.
33. Orr K. Structured system development / Orr K. - NY: Yourdon Press, 1971.
34. Schmidt D. C. Model-driven engineering, / Schmidt D. C. - IEEE Computer, vol. 39, February 2006. - pp. 25-31.
35. Per Kroll. Rational Unified Process Made Easy-A Practitioner's Guide to the RUP / Per Kroll, Philippe Kruchten. - Addison-Wesley, 2003. – 464 p.
36. Oscar Pastor. Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling / Oscar Pastor, Juan Carlos Molina. – Berlin Heilelberg: Springer, 2007. - 302 p.