НАЦІОНАЛЬНИЙ УНИВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ ТА ХАРДЕНІНГУ CMS WORDPRESS»

Виконав: здобувач 5 курсу
Рівень: бакалавр
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 125 «Кібербезпекаї»
Плаксін Олександр Андрійович

Керівник: к.т.н., доцент
Бойко Віктор Дмитрович

Рецензент: к.т.н., доцент
Чепурна Олена Євгенівна

Одеса, 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ”

Факультет **кібербезпеки та інформаційних технологій**

Кафедра **кібербезпеки**

Галузь знань: **12 Інформаційні технології**

Спеціальність: **125 Кібербезпека**

Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Плаксіну Олександру Андрійовичу

1. Тема роботи: «Автоматизація тестування на проникнення та харденінгу CMS WordPress»

Керівник роботи: к.т.н, доцент Віктор Дмитрович Бойко

затверджені наказом НУ ОЮА від «02» квітня 2024 року № 809-06

2. Строк подання здобувачем роботи: «20» травня 2024 рік

3. Дата видачі завдання: «15» вересня 2023 рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз вимог до проекту	20.02.2024-01.03.2024	

2	Аналіз конкурентів	02.03.2024- 09.03.2024	
3	Розробка функціональних вимог	14.03.2024- 21.03.2024	
4	Проектування вебзастосунку	22.03.2024- 28.03.2024	
5	Реалізація проєкту	01.04.2024- 08.04.2024	
6	Реалізація сутностей та бази даних	10.04.2024- 15.04.2024	
7	Аналіз результатів та висновки	16.04.2024- 19.04.2024	
8	Оформлення пояснювальної записки	20.04.2024- 27.04.2024	

Здобувач _____

(підпис) (прізвище та ініціали)

Керівник роботи _____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему “Автоматизація тестування на проникнення та харденінгу CMS Wordpress” на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 “Кібербезпека”, освітньою програмою: Кібербезпека, містить: 16 фотографій, 20 літературних джерел із переліком посилань.

Робота виконана на 32 сторінках загального тексту і 27 сторінках основного тексту. У процесі роботи були проаналізовані загальні поняття системи керування контентом WordPress, виявлена аудиторія використання, основні переваги та недоліки.

Далі, був проаналізований ринок безпекових рішень, пов’язаних із тестуванням WordPress на типові вразливості та місконфігурації та розроблене власне програмне рішення, що дозволяє в автоматизованому порядку проводити локальне сканування веб-серверу на WordPress на предмет місконфігурацій та вразливостей.

Результати аналізу показали, що на ринку програмного забезпечення відсутні локальні сканери вразливостей для WordPress, що робить новостворене програмне забезпечення унікальним в своєму роді.

В результаті роботи створене програмне забезпечення, що в автоматизованому порядку проводить швидке сканування локального веб-серверу із WordPress, після чого формує звіт в HTML та текстовому форматі, де описує результати сканування та рекомендації по їх виправленню.

ABSTRACT

Qualification work on the topic "Automation of penetration testing and hardening of CMS Wordpress" for the first (bachelor's) level of higher education in the specialty 125 "Cybersecurity", educational programme: Cybersecurity contains: 16 images, 20 literary sources with a list of references.

The work is done on 32 pages of the general text and 27 pages of the main text. In the course of the work, the general concepts of the WordPress content management system were analysed, the audience of use, the main advantages and disadvantages were identified.

Next, we analysed the market of security solutions related to testing WordPress for common vulnerabilities and misconfigurations and developed our own software solution that allows for automated local scanning of a WordPress web server for misconfigurations and vulnerabilities.

The results of the analysis showed that there are no local vulnerability scanners for WordPress on the software market, which makes the newly developed software unique in its kind.

As a result of the work, the software was created that conducts a quick scan of a local web server running WordPress in an automated manner, and then generates a report in HTML and text format describing the scan results and recommendations for their correction.

ЗМІСТ

ВСТУП

WordPress - це система керування контентом (Content Management System, CMS), яку використовують для створення і управління веб-сайтами, блогами та іншими типами веб-сайтів. [1]

WordPress поєднує в собі зручний інтерфейс для користувачів з потужною функціональністю, що робить його популярним як серед початківців, так і серед досвідчених веб-розробників.

Користувачі WordPress можуть легко створювати, редагувати та публікувати контент на своєму веб-сайті без необхідності великих знань програмування. Він надає широкий вибір тем та плагінів, що дозволяє користувачам налаштовувати свій сайт з урахуванням їхніх унікальних потреб та має власний програмний інтерфейс (API), що дозволяю більш досвідченим користувачам розробляти власні теми та плагіни “на свій смак та колір”.

WordPress може бути корисний для різних груп користувачів, в основному для:

• Блогерів: WordPress надає інтуїтивно зрозумілий інтерфейс для створення та управління блогами, дозволяючи легко публікувати контент, проводити трансляції, опитування тощо

• Підприємців: Від веб-сайтів-візитівок до онлайн-магазинів, WordPress є ідеальним вибором для малого бізнесу, оскільки він дозволяє швидко створювати та підтримувати професійний веб-присутність без великих витрат, а також легко інтегрується із CRM-системами, якими користуються інтернет-магазини для обробки замовлень та для роботи із клієнтами

• Веб-розробників: WordPress має величезну спільноту розробників та підтримується широким спектром тем і плагінів, що робить його доволі потужним інструментом для веб-розробників, які створюють веб-сайти для своїх клієнтів.

• Державних структур: Навіть деякі державні підприємства, органи та структури використовують WordPress для своїх ресурсів (приклад: веб-сайт Одеської ОДА [2] – створений на WordPress)

Таким чином, WordPress - це потужний інструмент, який може задовольнити потреби різних користувачів, починаючи від особистих блогерів і закінчуючи великим бізнесом та навіть державними структурами.

Незважаючи на свою популярність, WordPress має деякі типові проблеми.

Із таких проблем, можна виділити наступні:

• Безпека: Однією з найбільших проблем WordPress є безпека. Велика популярність платформи робить її цілью для зловмисників, які намагаються використовувати різноманітні вразливості для отримання доступу до веб-сайтів та інформації користувачів.

• Швидкодія: Неправильно налаштований веб-сайт WordPress може страждати від поганої швидкодії завантаження сторінок, що може вплинути на користувацький досвід та SEO-рейтинг. [3]

• Оновлення і сумісність: Підтримка WordPress вимагає регулярних оновлень ядра, тем і плагінів. Проблеми можуть виникати, коли оновлення не сумісні з існуючими компонентами веб-сайту. Є випадки, коли розробники мають повністю переписувати плагін через новий синтаксис бібліотеки мови PHP, на базі якої створювався плагін.

• Спам і безпека коментарів: Веб-сайти WordPress можуть стати мішенню для автоматичних спам-коментарів, що може стати проблемою для адміністраторів сайту.

• Надмірність плагінів: Використання занадто багато плагінів може призвести до перенавантаження веб-сайту та збільшення його витрат ресурсів, що може вплинути на продуктивність.

• Резервне копіювання і відновлення: Відсутність автоматичного резервного копіювання може призвести до втрати даних в разі виникнення проблем або вторгнень.

Хоча багато з цих проблем можуть бути вирішені за допомогою правильного налаштування та підтримки, вони все ще залишаються актуальними для багатьох користувачів WordPress.

Виходячи із вищенаведених проблем, визначень та аспектів, можна виділити об’єкт та предмет досліджень даної кваліфакаційної роботи.

Об'єктом дослідження є система управління контентом WordPress. Це включає в себе всі аспекти роботи WordPress, починаючи від огляду його как програмного продукту, закінчуючи його встановленням та тестуванням.

Предметом дослідження є процеси автоматизації тестування та хардерингу (зміцнення безпеки) WordPress. Це стосується як рішень, які будуть розглянуті при огляді ринку програмного забезпечення та методологій, пов’язаних із WordPress, так і тих, що будуть створені та розглянуті в даній кваліфікаційній роботі.

Зважаючи на наявність реальних вразливостей (в тому числі і вразливостей людського фактору), метою кваліфікаційної роботи є створення програмного забезпечення, яке в автоматичному режимі буде знаходити різні вразливості веб-сайтів на CMS WordPress та негайно повідомляти про них адміністратора.

Завдання роботи включають:

• Розробити таке програмне забезпечення, яке буде в автоматичному режимі проводити сканування, перевірку та формування звіту про стан веб-сайту на WordPress та повідомляти його адміністратора

• Протестувати його працездатність в різних умовах

• Зробити інструкцію для даної програми

1 РИЗИКИ ТА ЗАГРОЗИ ЕКСПЛУАТАЦІЇ CMS WORDPRESS

1.1 CMS Wordpress як об'єкт кібератак

WordPress – дуже популярна CMS.

Для розуміння масованості її використання, слід тільки переглянути статистику:

• Згідно дослідженням ресурсу w3techs [4] , 43% усіх існуючих веб-сайтів використовують WordPress

• Згідно даним ресурсу Hostinger [5] – щодня створюється понад 500 вебсайтів на WordPress

• В пошуковій мережі IoT Shodan [6] існує понад 1.500.000 веб-сайтів на WordPress, що вільно індексуються (багато корпоративних сайтів мають захист від індексування подібними системами накшталт Shodan, тому результатів лише півтора мільони)

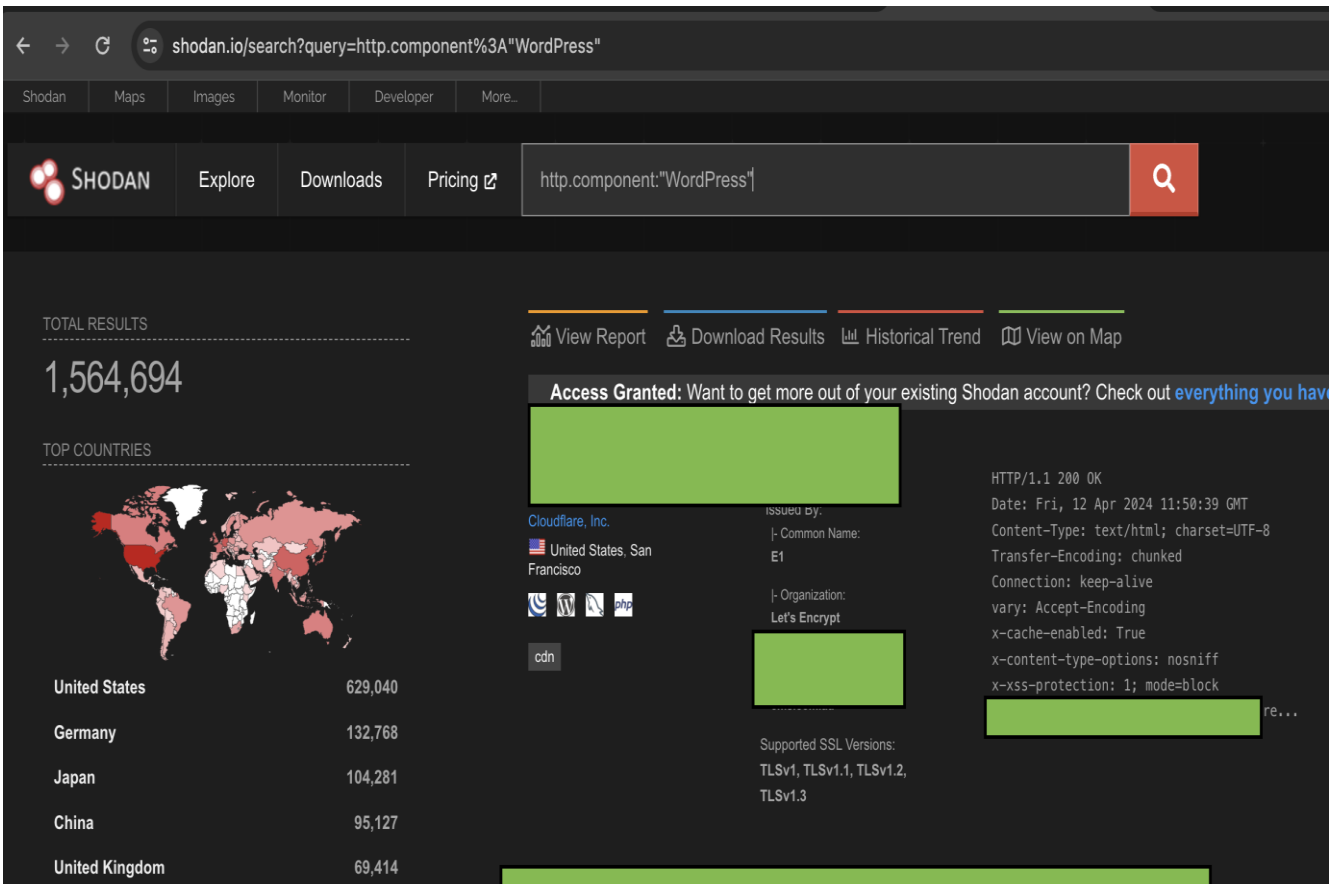


Рисунок 1.1 — статистика WordPress із пошукової системи IoT «Shodan»

• Подібна Shodan пошукова система Censys [7] показує приблизно той самий результат:

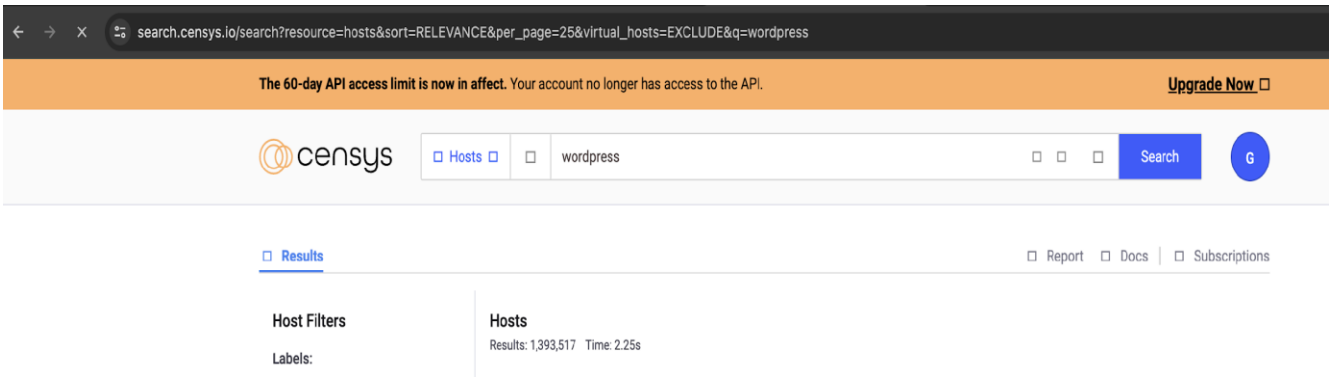


Рисунок 1.2 — статистика WordPress із пошукової системи IoT «Censys»

• Згідно даним з офіційного веб-сайту WordPress [1], CMS має майже 60.000 безкоштовних плагінів (додадків до CMS, які покращують досвід використання)

Таким чином, WordPress сміливо може займати перше місце серед інших існуючих CMS як сама популярна, проста, зручна та багатофункціональна CMS у світі.

Але завжди є “але”.

Проста в налаштуванні CMS WordPress так само проста й в місконфігурації (недолік із класу вразливостей через людський фактор, коли адміністративний користувач банально забуває / не є компетентним, або неправильно із точки зору безпеки здійснює налаштування веб-сайту, що може привести до швидкого зламу ресурсу із подальшим витоком чутливих даних).

Найбанальніший приклад місконфігурації – це залишення файлу wpconfig.php (конфігураційного файлу WordPress) із таким поширенням, через який його можна буде “прочитати” будь-якому користувачу.

Наприклад: якщо у конфігурації поширення “.php”, то веб-сервер не дасть його прочитати звичайному користувачу, однак деякі “адміністратори” роблять резервні копії конфігурацій та присвоюють їм інші поширення, які із легкістю можуть бути прочитані ким завгодно (наприклад, wp-config.php.bak, wpconfig.php.txt тощо)

В такій конфігурації міститься наступна інформація:

• Інформація про базу даних (де розташована, логін та пароль)

• Інформація про адміністративних користувачів (логін/пароль від поштового серверу)

• “солі” (елемент, що посилює паролі) та додадкові ключи від WordPress API

Вся ця інформація може бути використана зловмисниками для отримання доступу до веб-сайту. Наслідки, вважаю, зрозумілі.

Таким чином, для мінімалізації таких ризиків, як наприклад, людський фактор, потрібно створити рішення, яке зможе в автоматичному режимі раз в деякий період часу проводити певне “сканування” веб-сайту із внутрішньої сторони (тобто із середини серверу, а не із зовні), знаходити вразливості (місконфігурації, застаріле ПЗ тощо) та сигналізувати адміністратору про це. Після сканування, адміністратор буде проінформований про ті чи інші вразливості, що були знайдені, а далі повинен сам прийняти рішення та зробити певні кроки для їх виправлення.

WordPress — це потужна та популярна платформа для створення веб-сайтів та блогів, яка має багато переваг для різних типів користувачів. Його простий у використанні інтерфейс, велика спільнота користувачів та розробників, а також широкий вибір тем і плагінів роблять його привабливим вибором для початківців і досвідчених веб-розробників.

Проте, разом із своїми перевагами, WordPress також має свої виклики та проблеми. Найпоширеніші проблеми включають в себе проблеми безпеки, швидкодію, оновлення та сумісність, спам і безпеку коментарів, надмірність плагінів та резервне копіювання і відновлення. Більшість цих проблем можуть бути вирішені за допомогою належного налаштування та підтримки, але вони залишаються актуальними для багатьох користувачів.Крім того, важливо бути уважними до найпоширеніших вразливостей WordPress, таких як крос-сайтовий скриптінг (XSS), SQL ін'єкція, витік інформації, вразливості в сторонніх темах та плагінах, а також витік адміністративнихправ. Захист від цих вразливостей потребує постійного оновлення тем, плагінів та ядра WordPress, а також вживання

інших заходів безпеки, таких як встановлення додаткових захисних плагінів та використання складних паролів.

1.2 Огляд CMS WordPress

Як було сказано вище, WordPress – це популярна система управління контентом, яка дозволяє користувачам легко створювати та керувати вебсайтами та блогами.

Вона була вперше випущена в 2003 році і з тих пір стала найпоширенішою CMS у світі, використовуваною мільйонами сайтів від особистих блогів до великих корпоративних порталів. WordPress розповсюджується під ліцензією GNU General Public License (GPL) версії 2 або пізнішої. Це означає, що програмне забезпечення є відкритим кодом, і користувачі мають право вільно використовувати, змінювати та розповсюджувати його, дотримуючись умов цієї ліцензії. WordPress написаний на мові програмування PHP.

Він також використовує базу даних MySQL для зберігання даних, хоча підтримує й інші бази даних через розширення (PostgreSQL, MSSQL тощо).

Програмні особливості:

- Відкритий код: WordPress повністю безкоштовний та відкритий до модифікацій.
- Теми та плагіни: WordPress має десятки тисяч безкоштовних тем та плагінів (доповнень), що сприяють вдосконаленню користувацького досвіду.
- Універсальність: WordPress підходить майже для будь-яких тематик вебсайтів – від маленьких блогів до великих інтернет-магазинів та корпоративних сайтів.
- Зручність: Інтерфейс WordPress максимально зрозумілий та простий, що спрощує його використання навіть у недосвідчених користувачів.
- Інтеграція з іншими сервісами: WordPress має власний API (Application Protocol Interface), що дозволяє інтегрувати власні програми із веб-сайтом.

Отже, WordPress — це потужна, гнучка та зручна платформа для створення та управління вебсайтами, що робить його популярним вибором для широкого кола користувачів

1.3 Огляд існуючих рішень захисту CMS Wordpress

Самий час розглянути ринок програмного забезпечення, пов’язаного із WordPress, з метою знайти “нишу” для своєї програми та проаналізувати ринок подібного програмного забезпечення загалом.

Платформа GitHub [8] (найбільша програмна платформа для розробників) нараховує понад 300.000 різноманітних програм/модулів/доповнень тощо, пов’язаних із WordPress

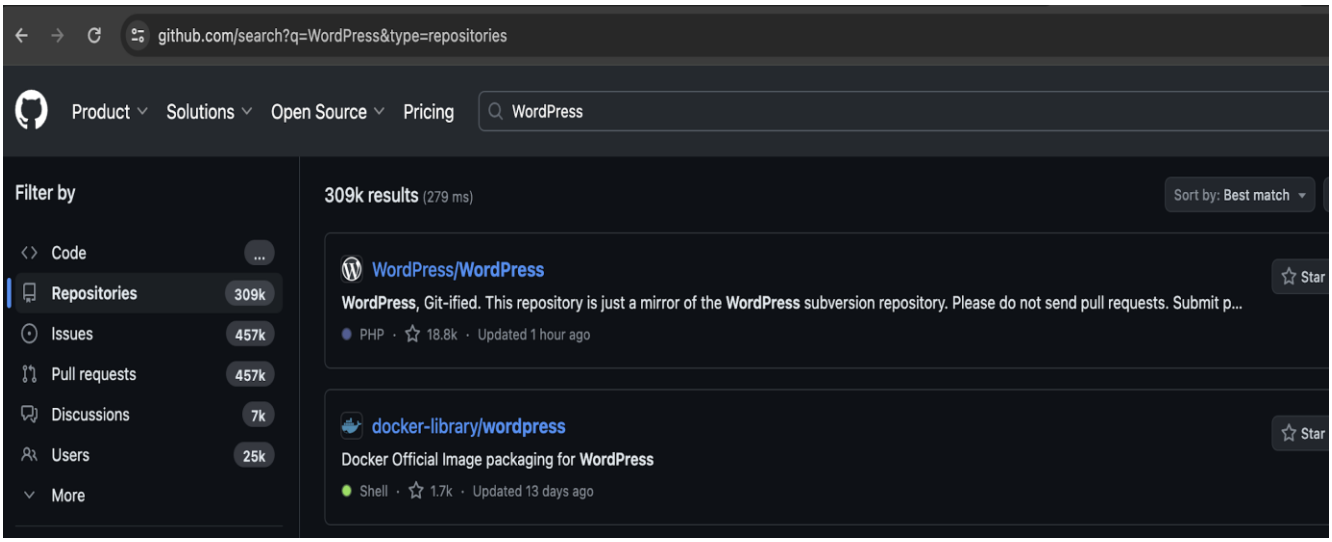


Рисунок 1.3 — статистика програмного забезпечення на платформі GitHub

Натоміж, рішень пов’язаних із безпекою WordPress меньш ніж 1000 (!) В цілому, по тестуванню безпеки сайту на WordPress існують тількі рішення, які працюють віддаленно, тобто тестують вже існуючий в інтернеті (або локальній мережі) веб-сайт без прямого доступу до нього.

Аналогічною ситуацією є й запити до пошукової системи Google [9], яка при запиті «WordPress security scanner» в показує тількі ті сканери[10], які працюють віддалено, тобто сканують вже існуючий веб-сайт на WordPress без прямого доступу до нього або сервера, де він розташований

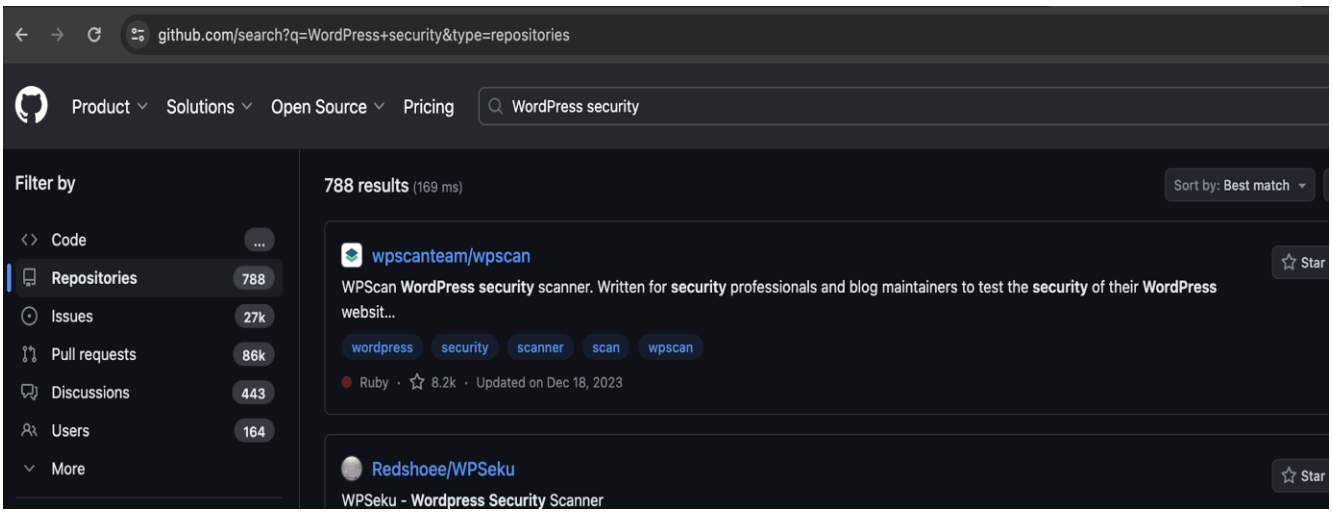


Рисунок 1.4 — статистика програмного забезпечення на платформі GitHub, пов’язаного із безпекою та скануванням веб-сайтів на WordPress.

Не зважаючи на свою широку популярність, WordPress-у бракує безпекових рішень, які працюють локально. Відсутність таких локальних рішень створює серйозні проблеми для користувачів цієї платформи, оскільки вони не мають можливості захистити свої сайти безпосередньо на сервері. На відміну від локальних рішень, в мережі є багато різних сканерів, які проводять тестування CMS WordPress віддаленим чином. Ці сканери зазвичай виконують перевірку сайтів на вразливості, аналізуючи їх ззовні. Основним клієнтом таких рішень є безпосередньо зловмисники, які користуються цими інструментами для пошуку слабких місць у захисті сайтів на базі WordPress.

Рішень, які працюють суто локально на тому самому сервері, на якому працює WordPress – немає. Це означає, що користувачі змушені покладатися на зовнішні сервіси для забезпечення безпеки своїх сайтів. Відсутність локальних інструментів ускладнює процес захисту, оскільки зовнішні сканери не завжди можуть забезпечити повну безпеку. Тож, час його створити!

Необхідно розробити рішення, яке працюватиме на тому ж сервері, де встановлено WordPress, і забезпечуватиме належний рівень захисту. Таке рішення могло б значно підвищити безпеку сайтів і зменшити ризик атак з боку зловмисників.

2 ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ CMS WORDPRESS

2.1 Вибір мови програмування

Основною мовою програмування обрано мову Python [11], на це є певні аргументи:

• Простота та гнучність

Python – проста, багатофункціональна та високорівнева мова програмування. Її функціонал в випадку написання потрібної програми ідеальний. Простота написання коду дозволить його легко досліджувати, писати та модифікувати. Його компактність та проста переносимість (Python є вбудованим майже в кожному дистрибутиві Linux) робить його “кишеньковою” мовою програмування.

• Швидкість:

Хоч і Python не варто порівнювати [12] в швидкості виконання коду із низькорівневими мовами програмування накшталт C, C++, Assembly, але в режимі роботи із базовим функціоналом без зайвих “важких” бібліотек, Python має достатньо високу швидкість виконання коду.

• Доступність

Навіть якщо на веб-сервері із WordPress якимось чином немає вбудованого Python, майже усі дистрибутиви Linux мають можливість установки через командну строку за допомогою пакетного менеджера, а на Windows та MacOS Python завантажується та встановлюється “в два кліки” з офіційного сайту.

• Широкий спектр можливостей

На відміну від інших мов програмування, Python має найвеличезнішу кількість бібліотек під будь-які потреби (під вебзастосунки, математику, логіку, штучний інтелект тощо). Завдяки такому розмаїттю бібліотек, на Python можна написати майже будь-яке програмне рішення, як просте, так и більш комплексне.

Із огляду на алгоритм та логіку програми, із “залежностей” виділяється лише дві бібліотеки:

1. requests[13]

Ця бібліотека дозволяє виконувати GET та POST запити на вебсервери та потрібна, щоб надсилати повідомлення адміністратору про статус виконання програми через Telegram.

2. cryptography [14]

Ця бібліотека має набір функцій та модулів, які використовуються у криптографічних операціях.

В данному випадку, із цієї бібліотеки буде використана функція, що реалізує шифрування файлу конфігурації програми за допомогою алгоритму AES, який на сьогоднішній час є найнадійнішим та використовується навіть у військовій сфері.

2.2 Специфікації програми

Головні специфікації програми:

• Програма має бути:

1. Проста та зрозуміла в використанні для користувачів будь-якого рівня. Чим простіше буде програма — тим буде краще.

2. Швидка, без зайвих процесів та часових витрат. Програма повинна робити свою справу максимально швидко, маючи відповідний функціонал.

3. Переносима. Програма повинна мати мінімум залежностей, виключивши можливість обмеження її функціоналу через оновлення сторонньої бібліотеки.

4. Безпечна. Програма повинна допомогати боротися проти кіберзлочинців, а не допомогати їм.

5. “Модифікабельна”. Досвідчені користувачі повинні мати змогу робити власні модифікації та правки для покращення користувацького досвіду.

6. Відповідати принципам SOLID [17]. Таким чином, програма зможе мати широку популярність та зрозумілість в середовищі досвідчених IT-фахівців.

Загальний функціонал програми:

• Программа повинна виконувати наступні функції:

1. Знаходити можливі місконфігурації.

2. Знаходити можливі вразливі плагіни та теми.

3. Формувати звіт в HTML та в текстовому форматах.

4. Інформувати адміністратора про початок, кінець та результат перевірки через доступний спосіб через Telegram [18]

5. Надати рекомендації із виправлення знайдених вразливостей адміністратору, якщо такі були знайдені.

2.3 Алгоритм та структура програми

Роботу програми можна поділити на 4 складові:

a) Запуск:

1. Завантаження та перевірка конфігурації програми, створення конфігурації якщо її немає

2. Шифрування конфігураційного файлу програми, якщо він не шифрований, попередження користувача про це

b) Ініціалізація:

1. Завантаження та перевірка модулей програми

2. Фінальне визначення об’єктів перевірки

c) Перевірка модулем безпеки:

1. Перевірка кожного модуля на предмет вразливості його самого

2. Відхилення потенційно небезпечних модулей, повідомлення адміністратора

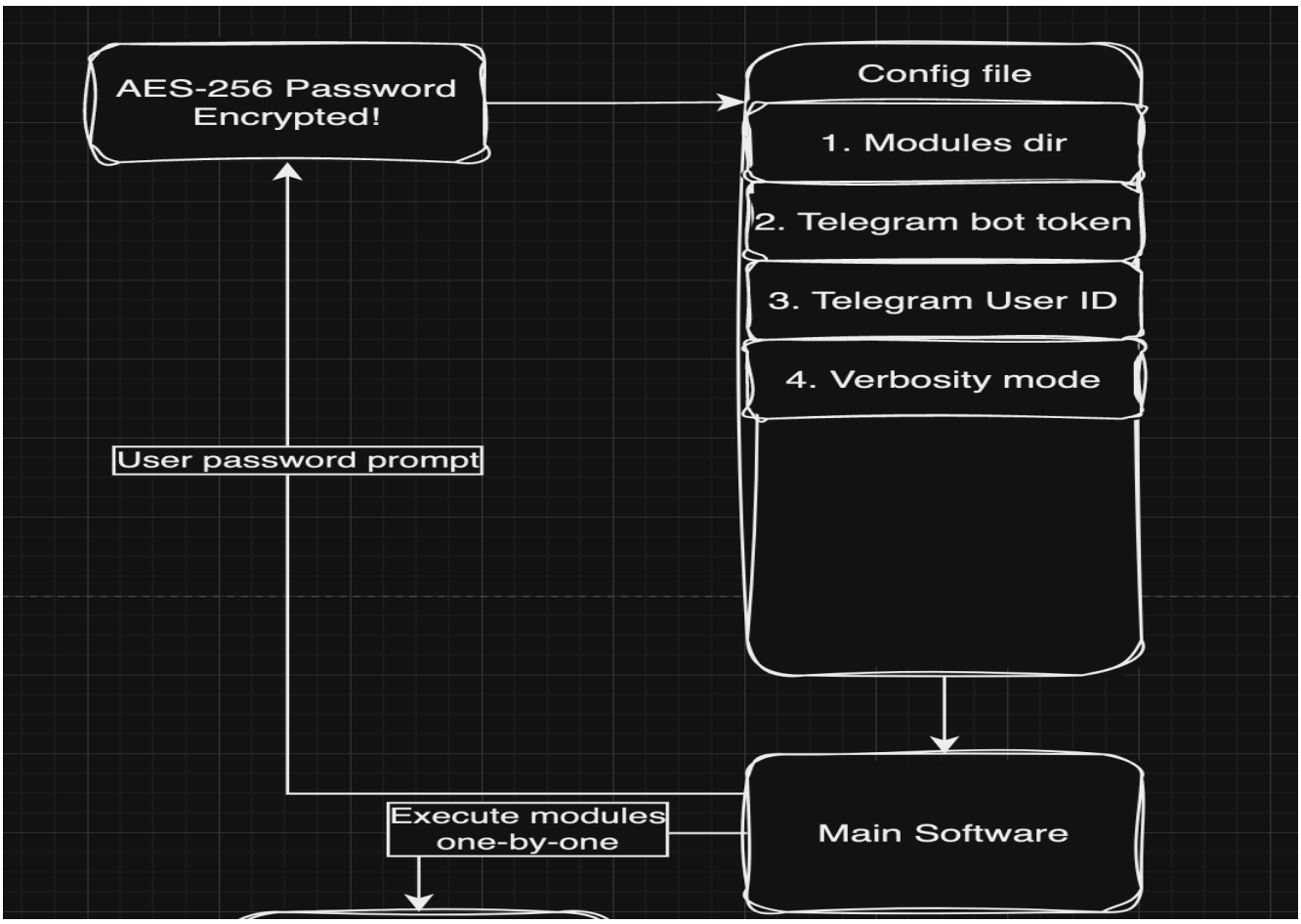


Рисунок 2.1 — демонстрація логіки завантаження конфігурації на блок-схемі

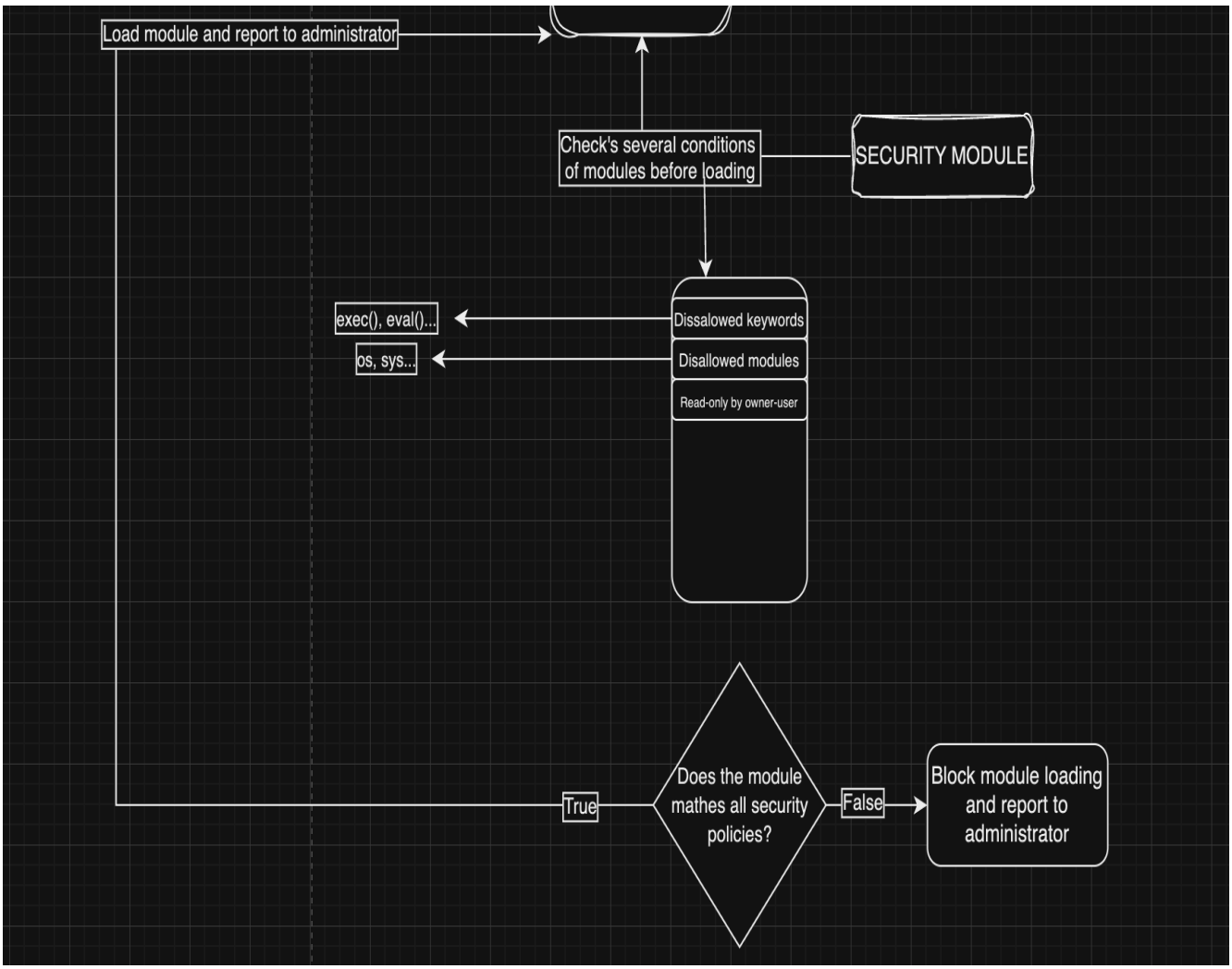


Рисунок 2.2 — демонстрація логіки роботи модуля безпеки на блок-схемі. Даний модуль не дозволяє завантажити модулі програми, що не відповідають правилам безпеки.

d) Виконання та звіт:

1. Виконання модулів один-за-одним, формування масиву вихідних даних

2. Реєстрація помилок (якщо вони є)

3. Формування звіту із отриманого масиву даних

4. Повідомлення адміністратора по закінчення сканування

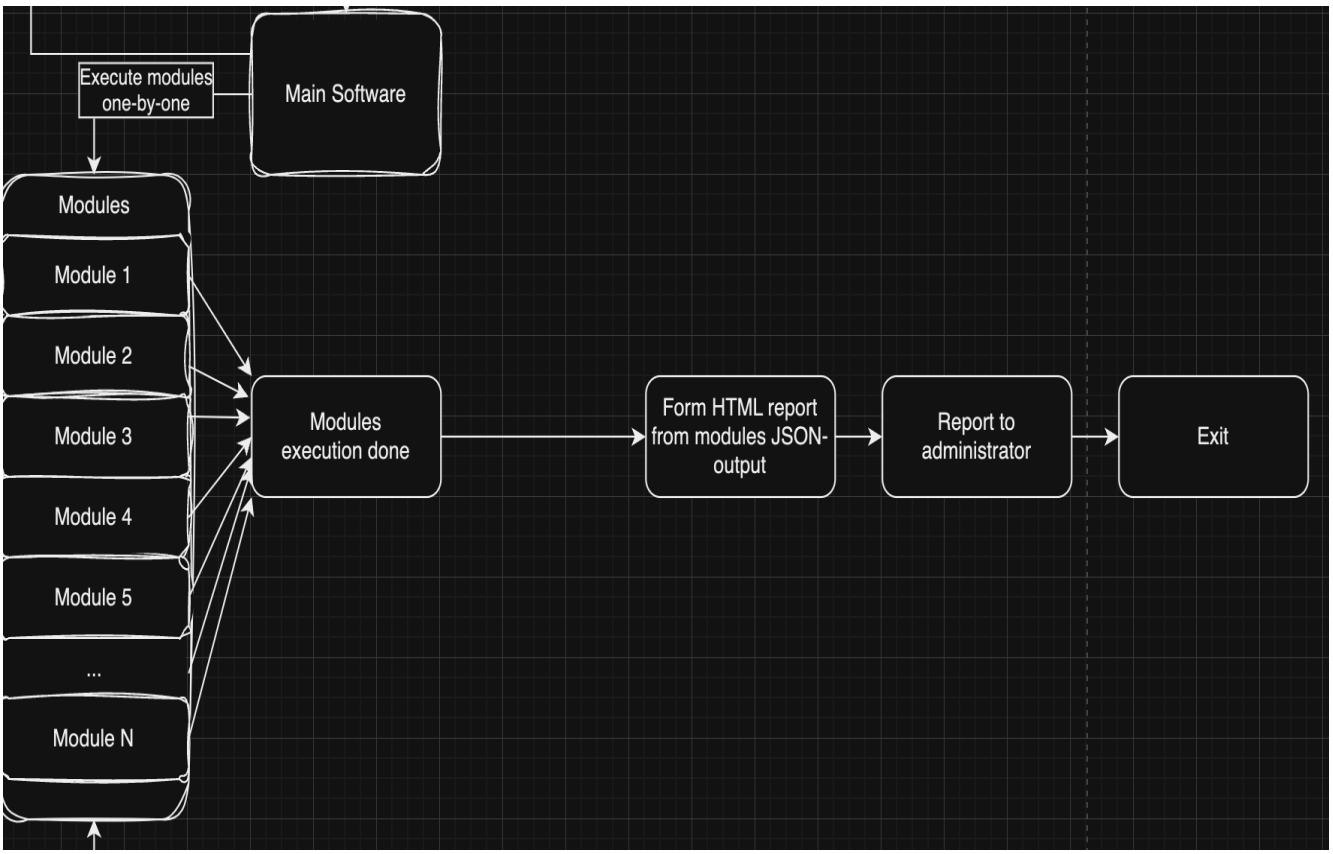


Рисунок 2.3 — демонстрація логіки виконання програми на блок-схемі.

За допомогою мови програмування Python, можна зробити багатофункціональну, зручну, швидку та безпечну програму, що зможе автоматизовано перевіряти налаштування WordPress та повідомляти адміністратора, якщо будуть знайдені вразливості або місконфігурації.

На виході отримаємо продукт, який стане у нагоді будь-якому адміністратору WordPress, що дбає про безпеку своїх ресурсів.

3 ПРАКТИЧНЕ ВИКОРИСТАННЯ ПРОГРАМИ

Тож, із алгоритмом роботи програми на теорії все зрозуміло, тепер час провести декілька тестів на практиці для чіткого розуміння, як програма працює.

Загалом, буде проведено 3 основні тести:

1. Перевірка реагування програми при ініціалізації потенційно шкідливого модуля

2. Перевірка роботи програми у випадку, якщо користувач допустив помилку

3. Практичне використання програми (запуск, відпрацювання модулів, збереження звіту)

3.1 Виявлення шкідливих модулів CMS Wordpress

Тест №1: Шкідливий модуль

Інструмент із забезпечення безпеки має виконувати функції забезпечення безпеки, а не навпаки, він не повинен допомогати зловмисникам. Якщо зловмисник якимось чином потрапляє до системи, де знаходиться WordPress, то програма не повинна стати тим інструментом, що допоможе зловмиснику підвисити привілегії та отримати доступ адміністративного рівня.

Проблема: Зловмисник потрапляє на Linux-сервер де знаходиться WordPress. Якщо він проник на веб-сервер через вразливість веб-серверу або самого WordPress, в нього будуть права користувача www-data, тобто дуже обмежені. Будь-який зловмисник буде прагнути підвищити свої права на веб-сервері до рівня root, тобто здійснити атаку типу Privilege Escalation [15].

Якщо адміністративний користувач залишив директорію із програмою HardPressor у директорії WordPress та поставив автоматичне сканування, подальша атака може виглядати так:

- Зловмисник створює шкідливий модуль, який буде завантажуватись із програмою та робити буквально “нічого”, окрім виконання шкідливого коду
- Програма запуститься автоматично (як сервіс або запланована робота cron), або її запустить адміністратор

- Програма завантажить шкідливий модуль
- Шкідливий модуль виконає код від імені адміністратора, тобто вже із підвищеними привілегіями
- Зловмисник підвищує привілегії на сервері та має більшу можливість потрапити далі в мережу

Рішення:

- Заборона запуску програми, якщо ідентифікатор (UID або EUID[16]) користувача дорівнює <1000 (це означає, що програма запускається якимось системним сервісом / користувачем або від імені root, що є небезпечним), обов'язкове повідомлення адміністратора

```
if os.getuid() < 1000 and OsPlatformName != "darwin":
    self.notifier.send_message("It seems that on your server where HardPressor is located, someone is trying to do something nasty
    exit("Running as root or any other service account is disabled for security reasons!")
```

Рисунок 3.1 — на самому запуску, програма перевіряє спеціальний ідентифікатор користувача в *nix-подібній системі. Якщо користувач системний або являється root — програма не запуститься із міркувань безпеки.

- Заборона ініціалізації модуля, якщо він містить "заборонені слова", такі як: exec, eval, system, sys, base64decode тощо. Модулі, які містять подібні ключові слова, можуть бути потенційно небезпечними та приводити до виконання шкідливого коду.

```
def security_module(self, module_file_path):
    banned_words = ['base64', 'exec', 'eval', 'system', 'sys', 'b64decode'] #add more banned words & obfuscation checks!
    with open(module_file_path, 'r') as module_file:
        module_file_data = module_file.read()
        for word in banned_words:
            if word in module_file_data:
                self.notifier.send_message(f"[ALERT] Banned words detected in module {module_file_path}! Banned word: {word}!")
                print(f"[VERBOSE][ALERT] Module {module_file_path} has been declined! Banned word detected: {word}") if self.verbosity_
                return {'success': False, 'message': f"Banned word detected: `{word}`", "data": None, "code": 403}
    return {'success': True, 'message': "", "data": None, "code": 0}
```

Рисунок 3.2 — якщо в модулі присутнє «заборонене слово» - модуль не буде ініціалізовано, а адміністратор буде повідомлений про таку спробу.

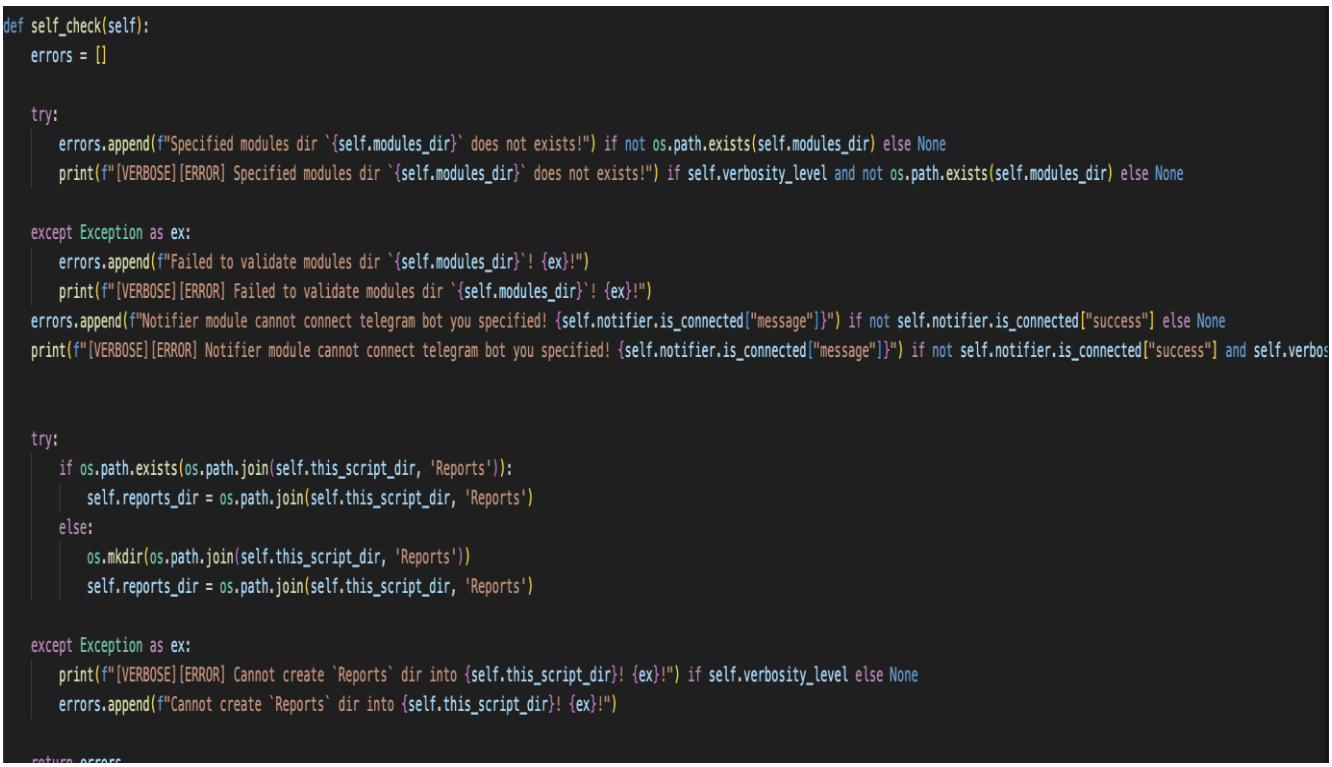
Розуміння базових понять та основ «безпечного коду» допомогає створити контрольований та корисний інструмент, а не "троянського коня"!

3.2 Виявлення місконфігурацій CMS Wordpress

Проблема: Задля створення максимально ефективного продукту із мінімалізацією можливих помилок збоку користувача, необхідно розуміти одну річ – не всі люди однакові та не всі мають високий рівень комп’ютерної грамотності. Через це, багато програмних рішень, які розраховані на користувачів із різним рівнем знань, треба “хардкодити”, тобто із самого початку закласти в програмний код такі алгоритми, які будуть обробляти кожну дію користувача та перевіряти правильність введення кожного аргумента та поля, де користувач потенційно може зробити помилку.

Рішення: Саме для цього, в програмі зроблена функція self_check(), яка перевіряє наявність помилок на самому початку використання. До таких помилок відносяться:

- Помилки заповнення конфігураційного файлу
- Помилки підключення модуля сповіщення через Telegram
- Відсутність потрібних для роботи програми директорій та/або файлів



```
def self_check(self):
    errors = []

    try:
        errors.append(f"Specified modules dir `{self.modules_dir}` does not exists!") if not os.path.exists(self.modules_dir) else None
        print(f"[VERBOSE][ERROR] Specified modules dir `{self.modules_dir}` does not exists!") if self.verbosity_level and not os.path.exists(self.modules_dir) else None

    except Exception as ex:
        errors.append(f"Failed to validate modules dir `{self.modules_dir}`! {ex}!")
        print(f"[VERBOSE][ERROR] Failed to validate modules dir `{self.modules_dir}`! {ex}!")
    errors.append(f"Notifier module cannot connect telegram bot you specified! {self.notifier.is_connected["message"]}") if not self.notifier.is_connected["success"] else None
    print(f"[VERBOSE][ERROR] Notifier module cannot connect telegram bot you specified! {self.notifier.is_connected["message"]}") if not self.notifier.is_connected["success"] and self.verbos


    try:
        if os.path.exists(os.path.join(self.this_script_dir, 'Reports')):
            self.reports_dir = os.path.join(self.this_script_dir, 'Reports')
        else:
            os.mkdir(os.path.join(self.this_script_dir, 'Reports'))
            self.reports_dir = os.path.join(self.this_script_dir, 'Reports')

    except Exception as ex:
        print(f"[VERBOSE][ERROR] Cannot create `Reports` dir into {self.this_script_dir}! {ex}!") if self.verbosity_level else None
        errors.append(f"Cannot create `Reports` dir into {self.this_script_dir}! {ex}!")

    return errors
```

Рисунок 3.3 - У випадку, якщо програма знаходить помилки під час “селф-тестування”, вона виводить їх користувачу на екран та закривається. Інструкції щодо виправлення помилок будуть надані користувачу.

Мінімалізація помилок та обробка кожного кроку виконання робить програму більш гнучку, просту та розумілу для користувачів будь-якого рівня, як професіональних адміністраторів веб-ресурсів, так і для початківців.

3.3 Інтерфейс та загальна послідовність роботи системи тестування

Тест №3: Робота програми в «бойових умовах»

Із теорією зрозуміло, настав час практичного використання та тестування програми. Розберемо практичне використання програми крок за кроком:

Крок 1: Створення конфігураційного файлу

Із самого початку, треба створити конфігураційний файл, приклад якого є в директорії Hardpressor/Core/sample_config_unencrypted.json, для початку розберемо із чого він складається:

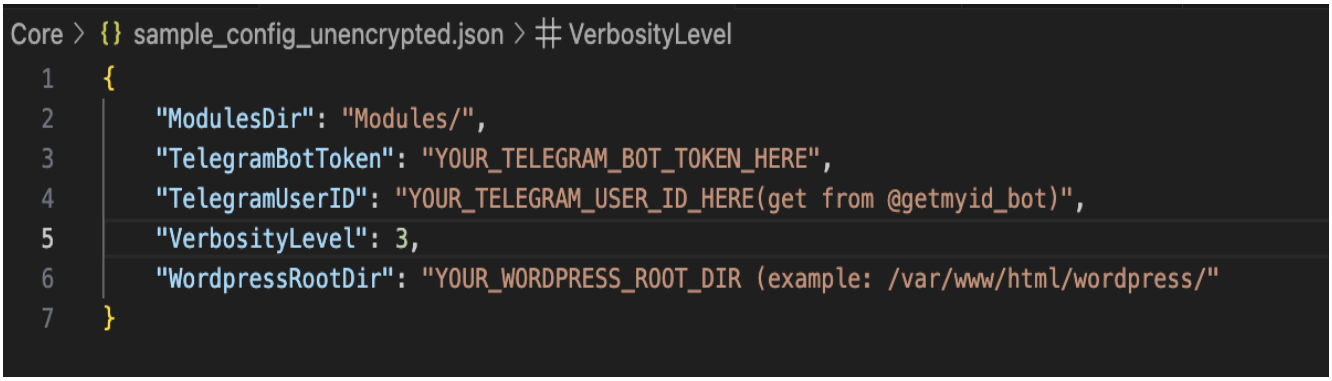


Рисунок 3.4 — приклад(шаблон) конфігураційного файлу програми

- ModulesDir : string – директорія, в якій розташовані модулі HardPressor.
- TelegramBotToken : string – Токен телеграм-боту, якій буде надсилати повідомлення про статус виконання. Отримати токен можна у “батька” ботів Телеграм [19].
- TelegramUserID : string – ID користувача, який буде отримувати повідомлення від боту. Отримати свій UserID можна через спеціального бота [20].

  Важливий момент! Після створення боту, треба почати з ним листування. Боти не вміють писати першими, якщо у вас немає діалогу із ним!
- VerbosityLevel : bool – Параметр, відповідаючий за “рівень деталізації” роботи програми. Якщо виставлений true – користувач буде отримувати більше повідомлень про статус виконання.
- WordPressRootDir : string – директорія, де розташований WordPress.

Приклад заповненого конфігураційного файлу:

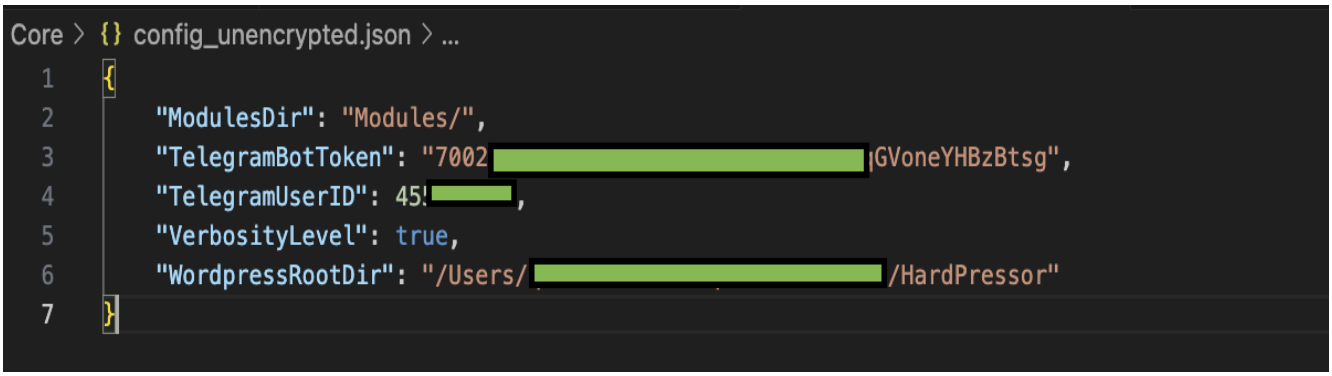


Рисунок 3.5 — заповнений конфігураційний файл програми, деякі дані сховані з міркувань безпеки.

Крок 2: Запуск програми

Програма запускається через командний інтерфейс за допомогою Python. Аргументом -c треба вказани шлях до конфігураційного файлу.

```
qwertec@CA38-LFQGX3439T HardPressor % python3.12 -u "/Users/qwertec/Desktop/Devel/InDev/HardPressor/hardpressor.py" -c Core/config_unencrypted.json
ALERT! Unencrypted config detected! Please, secure your config with password!
>>>
Successfully loaded and encrypted your config into new path: Core/config_unencrypted.json.bin!
Please, backup and remove your old unencrypted config (Core/config_unencrypted.json.bin)!
Successfully loaded your config
```

Рисунок 3.6 — запуск програми із командного інтерфейсу із вказанням конфігураційного файлу в якості аргументу (-c)

При першому запуску, програма скаже вам, що ваш конфігураційний файл не зашифрований та запропонує встановити пароль для його шифрування. Після цього, в тій самій директорії, де знаходиться конфігураційний файл, буде створена зашифрована версія вашої конфігурації із поширенням .bin. Не зашифрований конфігураційний файл тепер можна (навіть потрібно) видалити.

Наступного разу, при вказанні зашифрованого конфігураційного файлу, програмі буде потрібен пароль шифрування, яким вона спробує розшифрувати пароль. Якщо пароль вірний – конфігурацію буде завантажено, якщо ні – буде виведена помилка.

Крок 3: Очікування виконання

Коли програма запуститься та виконає роботу, телеграм-бот повідомить про це

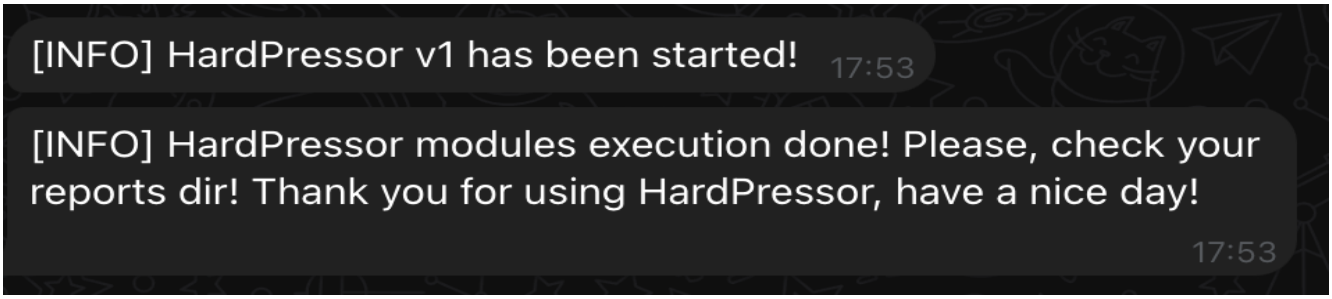


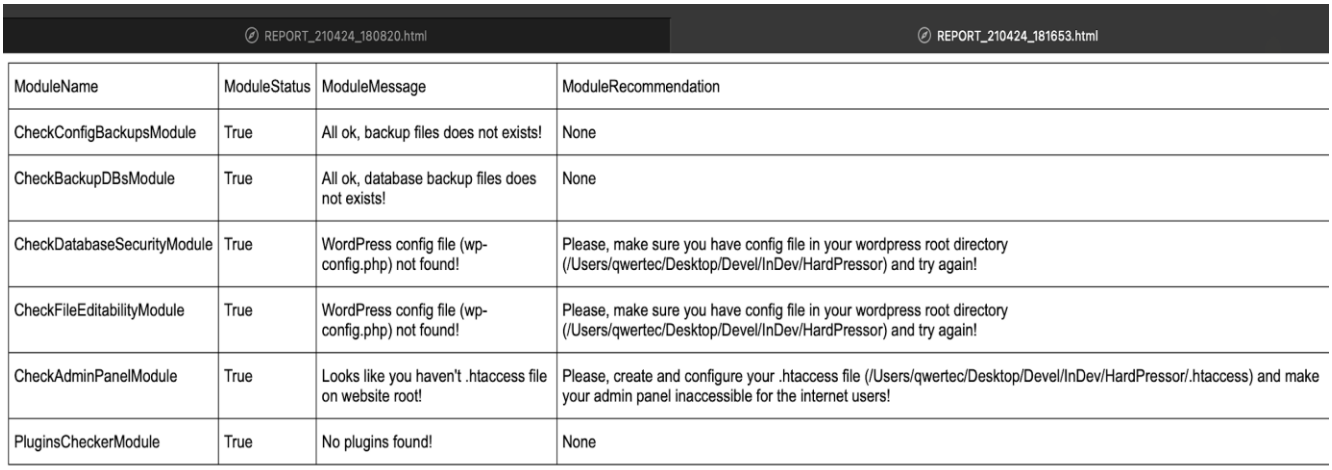
Рисунок 3.7 — програма повідомляє через телеграм про статус виконання

Як можна побачити, швидкість виконання програми дуже помітна. На повну перевірку пішло меньше хвилини. Результати роботи та більш детальна інформація буде виведена в консоль:

Рисунок 3.8 — програма завершила своє виконання та вивела інформацію в консоль

Як можна побачити, в текстовому варіанті виводу є детальна інформація про використані модулі, їх статус виконання (успіх або невдача), повідомлення від модуля та його рекомендації по виправленню тієї чи іншої вразливості та/або місконфігурації, якщо такі були знайдені.

Також, в директорії із програмою, буде створена тека “Reports”, у якій в форматі html буде збережена графічна форма звіту із часовою відміткою. Виглядає вона так:



ModuleName	ModuleStatus	ModuleMessage	ModuleRecommendation
CheckConfigBackupsModule	True	All ok, backup files does not exists!	None
CheckBackupDBsModule	True	All ok, database backup files does not exists!	None
CheckDatabaseSecurityModule	True	WordPress config file (wp-config.php) not found!	Please, make sure you have config file in your wordpress root directory (/Users/qwertec/Desktop/Devel/InDev/HardPressor) and try again!
CheckFileEditabilityModule	True	WordPress config file (wp-config.php) not found!	Please, make sure you have config file in your wordpress root directory (/Users/qwertec/Desktop/Devel/InDev/HardPressor) and try again!
CheckAdminPanelModule	True	Looks like you haven't .htaccess file on website root!	Please, create and configure your .htaccess file (/Users/qwertec/Desktop/Devel/InDev/HardPressor/.htaccess) and make your admin panel inaccessible for the internet users!
PluginsCheckerModule	True	No plugins found!	None

Рисунок 3.9 — звіт про виконане сканування програмою, збережений в формат HTML.

Звіт поділений на стовбці, а саме:

• ModuleName – назва модулю, що виконував перевірку

• ModuleStatus – статус виконання модулю (True якщо модуль відпрацював без помилок)

• ModuleMessage – повідомлення від модулю

• ModuleRecommendation – рекомендація від модулю (що треба зробити, щоб усунути помилку, якщо вона є)

Як можна побачити, в даному випадку програма не знайшла конфігураційний файл WordPress, зробила зауваження через доступність адміністративної панелі для користувачів інтернету (це небезпечно), а також здійснила пошук резервних копій баз даних, плагінів та конфігурацій.

Користувачі можуть писати власні модулі, залежачи від їх загальних потреб, посібник із написання модулів можна знайти стандартній директорії модулів “Modules/MODULE WRITING DOCS.pdf”

Із “вбудованих” модулів, що йдуть із самого початку, присутні наступні:

- Модуль CheckConfigBackupsModule – модуль, який шукає резервні копії конфігураційних файлів, яки може знайти та прочитати будь-який користувач

із інтернету, що є вкрай небезпечним та може привести до витоку чутливих даних.

- Модуль CheckBackupDBsModule – модуль, який шукає резервні копії баз даних в форматах sql, sqlite, sqlite3 та db. В таких базах даних можуть бути логіни та паролі адміністративних та інших користувачів.
- Модуль CheckDatabaseSecurityModule – модуль, який перевіряє безпечність бази даних WordPress. Якщо база даних розташована не на “локальному хості”, тобто є доступною із інтернету чи локальної мережі – це свідчить про те, що до такої бази даних могуть під’єднатись інші користувачі, що не є безпечним рішенням.
- Модуль CheckFileEditabilityModule – модуль, який перевіряє, чи є можливість редагування файлів із адміністративної панелі. Якщо зловмисник зможе отримати доступ до адміністративної панелі, він зможе “змінити” будь-який існуючий файл та записати туди шкідливий код, після чого банально перейти за посиланням до цього файлу, тим самим, виконавши шкідливий код збоку серверу (Remote Code Execution).
- Модуль CheckAdminPanelModule – модуль, який перевіряє доступність адміністративної панелі із інтернету. Відкрита адміністративна панель – це шлях до того, що її може спробувати зламати будь-який бажаючий. Ідеальний варіант – це коли адміністративна панель доступна лише із під локальної мережі, та не доступна із інтернету.
- Модуль PluginsCheckerModule – модуль, який перевіряє встановлені плагіни та їх версії, для послідуючої перевірки актуальності версій плагінів та їх можливих вразливостей.

Створення програми для сканування локальних сайтів на WordPress і виявлення потенційних помилок конфігурації є значущим кроком у напрямку автоматизації харденінгу цієї популярної платформи.

HardPressor не тільки полегшує процес виявлення вразливостей/місконфігурацій, а й сприяє підвищенню безпеки веб-сайтів на WordPress в умовах постійно зростаючої загрози кібератак.

Ця програма розроблена для того, щоб допомогти адміністраторам сайтів на WordPress своєчасно виявляти і виправляти потенційно небезпечні налаштування, що можуть стати ціллю для зловмисників. Завдяки цьому інструменту, користувачі можуть не тільки швидше реагувати на нові загрози, але й значно зменшити ризики, пов'язані з вразливостями, які можуть виникнути в процесі експлуатації веб-сайтів.

Автоматизація процесу харденінгу з використанням HardPressor дозволяє адміністраторам економити час і ресурси, оскільки програма автоматично проводить детальний аналіз конфігурації та пропонує рекомендації щодо підвищення безпеки. Це особливо важливо в умовах сучасного кіберпростору, де атаки стають все більш складними і витонченими.

Використання такого інструменту забезпечує більш високий рівень захисту даних користувачів, що є критично важливим для збереження довіри клієнтів і підтримки позитивної репутації веб-сайтів.

HardPressor також сприяє підвищенню обізнаності адміністраторів щодо потенційних загроз і найкращих практик безпеки. Інформація, що надається програмою, допомагає не лише вчасно виявляти проблеми, але й навчатися, як їх запобігати в майбутньому. Це створює додатковий рівень захисту, що знижує ймовірність успішних атак на сайти, розгорнуті на WordPress.

Таким чином, розробка та впровадження програмного забезпечення, яке забезпечує комплексний підхід до безпеки веб-додатків, є важливим кроком у боротьбі з кіберзагрозами. HardPressor є відмінним прикладом такого підходу, і його використання може значно підвищити рівень безпеки веб-сайтів на WordPress, забезпечуючи користувачам впевненість у захищеності їхніх даних.

ВИСНОВКИ

WordPress – найпопулярніша (станом на зараз) система керування контентом (CMS) у світі.

Її сфера використання величезна: від маленьких блогів до великих інтернет-магазинів та форумів.

Метою даної роботи стала розробка програмного рішення, яке дозволить адміністраторам WordPress всіх рівнів знань проводити комплексну перевірку своїх ресурсів на предмет місконфігурацій та вразливостей.

Мета даної кваліфікаційної роботи досягнена, а саме:

Розроблене програмне рішення HardPressor, яке дає можливість проводити перевірку сайтів на CMS WordPress

Створені базові модулі програми, які проводять сканування та зберігають вивід із рекомендаціями

Створена система самозахисту програми від зловмисних дій

Створена система сповіщення адміністратора програми про ті чи інші її дії

Таким чином, усі поставлені завдання успішно виконані та спільнота WordPress скоро зможе отримати потужний інструмент, що полегшить роботу із безпечного налаштування сайтів на найпопулярнішій CMS у світі!

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний веб-сайт CMS WordPress, URL: https://wordpress.com/

2. Офіційний веб-сайт Одеської ОДА, URL: https://oda.od.gov.ua/

3. Що таке SEO-оптимізація веб-сайту, URL: https://hostiq.ua/blog/ukr/what-is-seo/

4. Статистика використання CMS WordPress від платформи проведення опитувань w3techs, URL: https://w3techs.com/technologies/overview/content_management

5. Статистика використання CMS WordPress від хостинг-провайдера Hostinger, URL: https://www.hostinger.com/tutorials/wordpress-statistics

6. Shodan — пошукова система Internet-of-Things (IoT), URL: https://shodan.io/

7. Censys — пошукова система Internet-of-Things (IoT), URL: https://censys.io/

8. GitHub — найбільша платформа для хостингу коду та комунікацій IT-фахівців у світі, URL: https://github.com/

9. Google — найбільша пошукова система у світі, URL: https://google.com/

10. Zamościński, P., & Kozieł, G. (2020). Analysis of security CMS platforms by vulnerability scanners. Journal of Computer Sciences Institute, 16, 261–268 URL: https://ph.pollub.pl/index.php/jcsi/article/view/2020

11. Офіційний веб-сайт мови програмування Python, URL: https://python.org/

12. Порівняння швидкості мов програмування, URL: https://kiev.neocities.org/langcmp

13. Офіційний репозіторій бібліотеки requests, URL: https://pypi.org/project/requests/

14. Офіційний веб-сайт бібліотеки cryptography, URL: https://cryptography.io/en/latest/

15. Атака типу «Privilege Escalation», URL: https://www.cynet.com/network-attacks/privilege-escalation/

16. Різниця між EUID, UID та UID, URL: https://book.hacktricks.xyz/linux-hardening/privilege-escalation/euid-ruid-suid

17. Принцип SOLID, URL: URL: https://dou.ua/lenta/articles/solid-principles/

18. Офіційний веб-сайт месенджеру Telegram, https://telegram.org/

19. Офіційний Telegram-бот, що створює інших Telegram-ботів, URL: https://t.me/BotFather

20. Офіційне посилання на Telegram-бота, за допомогою якого можна дізнатись власний ID користувача, URL: https://t.me/getmyid_bot