Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

студента Шаталова Владислава Вадимовича
академічної групи 125м-21-1
спеціальності 125 Кібербезпека
спеціалізації
за освітньо-професійною програмою Кібербезпека
на тему Методи підвищення рівня захисту адміністративної панелі вебдодатків на Django

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		Рейтинговою	**інституційною**	
кваліфікаційної роботи	Сафаров О.О.			
розділів:	Сафаров О.О.			
спеціальний	Сафаров О.О.			
економічний	Пілова Д.П.			

Рецензент				

Нормоконтролер	Мєшков В.І.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.
«___» _____ 20__ року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту ***Шаталову Владиславу Вадимовичу*** **академічної групи** ***125м-21-1***
(прізвище ім'я по-батькові) (шифр)

спеціальності ***125 Кібербезпека***
(код і назва спеціальності)

на тему Методи підвищення рівня захисту адміністративної панелі вебдодатків на Django

затверджену наказом ректора НТУ «Дніпровська політехніка»
від 31.10.2022 № 1200-с

Розділ	Зміст	Термін виконання
Розділ 1	Стан питання. Постановка задачі	01.11.22-15.11.22
Розділ 2	Спеціальна частина	16.11.22-30.11.22
Розділ 3	Економічний розділ	01.12.22-15.12.22

Завдання виконано _____ _____
(підпис керівника) (прізвище, ініціали)

Дата видачі: 05.09.2022

Дата подання до екзаменаційної комісії: 16.12.2022

Прийнято до виконання _____ _____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 65 с., 17 рис., 1 табл., 1 додатків, 9 джерел.

Об`єктом дослідження є результати аналізу проблем та загроз адміністративної панелі вебдодатків на Django.

Мета роботи - дослідження результатів аналізу проблем та загроз адміністративної панелі вебдодатків на Django, аналіз підходів щодо забезпечення захищеності автентифікації адміністративної панелі вебдодатків на Django та розробка засобів, спрямованих на досягнення захисту автентифікації за допомогою двофакторної автентифікації.

Методом роботи було забезпечення безпечної взаємодії в рамках Django вебфреймворку, а саме використання Django Admin Panel. Розроблено програмний засіб, що реалізує зазначений підхід, та підтримує весь необхідний для безпечної взаємодії функціонал, а саме: фіксація спроби автентифікації зі збереженням двофакторного токену, надсилання токену двофакторної автентифікації користувачеві, встановлення сеансового ключа за класичним протоколом Django; переадресація на головну сторінку адмін-панелі.

Практичне застосування результатів даної роботи може бути використано для безпечної автентифікації у Django Admin Panel у будь-якому Django вебдодатку саме за допомогою двофакторної автентифікації.

DJANGO, DJANGO WEB APPLICATION, TWO-FACTOR AUTHENTICATION, PYPI, PYTHON DJANGO_ADMIN_PROTECT.

ABSTRACT

Explanatory note: 65 p., 17 fig., 1 table, 1 appendices, 9 sources.

The object of the research is the results of the analysis of problems and threats of the administrative panel of web applications on Django.

The purpose of the work is to study the results of the analysis of problems and threats of the administrative panel of web applications on Django, the analysis of approaches to ensure the security of authentication of the administrative panel of web applications on Django, and the development of means aimed at achieving authentication protection using two-factor authentication.

The working method was to ensure secure interaction within the Django web framework, namely the use of the Django Admin Panel. A software tool has been developed that implements the specified approach and supports all the functionality necessary for secure interaction, namely: recording an authentication attempt with saving a two-factor token, sending a two-factor authentication token to the user, setting a session key according to the classic Django protocol; redirection to the main page of the admin panel.

Practical application of the results of this work can be used for secure authentication to the Django Admin Panel in any Django web application using two-factor authentication.

DJANGO, DJANGO WEB APPLICATION, TWO-FACTOR AUTHENTICATION, PYPI, PYTHON DJANGO_ADMIN_PROTECT.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

2FA - Two Factor Authentication;

DAP - Django Admin Panel;

OTP - One Time Password;

PYPI - Python Package Index;

SQL - Structured Query Language;

SWD - Selenium Web Driver;

XSS - Cross Site Scripting.

ЗМІСТ

ВСТУП

Інформаційні технології є невід'ємною частиною ведення сучасного бізнесу для вирішення багатьох завдань, які лягають на вебсервіси, що надають зручне, віддалене та безпечне використання для інтернет-користувачів. Так як сучасні інформаційні технології розвиваються з великою швидкістю, в наявності у розробників з'являються нові, безпечні та швидко розроблювані фреймворки.

На початкових етапах вебфреймворки створювали і використовували великі комерційні компанії, але з часом усі переваги даної технології оцінили й інші учасники ринку та вебфреймворки стали розроблятися для загального доступу та випередили у своїх потужностях комерційні аналоги та попередні CMS системи.

В наш час вимоги користувачів до вебсервісів стрімко зростає і вебфреймворки намагаються встигати за вимогами, при цьому часто забувають і не приділяють уваги важливим аспектам, наприклад, кібербезпеці.

У цій роботі буде розглянуто захист вебфреймворку Django, який на відміну від своїх конкурентів фреймворків на мові програмування Java/TypeScript, таких як ExpressJS, NodeJs, NestJs, які не надають якого-небудь захисту "з коробки", надає певні рішення, які допомагають закрити більшість кібер небезпек.

Django має багаторівневий захист від більшості атак, які будуть висвітлені в першому розділі.

Другий розділ буде присвячений розробці додатку для захисту Django Admin Panel як способу керування вебсервісом та легкому доступу до чутливих даних.

У третьому розділі йдеться мова про економічну складову кваліфікаційної роботи.

У висновку підводяться підсумки дослідження, формуються остаточні висновки по даній темі.

РОЗДІЛ 1 СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття бекенд вебфреймворку

Бекенд вебфреймворк - програмне середовище - це основа, на якій розробники можуть створювати програми більш швидким та стандартизованим способом.

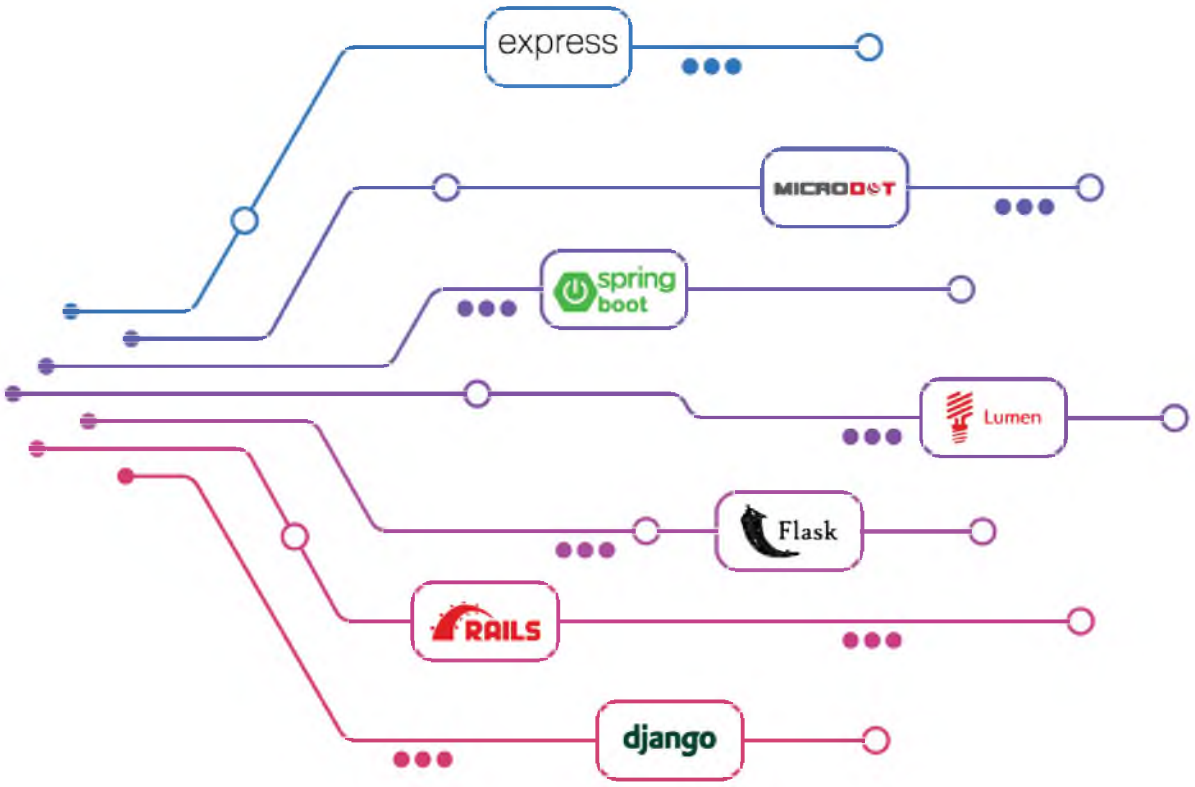

Рисунок 1.1. Вебфреймворки

Основною метою фреймворку є автоматизація накладних витрат, пов'язаних із діяльністю з розробки програмного забезпечення. Основні переваги використання фреймворку для розробки:

- Збереження часу
- Масштабованість
- Надійність
- Безпека
- Інтеграція

Крім того, більшість фреймворків мають відкритий вихідний код. [1]

1.2 Поняття Django бекенд вебфреймворку

Django - високорівневий відкритий Python-фреймворк (програмний каркас) для розробки вебсервісів. Названо його було на честь джазмена Джанго Рейнхардта (відповідно до музичних смаків одного із засновників проекту). Початкова розробка Django, як засобу для роботи новинних ресурсів, досить сильно позначилася на його архітектурі: він надає ряд засобів, які допомагають у швидкій розробці.

Django - це провідне серверне середовище з відкритим кодом, засноване на мові програмування Python. Він слідує шаблону контролера представлення моделі (MVC).

Django підходить для розробки складних та багатофункціональних вебсайтів, керованих базами даних, і є одним із найпростіших серверних фреймворків. Django вважається одним із найкращих серверних фреймворків для веброзробки.

Цей серверний фреймворк забезпечує оптимальне підключення, скорочення коду, більшу можливість повторного використання та швидшу розробку. Він використовує Python для всіх операцій у Django і надає додатковий інтерфейс адміністратора, який допомагає створювати, читати, оновлювати та видаляти операції. Django використовується багатьма відомими вебсайтами, такими як Disqus, Instagram, Pinterest, Coursera, Mozilla, The Washington Times та інші. [1]

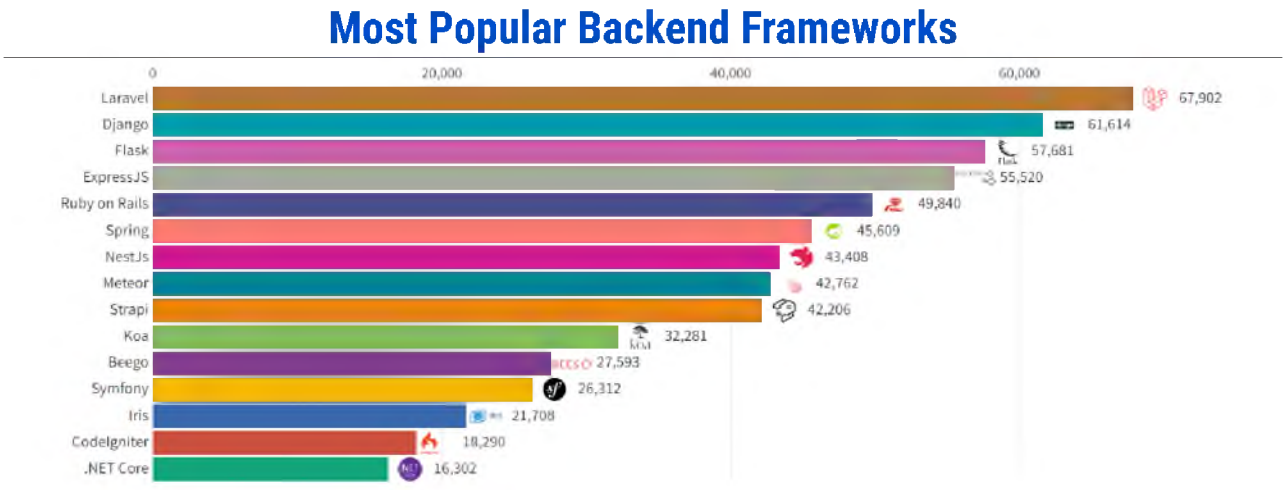

Рисунок 1.3 Топ бекенд фреймворків станом на липень 2022 року за данними statisticsanddata.org

Так, наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, в Django є вбудований модуль для керування вмістом, який можна включити в будь-який сайт, зроблений на Django, і який може керувати відразу декількома сайтами на одному сервері. Адміністративний модуль дозволяє створювати, змінювати і вилучати будь-які об'єкти наповнення сайту, протоколюючи всі дії, а також надає інтерфейс для управління користувачами і групами (з призначенням прав).

Переваги Django

1) Швидка структура Django проста у використанні, а структура з низькою кривою навчання створена для того, щоб допомогти розробникам прискорити весь процес розробки від початку до кінця. Це вважається простим бекендом для вебсайтів.

2) Багатофункціональний Django надає широкий спектр функцій, які допомагають користувачам виконувати деякі загальні вимоги веброзробки. Це допомагає в таких завданнях, як автентифікація користувачів, карти сайту, адміністрування контенту та багато іншого.

3) Оптимальна безпека. Django – це безпечна платформа, яка допомагає користувачам запобігти ряду проблем із безпекою, включаючи міжсайтові сценарії, клікджекінг, впровадження SQL та підробку запитів. Він надає систему автентифікації користувачів, яка дозволяє користувачам безпечно зберігати паролі та облікові записи та керувати ними.

4) Висока масштабованість. Django пропонує своїм користувачам високий рівень масштабованості. Ось чому багато провідних вебсайтів світу покладаються на нього, щоб легко задовольнити свої високі операційні вимоги.

5) Django — це середовище, яке можна використовувати для розробки широкого спектру типів додатків. Деякі з них включають програми для соціальних мереж, системи керування контентом та обчислювальні платформи. [1]

1.3 Основні заходи з кібербезпеки Django вебдодатків

Захист даних - важлива частина проектування будь-якого вебсайту. Безпека сайту вимагає пильності у всіх аспектах дизайну та використання сайту.
Інтернет – небезпечне місце! Регулярно стає відомою інформація про те, що вебсайти стають недоступними через атаки, наприклад: відмовлено в обслуговуванні, або відображення зміненої (і часто пошкодженої) інформації на їхніх сторінках. В інших випадках мільйони паролів, адрес електронної пошти та дані кредитних карток ставали загальнодоступними, наражаючи користувачів вебсайту на особисте збентеження або на фінансові ризики.

Мета веббезпеки полягає у запобіганні цим (або іншим) видам атак. Більш формальним визначенням веббезпеки є способи захисту вебсайтів від несанкціонованого доступу, використання, зміни, знищення або порушення роботи [2].

Для ефективної безпеки вебсайту необхідно приділяти особливу увагу до розробки всього вебсайту: до вашого вебдодатку, конфігурації вебсервера, при написанні політики створення та оновлення паролів, а також коду на стороні клієнта. Хоча все це звучить дуже зловісно, хороша новина полягає в тому, що

фреймворк Django забезпечить «за замовчуванням» надійні та продумані механізми захисту від ряду найбільш поширених атак, таких як:

1) Захист від міжсайтового скриптингу (XSS)
2) Захист від підробки міжсайтових запитів (CSRF)
3) Захист від SQL-ін'єкцій
4) Захист від клікджекінгу
5) Безпека сеансів

1.3.1 Захист від межсайтового скриптингу (XSS)

XSS-атаки дозволяють користувачу впроваджувати сценарії на стороні клієнта у браузерах інших користувачів. Зазвичай це досягається шляхом збереження шкідливих сценаріїв у базі даних, звідки вони будуть вилучатися та відображатися для інших користувачів, або шляхом надання користувачам можливості клацнути посилання, яке призведе до виконання JavaScript-коду зловмисника у браузері користувача. Проте, XSS-атаки можуть виходити з будь-якого ненадійного джерела даних, такого як файли cookie або вебслужби, якщо дані недостатньо очищені перед додаванням на сторінку.

Основна мета міжсайтового скриптингу – крадіжка cookies користувачів за допомогою вбудованого на сервері скрипту з подальшою вибіркою необхідних даних та використанням їх для наступних атак та зломів. Зловмисник здійснює атаку на користувачів не безпосередньо, а з використанням вразливостей вебсайту, який відвідують жертви, та впроваджує спеціальний JavaScript. У браузері в користувачів цей код відображається як єдина частина сайту. При цьому відвідуваний ресурс є співучасником XSS-атаки.

Якщо порівнювати з SQL-ін'єкціями, XSS безпечний для сервера, але несе загрозу для користувачів зараженого ресурсу або сторінки. Однак, якщо до зловмисника потрапляють cookies адміністратора, можна отримати доступ до панелі керування сайтом та його вмісту.

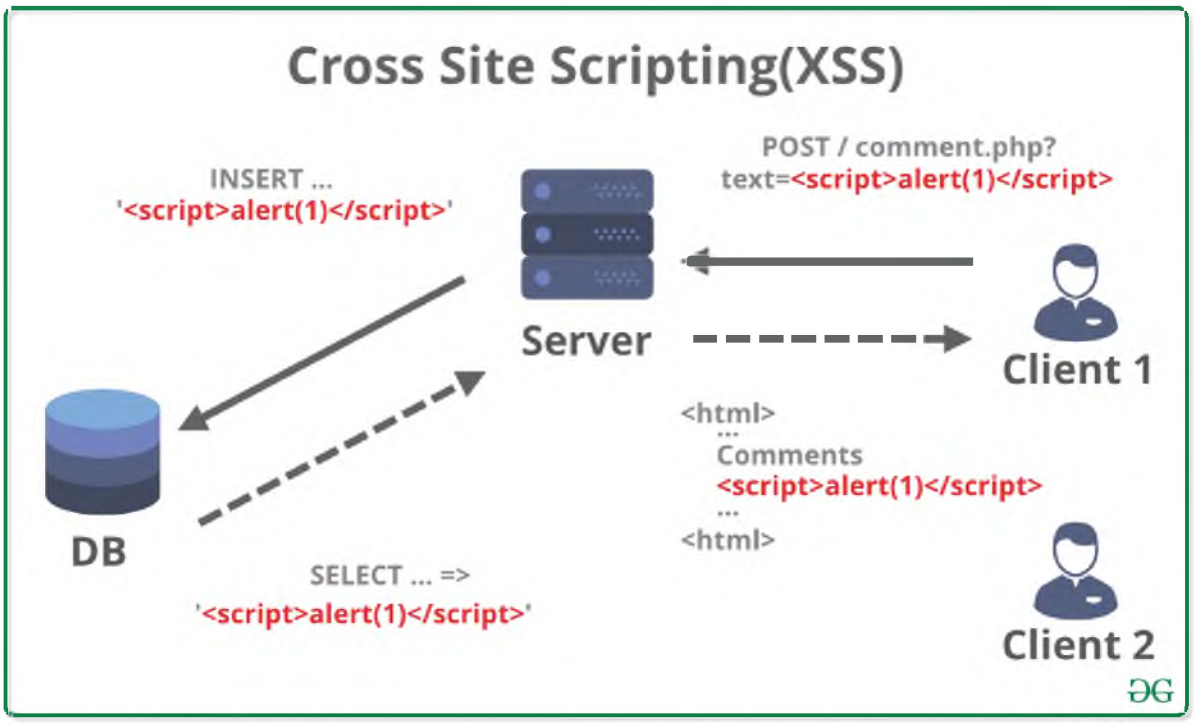

Рисунок 1.4 [5] Загальна схема XSS атаки

Запуск шкідливого коду JavaScript можливий тільки в браузері жертви, тому сайт, на який користувач зайде, повинен мати вразливість до XSS. Для атаки зловмисник спочатку перевіряє ресурси на наявність вразливостей через XSS, використовуючи автоматизовані скрипти або ручний режим пошуку. Зазвичай це стандартні форми, які можуть надсилати та приймати запити (коментарі, пошук, зворотний зв'язок).

Проводиться повний збір сторінок із формами введення, і кожна сканується на наявність вразливостей. Наприклад, є сторінка "Пошук" на сайті. Для перевірки вразливості XSS достатньо ввести запит:

<script>alert(document.cookie)</script>

Якщо на екрані з'явиться повідомлення - виявлено пролом у безпеці. В іншому випадку система відобразить сторінку з результатами пошуку. Найчастіше XSS-уразливості перевіряються у браузері Internet Explorer.

Ще один можливий варіант пошуку – використання сторінок, які опрацьовують GET-запити. Допустимо, є посилання виду:

https://site.com/catalog?p=8

В адресному рядку замість ідентифікатора (8) додається скрипт – "<script>alert(document.cookie)</script>, в результаті чого отримується посилання такого виду:

https://site.com/catalog?p
="><script>alert(document.cookie)</script>

Якщо сторінка має вразливість XSS, на екрані з'явиться повідомлення такого самого плану, як у першому випадку [9].

Використання шаблонів Django захищає від більшості XSS-атак. Шаблони Django екранують певні символи, особливо небезпечні для HTML. Хоча це захищає користувачів від більшості зловмисних дій, це не зовсім надійно [3].

1.3.2 Захист від підробки міжсайтових запитів (CSRF)

Атаки CSRF дозволяють зловмиснику виконувати дії з використанням облікових даних іншого користувача без відома чи згоди цього користувача.

Наприклад, користувач заходить на шахрайську вебсторінку, з якої від його особи надсилається команда на інший сервер. Браузер пам'ятає логін та пароль відвідувача цього сайту, тому операція здійснюється без його відома. Шахраї спеціально надсилають такі команди, які не вимагають підтвердження з боку користувача, або ж підробляють його відповідь за допомогою скрипту.

Щоб змусити користувача зайти на шкідливий сайт, зловмисники можуть надіслати посилання на нього у листі на електронну пошту. А щоб вона не виглядала підозріло, її зовнішній вигляд змінюють за допомогою скорочувача посилань.

Шахраї можуть за допомогою CSRF-атаки проводити переказ грошей з карт, вимагати скидання пароля до облікового запису, відновлювати доступ до сервісів. Браузери не можуть розпізнати обман, тому що вони звертаються до файлів cookie і вважають ненавмисний запит справжнім - для них це той самий користувач. Тому на міжсайтову атаку піддаються будь-які вебресурси, які використовують файли cookie, браузерну автентифікацію, клієнтські сертифікати авторизації.

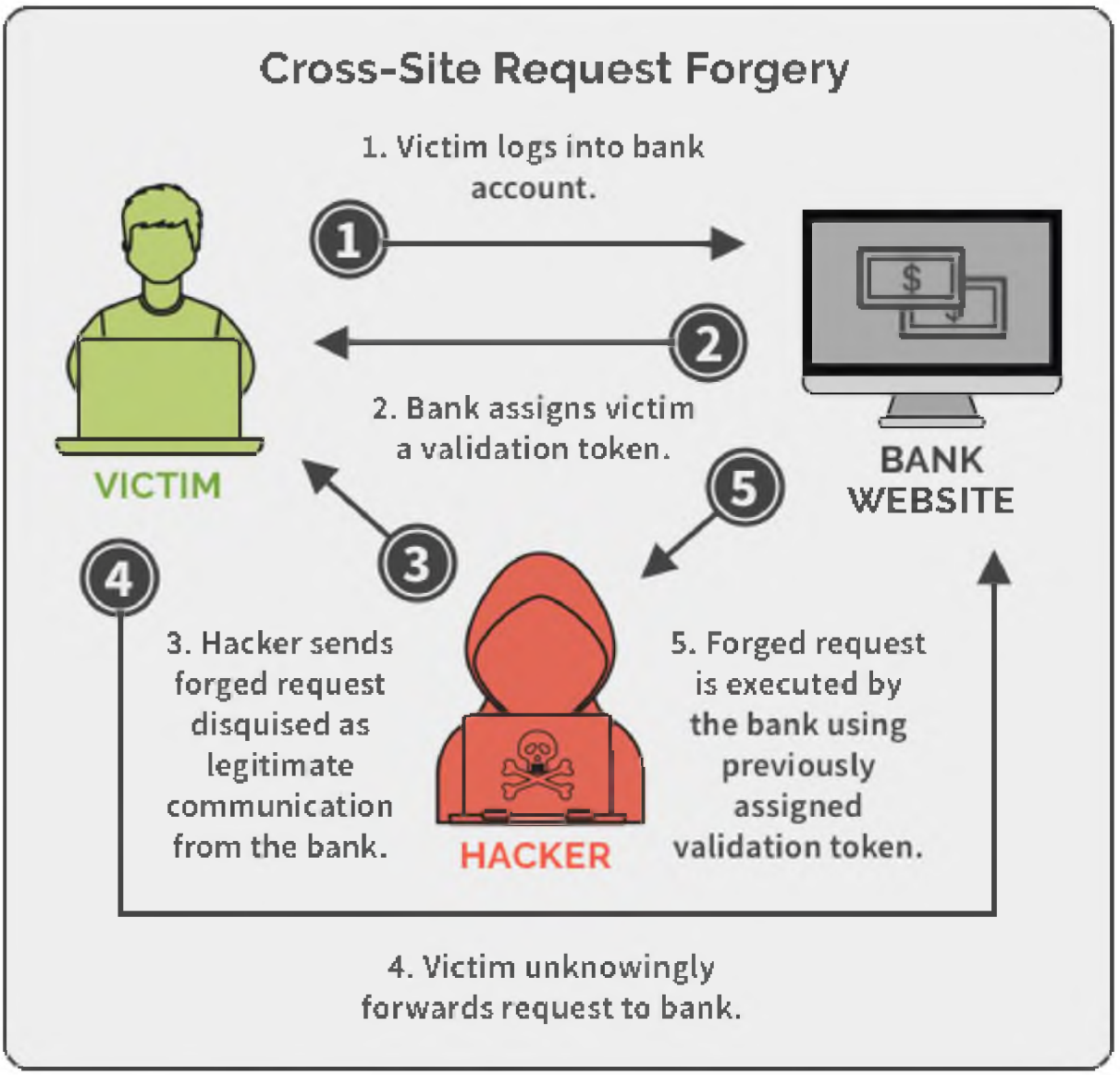

Рисунок 1.5 [6] Загальна схема CSRF атаки

Захист CSRF працює шляхом перевірки секрету у кожному POST-запиті. Це гарантує, що зловмисник не зможе просто «повторити» форму POST на ваш вебсайт і змусити іншого користувача, який увійшов до системи, ненавмисно надіслати цю форму. Зловмисник повинен знати секрет, який залежить від користувача (з використанням файлу cookie).

Django має вбудований захист від більшості типів CSRF-атак. Проте, як і будь-якого методу пом'якшення наслідків, існують обмеження. Наприклад, можна вимкнути модуль CSRF глобально або для певних уявлень [3].

1.3.3 Захист від SQL-ін'єкцій

SQL-ін'єкції (SQL injections, SQLi) — найдосвідченіший і найпростіший для розуміння тип атаки на вебсайт або вебдодаток. Проте він дивним чином залишається дуже поширеним і в наші дні. Організація OWASP (Open Web Application Security Project) згадує SQL-ін'єкції у своєму документі OWASP Top 10 2017 як загрозу номер один для безпеки вебдодатків, і навряд чи ситуація сильно змінилася за п'ять років.

SQL-ін'єкція - це тип атаки, коли зловмисник може виконати довільний код SQL у базі даних. Це може призвести до видалення записів або витоку даних.

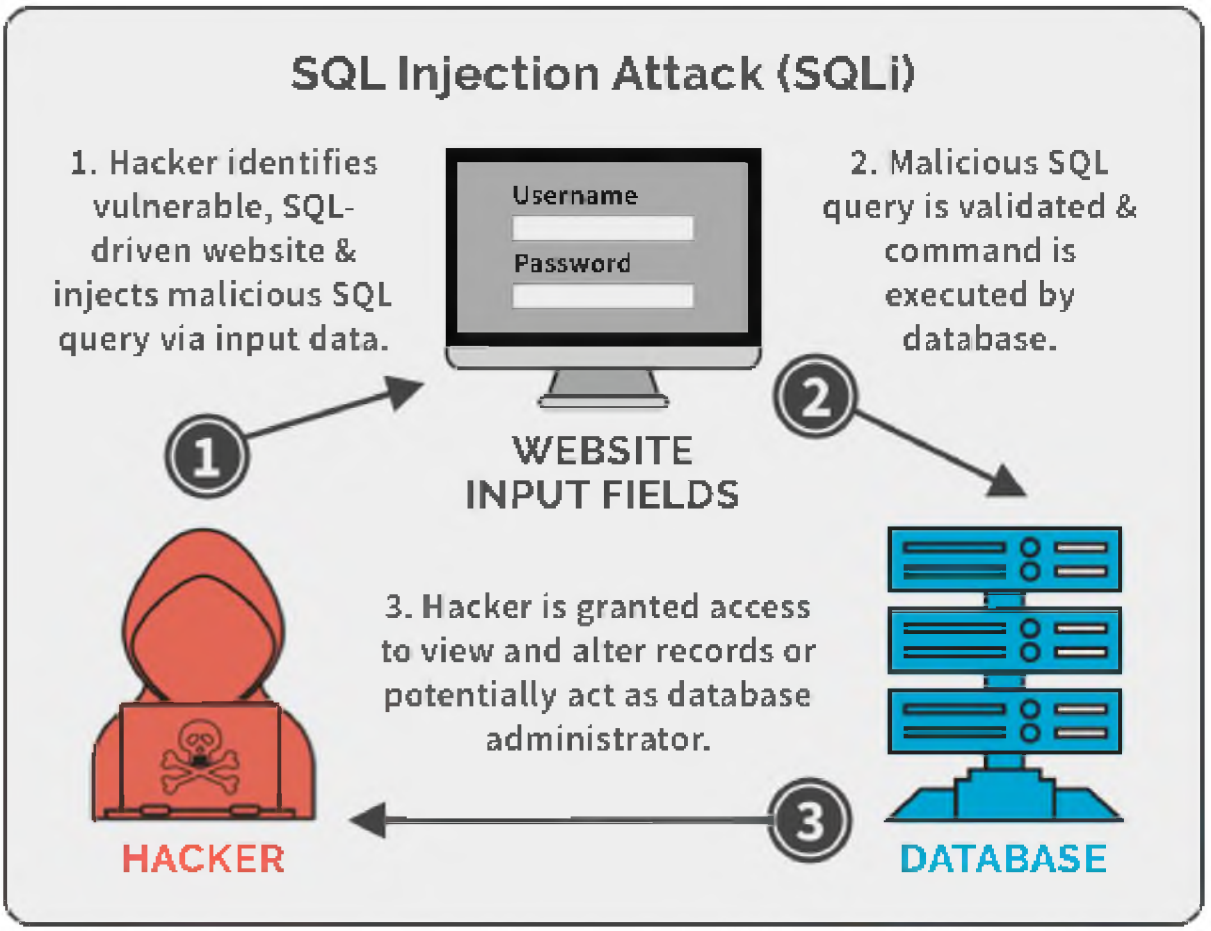

Рисунок 1.6 [7] Загальна схема атаки шляхом SQL-ін'єкцій

Наприклад, запит може надсилати облікові дані користувача через вебформу для доступу до сайту. Зазвичай такі вебформи налаштовані таким чином, щоб приймати лише певні типи даних, такі як ім'я користувача та пароль. Введена інформація звіряється з базою даних. Якщо все збігається, користувач може увійти на сайт. А якщо ні – у доступі буде відмовлено.

Ситуація небезпечна тим, що більшість вебформ не мають механізмів, які б унеможливлювали введення додаткової інформації в поле. Це дає зловмисникам можливість передати до бази даних власні запити через поля введення форми. Вони можуть використовувати цю вразливість у різних злочинних цілях, починаючи з крадіжки конфіденційних даних та закінчуючи маніпулюванням відомостями в базі.

Набір запитів Django захищений від SQL-ін'єкцій, оскільки їх запити створюються за допомогою параметризації запитів. Код запиту SQL визначається

окремо від параметрів запиту. Оскільки параметри можуть бути надані користувачем, отже, можуть містити у собі небезпеку, вони екрануються базовим драйвером бази даних.

Django також дає розробникам можливість писати необроблені запити або виконувати користувальницький SQL. Ці можливості слід використовувати з обережністю [3].

1.3.4 Захист від клікджекінгу

Клікджекінг - це тип атаки, при якому шкідливий сайт укладає інший сайт у кадр. Ця атака може призвести до того, що користувач, який нічого не підозрює, буде обманом змушений виконувати не навмисні дії на цільовому сайті.

Мета клікджекінгу може бути будь-якою – від більш-менш невинної накрутки лайків у соціальних мережах або передплатників до прихованого отримання персональних даних, здійснення покупок за чужий рахунок тощо. даних із профілю.

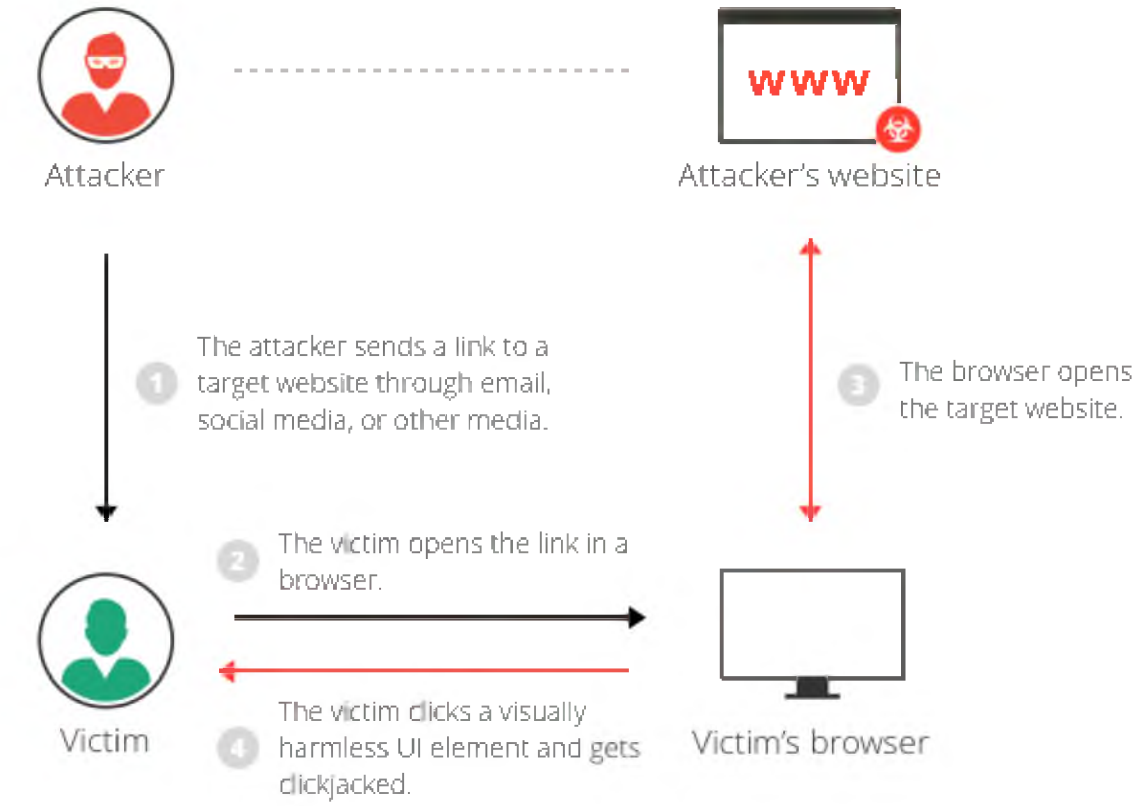

Рисунок 1.7 [8] Загальна схема клікджекінг-атаки

Django містить захист від клікджекінгу, який у підтримуючому браузері може запобігти відображенню сайту всередині кадру. Можна вимкнути захист для кожного представлення або налаштувати точне значення заголовка X-Frame-Options middleware.

Проміжне програмне забезпечення настійно рекомендується для будь-якого сайту, якому не потрібно, щоб його сторінки були укладені у кадр сторонніми сайтами, або необхідно дозволити це тільки для невеликої частини сайту [3].

1.3.5 Безпека сеансів

Django забезпечує повну підтримку анонімних сеансів. Платформа сеансу дозволяє зберігати та витягувати довільні дані для кожного відвідувача сайту. Він зберігає дані на стороні сервера та абстрагує відправлення та отримання файлів cookie. Файли cookie містять ідентифікатор сеансу, а не дані [3]. Дані зберігаються

в базі даних у зашифрованому вигляді із секретним ключем додатка. Без секретного ключа розшифрувати дані неможливо. Кожен Django додаток має свій таємний ключ.

1.4 Автентифікація та авторизація у Django вебдодатках

Django надає систему автентифікації та авторизації користувача, реалізовану на основі фреймворку роботи з сесіями. Система аутентифікації та авторизації дозволяє перевіряти облікові дані користувачів та визначати які дії який користувач може виконувати. Цей фреймворк включає вбудовані моделі для Користувачів і Груп (основний спосіб застосування прав доступу для більш ніж одного користувача), безпосередньо саму систему прав доступу, які визначають чи може користувач виконати завдання, з якою формою та відображенням для авторизованих користувачів, а також отримати доступ до контенту з обмеженим доступом.

Django також підтримує інші способи автентифікації користувачів, такі як:

- Token Authentication - ця схема автентифікації використовує просту схему автентифікації HTTP на основі токенів. Перевірка справжності за допомогою токена підходить для налаштувань клієнт-сервер, таких як власні настільні та мобільні клієнти.
- Basic Authentication - ця схема автентифікації використовує базову автентифікацію HTTP, підписану з використанням імені користувача та пароля. Звичайна автентифікація зазвичай підходить лише тестування.
- Remote User Authentication - ця схема автентифікації дозволяє делегувати автентифікацію вебсерверу, який встановлює REMOTE_USER змінну середовище.
- JSON Web Token Authentication - досить новий стандарт, який можна використовувати для аутентифікації на основі токенів. На відміну від вбудованої схеми Token Authentication, JWT Authentication не вимагає використання бази даних для перевірки токена.

- OAuth2 - стандарт який допомагає вам автентифікуватися за допомогою основних постачальників соціальних мереж oauth2, таких як Facebook, Google, Twitter, Orcid та інші. Він генерує токени у форматі JWT із простим налаштуванням.

1.5 Висновок

У першому розділі було розглянуто Django бекенд фреймворк для створення вебсервісів. Також були розглянуті основні вектори кібератак та методи захисту від них, передбачені цим фреймворком. Виходячи з вищевикладеного, можна встановити, що дана технологія є добре захищеним програмним забезпеченням, готовим покрити більшість кіберзагроз.

Однак, варто зазначити, що жодна система не гарантує повного захисту. Навіть використовуючи сучасні, добре захищені технології можна знайти серйозні вразливості. Одну з таких вразливостей було знайдено, виправлено та буде детально розібрано у другому розділі.

РОЗДІЛ 2 СПЕЦІАЛЬНА ЧАСТИНА

2.1 Аналіз вразливостей Django

2.1.1 Django Admin Panel

Однією з найпотужніших елементів Django є автоматичний інтерфейс адміністратора. Він зчитує метадані моделей, щоб надати швидкий, орієнтований модельний інтерфейс, у якому довірені користувачі можуть керувати контентом сайту. Рекомендоване використання адміністратором обмежено внутрішнім інструментом управління [4].

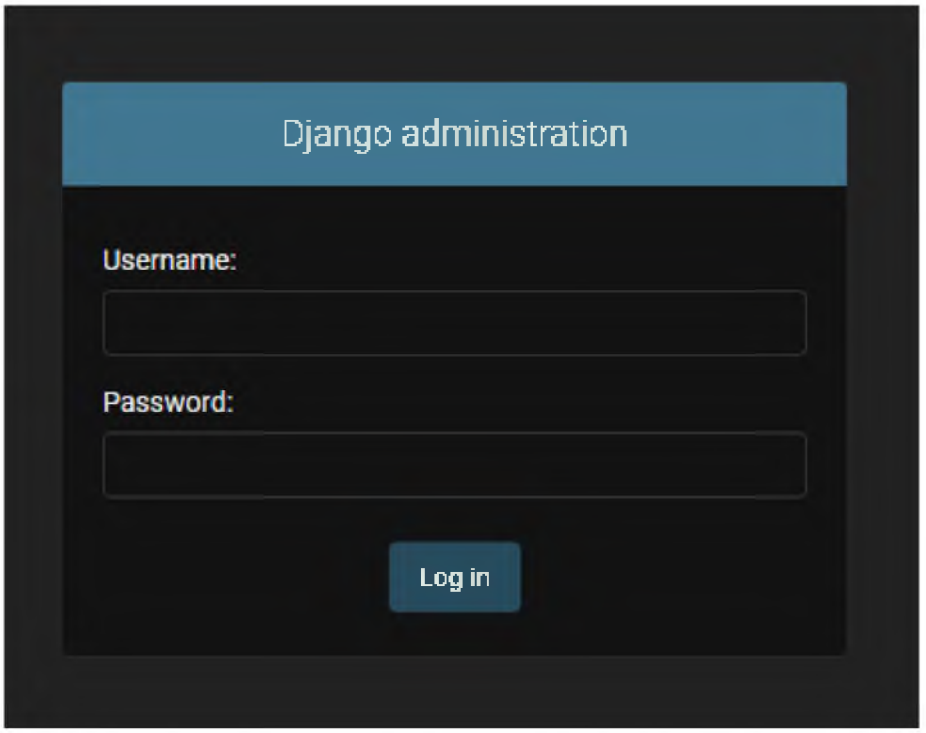

Рисунок 2.1 Стандартна форма для входу в адмін-панель

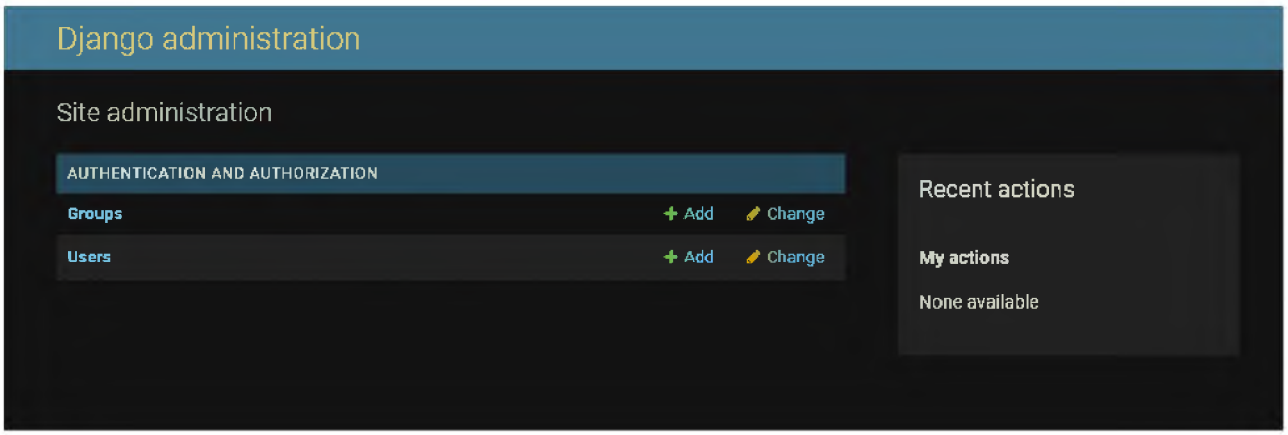

Рисунок 2.2 Базове уявлення Django Admin Panel

2.1.2 Об'єкт дослідження

Django Admin Panel є стратегічно важливим напрямом для захисту від кібератак, проте варто зазначити, що захищена вона тільки від CSRF атак. Але CSRF захист можна легко обійти, наприклад використовуючи ботів з Selenium Web Driver.

У разі несанкціонованого проникнення, атакуючий може отримати доступ до чутливих даних, змінити контент сайту, керуючи записами в базі даних, скомпрометувати дані користувачів сайту наражаючи їх на фінансові ризики.

Враховуючи сказане вище, об'єктом дослідження буде система автентифікації авторизації до даного застосунку.

2.1.3 Selenium WebDriver

Selenium WebDriver - інструмент для автоматизації дій веббраузера. У більшості випадків використовується для тестування Web-додатків, але цим не обмежується. Зокрема, він може бути використаний для вирішення рутинних завдань адміністрування сайту або регулярного отримання даних із різних джерел (сайтів). Webdriver - це популярний інструмент управління браузером, який найближче імітує дії користувача, та підтримує багато мов програмування (C#, Java, JavaScript, PHP, Python, Ruby, R, Perl та ін.)

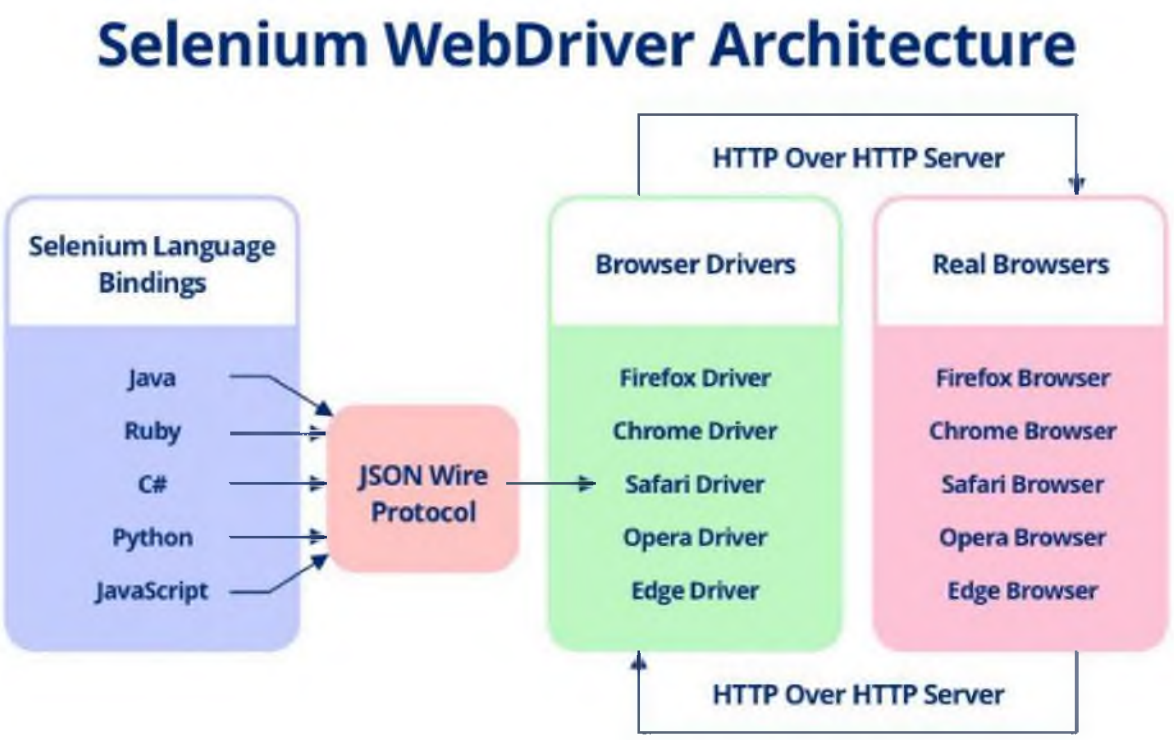

Рисунок 2.3 Архитектура Selenium WebDriver

Варто зазначити, що Selenium WebDriver можна використовувати для кібератак. Оскільки WebDriver повністю імітує дії користувача, хакери можуть використовувати цей інструмент для атаки методом брутфорсу.

2.1.4 Брутфорс

Брутфорсом називається метод злому облікових записів шляхом добору паролів до них. Термін утворений від англомовного словосполучення "brute force", що означає в перекладі "груба сила". Суть підходу полягає у послідовному автоматизованому переборі всіх можливих комбінацій символів з метою рано чи пізно знайти правильну. З цієї точки зору пошук пароля можна розглядати як математичне завдання, розв'язання якого знаходиться за досить великої кількості спроб. Програмне забезпечення для брутфорсу генерує варіанти паролів та перевіряє кожен із них. З погляду математики вирішити завдання у такий спосіб

можна завжди, але витрати часу на пошуки не завжди виправдовують мету, оскільки поле пошуку рішень величезне.

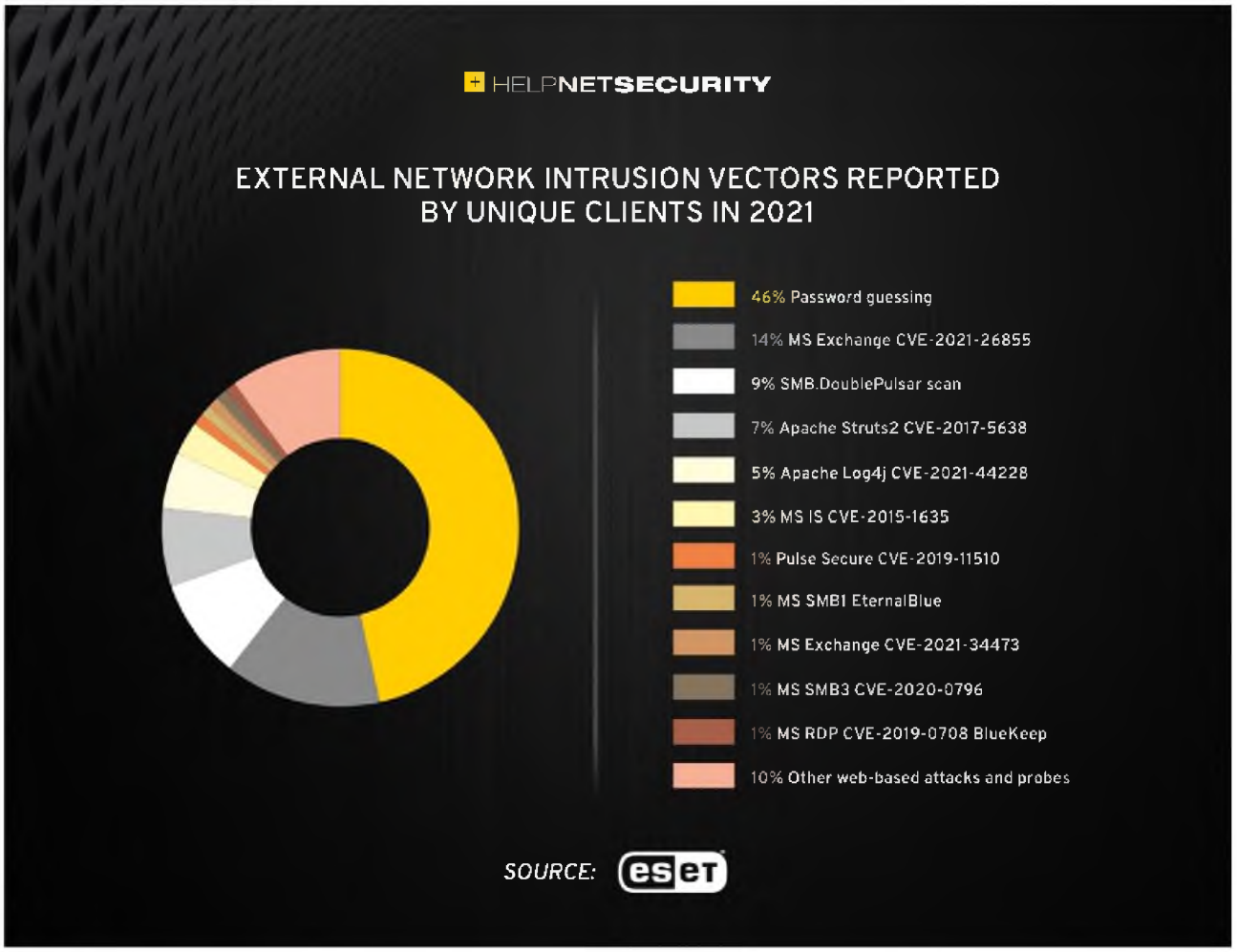

Рисунок 2.4 Брутфорс - найпопулярніший метод кібератаки

Брутфорс - найпопулярніший метод злому паролів до облікових записів онлайн-банків, платіжних систем та інших вебсайтів. Втім, зі збільшенням довжини пароля цей спосіб стає незручним, оскільки зростає час, що необхідно на перебір всіх можливих варіантів. Також за його допомогою можна перевіряти криптостійкість пароля. Брутфорс ще називають методом вичерпування, тому що вірна комбінація виявляється шляхом аналізу всіх можливих варіантів та відкидання кожного невідповідного поєднання.

2.2 Забезпечення безпеки автентифікації та авторизації у Django Admin Panel

В результаті аналізу вразливостей вебдодатків на Django, буде реалізований модуль для захисту контролера авторизації в Django Admin Panel від атаки методом брутфорсу, що використовує Selenium WebDriver. Для захисту модуль буде отримувати валідовані та вірні персональні дані користувача, генерувати унікальний токен з певним періодом життя та відправляти його на пошту користувача з метою підтвердження та завершення авторизації. Також у модуль закладено логіку гнучкого перевизначення шляху до Django Admin Panel з повідомленням адміністраторів про оновлення.

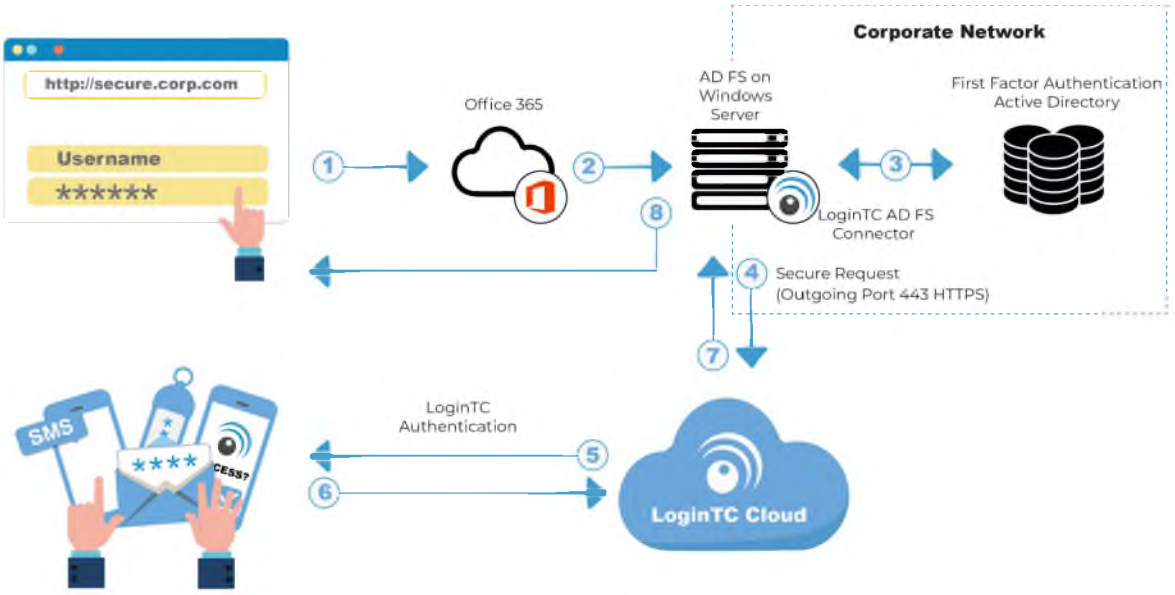

Рисунок 2.5 Архітектура роботи модулю

2.3 Створення проєкту та особливості програмної реалізації.

Для забезпечення безпеки будемо створювати модуль, який можна буде завантажити і встановити для будь-якого Django додатку.

Насамперед створюється проєкт з назвою django_protect. Створити проєкт на Django можна за допомогою команди у командному рядку, передбачливо завантаживши фреймворк:

```
>>> pip3 install django
>>> django-admin startproject django_protect
```

Створюється модуль з назвою django_admin_protect

```
>>> python manage.py startapp django_admin_protect
```

Утворилася папка django_admin_protect с наступним змістом:

```
+---django_admin_protect
|       admin.py
|       apps.py
|       forms.py
|       models.py
|       tests.py
|       urls.py
|       utils.py
|       views.py
|       __init__.py
```

Apps.py - є виконуваним файлом модуля, який ініціює додаток. Файл несе конфігурацію та базові налаштування програми, а також встановлює свої налаштування під час компіляції проєкту.

Створимо базові налаштування модуля, які можна буде змінити під час використання в settings.py

```
DEFAULT_SETTINGS = {
    "TEMPLATE_TITLE": "Django Admin",
    "UPDATE_ADMIN_PATH": {
        "INCLUDING": False,
        "EVERY": 60 * 60 * 2
    },
    "OTP": {
        "DATE_ISSUED": 60 * 15,
        "GENERATOR": None,
        "WIDTH": 6,
        "MESSAGE_TITLE": "OTP",
        "MESSAGE_BODY": "OTP: %s",
        "MAIL_FIELDS": ["email"]
    }
}
```

Models.py - зберігає в собі класи, які будуть використані Django ORM для подальшої взаємодії з базою даних. По суті, дані класи є описом таблиць/документів баз даних.

Створимо модель AdminProtectUserAuth в якій фіксуватимемо токени двофакторної автентифікації з зовнішнім ключем на об'єкт користувача, а також з полями дати створення токена, дати терміну придатності токена та дати успішної автентифікації.

```
class AdminProtectUserAuth(models.Model):
    user = models.ForeignKey(get_user_model(), on_delete=models.CASCADE, related_name="admin_protect_auths")
```

```
token = models.TextField(default=utils.get_token)
dt_create = models.DateTimeField(auto_now_add=True)
dt_issued = models.DateTimeField(default=utils.get_dt_issued)
dt_auth = models.DateTimeField(blank=True, null=True, default=None)
```

Forms.py зберігає класи які описують вебформу та вміють валідувати вхідні дані, очищати та надавати їх для подальшого використання.

Створимо клас-форму, яку будуть отримувати користувачі при спробі авторизуватися. При створенні успадковуємо базовий клас-форми Django авторизації, це допоможе не описувати логіку валідації та автентифікації користувача.

Views.py зберігає класи та функції, які приймають вебзапит і повертають вебвідповідь. Ця відповідь може бути HTML-вмістом вебсторінки, перенаправленням, помилкою 404, XML-документом або JSON-документом, файлом або зображенням.

Створимо клас, який буде приймати GET запит на отримання форми для автентифікації в Django Admin Panel та POST запит для отримання даних з форми та безпосередньої авторизації користувача.

Коли всі форми та класи представлення готові, потрібно перевизначити базовий урл адресу для авторизації у Django Admin Panel. Так як Django Admin Panel є внутрішнім модулем і всі посилання цього модуля знаходяться безпосередньо в його виконуваних файлах ми не можемо замінити базове уявлення.

```
urlpattents = [
    path('admin/login/<uuid:token>/', views.AdminProtectView.as_view(),
    name="django_admin_protect_login_token"),
    path('admin/login/', views.AdminProtectView.as_view(),
    name="django_admin_protect_login"),
]
```

Проте можна використовувати хитрість для перевизначення адреси. При запуску сервера Django створює дерево посилань, якщо у проєкті є два однакових шляхи з різними уявленнями, джанго обере те уявлення, яке було зареєстроване в масиві посилань першим. Тому треба визначити наші посилання перед посиланнями Django Admin Panel. Для цього при складанні проєкту додамо всі наші посилання на початок кореневого масиву посилань.

2.4 Використання модуля у вже створених проєктах

Для того, щоб використовувати мінімальні можливості модуля, потрібно виконати дві дії.

1) Завантажити модуль з менеджера модулів Python

```
>>> pip3 install django-admin-protect
```

2) Активувати модуль у налаштуваннях проєкту, прописавши його в масиві INSTALLED_APPS

```
INSTALLED_APPS = [
    …
    …
    'django_admin_protect'
    …
    …
]
```

Для більш гнучкого налаштування необхідно визначити словник DJANGO_ADMIN_PROTECT у налаштуваннях проєкту та визначити певні ключі:

- TEMPLATE_TITLE - Назва шаблону
- OTP - Словник із наступними ключами
- DATE_ISSUED - Час життя токена (у секундах)

- GENERATOR - Функція яка повертатиме токен
- WIDTH - Довжина токена для базового генератору
- MESSAGE_TITLE - Текст теми у повідомленні на пошту при надсиланні токена користувачеві
- MESSAGE_BODY - Текст тіла повідомлення на пошту при надсиланні токена користувачеві (обов'язково повинен мати "%s" для форматування токена в рядок)
- MAIL_FIELDS - Масив із назв полів для пошти в об'єкті користувача. Якщо у користувача кілька поштових імейлів у різних полях використовуватиметься те поле, яке прописане в масиві першим.

2.5 Результати

Коли всі налаштування зроблено і сервер запущений, переходимо за базовим посиланням для авторизації в Django Admin Panel і потрапляємо на нову форму для авторизації.

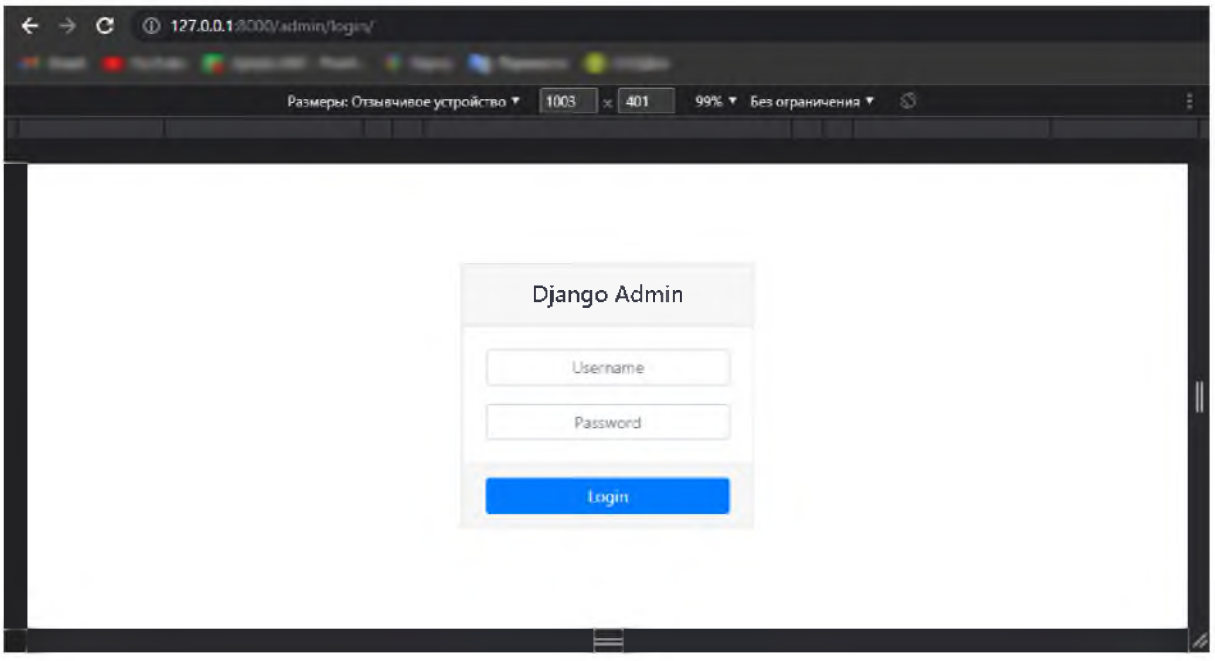

Рисунок 2.6 Форма входу до Django Admin Panel

Вводимо дані для авторизації та прямуємо на сторінку підтвердження з запитом пароля двофакторної автентифікації.

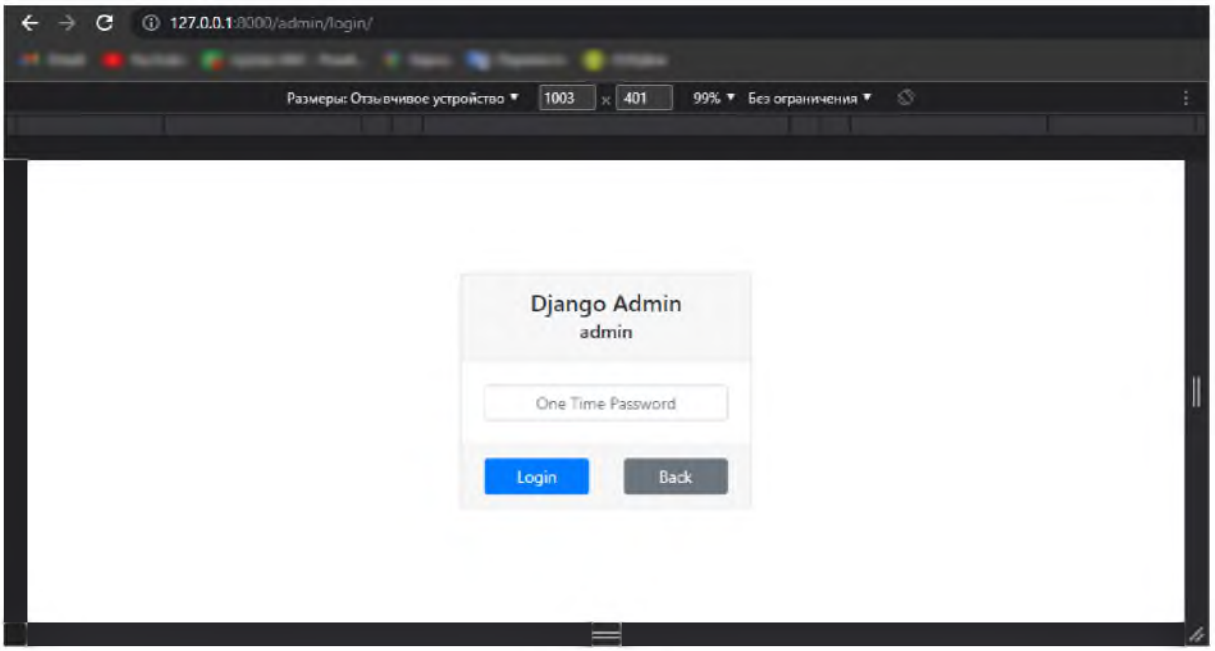

Рисунок 2.7 Запит унікального токена

Отримуємо пароль двофакторної автентифікації на пошту, вказану в об'єкті користувача.

Рисунок 2.8 Унікальний токен на пошті

Вводимо отриманий пароль

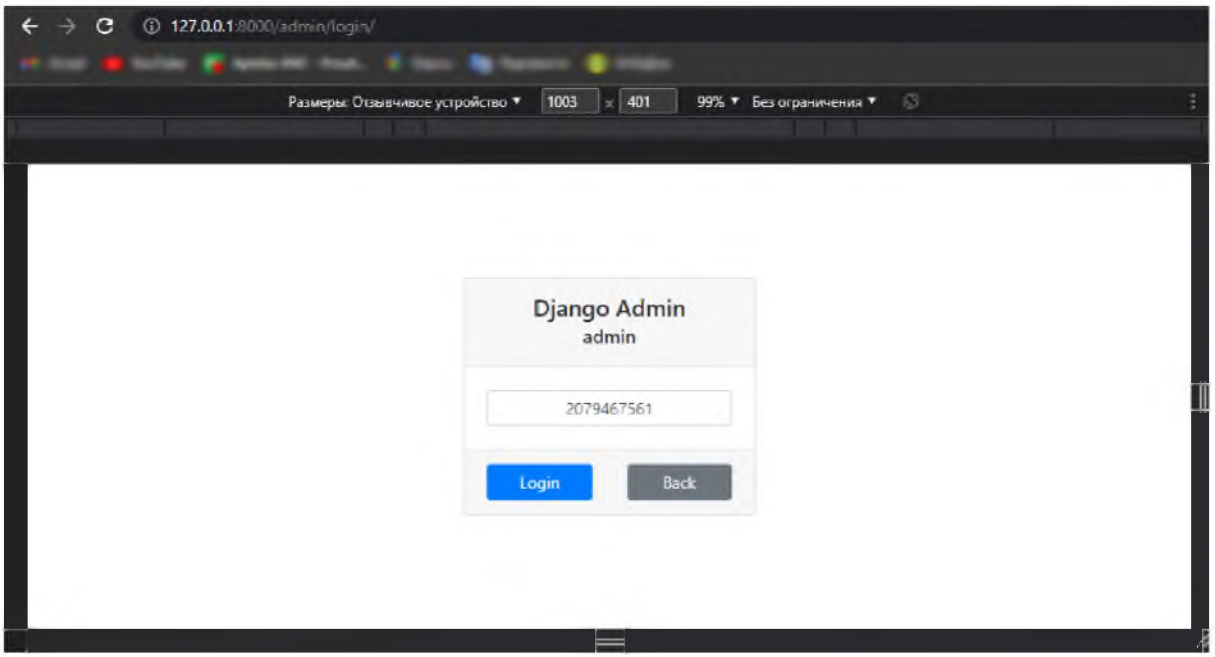

Рисунок 2.9 Введення токена у поле

Натискаємо кнопку Login і прямуємо безпосередньо в базову Django Admin Panel.

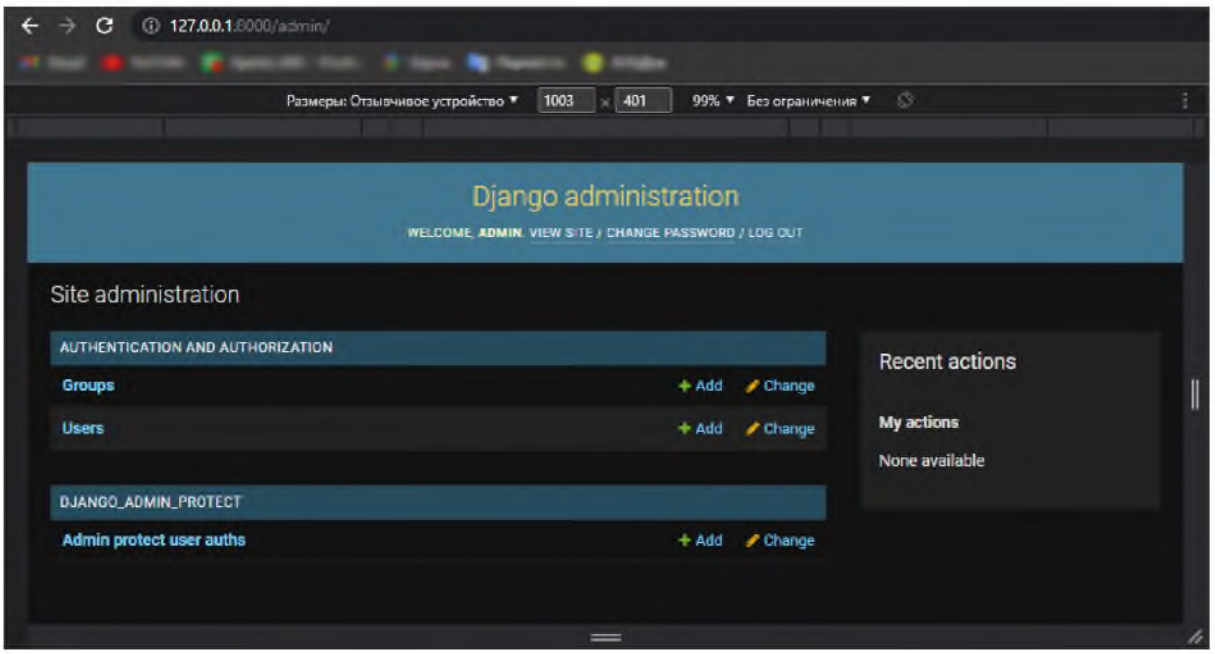

Рисунок 2.10 Автентифікацію успішно завершено

2.6 Висновок

Двофакторна аутентифікація є відмінним захистом від кібератак, проте варто відзначити, що даний модуль може підійти не всім через його архітектуру. Деякі компанії можуть відмовитися від використання цього захисту через брак ресурсів на виділення корпоративної пошти для співробітників або внутрішньої політики. У представленому в цій роботі модулі закладено додаткову логіку захисту від кібератак на Django Admin Panel, яка не змогла бути реалізована через технічні обмеження. Суть даної логіки в тому, щоб сервер мав можливість автоматично змінювати шлях до подання, попередньо запитавши певний токен в частині шляху, який буде відомий тільки адміністраторам сервісу. Такий захист допоможе динамічно перевизначати шлях до подання, що очікує даних користувача, що стане неможливим для кібератак методом брутфорсу.

РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою розділу є обґрунтування економічної доцільності розробки модулю забезпечення захищеності авторизації у Django Admin Panel, та визначенням витрат на проєктування та експлуатацію цієї системи.

3.1 Визначення витрат на розробку модулю

3.1.1 Визначення та розрахунок витрат на придбання та збирання комп'ютера для програмування

Капітальні витрати - грошові видатки, пов'язані з вкладенням в основний капітал чи в приріст виробничих запасів.

$$К_е = К_{пр} + К_м = 65500 + 800 = 66300 \text{ грн}$$

де $К_е$ - вартість витрат на придбання та встановлення системи відеоспостереження;

$К_{пр}$ - вартість придбання (таблиця 3.1);

$К_м$ - вартість монтажу

Таблиця 3.1 - Витрати на матеріали і обладнання

Найменування обладнання та матеріалів	Кількість, ш	Вартість, ш	Сума, грн
Ноутбук ASUS TUF Dash F15 FX517ZM	1	50800	50800
Монітор MSI OPTIX 29.5" MAG301CR2	1	14700	14700
Разом:			65500

Вище наведено таблицю з вартістю обладнання. Вартість збирання комплектуючих комп'ютера становить 800 грн.

Виходячи з розрахунків, загальні витрати на придбання та збирання комп'ютера для програмування складе 66300 грн.

3.1.2 Визначення трудомісткості розробки та розрахунок витрат на створення вимог

Трудомісткість створення вимог визначається тривалістю кожної робочої операції, починаючи з складання технічного завдання і закінчуючи оформленням документації за умови роботи одного робітника:

$$T = t_{тз} + t_{в} + t_{а} + t_{вз} + t_{озб} + t_{овр} + t_{д} = 2 + 3 + 2 + 2 + 2 + 3 + 1 = 15 \text{ годин}$$

де $t_{тз}$ - тривалість складання технічного завдання на розробку модулю;

$t_{в}$ - тривалість розробки концепції у організації;

$t_{а}$ - тривалість процесу аналізу ризиків;

$t_{вз}$ - тривалість визначення вимог до заходів, методів та засобів забезпечення;

$t_{озб}$- тривалість вибору основних рішень з забезпечення;

$t_{овр}$- тривалість організації виконання відновлювальних робіт і забезпечення неперервного функціонування організації;

$t_{д}$ - тривалість документального оформлення політики захищеності;

Витрати на створення вимог $К_{рп}$ складаються з витрат на заробітну плату спеціаліста $З_{зп}$ і вартості витрат машинного часу, що необхідний для опрацювання програми на персональному комп'ютері $З_{мч}$:

$$К_{рп} = З_{зп} + З_{мч} = 5179,5 + 40 = 5219,5 \text{ грн}$$

Заробітна плата виконавця враховує основну і додаткову заробітну плату, а також відрахування на соціальні потреби і визначається формулою:

$$З_{зп} = t \cdot З_{пр} = 15 \cdot 345,3 = 5179,5 \text{ грн}$$

де $З_{пр}$ - середньогодинна заробітна плата програміста з нарахуваннями, грн/годину.

Вартість машинного часу для налагодження програми на ПК визначається за формулою:

$$З_{мч} = t_{опр} \cdot С_{мч} + t_{д} = 2 \cdot 19,5 + 1 = 40 \text{ грн}$$

де $t_{опр}$ - трудомісткість налагодження програми на ПК, годин;

$С_{мч}$ - вартість 1 години машинного часу ПК, грн/година;

$t_{д}$ - трудомісткість підготовки документації на ПК, годин;

Вартість 1 години машинного часу ПК визначається за формулою:

$$С_{мч} = Р \cdot t_{нал} \cdot C_{е} + \frac{Ф_{зал} \cdot Н_{а}}{F_{р}} + \frac{К_{лпз} \cdot Н_{апз}}{F_{р}} =$$

$$= 0,45 \cdot 15 \cdot 1,68 + \frac{25000 \cdot 0,6}{1920} + \frac{3500 \cdot 0,2}{1920} = 19,5 \text{ грн/год}$$

де Р - встановлена потужність ПК, кВт;

$C_{е}$ - тариф на електричну енергію, грн/кВт-година;

$Ф_{зал}$ - залишкова вартість ПК на поточний рік, грн;

$Н_{а}$ - річна норма амортизації на ПК, частки одиниці;

$Н_{апз}$ - річна норма амортизації на ліцензійне програмне забезпечення, частки одиниці;

$К_{лпз}$ - вартість ліцензійного програмного забезпечення, грн;

$F_{р}$ - річний фонд робочого часу (1920);

Капітальні (фіксовані) витрати на розробку та впровадження програми підвищення обізнаності складають:

$$К = К_{пр} + К_{зпз} + К_{рп} + К_{аз} + К_{дм} + К_{навч} + К_{н}$$

де $К_{пр}$ – вартість розробки програми підвищення рівня обізнаності та залучення для цього зовнішніх консультантів, тис.грн. Сторонні організації не наймалися, тому даний коефіцієнт не враховується при розрахунках;

$К_{зпз}$ – вартість закупівель ліцензійного основного та додаткового ПЗ, складає 0 грн (фреймворк Django);

$К_{рп}$ – вартість розробки програми підвищення обізнаності складає 10 000 грн;

$К_{аз}$ – вартість закупівлі апаратного забезпечення, складає 65500 грн.

$К_{дм}$ – вартість допоміжних матеріалів: 0 грн;

$К_{навч}$ – витрати на навчання технічних фахівців і обслуговуючого персоналу, грн. Дані витрати не враховуються під час розрахунку формули, тому що фахівці не проходили платного навчання.

$К_{н}$ – витрати на встановлення обладнання та налагодження системи ІБ, грн. Даних витрат не було, оскільки програма націлена на підвищення рівня знань у працівників підприємства.

$$К = 5219,5 + 10000 + 65500 = 80719,5$$

Таким чином, капітальні витрати на розробку та впровадження вимог складають 80719,5 грн.

Розрахунок річних експлуатаційних витрат на утримання і обслуговування програми підвищення обізнаності персоналу

Річні поточні (експлуатаційні) витрати на функціонування програми підвищення обізнаності складають:

$$С = Св + Ск + Сак \text{ , грн}$$

де Св – вартість відновлення й модернізації системи;

Ск – витрати на керування програмою в цілому;

Сак – витрати, викликані активністю користувачів.

Витрати на керування програмою підвищення обізнаності персоналу складають:

$$Ск = Сн + Са + Сз + Сел + Сєв + Стос, \text{ грн}$$

Річний фонд заробітної плати персоналу, що обслуговує програму (Сз) складає:

$$Сз = Зосн + Здод, \text{ грн}$$

Основна заробітна плата спеціаліста з інформаційної безпеки на місяць – 25 000 грн, додаткова заробітна плата – 8% від основної зарплати:

$$Сз = 25000 * 12 + 25000 * 12 * 0,08 = 324000 \text{ грн}$$

Ставка ЄСВ для всіх категорій платників складає 22%:

$$Сєв = 324000 * 0,22 = 71280 \text{ грн}$$

Вартість електроенергії, що споживається ноутбуками протягом року (Сел), визначається за формулою:

$$Сел = P * Fp * Це \text{ , грн}$$

де Р – встановлена потужність ПК, кВт;

Це – тариф на електричну енергію, грн/кВт*год .

F – річний фонд робочого часу.

$$Сел = 1 * 1920 * 2,14 = 4108,8 \text{ грн}$$

Витрати на технічне й організаційне адміністрування програми визначаються в відсотках від капітальних витрат – 2% (Стос = 80719,5 * 0,02 = 1614,39 грн). Витрати на керування програмою підвищення обізнаності персоналу (Ск) дорівнюють:

$$Ск = 34000 + 324000 + 71280 + 4108,8 + 1614,39 = 435003,19 \text{ грн}$$

Таким чином, річні поточні витрати складають:

$$С = 5000 + 435003,19 = 440003,19 \text{ грн}$$

Визначення річного економічного ефекту від впровадження програми підвищення обізнаності персоналу.

Загальний ефект від впровадження програми ІБ визначається з урахуванням ризиків порушення ІБ підприємства і становить:

$$E = B * R - C$$

де В – загальний збиток від атак на мережу підприємства, грн;

R – очікувана ймовірність атаки на мережу підприємства, частки одиниці;

С – щорічні витрати на оновлення програми підвищення обізнаності персоналу, грн.

Загальний збиток від атаки на мережу підприємства складає:

$$\text{В} = \sum i \sum n U, \text{ грн}$$

де I – число атакованих мереж підприємства;

N – середнє число атак на рік;

U – упущена вигода від простою атакованої мережі підприємства.

Упущена вигода від простою атакованою мережі підприємства становить:

$$U = \text{Пп} + \text{Пв} + V$$

де Пп – оплачувані втрати робочого часу та простої співробітників атакованої мережі підприємства, грн;

Пв – вартість відновлення працездатності мережі підприємства;

V – втрати від зниження обсягу продажів за час простою атакованої мережі підприємства, грн.

Втрати від зниження продуктивності співробітників атакованої мережі підприємства являють собою втрати їхньої заробітної плати за час простою внаслідок атаки:

$$\text{Пп} = (\sum \text{Зс}) / F * t\text{п}$$

де F – місячний фонд робочого місяця (при 40–а годинному робочому тижні становить 176 ч);

Зс – заробітна плата співробітників атакованої мережі на підприємства, грн на місяць;

*t*п – час простою мережі підприємства внаслідок атаки, год.

Витрати на відновлення працездатності мережі підприємства включають:

$$Пв = Пви + Ппв + Пзч$$

де Пви – витрати на повторне введення інформації, грн;

Ппв – витрати на відновлення мережі підприємства, грн;

Пзч – вартість заміни устаткування або запасних частин, грн.

Витрати на повторне введення інформації розраховуються:

$$Пви = (\sum Зс)/F * tви$$

де *t*ви – час повторного введення загубленої інформації співробітниками атакованої мережі підприємства, год.

Витрати на відновлення мережі підприємства визначаються:

$$Ппв = (\sum Зо)/F * tв$$

де *t*в – час відновлення після атаки персоналом, що обслуговує мережу підприємства, год;

Зо – заробітна плата обслуговуючого персоналу, грн на місяць.

Втрати від зниження очікуваного обсягу продажів за час простою атакованої мережі підприємства визначаються:

$$V = (O/F\text{г}) * (t\text{в} + t\text{п} + t\text{ви})$$

де О – обсяг продажів атакованої мережі підприємства, грн у рік;

Fг – річний фонд часу роботи підприємства становить 2080 ч.

Визначення річного економічного ефекту:

$$V = (7000000/2080) * (3 + 2 + 2) = 23555 \text{ грн}$$

$$\text{Ппв} = (250000/176) * 3 = 4261{,}3 \text{ грн}$$

$$\text{Пви} = (200000/176) * 2 = 2272{,}7 \text{ грн}$$

$$\text{Пв} = 4261{,}3 + 2272{,}7 = 6534 \text{ грн}$$

$$\text{Пп} = (200000/176) * 2 = 2272{,}7 \text{ грн}$$

$$U = 6534 + 2272{,}7 + 23555 = 32361{,}7 \text{ грн}$$

$$\text{В} = \sum 2 \sum 20\ 32361{,}7 = 2 * 20 * 32361{,}7 = 1294468 \text{ грн}$$

$$E = 1294468 * 0{,}7 - 440003{,}19 = 466124{,}4 \text{ грн}$$

Визначення та аналіз показників економічної ефективності запропонованого в кваліфікаційній роботі проєктного рішення.

Оцінка економічної ефективності програми підвищення рівня обізнаності, здійснюється на основі визначення та аналізу наступний показників:

1. Сукупна вартість володіння (ТСО);
2. Коефіцієнт повернення інвестицій (ROSI);
3. Термін окупності капітальних інвестицій То.

Коефіцієнт повернення інвестицій (ROSI) показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження програми підвищення обізнаності персоналу.

$$ROSI = E/K \text{ , частки одиниці}$$

де Е – загальний ефект від впровадження програми підвищення обізнаності персоналу, грн;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн.

$$ROSI = 466124{,}4 / 80719{,}5 = 5{,}7$$

Термін окупності капітальних інвестицій То показує, за скільки років капітальні інвестиції окупаються за рахунок загального ефекту від впровадження програми підвищення обізнаності персоналу:

$$\text{То} = K/E = 1/ROSI \text{ , років}$$

$$\text{То} = 180719{,}5/466124{,}4 = 0{,}4 \text{ рока (5 місяців)}$$

3.2 Висновки економічного розділу

В результаті проведення економічних розрахунків, з метою оцінки вартості та рентабельності впровадження Django Admin Protect, було розраховано показники капітальних витрат (К). Капітальні витрати на впровадження обраних програмно-апаратних засобів склали 66300 гривень.

Розрахувати доцільність впровадження на даному етапі не є можливим, оскільки ефект, більшою мірою, є соціально значущим, та для отримання точних даних потребує всебічного окремого вивчення з урахуванням показників як приватних надавачів послуг так і державних інститутів.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи, було проаналізовано наявні методи забезпечення кіберзахисту Django вебдодатків. Було приділено увагу до основних методів забезпечення безпеки вебдодатків. Висвітлено проблему у способі автентифікації у Django Admin Panel.

Головною задачею поставлено «Запропонувати методи підвищення рівня захисту адміністративною панелі вебдодатків на Django».

У спеціальній частині роботи виконано аналіз наявної моделі підвищення рівня захисту адмін-панелі на базі власної електронно-обчислювальної машини.

Висновки та результати тестування, отримані в результаті вивчення вищезазначеного методу вказали на можливість до виправлення більшості проблем які наявні у вебдодатків на Django. Результатами стала потреба у впровадженні модулю django-admin-protect до вебдодатку. Окремо був наданий перелік правил та рекомендацій при використанні модуля в інших проєктах.

Під час виконання економічного розділу було розраховано показники капітальних витрат у разі комерційної розробки модулю для забезпечення захисту адмін-панелі у вебдодатках на Django.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is backend framework
[Електронний ресурс] – Режим доступу до ресурсу: https://blog.back4app.com/backend-frameworks/#What_is_a_backend_framework
2. Web безпека Django admin site
[Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Website_security#:~:text=%D0%A6%D0%B5%D0%BB%D1%8C%20%D0%B2%D0%B5%D0%B1%2D%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%D0%B7%D0%B0%D0%BA%D0%BB%D1%8E%D1%87%D0%B0%D0%B5%D1%82%D1%81%D1%8F%20%D0%B2,%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D1%83%D0%BD%D0%B8%D1%87%D1%82%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B8%D0%BB%D0%B8%20%D0%BD%D0%B0%D1%80%D1%83%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B
3. Official documentation Django security
[Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/2.0/topics/security/
4. Official documentation Django admin
[Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.1/ref/contrib/admin/
5. What is Cross Site Scripting
[Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/what-is-cross-site-scripting-xss/
6. What is Cross Site Request Forgery (CSRF)
[Електронний ресурс] – Режим доступу до ресурсу: https://creativegroundtech.com/what-is-cross-site-request-forgery-csrf/
7. SQL Injection Attacks

[Електронний ресурс] – Режим доступу до ресурсу:
https://spanning.com/blog/sql-injection-attacks-web-based-application-security-part-4/
8. What is clickjacking
[Електронний ресурс] – Режим доступу до ресурсу:
https://www.imperva.com/learn/application-security/clickjacking/
9. XSS attack
[Електронний ресурс] – Режим доступу до ресурсу:
https://wiki.rookee.ru/cross-site-scripting/
10. База знаний Django
[Електронний ресурс] – Режим доступу до ресурсу:
https://django.fun/ru/
11. Вебфреймворк Django
[Електронний ресурс] – Режим доступу до ресурсу:
https://developer.mozilla.org/ru/docs/Learn/Server-side/Django
12. Selenium WebDriver
[Електронний ресурс] – Режим доступу до ресурсу:
https://www.selenium.dev/documentation/webdriver/
13. Selenium WebDriver Python
[Електронний ресурс] – Режим доступу до ресурсу:
https://selenium-python.readthedocs.io/getting-started.html
14. Брутфорс
[Електронний ресурс] – Режим доступу до ресурсу:
https://encyclopedia.kaspersky.ru/glossary/brute-force/

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітки
Документація				
1	А4	Реферат	1	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	2	
5	А4	Стан питання. Постановка задачі	13	
6	А4	Спеціальна частина	13	
7	А4	Економічний розділ	10	
8	А4	Висновки	1	
9	А4	Перелік посилань	1	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	
14	А4	Додаток Ґ	12	

ДОДАТОК Б. Перелік матеріалів на оптичному носії

- Шаталов В.В._125м–21–1.docx
- Шаталов В.В._125м–21–1.pdf
- Шаталов В.В._125м–21–1.pptx

ДОДАТОК В. Відгук керівника економічного розділу

Відгук керівника економічного розділу

Економічний розділ виконаний відповідно до вимог, які ставляться до кваліфікаційних робіт, та заслуговує на оцінку __ б. (_____).

Керівник розділу _____ доц. Пілова Д.П.

(підпис) (ініціали, прізвище)

ДОДАТОК Г

ВІДГУК

на кваліфікаційну роботу студента групи 125м-21-1
Шаталова Владислава Вадимовича
на тему: «Методи підвищення рівня захисту адміністративної панелі вебдодатків на Django»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, викладених на 65 сторінках.

Метою кваліфікаційної роботи є дослідження вразливостей адміністративної панелі вебдодатків розроблених на фреймворку Django та розробка методів захисту процесів автентифікації в ній.

Для досягнення поставленої мети в кваліфікаційній роботі вирішуються наступні задачі: аналіз актуальних типів атак на вебдодатки, їх вразливостей та особливостей забезпечення безпеки Django вебдодатків, дослідження методів забезпечення захисту адміністративної панелі. Розроблено програмну реалізацію двохфакторної автентифікації.

Практичне значення результатів кваліфікаційної роботи полягає у отриманих результатах дослідження актуальних вразливостей Django вебдодатків та розробленій методиці підвищення захисту адміністративної панелі.

За час дипломування Шаталов В.В. проявив себе фахівцем, здатним самостійно вирішувати поставлені задачі та заслуговує присвоєння кваліфікації магістра за спеціальністю 125 Кібербезпека, освітньо-професійна програма «Кібербезпека».

Кваліфікаційна робота заслуговує оцінки «відмінно».

Керівник
кваліфікаційної роботи доц. каф. БІТ, к.т.н. Сафаров О.О.

ДОДАТОК Ґ

django_admin_protect/admin.py

```
from django.contrib import admin

from .models import AdminProtectUserAuth

admin.site.register(AdminProtectUserAuth)
```

django_admin_protect/apps.py

```
from django.apps import AppConfig

import importlib

class DjangoAdminProtectConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'django_admin_protect'

    DEFAULT_SETTINGS = {
        "TEMPLATE_TITLE": "Django Admin",
        "UPDATE_ADMIN_PATH": {
            "INCLUDING": False,
            "EVERY": 60 * 60 * 2
        },
        "OTP": {
            "DATE_ISSUED": 60 * 15,
            "GENERATOR": None,
            "WIDTH": 6,
```

```
            "MESSAGE_TITLE": "OTP",
            "MESSAGE_BODY": "OTP: %s",
            "MAIL_FIELDS": ["email"]
        }
    }
    SETTINGS = DEFAULT_SETTINGS.copy()

    def ready(self):
        super().ready()

        from django.conf import settings
        from . import urls

        urlconf = importlib.import_module(settings.ROOT_URLCONF)

        for url in urls.urlpattents:
            urlconf.urlpatterns.insert(0, url)

        if hasattr(settings, "DJANGO_ADMIN_PROTECT"):
            self.update_settings(DjangoAdminProtectConfig.SETTINGS,
settings.DJANGO_ADMIN_PROTECT)

        if
(DjangoAdminProtectConfig.SETTINGS.get("UPDATE_ADMIN_PATH",
{}).get("INCLUDING") or \

DjangoAdminProtectConfig.DEFAULT_SETTINGS["UPDATE_ADMIN_PATH
"]["INCLUDING"]):
            ...
```

```
    def update_settings(self, default_settings, settings):
        for key, value in default_settings.items():
            if new_value := settings.get(key):
                if isinstance(value, dict):
                    if isinstance(new_value, dict):
                        self.update_settings(value, new_value)
                else:
                    default_settings[key] = new_value
```

django_admin_protect/forms.py

```
from django import forms
from django.contrib.auth.forms import AuthenticationForm


class AdminProtectForm(AuthenticationForm):
    def __init__(self, admin_protect=None, *args, **kwargs):
        super().__init__(*args, **kwargs)

        self.fields["username"].widget.attrs.update(
            {"class": "form-control form-control-sm text-center", "placeholder":
"Username"}
        )

        self.fields["password"].widget.attrs.update(
            {"class": "form-control form-control-sm text-center", "placeholder":
"Password"}
        )

        if admin_protect:
```

```
            del self.fields["username"]
            del self.fields["password"]

            self.fields["otp"] = forms.CharField(
               required=False,
               widget=forms.TextInput({
                  "class": "form-control form-control-sm text-center",
                  "placeholder": "One Time Password"
               })
            )

            self.admin_protect = admin_protect
            self.user = self.admin_protect.user
```

django_admin_protect/forms.py

```
from django.db import models
from django.contrib.auth import get_user_model

from . import utils


class AdminProtectUserAuth(models.Model):
   user = models.ForeignKey(get_user_model(), on_delete=models.CASCADE,
related_name="admin_protect_auths")
   token = models.TextField(default=utils.get_token)
   dt_create = models.DateTimeField(auto_now_add=True)
   dt_issued = models.DateTimeField(default=utils.get_dt_issued)
   dt_auth = models.DateTimeField(blank=True, null=True, default=None)
```

django_admin_protect/urls.py

```
from django.urls import path
from . import views

urlpattents = [
   path('admin/login/<uuid:token>/', views.AdminProtectView.as_view(),
name="django_admin_protect_login_token"),
   path('admin/login/', views.AdminProtectView.as_view(),
name="django_admin_protect_login"),
]
```

django_admin_protect/utils.py

```
from django.http import Http404
from django.shortcuts import redirect
from django.core.mail import EmailMessage

from .apps import DjangoAdminProtectConfig

import random
import importlib
from datetime import datetime, timedelta



DEFAULT_SETTINGS = DjangoAdminProtectConfig.DEFAULT_SETTINGS
SETTINGS = DjangoAdminProtectConfig.SETTINGS



def token_required(func):
```

```
    def wrapper(request, token=None, *args, **kwargs):
        need_token = SETTINGS.get("UPDATE_ADMIN_PATH",
{}).get("INCLUDING") or
DEFAULT_SETTINGS["UPDATE_ADMIN_PATH"]["INCLUDING"]
        if need_token and not token:
            raise Http404

        if not need_token and token:
            return redirect("django_admin_protect_login")

        return func(request, token=token, *args, **kwargs)

    return wrapper



def get_token(*args, **kwargs):
    if generator := SETTINGS.get("OTP", {}).get("GENERATOR"):
        if func := importlib.import_module(generator):
            return func()

    token_width = SETTINGS.get("OTP", {}).get("WIDTH") or
DEFAULT_SETTINGS["OTP"]["WIDTH"]
    return ''.join(map(str, random.choices(range(0, 10), k=token_width)))



def get_dt_issued(*args, **kwargs):
    if dt_issued := SETTINGS.get("OTP", {}).get("DATE_ISSUED"):
        return datetime.now() + timedelta(seconds=dt_issued)

    dt_issued = DEFAULT_SETTINGS["OTP"]["DATE_ISSUED"]
```

```
    return datetime.now() + timedelta(seconds=dt_issued)


def send_otp(admin_protect):
    title = SETTINGS.get("OTP", {}).get("MESSAGE_TITLE") or
DEFAULT_SETTINGS["OTP"]["MESSAGE_TITLE"]
    body = (SETTINGS.get("OTP", {}).get("MESSAGE_BODY") or
DEFAULT_SETTINGS["OTP"]["MESSAGE_BODY"]) % admin_protect.token
    mail_fields = SETTINGS.get("OTP", {}).get("MAIL_FIELDS") or
DEFAULT_SETTINGS["OTP"]["MAIL_FIELDS"]

    for field in mail_fields:
        if hasattr(admin_protect.user, field) and (mail := getattr(admin_protect.user,
field)):
            return EmailMessage(
                subject=title,
                body=body,
                to=[mail]
            ).send()
```

django_admin_protect/views.py

```
from django.views.generic import FormView
from django.shortcuts import redirect
from django.contrib.auth import authenticate, login
from django.utils import timezone
from django.utils.decorators import method_decorator

from datetime import datetime
```

```
from .forms import AdminProtectForm
from .apps import DjangoAdminProtectConfig
from .models import AdminProtectUserAuth
from .utils import token_required, send_otp


@method_decorator(token_required, name="dispatch")
class AdminProtectView(FormView):
    template_name = "django_admin_protect/login.html"
    form_class = AdminProtectForm

    def post(self, request, *args, **kwargs):
        if self.request.POST.get("back") and
self.request.session.get("admin_protect_id"):
            del self.request.session["admin_protect_id"]
            return redirect("django_admin_protect_login")

        return super().post(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        context["SETTINGS"] = DjangoAdminProtectConfig.SETTINGS

        return context

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
```

```
        if admin_protect_id := self.request.session.get("admin_protect_id"):
            admin_protect =
AdminProtectUserAuth.objects.get(id=admin_protect_id)

            if admin_protect.dt_issued >= timezone.now():
                kwargs["admin_protect"] = admin_protect

        return kwargs

    def form_valid(self, form):
        username = form.cleaned_data.get("username")
        password = form.cleaned_data.get("password")
        otp = form.cleaned_data.get("otp")

        if username and password:
            user = authenticate(request=self.request, username=username,
password=password)

            if user is not None:
                admin_protect = AdminProtectUserAuth.objects.create(user=user)
                self.request.session["admin_protect_id"] = admin_protect.id

                sended = None

                try:
                    sended = send_otp(admin_protect)
                except: pass

                if not sended:
                    del self.request.session["admin_protect_id"]
```

```
                form.errors["otp"] = ["OTP not sended"]
                return super().form_invalid(form)


        elif otp and hasattr(form, "admin_protect"):
            if form.admin_protect.token == otp:
                form.admin_protect.dt_auth = datetime.now()
                form.admin_protect.save()

                login(request=self.request, user=form.admin_protect.user)

                return redirect("admin:index")

            form.errors["otp"] = ["Not valid OTP"]
            return super().form_invalid(form)

        return redirect("django_admin_protect_login")
```

django_admin_protect/templates/django_admin_protect/login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ SETTINGS.TEMPLATE_TITLE }}</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css"
```

```
integrity="sha384-xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0
nLMwNLD69Npy4HI+N" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
bbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Fy6S3B9q64WdZWQUiU+q4/2Lc9npb8tCaSX9FK7E8HnRr
0Jz8D6OP9dO5Vg3Q9ct" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
bbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU
+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.min.js"
integrity="sha384-+sLIOodYLS7CIrQpBjl+C7nPvqq+FbNUBDunl/OZv93DB7
Ln/533i8e/mZXLi/P+" crossorigin="anonymous"></script>
</head>
<body>
  <div class="container-fluid d-flex" style="width: 100vw; height: 100vh;
overflow: hidden;">
    <div class="card m-auto">
      <form action="." method="POST">
        {% csrf_token %}
        <div class="card-header d-flex flex-column">
          <h5 class="mx-auto mb-1">{{ SETTINGS.TEMPLATE_TITLE
}}</h5>
```

```
            {% if form.user %}
              <h6 class="mx-auto mb-1">{{ form.user }}</h6>
            {% endif %}
          </div>
          <div class="card-body d-flex flex-column" style="width:
25vw!important; min-width: 25vw!important; max-width: 25vw!important;">
            {% if form.username %}
              <div>
                {{ form.username }}
              </div>
            {% endif %}
            {% if form.password %}
              <div class="mt-3">
                {{ form.password }}
              </div>
            {% endif %}
            {% if form.otp %}
              <div>
                {{ form.otp }}
              </div>
            {% endif %}
            {% if form.errors %}
              <div class="mt-3">
                <strong>Errors:</strong>
                {% for field, errors in form.errors.items %}
                  {% for error in errors %}
                    <p class="m-0">{{ error }}</p>
                  {% endfor %}
                {% endfor %}
              </div>
```

```
                {% endif %}
            </div>
            <div class="card-footer">
                <div class="row">
                    <div class="col-lg">
                        <input class="btn btn-sm btn-primary btn-block my-auto"
type="submit" value="Login">
                    </div>
                    {% if form.otp %}
                        <div class="col-lg">
                            <input class="btn btn-sm btn-secondary btn-block my-auto"
type="submit" value="Back" name="back">
                        </div>
                    {% endif %}
                </div>
            </div>
        </form>
    </div>
  </div>
</body>
</html>
```