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 бакалавра

студента *Ткаченко Євгенія Валентиновича*

академічної групи *125-20-2*

спеціальності *125 Кібербезпека*

спеціалізації[1]

За освітньо-професійною програмою *Кібербезпека*

на тему *Система захисту інформації вебсерверу з функцією моніторингу та реагування на інциденти*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Ткач М.О.			
розділів:				
спеціальний	доц. Ткач М.О.			
економічний	к.е.н., доц. Пілова Д.П.	92б	відмінно	

Рецензент				

Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту ***Ткаченко Євгенію Валентиновичу*** **академічної групи** ***125-20-2***
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

на тему ***Система захисту інформації вебсерверу з функцією моніторингу та реагування на інциденти***

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № 469-с

Розділ	**Зміст**	**Термін виконання**
Розділ 1	Загальне використання вебсерверів Nginx та Apache. Сучасні методи захисту інформації на вебсерверах. Системи моніторингу та реагування на інциденти.	15.03.2024
Розділ 2	Засоби для створення надійної системи захисту інформації вебсервера з функцією моніторингу та реагування на інциденти. Використання Zabbix та Python-скрипту для моніторингу. Зовнішній пристрій на основі плати ESP32 з розробленим програмним забезпеченням для віддаленого реагування на інциденти.	10.05.2024
Розділ 3	Розрахунок капітальних витрат, річних експлуатаційних витрат, аналіз економічної ефективності.	11.06.2024

Завдання видано _____ Максим ТКАЧ
(підпис керівника) (ім'я, прізвище)

Дата видачі: 01.04.2024р.

Дата подання до екзаменаційної комісії: 28.06.2024р.

Прийнято до виконання _____ Євгеній ТКАЧЕНКО
(підпис студента) (ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 97 с., 4 рис., 1 табл., 5 додатка, 14 джерел.

Об'єкт розробки: Система захисту інформації вебсерверу з функцією моніторингу та реагування на інциденти.

Предмет розробки: програмно-апаратний комплекс, який включає в себе інструменти для моніторингу вебсервера (Zabbix, скрипт на Python), систему сповіщення через Telegram та зовнішній пристрій на плати ESP32 з розробленим програмним забезпеченням для віддаленого керування живленням сервера.

Мета роботи: забезпечення безпрервної роботи вебсервера за рахунок збільшення часу віддаленого моніторингу за його станом та розширення можливостей для віддаленого керування вебсервером.

У першому розділі розглянуто загальне використання вебсерверів Nginx та Apache, оглянуто сучасні методи захисту інформації на вебсерверах та системи моніторингу та реагування на інциденти.

У другому розділі розглянуто засоби для створення надійної системи захисту інформації вебсерверу з функцією моніторингу та реагування на інциденти.

У третьому розділі здійснено аналіз економічної ефективності впровадження системи моніторингу та реагування на інциденти, оцінку витрат на розробку, впровадження та підтримку, а також потенційні економічні вигоди для організацій.

Практична цінність розробки полягає у підвищенні надійності працездатності вебсервера за допомогою системи моніторингу Zabbix та скриптів на Python; своєчасного сповіщення за допомогою інтеграції з Telegram; віддаленому керуванні мережею вебсервера зовнішнім пристроєм на основі плати ESP32 з розробленим програмним забезпеченням; підвищенні загального рівня безпеки сервера.

СИСТЕМА МОНІТОРИНГУ, РЕАГУВАННЯ НА ІНЦИДЕНТИ, ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ, ESP32, ВВІДДАЛЕНЕ РЕАГУВАННЯ, ВІДДАЛЕНИЙ ДОСТУП, ІНФОРМАЦІЙНА СИСТЕМА, КІБЕРБЕЗПЕКА, УПРАВЛІННЯ ІНФОРМАЦІЙНОЮ БЕЗПЕКОЮ.

ABSTRACT

Explanatory note: 97 pp., 4 pic., 1 table, 5 app, 14 sources.

Object of development: An information security system for a web server with incident monitoring and response functionality.

The subject of the development: Hardware-software complex, which includes tools for monitoring the web server (Zabbix, Python script), a notification system via Telegram, and an external device based on the ESP32 board with custom-developed software for remote power control of the server.

The purpose of the work: Ensuring the continuous operation of the web server by increasing the time of remote monitoring of its condition and expanding the possibilities for remote management of the web server.

In the first chapter: General use of Nginx and Apache web servers, a review of modern methods of information protection on web servers, and systems for monitoring and incident response.

In the second section: Tools for creating a reliable information security system for a web server with monitoring and incident response functionality.

In the third chapter: An analysis of the economic efficiency of implementing the monitoring and incident response system, including cost assessment for development, implementation, and maintenance, as well as potential economic benefits for organizations.

The practical value of the development lies in increasing the reliability of server performance using the Zabbix monitoring system and Python scripts; timely instant notification using integration with Telegram; remote management of the web server network using an external device based on the ESP32 board with custom-developed software; increasing the overall level of server security.

MONITORING SYSTEM, INCIDENT RESPONSE, ECONOMIC EFFICIENCY, ESP32, REMOTE RESPONSE, REMOTE ACCESS, INFORMATION SYSTEM, CYBER SECURITY, INFORMATION SECURITY MANAGEMENT.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

AES - Advanced Encryption Standard

APM - Application Performance Management

CSRF - Cross-Site Request Forgery

DDoS - Distributed Denial of Service

DoS - Denial of Service

ECC - Elliptic Curve Cryptography

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

IAM - Identity and Access Management

IDS/IPS - Intrusion Detection System / Intrusion Prevention System

IMAP - Internet Message Access Protocol

IP - Internet Protocol

NGFW - Next-Generation Firewall

POP3 - Post Office Protocol 3

RSA - Rivest-Shamir-Adleman

RTMP - Real-Time Messaging Protocol

SMTP - Simple Mail Transfer Protocol

SQL - Structured Query Language

SSL/TLS - Secure Sockets Layer / Transport Layer Security

URL - Uniform Resource Locator

VPN - Virtual Private Network

WAF - Web Application Firewall

ЗМІСТ

с.

ВСТУП

Вебсервер - це програмне забезпечення або апаратний пристрій, який обробляє запити від клієнтів, як-от веббраузери, і надає їм доступ до вебсторінок, файлів або інших ресурсів через Інтернет. Основне завдання вебсервера - передавати вебсторінки користувачам, які запитують їх за допомогою браузера.

Основні функції вебсервера:

1. Вебсервер обробляє запити від клієнтів за протоколом HTTP. Ці запити можуть бути запитом на отримання вебсторінки, відправлення форми або завантаження файлу;
2. Відповідно до запиту, вебсервер надсилає відповідь клієнту. Це може бути HTML-сторінка, зображення, відео, документи або будь-який інший контент. Відповідь також може містити коди статусу HTTP, які повідомляють клієнту про результат обробки запиту (наприклад, 200 OK, 404 Not Found);
3. Обробка статичних і динамічних контентів:

- вебсервери можуть надавати файли, що не змінюються (HTML, CSS, JavaScript, зображення);
- вебсервери можуть працювати разом з іншими програмами або скриптами (наприклад, PHP, Python, Ruby), щоб генерувати динамічний контент на основі даних або запитів користувачів;

4. Вебсервери можуть вимагати від користувачів проходити аутентифікацію та надавати доступ до певних ресурсів лише авторизованим користувачам;
5. Вебсервери ведуть журнали відвідувань та дій, які можуть використовуватися для аналізу роботи сайту, виявлення помилок або атак на сервер;
6. Вебсервери можуть використовувати різні засоби захисту для запобігання несанкціонованому доступу, DDoS-атакам та іншим загрозам.

На сьогоднішній день існує багато технологій та методів для забезпечення захисту інформації на вебсерверах. Серед них можна виділити:

- міжмережеві екрани (фаєрволи), які контролюють трафік між внутрішніми і зовнішніми мережами, запобігаючи несанкціонованому доступу;

- системи виявлення та запобігання вторгнень (IDS/IPS), що аналізують мережевий трафік та виявляють підозрілі дії, надаючи можливість негайного реагування;
- антивірусне програмне забезпечення, яке сканує та видаляє шкідливі програми;
- шифрування даних, що забезпечує захист інформації при її передачі та зберіганні;
- політики доступу та управління правами, які визначають рівні доступу користувачів до різних ресурсів.

Незважаючи на це, існуючі рішення мають обмеження і не завжди можуть забезпечити необхідний рівень безпеки. Однією з основних проблем є недостатня швидкість реагування на інциденти, що може призвести до значних втрат навіть при наявності захисних засобів.

Ефективний захист вебсервера вимагає не лише наявності засобів запобігання атакам, але й постійного моніторингу його роботи для своєчасного виявлення підозрілих дій та інцидентів. Моніторинг дозволяє в реальному часі відстежувати активність на сервері, аналізувати поведінку користувачів та мережевий трафік, виявляти аномалії та потенційні загрози.

Реагування на інциденти включає швидке вжиття заходів для нейтралізації загрози, мінімізації шкоди та відновлення нормальної роботи системи. Це може включати автоматичне блокування підозрілої активності, інформування адміністраторів про інцидент, запуск процесів відновлення даних та інші дії.

Завдання роботи включають:

1. Обрати моніторинговий сервіс для виконання моніторингу стану вебсерверу та інтеграція сповіщень через Telegram;
2. Опис використання Python-скрипта для моніторингу системи із сповіщеннями через Telegram-бота;
3. Розробка зовнішнього пристрою для контролю живлення машини для вебсерверу на основі плати ESP32 з розробленим програмним забезпеченням із віддаленим керуванням через Telegram.

Практичне значення роботи полягає у запровадженні комплексної системи моніторингу за станом вебсервера за допомогою багатофункціонального програмного забезпечення Zabbix та Python-скрипта з використання бібліотеки psutil, миттєвого сповіщення через Telegram та реагування на інциденти за допомогою управлінням живлення зовнішнім пристроєм на основі плати ESP32 з розробленим програмним забезпеченням.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Загальне використання вебсерверів

У сучасному світі, де цифрові технології є невід'ємною частиною бізнесу, освіти, комунікацій та багатьох інших сфер, вебсервери відіграють важливу роль у забезпеченні доступу до інформаційних ресурсів та наданні різноманітних послуг. З кожним роком обсяги інформації, що передаються та зберігаються на вебсерверах, значно зростають, що робить їх привабливими цілями для зловмисників. Кібератаки на них можуть призвести до значних фінансових втрат, порушення роботи організацій, компрометації конфіденційних даних та шкоди репутації.

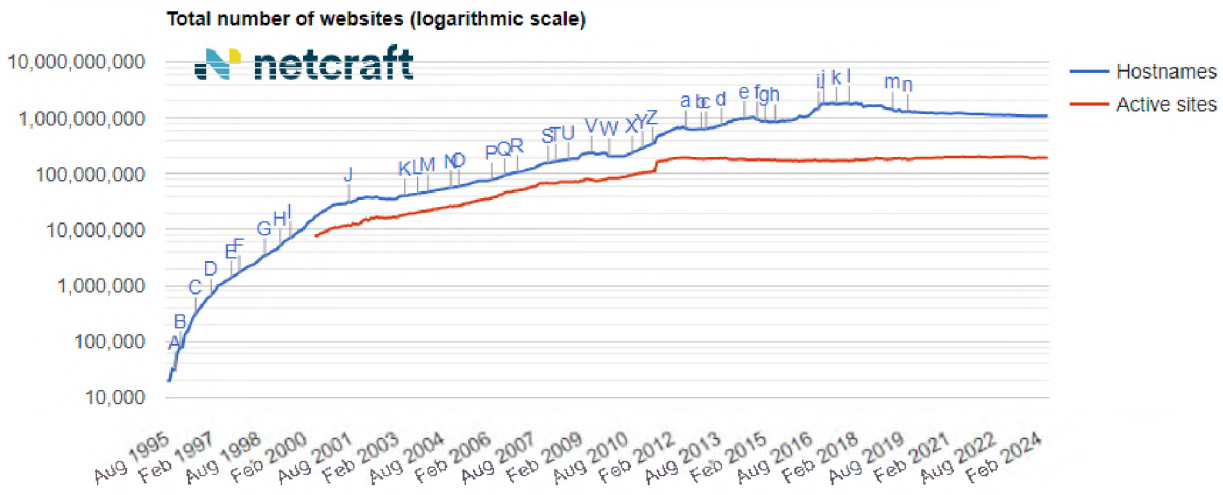

Рисунок 1.1 - Статистика росту кількості вебсайтів

Станом на травень 2024 року[14]:

- кількість вебсайтів - 1 097 398 145 сайтів;
- кількість доменів - 268 0137 699 доменів;
- кількість вебсерверів - 12 898 459 вебсерверів.

Ці цифри свідчать про величезний обсяг інформації та послуг, що надаються через інтернет. Для ефективного обслуговування такого масштабного трафіку використовуються різні вебсервери. Найпопулярнішими серед них є Nginx та Apache.

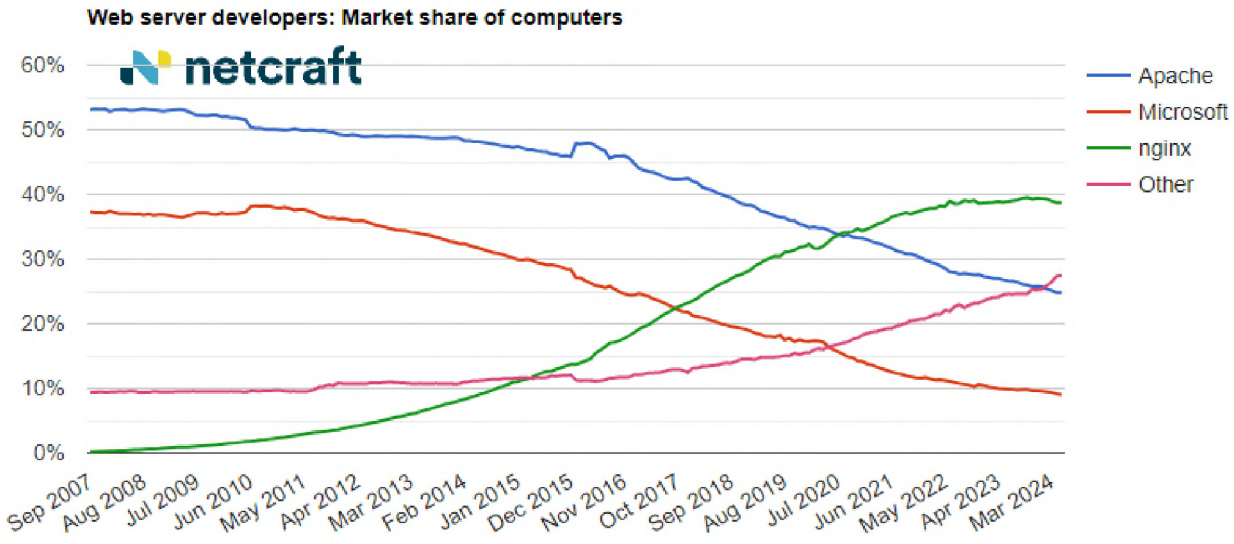

Рисунок 1.2 - Процент використання вебсервісів різних виробників

За даними на травень 2024 з рисунку 1.1.2 - більша частина(38,7%) вебсерверів працює на nginx - 4 991 558 серверів, після нього одразу йде Apache - 3 200 253 машини з ним.

1.1.1 Вебсервер Nginx

NGINX — це вебсервер і поштовий проксі, який працює під управлінням операційних систем сімейства Linux/Unix і Microsoft[4].

Спочатку продукт розроблявся тільки під Unix-системи. Перші релізи тестувалися на FreeBSD, Linux, Solaris, але пізніше розробник додав сумісність із платформою Windows.

Nginx є лідером на ринку вебсерверів. Багато великих компаній використовують його на своїх порталах.

На відміну від інших продуктів цього сегмента, Nginx використовує інший принцип обробки вхідних даних. ПО розбиває кожен запит користувача на кілька дрібних, спрощуючи таким чином обробку кожного. У термінології Nginx вони отримали назву робоче з'єднання.

Після обробки кожне з'єднання збирається в одному віртуальному контейнері, щоб трансформуватися в єдиний початковий запит, а після відправляється

користувачу. Одне з'єднання може одночасно обробляти до 1024 запитів кінцевого користувача.

Для зменшення навантаження на оперативну пам'ять вебсервер використовує виділений сегмент пам'яті, який називається «пул» (pool). Він динамічний і розширюється при збільшенні довжини запиту.

Вебсервер застосовується в наступних ситуаціях:

- виділений порт або IP-адреса, якщо на сервері присутня велика кількість статичного матеріалу (зображення, тексти тощо) або файлів для завантаження користувачами, то Nginx використовують, щоб виділити під ці операції окрему IP-адресу або порт. Таким чином навантаження на сервер розподіляється;
- проксі-сервер, коли користувач завантажує сторінку сайту, на якій розташований статичний контент, Nginx спочатку кешує дані у себе, а потім повертає результат. При наступних запитах цієї сторінки відповідь відбувається в рази швидше;
- розподіл навантаження, при запиті сторінки сайту користувачу видається відповідь у синхронній послідовності. Nginx використовує асинхронний режим. Всі запити обробляються на різних етапах. Такий підхід підвищує швидкість обробки;
- поштовий сервер, оскільки у вебсервер вбудовані механізми автентифікації, його часто використовують для перенаправлення на поштові сервіси після проходження авторизації клієнтом.

У Nginx вбудовані механізми захисту. Інформація передається по зашифрованому каналу через протоколи SSL/TLS.

Вебсервер Nginx ідеально підходить для сайтів, на яких міститься в основному статичний контент. Він також здатний виступати як редирект для поштових сервісів або в ролі проксі-сервера. Простота і гнучкість налаштування дозволяє масштабувати продукт без особливих зусиль.

1.1.2 Вебсервер Apache

Apache — вебсервер, що розповсюджується безкоштовно. Програмне забезпечення є кросплатформним продуктом, тобто працює на різних операційних системах (Linux, Windows, Solaris тощо)[5].

Основні відмінності від конкурентів — надійність і гнучкість. Apache працює за принципом модулів. Клієнт спочатку встановлює ядро, а потім підключає необхідні моделі під свої задачі.

Apache обробляє запити клієнт-сервер за протоколом HTTP. Вебсервер має 3 модулі мультипроцесингу, які відповідають за обробку запиту користувача:

- mpm_prefork створює на кожен запит клієнта окремий процес. Чим менше процесів, тим швидше обробляються задачі користувача. Під кожен запит на сервері виділяється об'єм оперативної пам'яті. Цей модуль зазвичай використовують у парі з іншими зовнішніми компонентами, які не вміють працювати в багатопоточному режимі, наприклад, mod_php;
- mpm_worker створює процеси, які працюють відразу з кількома потоками. Один потік обробляє одне з'єднання користувача. Модуль швидше обробляє запити користувача і менше навантажує ресурси сервера;
- mpm_event розроблений для роботи з постійними (keep-alive) з'єднаннями. Розділяє потоки на постійні і активні, що дозволяє підвищити швидкість обробки запитів користувачів;

Існують модулі, які оптимізовані під сімейство операційних систем. Наприклад, модуль mpm_winnt працює з ОС Windows, mpm_netware — з Netware.

Розробники зробили Apache максимально доступним і простим в експлуатації. Спільнота регулярно випускає патчі та оновлення.

Apache має функцію віртуалізації. На одній IP-адресі може існувати багато віртуальних сайтів. Кожен хост налаштовується під індивідуальні вимоги клієнта: налаштування прав доступу до файлів, обмеження споживання обчислювальних потужностей фізичного сервера тощо.

Для вебсервера існують модулі, які додають у ядро Apache підтримку інших (СУБД написана на С) мов, наприклад, PHP, Ruby, Perl.

Apache має низку вбудованих функцій безпеки. До них належать система авторизації користувачів, обмеження доступу за IP-адресами, розмежування прав доступу до каталогів і файлів на сервері тощо. Доступна функція запуску процесів на підставі ідентифікаторів користувачів або груп.

За необхідності користувач може зашифрувати дані, що передаються між сервером і клієнтом, використовуючи протокол SSL. Додаткові функції захисту підключаються через модуль mod_security.

Вебсервер Apache розроблений для запуску сайту без зайвих зусиль. Інтуїтивний інтерфейс, модульна система, вбудовані механізми захисту та багато інших функцій роблять його одним із найпопулярніших вебсерверів.

1.2 Огляд методів та засобів захисту інформації на вебсерверах

Захист вебсерверів є критично важливим аспектом забезпечення безпеки в сучасному цифровому світі. Кібератаки можуть призвести до значних фінансових втрат, компрометації конфіденційних даних, порушення роботи організацій та шкоди репутації. Розглянемо основні методи та засоби захисту інформації на вебсерверах[7].

1. Шифрування Даних (SSL/TLS)

SSL/TLS — це криптографічні протоколи, які забезпечують захищений канал зв'язку між вебсервером і клієнтом. Основні переваги:

- конфіденційність: захищає дані від перехоплення під час їх передачі;
- цілісність: гарантує, що дані не були змінені під час передачі;
- аутентифікація: підтверджує справжність вебсервера.

2. Вебфаєрволи (WAF)

Web Application Firewalls (WAF) — це рішення, які захищають вебдодатки від різних типів атак, таких як SQL-ін'єкції, XSS (Cross-Site Scripting), DDoS-атаки тощо. Основні функції:

- фільтрація та моніторинг http/https трафіку: виявлення та блокування шкідливих запитів;
- політики безпеки: налаштування політик для специфічних вебдодатків.

3. Антивірусне ПЗ та Сканери Вразливостей

Антивірусні рішення та сканери вразливостей допомагають виявляти та усувати загрози на вебсервері. Основні функції:

- антивірусні програми: захищають сервери від шкідливого програмного забезпечення, що може бути завантажене або запущене;
- сканери вразливостей: автоматично перевіряють сервери на наявність вразливостей та конфігураційних помилок.

4. Інструменти Контролю Доступу та Управління Ідентичностями

IAM (Identity and Access Management) рішення допомагають керувати доступом до ресурсів вебсервера. Основні функції:

- мультифакторна автентифікація (mfa): використання декількох факторів для підтвердження особи користувача;
- ролі та дозволи: налаштування ролей та доступів на основі принципу найменших привілеїв.

5. Системи Виявлення та Запобігання Вторгненням (IDS/IPS)

IDS/IPS призначені для виявлення та запобігання несанкціонованих доступів або атак. Основні функції:

- IDS: аналізує мережевий трафік на предмет підозрілих дій та сповіщає адміністратора;
- IPS: активно блокує виявлені загрози.

6. Журналювання та Моніторинг

Регулярний моніторинг і аналіз логів допомагають виявляти аномалії та потенційні загрози. Основні функції:

- логування - збереження подій для подальшого аналізу;
- моніторинг - відстеження активності та продуктивності вебсервера в реальному часі.

7. Безпека на Рівні Додатків

Захист коду та належне налаштування додатків є критично важливими для запобігання атак. Основні функції:

- регулярні оновлення: вчасне оновлення серверного програмного забезпечення та додатків;
- захист від XSS та CSRF атак: впровадження відповідних заходів безпеки в коді додатків.

8. Резервне Копіювання та Відновлення

Регулярне резервне копіювання даних забезпечує можливість відновлення після інцидентів безпеки. Основні функції:

- автоматичні резервні копії: налаштування автоматичних резервних копій важливих даних;
- план відновлення: відпрацьовані процедури відновлення після аварій.

Захист інформації на вебсерверах вимагає комплексного підходу, що включає використання різноманітних технологій та методів. Використання SSL/TLS для шифрування даних, WAF для захисту від вебатак, антивірусного ПЗ та сканерів вразливостей, інструментів IAM, систем IDS/IPS, моніторингу та журналювання, безпеки на рівні додатків, а також регулярного резервного копіювання забезпечує надійний рівень захисту інформації на вебсерверах.

1.2.1 Межмережеві екрани

Міжмережевий екран (МЕ, брандмауер або Firewall) - це програмно-апаратний або програмний комплекс, який відстежує мережеві пакети, блокує або дозволяє їх проходження. У фільтрації трафіку брандмауер базується на встановлених параметрах - частіше всього їх називають правилами МЕ.

Сучасні міжмережеві екрани розташовані на периферії мережі, обмежують транзит трафіку, встановлення небажаних з'єднань та подібні дії за допомогою засобів фільтрації та аутентифікації.

Основне завдання МЕ - це фільтрація трафіку між зонами мережі. Він може використовуватися для розмежування прав доступу в мережу, захисту від

сканування мережі компанії, проведення мережевих атак. Проще кажучи, міжмережевий екран - це один з пристроїв, за допомогою якого забезпечується мережева безпека компанії.

Функції МЕ:

1. Зупинити підроблення трафіку: міжмережевий екран виявляє спроби замаскувати трафік під дані вашої компанії, але відправлені з іншого ip-адреси. він виявить це як підставну адресацію і перешкодить спробі вторгнутися в вашу мережу;

2. Захист від DDoS-атак: міжмережевий екран здатний розпізнати атаки типу DDoS, коли зловмисники намагаються перевантажити ресурси компанії, надсилаючи багато запитів з інфікованих пристроїв. Він виявить такі атаки, визначить їх характеристики і передасть інформацію про них брандмауеру для подальшої фільтрації шкідливого трафіку;

3. Заборонити передачу даних на невідомий IP-адрес: якщо співробітник компанії завантажив шкідливий файл і заразив свій комп'ютер, що призвело до витоку корпоративної інформації, міжмережевий екран автоматично зупинить спроби віруса передати інформацію на невідомий IP-адрес, запобігаючи таким чином втратам даних.

Мережевий трафік, який проходить через брандмауер, порівнюється з правилами, щоб визначити, чи потрібно його пропускати.

Правило міжмережевого екрана складається з умови (IP-адреса, порт) і дії, яку необхідно застосувати до пакетів, які відповідають заданій умові. До дій відносяться команди дозволити (accept), відхилити (reject) і відкинути (drop). Ці умови вказують МЕ, що саме потрібно зробити з трафіком:

- дозволити - пропустити трафік;
- відхилити - не пропускати трафік і вивести користувачу повідомлення-помилку "недоступно";
- відкинути - заблокувати передачу і не виводити відповідне повідомлення.

МЕ поділяються на два основні типи: апаратні та програмні.

Апаратний міжмережевий екран - це, як правило, спеціалізоване обладнання, компоненти якого (процесори, плати і т.д.) спроектовані спеціально для обробки трафіку.

Вони працюють на спеціальному програмному забезпеченні - це необхідно для підвищення продуктивності обладнання. Прикладами апаратних міжмережевих екранів є такі пристрої, як Cisco ASA, FortiGate, Cisco FirePower, UserGate та інші.

Апаратні міжмережеві екрани більш потужні порівняно з програмними, однак це впливає на вартість рішень. Нерідко вона у декілька разів вища, ніж у програмних аналогів.

Програмний міжмережевий екран - це програмне забезпечення, яке встановлюється на реальний або віртуальний пристрій.

Через такий міжмережевий екран весь трафік перенаправляється в мережу компанії. До програмних відносяться брандмауер у Windows і iptables у Linux.

Програмні міжмережеві екрани, як правило, дешевше і можуть встановлюватися не тільки на межах мережі, але і на робочих станціях користувачів. З основних недоліків - менша пропускна здатність і складність налаштування у деяких випадках.

Міжмережевий екран з контролем стану сеансів аналізує всю активність користувачів від початку до кінця - кожну встановлену користувальницьку сесію. На основі цих даних він визначає типову та нетипову поведінку користувача. Якщо поведінка в межах сесії здається йому нетиповою, МЕ може заблокувати трафік.

Отже, рішення про схвалення або блокування вхідного трафіку приймається не лише на основі заданих адміністратором правил, але й з урахуванням контексту - даних, отриманих з попередніх сесій. Брандмауери з відстеженням стану сесій вважаються набагато більш гнучкими, ніж класичні міжмережеві екрани.

Новітньою технологією в сфері МЕ є Межмережевий екран наступного покоління (NGFW).

Next-generation firewall - це файрвол, який має основну відмінність у тому, що він може здійснювати фільтрацію не лише на рівні протоколів і портів, але й на рівні додатків та їх функцій. Це дозволяє ефективніше відбивати атаки та блокувати

шкідливу активність. Також NGFW має більш детальну настройку політик безпеки та рішень для великого бізнесу.

Всі NGFW мають основні загальні функції антивірус, антиспам, IDS/IPS, а також спеціальні:

- Deep Packet Inspection (DPI) - ця технологія виконує детальний аналіз пакетів на верхніх рівнях моделі OSI, що дозволяє проводити аналіз трафіку на основі поведінки і розпізнавати додатки, які не використовують заздалегідь відомі заголовки і структури даних;
- Sandboxing - це метод, за яким файл автоматично поміщається в ізольоване середовище для тестування, відоме як "пісочниця". У цьому середовищі можна ініціювати виконання підозрілої програми або перехід за URL-адресою, яку зловмисник може додати до листа. Пісочниця створює безпечне місце для встановлення та виконання програми, не піддаваючи небезпеці решту системи;
- фільтрація за URL-адресою, або вебфільтр, - це можливість блокування доступу до вебсайтів або інших вебдодатків за ключовим словом у адресі;
- інспектування SSL дозволяє міжмережевому екрану нового покоління встановлювати SSL-сесію з клієнтом та сервером. Завдяки цьому існує можливість переглядати зашифрований трафік і застосовувати до нього політики безпеки;
- контроль додатків (Application Control) використовується для обмеження доступу до додатків, їх функцій або до цілих категорій додатків. Все це використовує функції відстеження стану додатків, запущених користувачем, в реальному часі;
- веббрандмауер для вебдодатків (Web Application Firewall) - це сукупність правил і політик, спрямованих на запобігання атак на вебдодатки;
- аутентифікація користувачів - можливість налаштовувати індивідуальні правила для кожного користувача або групи.

Міжмережеві екрани (фаєрволи) є невід'ємною частиною комплексної системи захисту інформації. Вони забезпечують надійний бар'єр між внутрішньою мережею та зовнішніми загрозами, захищаючи від несанкціонованого доступу та мережевих атак. Використання сучасних фаєрволів, таких як pfSense, Cisco ASA та

Check Point FireWall-1, дозволяє забезпечити високий рівень безпеки, ефективність та гнучкість управління мережею.

1.2.2 Системи виявлення та запобігання вторгнень (IDS/IPS)

Система виявлення вторгнень (IDS) і система запобігання вторгнень (IPS) забезпечують більш високий рівень захисту мережі порівняно з традиційними засобами захисту, такими як антивіруси, спам-фільтри та файерволи[8].

Антивірус аналізує файли, спам-фільтр аналізує листи, файервол - з'єднання за IP-адресою. IDS/IPS аналізують дані і мережеву поведінку. Продовжуючи аналогію з правоохоронцями, файервол, поштові фільтри та антивіруси - це рядові працівники, які працюють "на вулиці", а системи виявлення та запобігання вторгнень - це старші офіцери, які працюють у відділі.

Принцип роботи системи виявлення вторгнень (IDS) полягає в визначенні загроз на основі аналізу трафіку, але подальші дії залишаються на відповідальність адміністратора. Системи IDS поділяються на типи залежно від місця встановлення та принципу дії.

Є два найпопулярніших видів IDS за місцем встановлення:

1. Network Intrusion Detection System (NIDS) - ця технологія надає можливість встановити систему на стратегічно важливих місцях мережі та аналізувати вхідний/вихідний трафік всіх пристроїв мережі. NIDS аналізує трафік на глибокому рівні, "заглядаючи" в кожний пакет від канального рівня до рівня додатків. NIDS відрізняється від мережевого екрана, або файерволу. Файервол зафіксовує лише атаки, що надходять ззовні мережі, тоді як NIDS може виявляти і внутрішню загрозу. Мережеві системи виявлення вторгнень контролюють увесь обсяг мережі, що дозволяє не витрачати кошти на додаткові рішення. Але є недолік: NIDS відстежує весь мережевий трафік, споживаючи велику кількість ресурсів. Чим більше обсяг трафіку, тим більше потреба в ресурсах CPU та RAM. Це призводить до помітних затримок у обміні даними та зниження швидкості роботи мережі. Великий обсяг інформації також може "засліпити" NIDS, змушуючи систему пропускати деякі пакети, що робить мережу вразливою;

2. Host-based Intrusion Detection System (HIDS) - ці системи встановлюються на один хост всередині мережі і захищають лише його. HIDS також аналізують всі вхідні та вихідні пакети, але лише для одного пристрою. Система HIDS працює на основі створення знімків файлів: вона робить знімок поточної версії і порівнює його з попереднім, таким чином виявляючи можливі загрози. HIDS найкраще встановлювати на критично важливі машини в мережі, які рідко змінюють конфігурацію.

Всі системи виявлення атак IDS працюють за одним принципом — пошук загрози шляхом аналізу трафіку. Відмінності полягають у самому процесі аналізу. Існує три основних види: сигнатурні, засновані на аномаліях і засновані на правилах.

Сигнатурні IDS працюють за подібним до антивірусного програмного забезпечення принципом. Вони аналізують сигнатури і порівнюють їх з базою даних, яка повинна постійно оновлюватися для забезпечення правильної роботи. Відповідно, основним недоліком сигнатурних IDS є те, що якщо з якихось причин база недоступна, мережа стає вразливою. Також, якщо атака нова і її сигнатура невідома, існує ризик того, що загроза не буде виявлена.

Сигнатурні IDS здатні відстежувати шаблони або стани. Шаблони — це ті сигнатури, які зберігаються в постійно оновлюваній базі. Стани — це будь-які дії всередині системи.

Початковий стан системи — нормальна робота, відсутність атаки. Після успішної атаки система переходить в скомпрометований стан, тобто зараження пройшло успішно. Кожна дія (наприклад, встановлення з'єднання за протоколом, який не відповідає політиці безпеки компанії, активізація ПЗ і т. д.) здатна змінити стан. Тому сигнатурні IDS відстежують не дії, а стан системи.

IDS, засновані на аномаліях, використовують машинне навчання для виявлення загроз, проте для їх правильної роботи необхідний період навчання. Адміністраторам рекомендується протягом перших кількох місяців повністю вимкнути сигнали тривоги, щоб система мала змогу навчитися. Після тестового періоду вона буде готова до роботи.

Система аналізує роботу мережі в поточний момент, порівнює з аналогічним періодом і виявляє аномалії. Аномалії поділяються на три категорії:

1. Статистичні аномалії виявляються, коли система IDS створює профіль звичайної активності (обсяг вхідного/вихідного трафіку, запускаємі програми і т.д.) і порівнює його з поточним профілем;

2. Для виявлення аномалій протоколів IDS-система аналізує комунікаційні протоколи, їх зв'язки з користувачами, програмами і створює профілі;

3. IDS також можуть виявляти аномалії, будь-яку небезпечну або навіть загрозливу активність у мережевому трафіку.

Система запобігання вторгненням (IPS) - це наступний крок в розвитку систем мережевого захисту. IPS не лише повідомляє про загрозу, але й може самостійно вживати заходів.

Наразі практично не існує чистих IPS, на ринку представлений широкий спектр систем IDPS (Intrusion Detection and Prevention Systems). IDPS виявляють атаки і вживають запрограмовані дії.

IDPS-системи допускають певний відсоток хибних негативних (false negative) та хибних позитивних (false positive) реакцій. Щоб мінімізувати помилкові спрацьовування, IDPS дозволяють задати граничні значення для реакцій - наприклад, встановити значення допустимого збільшення трафіку в будні дні. Адміністратор, відповідальний за IDS, задає їх у консолі управління.

Якщо ви вирішуєте встановити в мережу захист, будь то IDS/IPS, UTM або NGFW, постає питання, де його ставити. Насамперед це залежить від типу обраної системи. Так, PIDS не має сенсу ставити перед фаєрволом, усередині мережі, а NGFW включає відразу всі елементи, тому її можна ставити куди завгодно.

Система виявлення вторгнень може бути встановлена перед фаєрволом з внутрішньої сторони мережі. У такому разі IDS буде аналізувати не весь трафік, а лише той, що не був заблокований фаєрволом. Це логічно: навіщо аналізувати дані, що блокуються. До того ж, це знижує навантаження на систему.

IDS ставлять також і на зовнішній межі мережі після фаєрволу. У такому випадку вона фільтрує зайвий шум глобальної мережі, а також захищає від

картування мережі ззовні. При такому розташуванні система контролює рівні мережі з 4 до 7 і відноситься до сигнатурного типу. Таке розгортання скорочує кількість хибнопозитивних спрацьовувань.

Інша часта практика — встановлення кількох копій системи виявлення вторгнень у критичних місцях для захисту мережі за пріоритетом важливості. Також допускається встановлення IDS усередині мережі для виявлення підозрілої активності.

Система запобігання вторгненням (IDPS) - це важливий інструмент кібербезпеки, який доповнює брандмауери та системи виявлення вторгнення (IDS). IDPS може виявляти та блокувати кібератаки, щоб захистити мережу від шкоди.

1.2.3 Антивірусне програмне забезпечення

Антивірус для вебсерверів — це спеціалізований тип програмного забезпечення, розроблений для захисту вебсерверів від різних типів кіберзагроз[9]. На відміну від стандартних антивірусних програм, призначених для персональних комп'ютерів або корпоративних мереж, антивірусні рішення для вебсерверів спеціально розроблені для захисту серверів, на яких розміщуються вебсайти та вебдодатки.

Основні загрози для вебсерверів включають атаки типу Distributed Denial of Service (DDoS), шкідливе програмне забезпечення, міжсайтовий скриптинг (XSS), SQL-ін'єкції та різні форми несанкціонованого доступу або витоку даних. Ці атаки можуть поставити під загрозу конфіденційні дані, порушити роботу сервісу та завдати шкоди репутації компанії.

Ключові функції антивірусних програм для вебсерверів:

- реальний моніторинг та сканування: постійно сканує сервер на наявність шкідливої активності та відомих вразливостей, забезпечуючи виявлення та усунення загроз своєчасно;
- розширене виявлення загроз: використовує складні алгоритми та розвідку загроз для виявлення та нейтралізації відомого шкідливого програмного забезпечення та нових, що з'являються загроз;

- налаштовувані політики безпеки: дозволяє адміністраторам встановлювати конкретні політики безпеки, адаптовані до потреб їхнього сервера, забезпечуючи баланс між безпекою та продуктивністю;
- регулярні оновлення: регулярно оновлює свою базу даних для включення останніх визначень вірусів та розвідки загроз, захищаючи сервер від нових вразливостей;
- системи запобігання та виявлення вторгнень: виявляє та блокує спроби злому сервера, включаючи несанкціонований доступ, sql-ін'єкції та xss-атаки;
- моніторинг цілісності файлів: забезпечує, щоб критичні системні файли не були змінені несанкціонованими процесами або користувачами.

Включення антивірусу для вебсерверів у стратегію кібербезпеки є важливим з кількох причин:

- захист даних: допомагає захистити конфіденційні дані, що зберігаються на вебсерверах, від крадіжки або пошкодження;
- безперервність роботи: запобігання атакам забезпечує безперервну роботу вебсервісів, уникаючи дорогих простоїв;
- відповідність вимогам: багато галузей мають нормативні вимоги щодо належного захисту даних, і антивірус для вебсерверів може бути частиною виконання цих нормативних вимог;
- управління репутацією: запобігає порушенням безпеки, які можуть завдати шкоди репутації бізнесу;
- вартість безпеки: хоча вартість кібератаки може бути величезною, інвестування у надійний антивірус для вебсерверів є економічно ефективним заходом для запобігання таким витратам.

1.2.4 Шифрування даних

В умовах швидкого оцифрування інформації та зростаючої залежності від вебплатформ для різних видів діяльності безпека конфіденційних даних стала першочерговим завданням. Захист інформації від несанкціонованого доступу та кіберзагроз ще ніколи не був таким важливим. Шифрування даних на вебресурсах

стає найважливішим заходом захисту, що забезпечує конфіденційність і цілісність переданих і збережених даних[10].

Шифрування даних — це фундаментальний механізм, який перетворює звичайний текст у формат, неможливий для читання, роблячи його недоступним для сторонніх осіб. Шифрування служить життєво важливою лінією захисту від кіберзагроз, включаючи злом, витік даних і крадіжку особистих даних. Шифруючи конфіденційні дані на вебресурсах, організації та приватні особи можуть захистити свою інформацію від перехоплення, маніпуляцій чи експлуатації.

Алгоритми шифрування формують основу шифрування даних на вебресурсах. Ці алгоритми визначають, як перетворюються і захищаються дані. Широко використовуваними алгоритмами шифрування є Advanced Encryption Standard (AES), RSA і Elliptic Curve Cryptography (ECC). AES широко відомий своєю безпекою та ефективністю, тоді як RSA і ECC популярні завдяки своїй здатності полегшувати безпечний обмін ключами та операції шифрування/дешифрування. Кожен алгоритм має свої сильні та слабкі сторони, і вибір алгоритму залежить від таких факторів, як потрібний рівень безпеки, обчислювальні ресурси та міркування щодо продуктивності.

Ефективне управління ключами має вирішальне значення для безпечного шифрування даних на вебресурсах. Алгоритми шифрування використовують ключі для шифрування та дешифрування даних. Генерація, розповсюдження, зберігання та відкликання ключів — це методи управління ключами, які забезпечують конфіденційність і цілісність ключів шифрування. Безпечне управління ключами включає в себе надійні механізми для генерації надійних ключів, безпечного розповсюдження їх серед уповноважених осіб, захисту ключів від несанкціонованого доступу та впровадження процедур для відкликання скомпрометованих або застарілих ключів. Правильне управління ключами має важливе значення для підтримання безпеки зашифрованих даних протягом усього їх життєвого циклу.

Захищені комунікаційні протоколи відіграють життєво важливу роль у шифруванні даних на вебресурсах. Протокол безпеки транспортного рівня (TLS) та

його попередник, Secure Sockets Layer (SSL), є широко використовуваними протоколами для встановлення захищених з'єднань між вебсерверами та клієнтами. Ці протоколи забезпечують шифрування даних під час передачі, гарантуючи, що вони залишаються конфіденційними та захищеними від перехоплення. Протоколи безпечного підтвердження зв'язку, взаємна автентифікація та використання центрів сертифікації є невід'ємними компонентами протоколів захищеного зв'язку. Вони встановлюють довіру і перевіряють особу сторін, які спілкуються, ще більше підвищуючи безпеку вебресурсів.

Хоча шифрування даних має важливе значення для підтримання безпеки, воно може вплинути на продуктивність вебресурсів. Процеси шифрування і дешифрування вимагають обчислювальних ресурсів, що потенційно призводить до збільшення накладних витрат на обробку та часу відгуку. Однак досягнення в області апаратного забезпечення та методів оптимізації дозволили знизити багато проблем з продуктивністю. Ефективні алгоритми шифрування, апаратне прискорення та інтелектуальні механізми кешування можуть значно знизити вплив на продуктивність Інтернету. Дотримання балансу між безпекою та продуктивністю має вирішальне значення для забезпечення безперебійної взаємодії з користувачем при збереженні надійного захисту даних.

Впровадження заходів щодо шифрування даних на вебресурсах має кілька практичних наслідків. Організації, які обробляють конфіденційну інформацію, такі як фінансові установи, постачальники медичних послуг та платформи електронної комерції, підпорядковуються правовим та нормативним вимогам, що стосуються захисту даних. Шифрування допомагає організаціям виконувати ці зобов'язання та уникати юридичних наслідків. Крім того, шифрування даних підвищує довіру клієнтів, покращує репутацію бренду та знижує ризик фінансових втрат у результаті витоку даних. Для окремих користувачів шифрування даних забезпечує душевний спокій, гарантуючи конфіденційність та безпеку їх особистої інформації при взаємодії з вебресурсами.

На завершення слід зазначити, що шифрування даних на вебресурсах є незамінним заходом для забезпечення безпеки і конфіденційності інформації. У

міру того, як цифрові платформи стають все більш поширеними, а кіберзагрози — все більш витонченими, шифрування служить найважливішим механізмом захисту від несанкціонованого доступу, витоку даних і крадіжки особистих даних. Використовуючи надійні алгоритми шифрування, впроваджуючи ефективні методи управління ключами та використовуючи захищені протоколи зв'язку, організації та приватні особи можуть значно підвищити захист своїх даних.

1.2.5 Політики доступу та управління правами

Управління вебдоступом (Web Access Management, WAM) є спеціалізованою системою управління ідентифікацією, призначеною для регулювання доступу користувачів до онлайн-ресурсів і послуг[11]. Управління вебдоступом виконує дві основні функції системи управління ідентифікацією та доступом:

- аутентифікація. WAM перевіряє особу користувача і забезпечує, що він є законним. Це може включати простий доступ за паролем, але також може включати багатофакторну аутентифікацію;
- авторизація. WAM застосовує індивідуальні або рольові привілеї для доступу до конкретних додатків або баз даних. Якщо хакери отримують доступ, їх можливості будуть обмеженими у пересуванні по мережевих ресурсах.

Поряд із цими основними функціями, WAM також може включати самостійне управління паролями та автоматизоване обслуговування для очищення облікових записів, які вже не використовуються.

Спочатку управління вебдоступом було задумане для керування доступом між зовнішніми користувачами і мережевими ресурсами. Але з розширенням дистанційної роботи та вебдоступу до офісів, воно стало важливою системою доступу для співробітників.

Проте WAM був розроблений для регулювання доступу до вебсерверів. У результаті, він не враховує потреби користувачів гібридної хмари. Більш ефективні рішення для управління ідентифікацією та доступом швидко замінюють його як безпечний варіант доступу.

Управління вебдоступом (WAM) контролює доступ користувачів до онлайн-ресурсів і послуг.

Розроблене в 1990-х роках разом з Всесвітньою мережею, WAM об'єднує аутентифікацію користувачів із контролем доступу і включає єдиний вхід (Single Sign-On, SSO) для спрощеного доступу до різних доменів.

WAM захищає вебресурси, аутентифікуючи та авторизуючи доступ для законних користувачів, блокуючи при цьому несанкціонованих.

Воно працює шляхом перевірки облікових даних користувачів та впровадження політик на основі авторизації для встановлення дозволів користувачів на мережевих ресурсах.

Сучасні технології управління доступом тепер перевершують WAM, надаючи більш широкі рішення для різних ідентифікацій користувачів і хмарних середовищ. WAM не створює ідентифікації користувачів і не відповідає стандартам аутентифікації в хмарі.

На певний час WAM був критичним інструментом безпеки.

Коли вебресурси стали повсюдними, рішення WAM дозволили контролювати доступ до критичних ресурсів. Системи аутентифікації та авторизації захищали від нелегітимних зовнішніх акторів, але дозволяли справжнім користувачам доступ до робочих навантажень і послуг.

WAM також полегшив об'єднання різних вебдоменів. Федеративні ідентичності та єдиний вхід (SSO) зробили можливим об'єднання доступу до різних вебресурсів. Це допомогло керувати безпекою і спростити доступ. Замість того, щоб мати справу з численними обліковими даними, користувачі могли отримувати доступ до вебресурсів через єдиний панель.

По-перше, програмне забезпечення управління вебдоступом контролює доступ, запитуючи облікові дані користувача. Зазвичай доступ надається, коли користувачі надають ім'я користувача та пароль. Але команди безпеки можуть додати додаткові фактори управління аутентифікацією, такі як одноразові паролі, цифрові сертифікати або тимчасовий токен.

Коли система підтверджує особу користувача, рішення WAM застосовують політику на основі авторизації. Це зіставляє кожного користувача з набором

дозволів. Ці дозволи визначають, які ресурси доступні користувачеві і рівень контролю, який вони мають над даними в мережі.

Більшість впроваджень WAM покладаються на локальну інфраструктуру. Агенти, пов'язані з вебсервісами, спілкуються з центральними серверами. Ці сервери аутентифікації схвалюють або відхиляють доступ на основі реєстрів мережевих користувачів. Однак проксі-сервери також можуть грати роль.

WAM був одним із найкращих типів IAM для управління доступом до вебмережевої архітектури 1990-х і 2000-х років. Однак він загалом був перевершений сучасними технологіями управління ідентифікацією та доступом (IAM).

Сучасне управління доступом будується на основі WAM у кілька важливих способів:

Системи WAM надають доступ користувачів до конкретних ресурсів, таких як вебдодатки або фізичні машини. Це менш актуально для хмарних середовищ із великою кількістю віртуальних машин та екземплярів додатків.

WAM зазвичай має обмежені функції управління ідентифікацією. Але сучасні мережі повинні задовольняти багато різних типів ідентичності. Системи управління доступом повинні приймати різні рівні співробітників. Можуть бути сторонні користувачі, клієнти або навіть нелюдські сервісні облікові записи. Усі вони потребують специфічних політик управління доступом.

WAM не може надати гранульоване управління привілеями для ефективного захисту даних у хмарі. Витоки даних є критичною загрозою безпеці. Але для захисту даних клієнтів, розміщених у хмарі, потрібна гранульована авторизація.

Сучасне управління доступом є більш комплексним, ніж старі вебпортали. Новіші системи можуть моніторити активність користувачів і калібрувати точні засоби контролю доступу для кожної ролі. І вони можуть робити це у гібридних хмарних середовищах. Просто управління вебдоступом недостатньо.

Найкращий спосіб зрозуміти, як змінився контроль доступу, це порівняти управління вебдоступом із сучасною технологією управління ідентифікацією (IdM).

Зазвичай базується на серверах аутентифікації та мережах агентів на вебсервісах. Запити на аутентифікацію передаються від пристроїв користувачів до агентів, а потім до серверів аутентифікації. Сервери розташовані на місцях.

WAM зазвичай не генерує ідентифікації користувачів. Послуги залежать від окремих інструментів управління ідентифікацією і не включають стандарти аутентифікації у хмарі.

Сучасні типи IAM включають генерацію ідентичності як основний компонент. Сучасні системи управління доступом призначають ідентичності користувачів і керують ними протягом усього їхнього життєвого циклу. Менеджери можуть призначати привілеї доступу на різних хмарних платформах та локальних ресурсах.

IdM працює з новітніми стандартами і протоколами аутентифікації. Вона базується у хмарі і добре працює з віддаленими пристроями та мобільними додатками. Майже немає потреби у локальному обладнанні.

Стара система управління вебдоступом має кілька недоліків у порівнянні з сучасним управлінням доступом. Наприклад, негативні аспекти включають:

- дорого використовувати у великих масштабах;
- не може ефективно керувати доступом зовнішніх користувачів до хмарних ресурсів. несумісний з популярними стандартами аутентифікації у хмарі;
- повільне введення або видалення ідентифікацій, негнучке управління привілеями або налаштуваннями аутентифікації;
- обмежені функції збору даних. не може створювати глибокі аудиторські сліди або генерувати потоки даних для допомоги у безпеці чи маркетингу;
- проблеми з управлінням користувачами та сумісністю з хмарою можуть створювати серйозні прогалини у безпеці. це може поставити під загрозу конфіденційні дані.

Фундаментально, WAM був ефективним способом управління ідентифікаціями в Інтернеті. Але це небезпечний варіант для захисту ідентифікацій у хмарі.

Хмарні обчислення потребують рішень IAM, які враховують віддалений доступ та постійно змінювані мережеві кінцеві точки. І вимоги до відповідності зараз виключають старі системи WAM, які вважаються небезпечними для управління конфіденційними даними

1.3 Моніторинг вебсерверів та реагування на інциденти

Моніторинг вебсервера — це відстеження показників здоров'я та продуктивності вебсерверів.

Деякі приклади показників здоров'я вебсерверів включають використання ЦП, обсяг жорсткого диска, стан вебзастосунку та сертифікати SSL/TLS.

Деякі приклади показників продуктивності вебсерверів включають активні з'єднання, вхідні запити, час завантаження та час відповіді. Моніторинг вебсерверів забезпечує миттєву видимість та дозволяє контролювати здоров'я та продуктивність вебсерверів. Це, в свою чергу, дозволяє:

1. Вживати заходів у разі відмови апаратного або програмного забезпечення;
2. Вживати проактивних заходів у разі перевантаження апаратного або програмного забезпечення та необхідності оновлення або заміни.

За допомогою моніторингу вебсерверів отримуются своєчасні сповіщення, що дозволяють вчасно діяти, підтримуючи роботу вебсерверів для забезпечення оптимальної продуктивності для кінцевих користувачів.

Послуги, що надаються через Інтернет, зазвичай не працюють з одного вебсервера. Для досягнення найкращої продуктивності, балансування навантаження та зменшення часу завантаження, постачальники послуг розгортають вебсервери в декількох місцях, щоб отримувати послугу від найближчого сервера. Моніторинг вебсерверів дозволяє відстежувати важливі показники продуктивності — такі як час завантаження та затримка — для регіональних та глобальних розгортань вебсерверів.

Моніторинг вебсерверів також допомагає визначити, які з вебсайтів та вебсервісів набирають популярність, тобто отримують більше запитів користувачів.

Можна планувати масштабування вебсайтів та вносити інші зміни, які допоможуть впоратися зі зростаючим попитом.

Те ж саме стосується вебсайтів, чия популярність знижується. За допомогою моніторингу вебсерверів можна виявляти та усувати проблеми, такі як збільшення часу завантаження, які змушують користувачів переходити до інших сервісів.

Якщо вебсервери не моніторяться, не має уявлення про їх здоров'я та продуктивність. Не має знання, чи працюють вебсервери належним чином, або чи страждають користувачі через погану продуктивність серверів. Це може призвести до зниження продуктивності або недоступності послуг. Можливими короткостроковими наслідками можуть бути: поганий клієнтський досвід, втрата клієнтів та доходів. Можливими довгостроковими наслідками можуть бути: втрата репутації та нестабільність бізнесу.

Найбільший виклик у моніторингу вебсерверів — забезпечення хорошого клієнтського досвіду. Правильний інструмент моніторингу, використаний належним чином, може допомогти вам подолати цей виклик.

Можна використовувати моніторинг вебсерверів для огляду загальної картини та деталей. Загальна картина показує, чи працює сервіс та яке загальне навантаження на різні апаратні та програмні компоненти. Якщо заглиблюватися у деталі, можна переглянути конкретні показники, які надають точне уявлення про досвід користувачів.

Наприклад, якщо інструмент моніторингу повідомляє, що сервіс не працює, це означає, що всі користувачі зазнають впливу. Можна дослідити можливі причини та працювати над вирішенням проблеми.

Якщо ж сервіс доступний, і подає сигнал тривоги, що вказує на те, що певна кількість користувачів зазнає високого часу завантаження, підхід вже буде іншим. Проблема може бути викликана проблемою з базою даних, кількома вебзастосунками, розміщеними на одному вебсервері, або надмірним навантаженням на один застосунок. Можна використовувати інструменти моніторингу, щоб глибше дослідити корінну причину для вирішення проблеми.

Тому найважливішим аспектом подолання цього виклику є вибір правильного інструменту моніторингу серверів, який забезпечує видимість усіх аспектів продуктивності.

Існує багато метрик серверів, які важливі через їх вплив на продуктивність. Виділити можна три основні показники:

Доступність сервісу - розгортання вебсерверів для надання послуги, такої як вебсторінка або вебзастосунки, клієнтам. Якщо сервіс доступний, це означає, що клієнти можуть отримати доступ до нього та використовувати його. Якщо сервіс недоступний - клієнти не можуть отримати до нього доступ або використовувати його.

Моніторинг вебсерверів надає видимість доступності сервісу. За допомогою сповіщень отримується повідомлення, як тільки виникає проблема, що впливає на сервіс. Можна вжити заходів для вирішення проблеми та відновлення сервісу.

Використання апаратних засобів - це використання ресурсів хоста, де хост — це машина, на якій вебсервер працює як сервіс. Ці метрики хоста включають використання ЦП, пам'яті та сховища. Якщо ЦП перевантажено або пам'ять і сховище близькі до заповнення, це вплине на продуктивність всієї системи. Користувачі страждатимуть від повільного часу відповіді або навіть недоступності сервісу.

Моніторинг метрик хоста дозволяє запобігати проблемам до їх виникнення. Також можна проактивно планувати оновлення апаратного забезпечення або, якщо потрібно, переміщувати сервіс на платформу з кращим апаратним забезпеченням.

Продуктивність - доступність сервісу показує, чи можуть клієнти отримати доступ до сервісу та використовувати його. Продуктивність занурюється у деталі та розглядає запити за секунду, середній та піковий час відповіді та інші метрики з'єднання, такі як запити та активні з'єднання, що надають кількісну оцінку фактичного досвіду користувачів.

Якщо продуктивність вебсервера відповідає ключовим показникам ефективності (KPI), не потрібно вживати коригувальних заходів. Якщо ж KPI не досягнуто, можна вжити відповідних заходів для поліпшення клієнтського досвіду.

Інструменти моніторингу вебсерверів — це спеціалізоване програмне забезпечення, яке надає уявлення про здоров'я та продуктивність вебсерверів з точки зору апаратного забезпечення, програмного забезпечення, застосунків та сервісів.

Ці інструменти збирають відповідні метрики та відображають їх у графічному форматі для легкого розуміння. Можна використовувати ці інструменти для створення детальних звітів, що показують тенденції. Це дозволяє оцінювати здоров'я та продуктивність протягом визначеного періоду часу.

Також можна визначати порогові значення для метрик на вибір і налаштовувати сповіщення. Таким чином, досягається поінформованість та можливість вжити відповідних коригувальних заходів.

Вебсервери, такі як Apache та Nginx, повідомляють внутрішні метрики через HTML-сторінку. Сервери також надають машинозчитувані версії цих сторінок. Інструменти періодично аналізують ці машинозчитувані сторінки для збору метрик. Зібрані метрики також можуть бути об'єднані для створення кастомних метрик. Метрики потім відображаються через різні варіанти візуалізації, такі як дашборди.

Кожна конфігурація вебсервера відрізняється та має свої операційні вимоги. Інструмент, який обирається для моніторингу вебсерверів, повинен задовольняти ваші конкретні вимоги. Крім того, інструмент повинен бути:

- легко конфігуруваним;
- інтуїтивно зрозумілим у використанні;
- підтримувати інтеграцію з сторонніми застосунками;
- масштабованим.

Найкращий спосіб реалізувати моніторинг вебсерверів — це використання інструменту, який пропонує всі необхідні функції моніторингу. Інструмент повинен бути здатен моніторити вебсервери незалежно від того, чи вони знаходяться у хмарі, в приміщеннях або на віртуальній машині. Інструмент повинен надавати необхідний рівень деталізації, щоб мати змогу приймати рішення на основі достовірної інформації.

Є можливість налаштування дашбордів для вебсерверів, які надають чітке та візуальне представлення найважливіших метрик. З одного погляду можна вирішити, чи все працює добре, чи потрібно вжити заходів для вирішення існуючої або проблеми, що розвивається.

Налаштування сповіщень та повідомлень, які відповідають вимогам, дозволяє підвищити ефективність та заощадити час. Можна налаштувати порогові значення для метрик, які оптимізовані відповідно до ваших операційних вимог.

Як сповіщення допомагають заощадити час? Коли отримується сповіщення, є точне знання, які метрики перевищують свої порогові значення. Це дозволяє зосередитись на пошуку та усуненні можливих причин цих конкретних сигналів тривоги та уникнути витрати часу на неважливі питання.

Як сповіщення допомагають підвищити ефективність? Без моніторингу на основі сповіщень деякі ресурси були б присвячені спостереженню за системою моніторингу 24/7. З моніторингом на основі сповіщень IT-ресурси можуть бути залучені до інших продуктивних заходів, таких як планування, розгортання та розширення мережі.

Коли налаштовуються сповіщення, важливо переконатися, що вони надсилаються IT-ресурсам, які розуміють їх значення та можуть вжити відповідних заходів. Важливо також надсилати сповіщення через відповідні канали зв'язку - SMS, електронну пошту тощо - щоб вони були отримані вчасно та належним чином.

Перш ніж говорити про реагування, важливо розібратися, що таке інцидент. В IT-сфері є три терміни, які іноді використовуються як синоніми, але насправді означають різні речі.

- подія — це безпечне, регулярно повторюване діяння, наприклад, створення файлу, видалення папки або відкриття електронного листа. Зазвичай подія сама по собі не є ознакою порушення безпеки. Але якщо вона відбувається одночасно з іншими подіями, це може сигналізувати про загрозу;
- оповіщення — це повідомлення про подію, яке може бути пов'язане із загрозою;

- інцидент — це група взаємопов'язаних оповіщень, які людина або автоматизована система вважає реальною загрозою. Кожне оповіщення окремо може не бути маркером серйозної загрози, але в сукупності вони вказують на можливе порушення безпеки.

Реагування на інциденти — це дії, які організація вживає, коли є підстави вважати, що відбувся злом IT-системи або витік даних. Наприклад, фахівці з безпеки будуть вживати заходів, якщо побачать ознаки неавторизованого доступу, запуску шкідливої програми або збою в системі захисту.

Цілі такого реагування — якомога швидше припинити кібератаку, відновити дані та повідомити клієнтів або державні органи відповідно до регіональних законів. Після цього важливо зрозуміти, як знизити ризик аналогічних зломів у майбутньому.

Реагування на інцидент зазвичай починається з того, що служба безпеки отримує достовірне оповіщення від системи управління інформаційною безпекою та подіями безпеки (SIEM).

Фахівці повинні переконатися, що подія кваліфікується як інцидент, а потім ізолювати уражені системи і усунути загрозу. Іноді інциденти призводять до серйозних наслідків і потребують багато часу для їх усунення. У таких випадках компанії змушені відновлювати дані з резервної копії, вирішувати проблеми з викупом або повідомляти клієнтів про те, що їхні дані були скомпрометовані.

З цієї причини до реагування на інциденти зазвичай залучаються не лише фахівці з кібербезпеки. Стратегію боротьби з наслідками атаки розробляють експерти з конфіденційності, юристи та керівники компанії.

Зловмисники різними способами отримують доступ до даних компанії або іншим чином ставлять під загрозу корпоративні системи і бізнес-процеси. Нижче наведені найпоширеніші методи:

- фішинг — це атака з використанням соціальної інженерії, при якій зловмисник зв'язується з жертвою за допомогою електронної пошти, SMS або по телефону і видає себе за знайому людину або представника відомого бренду. Під час типової фішингової атаки одержувача намагаються обманом змусити

завантажити шкідливу програму або надати пароль. Зловмисники експлуатують довірливість людей, а також змушують їх щось зробити за допомогою психологічного тиску, наприклад, викликаючи страх. Більшість таких атак спрямовані на тисячі людей в надії на те, що когось вдасться обдурити. Але буває і більш складний підхід, який називається цільовим фішингом. У такому разі проводиться ретельний аналіз і створюється повідомлення, яке повинно бути переконливим для конкретної людини;

- шкідлива програма — це будь-яке програмне забезпечення, яке призначене для крадіжки даних або нанесення шкоди комп'ютерній системі. Наприклад, це можуть бути віруси, програми-вимагачі, програми-шпигуни і трояни. Щоб встановити шкідливу програму, зловмисники використовують уразливості апаратного і програмного забезпечення або переконують співробітника зробити це за допомогою соціальної інженерії;

- під час атак з використанням програм-вимагачів зловмисники використовують шкідливе ПЗ для шифрування важливих даних і систем, а потім погрожують розкрити або знищити інформацію, якщо жертва не заплатить викуп;

- під час розподілених атак типу "відмова в обслуговуванні" (DDoS) зловмисники перевантажують трафіком мережу або систему, щоб уповільнити її роботу або викликати збій. Зазвичай зловмисники атакують соціально значущі організації, наприклад банки або державні установи, щоб перервати їх роботу і позбавити грошей. Але жертвою атаки такого типу може стати будь-яка компанія;

- ще один спосіб крадіжки персональних даних — це втручання в онлайн-спілкування між людьми, які вважають, що їхня бесіда приватна. Кіберзлочинці перехоплюють повідомлення і копіюють або змінюють їх перед відправкою одержувачу. Вони намагаються маніпулювати одним з учасників розмови, щоб той передав їм цінні дані;

- більшість атак організовують сторонні люди, не працюючи в компанії. Але фахівцям служби безпеки важливо стежити і за внутрішніми загрозами. Співробітники та інші особи, які використовують в роботі ресурси з обмеженим доступом, можуть випадково або навмисно розкрити конфіденційну інформацію;

- часто причиною появи бреші в системі безпеки стають вкрадені облікові дані. Неважливо, як зловмисники отримали пароль: організували фішингову кампанію або вгадали стандартну комбінацію. Коли у них є доступ до системи, вони можуть встановити шкідливі програми, провести рекогносцировку мережі або підвищити свої привілеї і дістатися до більш конфіденційних систем і даних.

Коли виникає проблема безпеки, фахівці повинні працювати спільно і діяти ефективно, щоб усунути загрозу і дотримати нормативні вимоги. У таких стресових ситуаціях легко розгубитися і зробити помилку. Тому багато компаній розробляють план реагування на інциденти. Такий план розподіляє ролі і обов'язки і містить інструкції щодо належного усунення проблем, документування і інформування про інцидент.

Серйозна атака не тільки порушує внутрішні бізнес-процеси, а й впливає на репутацію компанії серед клієнтів і спільноти, а також може спричинити юридичні наслідки. Підсумковий збиток залежить від поведінки всіх учасників процесу: починаючи зі швидкості реакції служби безпеки і закінчуючи інформуванням з боку керівників.

Якщо компанія приховує факт злому від клієнтів і держорганів або недостатньо серйозно ставиться до загрози, це може бути порушенням нормативних вимог. Такі помилки виникають, коли у учасників процесу немає плану дій. Люди бояться наслідків атаки і на емоціях приймають необдумані рішення, які в підсумку шкодять організації.

Якщо у співробітників є продуманий план, вони знають, що потрібно робити на кожному етапі атаки, і їм не доводиться вигадувати щось на ходу. Коли робота системи відновлена, до компанії можуть виникнути питання. Але якщо всі діяли за планом, фахівцям не складе труднощів показати, як саме вони відреагували на загрозу. Це дозволить переконати клієнтів, що ви серйозно поставилися до інциденту і вжили заходів для запобігання більш тяжких наслідків.

Існує кілька підходів до реагування на інциденти. У цій сфері багато компаній покладаються на керівництва від організацій, які розробляють стандарти забезпечення безпеки. Інститут SANS — це приватна організація, яка склала

описану нижче шестиступеневу схему реагування на інциденти. Компанії також орієнтуються на рекомендації з відновлення роботи системи після інцидентів від Національного інституту стандартів і технологій (NIST):

1. До виникнення інциденту важливо зменшити кількість вразливостей, налаштувати політики безпеки і визначити процедури захисту. На етапі підготовки організації оцінюють ризики, щоб виявити слабкі місця і пріоритизувати активи. Цей етап включає в себе написання і уточнення процедур безпеки, розподіл ролей і обов'язків, а також оновлення систем для зниження ризику. Технології змінюються, і фахівці витягають уроки з минулих атак. Тому більшість компаній регулярно повертаються до цього етапу і покращують політики, процедури і системи;

2. Кожен день служба безпеки може отримувати тисячі оповіщень про підозрілу активність. Деякі з них є помилковими спрацюваннями або незначними подіями, які не ескалуються до рівня інциденту. Якщо ж фахівці виявляють інцидент, то вони вивчають характер порушення і документують зібрану інформацію про джерело загрози, тип атаки і цілі зловмисників. На цьому етапі команда також повинна проінформувати зацікавлених осіб і обговорити подальші кроки;

3. Наступний крок — максимально швидко обмежити поширення загрози. Чим довше у зловмисників є доступ до системи, тим більше шкоди вони можуть завдати. Служба безпеки повинна швидко ізолювати від решти мереж додатки або системи, які зазнали атаки. Це допомагає запобігти доступу зловмисників до інших частин корпоративної інфраструктури;

4. Коли загроза ізольована, фахівці видаляють зловмисників і всі шкідливі програми з уражених систем і ресурсів. Іноді може знадобитися відключити ці системи від мережі. При цьому служба безпеки продовжує інформувати зацікавлених осіб про хід робіт;

5. Після інциденту відновлення нормальної роботи може зайняти кілька годин. Коли загроза усунена, служба безпеки реанімує систему і відновлює дані з резервної копії. Також важливо контролювати стан уражених атакою ресурсів, щоб переконатися, що зловмисник не повернувся;

6. Після усунення інциденту служба безпеки аналізує те, що сталося, і визначає, як можна поліпшити процес реагування. На цьому етапі команда витягає уроки, які допомагають посилити захист організації;

Команду реагування на інциденти називають по-різному: групою реагування на інциденти у сфері комп'ютерної безпеки (CSIRT), командою реагування на кіберінциденти (CIRT) або службою реагування на порушення комп'ютерної безпеки (CERT). Це крос-функціональна команда, яка відповідає за реалізацію плану з реагування на інциденти. До неї входять не тільки люди, які усувають загрози, але і ті, хто приймають комерційні або юридичні рішення, пов'язані з інцидентом. Зазвичай така команда складається з перерахованих нижче фахівців.

Керівник команди реагування на інциденти (часто це IT-директор) контролює всі етапи і інформує зацікавлених осіб всередині компанії.

Аналітики інформаційної безпеки вивчають інцидент і намагаються з'ясувати, що сталося. Також вони документують результати своїх досліджень і збирають дані для подальших експертиз.

Дослідники загроз аналізують ситуацію за межами організації і збирають інформацію, яка дасть додатковий контекст.

Директор з інформаційної безпеки або директор з інформаційних технологій дає загальні вказівки і служить зв'язуючою ланкою між співробітниками і іншими представниками керівництва.

Фахівці з кадрів допомагають контролювати внутрішні загрози.

Головний юрисконсульт компанії допомагає команді розібратися в питаннях відповідальності за інцидент і забезпечує збір інформації для судової експертизи.

Фахівці зі зв'язків з громадськістю координують комунікацію із ЗМІ, клієнтами і іншими зацікавленими сторонами.

Команда реагування на інциденти може бути частиною центру інформаційної безпеки (SOC), який забезпечує захист активів компанії в цілому.

Часто в організаціях мережі і рішення для забезпечення безпеки генерують велику кількість оповіщень, які команда реагування на інциденти не в змозі обробити. Щоб фахівці могли зосередитися на реальних загрозах, багато компаній

автоматизують реагування на інциденти. В рамках автоматизації застосовуються інструменти на базі штучного інтелекту і машинного навчання. Вони пріоритизують оповіщення, виявляють інциденти і усувають загрози за допомогою схем реагування на основі програмних сценаріїв.

Системи оркестрації, автоматизації і реагування на інциденти безпеки (SOAR) — це категорія інструментів, які підприємства використовують для автоматизації цих процесів. Такі рішення володіють наступним функціоналом:

- співставлення даних з декількох кінцевих точок і засобів забезпечення безпеки для виявлення інцидентів, з якими повинні розбиратися фахівці;
- запуск готових сценаріїв реагування для ізоляції і усунення загроз відомих типів;
- створення графіка розслідування, який включає дії, рішення і збір доказів для аналізу;
- збір релевантних зовнішніх даних для аналізу фахівцями.

Розробка плану реагування на інциденти може здатися складним завданням. Але такий план значно знижує ризик того, що компанія виявиться непідготовленою до серйозної загрози. Ось з чого варто почати роботу.

1. Перший крок в плані реагування на інциденти — з'ясувати, які саме активи ви захищаєте. Задокументуйте критично важливі корпоративні дані, в тому числі відомості про їх місцезнаходження і ступінь важливості для бізнесу;

2. У кожної організації є свої ризики. Вивчіть найсерйозніші уразливості компанії і оцініть, як зловмисник може їх використовувати;

3. Під час стресової ситуації чіткі процедури допоможуть швидко і ефективно впоратися з інцидентом. Для початку визначте, що ви кваліфікуєте як інцидент. Потім опишіть кроки, які повинна зробити служба безпеки для виявлення атаки, ізоляції загрози і подальшого відновлення системи. Сюди ж відносяться процедури документування рішень і збору доказів;

4. Створіть крос-функціональну команду, яка буде відповідати за процедури реагування і їх реалізацію у разі виникнення інциденту. Чітко визначте ролі і не забувайте про нетехнічних фахівців, які допоможуть прийняти рішення, пов'язані з

інформуванням і юридичною відповідальністю. Також бажано включити до команди когось із керівників, щоб ця людина відстоювала інтереси колег на рівні директорів компанії;

5. З планом інформування вам не доведеться гадати, коли і як повідомляти про інцидент людям всередині і за межами організації. Продумайте різні сценарії і визначте, за яких обставин потрібно інформувати керівництво, всю компанію, клієнтів, ЗМІ або інших сторонніх зацікавлених осіб;

6. Жертвою зловмисників може стати будь-який співробітник. Тому важливо, щоб всі в компанії розуміли план реагування і знали, що робити при підозрі на атаку. Періодично перевіряйте своїх працівників, щоб переконатися, що вони вміють розпізнавати фішингові листи. Створіть умови, в яких вони зможуть легко повідомити команду реагування на інциденти, якщо випадково натиснуть на шкідливе посилання або відкриють заражене вкладення.

1.3.1 Моніторингові сервіси для вебсерверів

Існує багато інструментів для моніторингу та реагування для вебсерверів. Далі приведені приклади найпопулярніших рішень для використання.

Prometheus і Grafana є одними з найпопулярніших інструментів для моніторингу, які використовуються разом для збору, зберігання та візуалізації метрик. Це потужне поєднання надає можливість отримати детальне уявлення про роботу Nginx та інших компонентів інфраструктури.

Prometheus — це система моніторингу з відкритим кодом, яка була розроблена компанією SoundCloud. Вона призначена для збору та зберігання часових рядів даних, таких як метрики та події.

Основні Особливості Prometheus:

1. Збір метрик:

- Pull Модель: Prometheus регулярно опитує (pull) метрики з кінцевих точок (endpoint);

- експортери: використовуються для збору метрик з додатків та систем. які не підтримують власний формат метрик Prometheus. Наприклад, для Nginx є nginx-prometheus-exporter;

2. Зберігання даних:

- часові ряди: дані зберігаються як часові ряди, що дозволяє ефективно обробляти великі обсяги метрик;

- локальне зберігання: дані зберігаються на локальному диску, що забезпечує швидкий доступ до них;

3. Запити та аналіз:

- PromQL: мова запитів, яка дозволяє виконувати складні аналітичні запити до метрик.

4. Алертинг:

- Alertmanager: компонент для управління та доставки сповіщень на основі заданих умов (алертів);

- Grafana — це інструмент для візуалізації даних з відкритим кодом, який широко використовується для створення інтерактивних дашбордів.

Основні особливості Grafana:

1. Дашборди та панелі:

- налаштування дашбордів: можливість створення кастомних дашбордів з різними панелями для візуалізації метрик;

- шаблони: наявність готових шаблонів дашбордів для nginx та інших сервісів;

2. Підключення джерел даних:

- Prometheus: Grafana легко інтегрується з Prometheus як джерело даних;

3. Алертинг:

- сповіщення: можливість налаштування сповіщень на основі метрик, що візуалізуються;

4. Візуалізація:

- графіки та діаграми: різноманітні типи візуалізацій, такі як лінійні графіки, гістограми, теплові карти та інші.

Поєднання Prometheus та Grafana надає потужні можливості для моніторингу Nginx, дозволяючи збирати, аналізувати та візуалізувати метрики в реальному часі.

Це допомагає вчасно виявляти проблеми, оптимізувати продуктивність та забезпечити стабільну роботу вашого вебсервера.

Іншим популярним інструментом моніторингу вебсервера Nginx є Datadog.

Datadog — це хмарна платформа для моніторингу та аналітики, яка надає комплексне рішення для відстеження метрик, логів та безпеки вашої інфраструктури та додатків. Datadog забезпечує високу деталізацію метрик, потужні інструменти для візуалізації та аналітики, а також зручний інтерфейс для управління оповіщеннями.

Основні Особливості Datadog:

1. Збір метрик:

- агенти Datadog: спеціальні агенти встановлюються на сервери для збору метрик з різних джерел, включаючи Nginx;

- автоматичне виявлення сервісів: агенти автоматично виявляють та збирають метрики з популярних сервісів;

2. Візуалізація:

- дашборди: налаштовувані дашборди, які дозволяють відображати метрики в різних форматах (графіки, гістограми, табличні дані тощо);

- шаблони: готові шаблони дашбордів для Nginx, що дозволяють швидко почати моніторинг;

3. Аналіз логів:

- лбір логів: можливість збору та аналізу логів Nginx для детального відстеження подій та збоїв;

- кореляція метрик та логів: здатність корелювати метрики з логами для глибшого аналізу проблем;

4. Алертинг:

- налаштування сповіщень: гнучкі налаштування сповіщень, які дозволяють створювати оповіщення на основі складних умов;

- інтеграція з месенджерами: підтримка інтеграції з популярними месенджерами та інструментами для оповіщення (Slack, PagerDuty тощо);

5. Інтеграції:

- широкий спектр інтеграцій: підтримка понад 400 інтеграцій з різними сервісами та платформами, включаючи хмарні провайдери, контейнери, бази даних тощо;
- Nginx інтеграція: легке підключення Nginx до Datadog для збору метрик.

Datadog надає комплексний підхід до моніторингу Nginx, дозволяючи збирати метрики, аналізувати логи та налаштовувати сповіщення. Це допомагає забезпечити стабільну роботу вашого вебсервера, швидко виявляти та усувати проблеми, а також оптимізувати продуктивність. Datadog є потужним інструментом для DevOps команд, забезпечуючи глибокий огляд інфраструктури та додатків в реальному часі.

Також для моніторингу використовують хмарну платформу New Relic.

New Relic — це потужна хмарна платформа для моніторингу та управління продуктивністю додатків (APM), яка забезпечує комплексне рішення для відстеження метрик, логів, подій та інцидентів в реальному часі. Вона надає інструменти для детального аналізу роботи додатків та інфраструктури, включаючи вебсервери, такі як Nginx.

Основні особливості New Relic:

1. Моніторинг продуктивності додатків (APM):

- трасування транзакцій: відстеження шляху кожної транзакції через додаток для виявлення вузьких місць;
- виявлення аномалій: автоматичне виявлення аномалій у продуктивності;

2. Інфраструктурний моніторинг:

- моніторинг системних ресурсів: збір метрик про використання cpu, пам'яті, дисків та мережевих ресурсів;
- контейнерний моніторинг: підтримка Docker та Kubernetes для моніторингу контейнеризованих додатків;

3. Аналіз логів:

- збір та аналіз логів: можливість збирати та аналізувати логи з різних джерел, включаючи nginx;
- кореляція логів та метрик: здатність корелювати логи з метриками для глибшого аналізу;

4. Алертинг і сповіщення:

- налаштування сповіщень: гнучкі налаштування сповіщень на основі метрик, логів та подій;

- інтеграція з месенджерами: підтримка інтеграції з популярними месенджерами та інструментами для оповіщення (slack, pagerduty тощо);

5. Аналітика та візуалізація:

- дашборди: інтерактивні дашборди для візуалізації метрик і логів в реальному часі;

- New Relic One: платформа для інтеграції всіх сервісів New Relic в єдину панель управління.

New Relic надає комплексний підхід до моніторингу Nginx, дозволяючи збирати метрики, аналізувати логи та налаштовувати сповіщення. Це допомагає забезпечити стабільну роботу вашого вебсервера, швидко виявляти та усувати проблеми, а також оптимізувати продуктивність. Завдяки інтеграції з іншими компонентами інфраструктури та додатками, New Relic забезпечує глибокий огляд вашої IT-екосистеми, дозволяючи ефективно управляти продуктивністю та забезпечити високу якість обслуговування.

Порівняння моніторингових сервісів за критеріями приведено в таблиці 1.1:

Таблиця 1.1 - Порівняння сервісів для моніторингу

Критерії	Prometheus + Grafana	Datadog	New Relic
По типу	Відкритий вихідний код	Хмарні платформи	Хмарні платформи
Основні можливості	Збір метрик та візуалізація	Збір метрик, логування, трасування	APM, збір метрик, логування, події, трасування
Засоби візуалізації	Використовує візуалізатор Grafana	Використовує адаптивні дашборди	Використовує інтерактивні дашборди

Продовження таблиці 1.1

Оповіщення про загрози	Alertmanager	Вбудовані оповіщення	Вбудовані оповіщення
Інтеграції	Експортери та плагіни Grafana	Має більше 400 інтеграцій	Має можливість інтеграції з багатьма мовами та технологіями
Вартість	Безкоштовне використання	Платно, залежить від обсягу використання	Платно, залежить від обсягу використання
Спеціалізовані знання	Вимагає технічні знання	Легкий у використанні, простий інтерфейс	Інтуїтивно зрозумілий інтерфейс
Підтримка	Широко підтримується у вільному доступі	має комерційну підтримкку	має комерційну підтримкку
Недоліки	Потребує технічні знання та потребує налаштування	Вартість за використання	Вартість за використання

1.3.2 Сервіси реагування на інциденти для вебсерверів

Далі приведено опис найпопулярніших сервісів реагування на інциденти для вебсерверів.

Splunk є потужною платформою для аналізу та моніторингу машинних даних в реальному часі. Вона дозволяє організаціям збирати, зберігати, аналізувати та

візуалізувати дані з різноманітних джерел, включаючи журнали подій, мережевий трафік, показники продуктивності та багато іншого.

Функціональна частина сервісу:

- моніторинг в реальному часі: Splunk збирає та аналізує дані в режимі реального часу, що дозволяє миттєво реагувати на інциденти;
- виявлення загроз: використовує машинне навчання та аналітику для виявлення аномалій та загроз, зокрема можливість створювати власні моделі для специфічних потреб організації;
- реагування на інциденти: платформа інтегрується з Splunk Phantom, що дозволяє автоматизувати реагування на інциденти за допомогою готових сценаріїв;
- аналітика: розширені можливості аналітики, включаючи візуалізацію даних через дашборди, створення аналітичних звітів та графіків;
- інтеграція: підтримка широкого спектру інтеграцій з іншими системами безпеки та управління інцидентами, що дозволяє легко інтегруватися в існуючу інфраструктуру;

Особливості Splunk:

- потужні можливості аналізу великих обсягів даних;
- гнучкість в налаштуванні і масштабуванні;
- високий рівень інтеграції з іншими інструментами безпеки.

Cortex XDR від Palo Alto Networks є рішенням для розширеного виявлення та реагування (XDR), яке об'єднує дані з різних джерел, таких як мережевий трафік, кінцеві точки та хмари, для детального аналізу та реагування на інциденти безпеки.

Функціональна частина сервісу:

- виявлення загроз: використовує машинне навчання для виявлення складних атак, що дозволяє ідентифікувати нові та відомі загрози;
- кореляція даних: збирає та корелює дані з різних джерел для створення єдиного огляду інцидентів безпеки;
- реагування на інциденти: автоматизовані дії для ізоляції та усунення загроз з використанням готових сценаріїв реагування;

- форензика: забезпечує детальний аналіз інцидентів, що дозволяє зрозуміти хронологію подій та вектори атак;
- інтеграція: підтримує інтеграцію з іншими продуктами Palo Alto Networks та сторонніми системами, забезпечуючи широкі можливості для побудови єдиного рішення безпеки.

Особливості Palo Alto Networks Cortex XDR:

- глибокий аналіз загроз з використанням машинного навчання;
- інтеграція даних з різних джерел для кореляції та аналізу;
- висока ефективність у виявленні та реагуванні на складні атаки.

Cisco SecureX є хмарною платформою для забезпечення безпеки, яка інтегрує та автоматизує різні аспекти кібербезпеки, включаючи виявлення та реагування на інциденти. SecureX об'єднує дані з різних джерел, щоб забезпечити централізований моніторинг та керування безпекою.

Функціональна частина сервісу:

- централізований моніторинг: SecureX об'єднує дані з різних джерел для загального огляду стану безпеки, забезпечуючи видимість і контроль над всією інфраструктурою;
- автоматизація: використовує автоматизовані робочі процеси для швидкого реагування на загрози, що дозволяє зменшити час реагування та підвищити ефективність роботи команди безпеки;
- візуалізація: інтуїтивно зрозумілі дашборди для моніторингу та аналізу загроз, що дозволяє швидко отримувати необхідну інформацію;
- спільна робота: забезпечує можливості для спільної роботи між різними командами безпеки, що сприяє ефективнішому реагуванню на інциденти;
- інтеграція: підтримує широку інтеграцію з іншими продуктами cisco та сторонніми рішеннями, що дозволяє створювати єдину екосистему безпеки.

Особливості Cisco SecureX:

- інтеграція з екосистемою cisco та сторонніми продуктами;
- потужні можливості автоматизації та оркестрації;

- інтуїтивний інтерфейс для швидкого доступу до інформації та управління інцидентами.

Splunk відзначається своєю потужною аналітикою та широкими можливостями інтеграції, що робить його ідеальним для великих організацій, які потребують гнучкого інструменту для моніторингу та аналізу великих обсягів даних.

Palo Alto Networks Cortex XDR забезпечує розширене виявлення та реагування з акцентом на машинне навчання та кореляцію даних з різних джерел, що підходить для організацій, які шукають передові можливості виявлення загроз та форензики.

Cisco SecureX відомий своєю інтеграцією з екосистемою Cisco та іншими продуктами, а також простотою використання та потужними можливостями автоматизації, що робить його привабливим для організацій, які вже використовують рішення Cisco або шукають комплексну платформу для управління безпекою.

РОЗДІЛ 2. СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Порівняння моніторингових систем Zabbix та Prometheus

Для успішного бізнесу необхідно впровадити ефективну систему моніторингу, яка охоплює всі сфери бізнесу та інфраструктури — сервери, бази даних, сервіси, загальний трафік і навіть зібрані доходи. Користувачами цієї системи моніторингу можуть бути системні адміністратори, програмісти, інженери з інформаційних технологій, а також усілякі аналітики. Сучасна ефективна система моніторингу повинна надавати послуги зі збору метрик, їх зберігання, обчислень/прогнозів, візуалізації та оповіщення. Знайти інструмент, який може досягти всього цього, є досить складним завданням.

Алі буде розглянуто двох основних постачальників послуг: Prometheus та Zabbix. Обидва Prometheus і Zabbix є чудовими інструментами для моніторингу тимчасових рядів, де Zabbix є інструментом старшого покоління, а Prometheus — сучасним. Обидва інструменти є відкритим програмним забезпеченням з доступними хостинг-опціями.

Короткий огляд Prometheus

Prometheus — це система моніторингу з відкритим вихідним кодом, яка надає потужну мову запитів, функції зберігання та візуалізації для своїх користувачів. Вона збирає реальні метрики та записує їх у базу даних тимчасових рядів. Prometheus забезпечує багатовимірну модель даних, що дозволяє визначати метрики за іменем та/або тегами, щоб ідентифікувати їх як частину унікального тимчасового ряду. Вона написана на Go та ліцензована за ліцензією Apache 2, з вихідним кодом, доступним на GitHub. Як проєкт з відкритим вихідним кодом, Prometheus має широку підтримку спільноти, тому має ряд клієнтських бібліотек, які дозволяють легко взаємодіяти з нею. Також Prometheus має рідну підтримку таких сервісів, як Docker і Kubernetes.

Короткий огляд Zabbix

Zabbix — це програмне забезпечення для моніторингу численних параметрів мереж, серверів, додатків, віртуальних машин і хмарних сервісів. Воно може збирати метрики, виявляти проблеми, візуалізувати, сповіщати та надсилати

повідомлення. Zabbix має вебінтерфейс, який забезпечує легку взаємодію з усією статистикою, візуалізаціями та налаштуваннями параметрів. Zabbix не зберігає дані самостійно, але може використовувати широкий спектр баз даних. Основна частина Zabbix написана на C, а вебфронтенд — на PHP.

Збір метрик

Prometheus є інструментом, який має широкий набір вбудованих функціональних можливостей, тому користувачам Prometheus не потрібно встановлювати різні плагіни або демонів для збору метрик. Виявлення сервісів також автоматичне, що робить процес швидким. Завдяки великій спільноті багато сервісів можуть надсилати метрики у форматі Prometheus. Якщо деякі сервіси не можуть це зробити, то існує багато бібліотек, які допомагають експортувати існуючі метрики з сторонніх систем у метрики Prometheus. Ці бібліотеки називаються Exporters. Варто зазначити, що популярні сервіси, такі як Kubernetes і Docker, підтримують метрики Prometheus.

Prometheus розроблений для періодичного збору метрик з цільової системи. Також можливо збирати метрики за допомогою механізму push. Це може бути необхідно в ситуаціях, коли збір метрик за допомогою pull неможливий. Наприклад, коли сервіси моніторингу захищені брандмауером або моніторингові сервіси підключені до мережі періодично та на короткий час. Для цього використовується спеціальний компонент pushgateway, який встановлюється окремо.

Оскільки Zabbix спочатку розроблявся для моніторингу серверів, він фокусується на хостах. З точки зору користувача, Zabbix ділиться на дві великі частини: сервер і агенти. Сервер розташований на одній машині, яка збирає та зберігає статистичні дані, а агенти розташовані на тих машинах, з яких збираються дані. Агенти Zabbix підтримують як пасивні (опитування), так і активні перевірки (уловлювання). Пасивні перевірки означають, що сервер Zabbix запитує значення у агента Zabbix, а агент обробляє запит і повертає значення серверу Zabbix. Активні перевірки означають, що агент Zabbix запитує список активних перевірок у серверу Zabbix, а потім періодично надсилає результати.

У випадках, коли немає можливості встановити агента, Zabbix пропонує базовий моніторинг без агента. З його допомогою ви можете перевіряти доступність мережевих сервісів, а також виконувати віддалені команди.

Зберігання даних

Prometheus зберігає дані у власній базі даних тимчасових рядів (TSDB). Маючи власну TSDB, Prometheus може отримувати та обробляти незрівнянно більше метрик, ніж багато інших систем моніторингу. Дані можуть записуватись Prometheus з тимчасовими мітками з мілісекундною роздільною здатністю. Сам Prometheus зберігає дані лише до 14 днів, що ускладнює збереження записів та прогнозування даних. Для тривалого зберігання даних можна налаштувати віддалене зберігання.

Zabbix використовує зовнішню базу даних для зберігання даних. База даних Zabbix повинна бути створена під час її встановлення. Наразі підтримуються наступні бази даних: MySQL, PostgreSQL, Oracle, IBM DB2 та SQLite.

Відмінність між Prometheus і Zabbix, яку необхідно враховувати, полягає в тому, що Prometheus зберігає лише значення тимчасових рядів. Він не підходить для текстів, журналів або журналів подій. Якщо використовувати Prometheus з Grafana, що є найбільш типовим способом, можна легко обійти обмеження Prometheus за допомогою Grafana's Loki. Хоча краще використовувати спеціалізовані продукти для журналів, Zabbix може надати базову функціональність для зберігання деяких текстових значень, їх аналізу та налаштування тригерів.

Запити

Prometheus надає власну функціональну мову запитів під назвою PromQL (Prometheus Query Language). PromQL є надзвичайно гнучкою, легкою та потужною. Вона може застосовувати функції та оператори до ваших запитів метрик, фільтрувати, групувати за мітками та використовувати регулярні вирази для покращеного співставлення та фільтрування. Результат виразу може бути показаний як графік, переглянутий як табличні дані в браузері виразів Prometheus або спожитий зовнішніми системами через HTTP API.

Zabbix не настільки гнучкий у запитах. Він використовує ключі елементів для отримання метрик.

Візуалізація

Prometheus має простий, але корисний інструмент візуалізації під назвою Expression Browser. Expression Browser не має функціональності повної панелі моніторингу. Ви можете використовувати його для виконання глибоких запитів у збережені метрики. Expression Browser не відображає метрики, які ви спостерігаєте протягом тривалого часу. Він працює як консоль досліджень, дозволяючи вам виконувати запити у вашу базу даних, змінювати запит на льоту та знаходити відповіді на ваші запитання.

Для того, щоб мати можливість повністю спостерігати та аналізувати ваші графіки, вам потрібно встановити повноцінний інструмент візуалізації, такий як Grafana. Grafana включає вбудовану підтримку Prometheus і є безкоштовною. Це дозволяє вам повністю використовувати обидва інструменти для створення ефективної системи моніторингу.

Zabbix включає рідний вебінтерфейс, який надає панель управління з гнучкими налаштуваннями. Ця функціональність присутня спочатку, і вам не потрібно встановлювати або налаштовувати щось інше. Вебінтерфейс Zabbix надає декілька способів представлення візуального огляду IT-середовища: панелі на основі віджетів, графіки, мережеві карти, слайд-шоу та детальні звіти. За замовчуванням, фронтенд Zabbix надає кілька попередньо визначених тем. Також користувачі можуть створювати власні теми.

Оповіщення

Для управління сповіщеннями з Prometheus потрібно встановити Alertmanager. Це тому, що сповіщення з Prometheus розділені на дві частини. Спочатку ви визначаєте правила сповіщень на сервері Prometheus, які будуть надсилати сповіщення до Alertmanager.

Тоді Alertmanager керує цими сповіщеннями шляхом зменшення, пригнічення, агрегування та надсилання повідомлень. Повідомлення можуть

надсилатися електронною поштою, системами сповіщень на виклик та платформами чату.

Alertmanager категоризує сповіщення подібного характеру в одне повідомлення, щоб уникнути дублювання. Це особливо корисно під час великих відмов, коли багато систем одночасно виходять з ладу. Також Alertmanager може пригнічувати повідомлення для певних сповіщень, якщо деякі інші сповіщення вже активовані.

Як і у випадку з візуалізацією, Zabbix має вбудовану функціональність сповіщень. З Zabbix можливо інформувати відповідальних осіб про виникнення подій, використовуючи багато різних каналів та опцій. Система сповіщень Zabbix дозволяє керувати подіями різними способами: надсиланням повідомлень, виконанням віддалених команд, ескалацією проблем згідно з гнучкими рівнями обслуговування, визначеними користувачем, тощо. Також можливо налаштувати повідомлення на основі ролі одержувача, вибираючи, яку інформацію включати, таку як дата, час, ім'я хоста, значення елементів, значення тригерів, профіль хоста, історія ескалацій тощо.

Обидва Zabbix і Prometheus є популярними системами моніторингу, але мають різні підходи та функціональні можливості. Далі приведено причини, за яких було обрано Zabbix для моніторингу вебсервера:

1. Інтегроване рішення - Zabbix є комплексною системою моніторингу, яка включає збір даних, зберігання, аналіз, сповіщення та візуалізацію "з коробки". Prometheus, навпаки, вимагає додаткових компонентів, таких як Grafana для візуалізації та Alertmanager для сповіщень;

2. Підтримка SNMP і агентів - Zabbix має вбудовану підтримку SNMP, що дозволяє легко інтегруватися з мережевим обладнанням. Також він пропонує власні агенти для збору даних з серверів і інших пристроїв. Prometheus більше орієнтований на збір метрик через HTTP(S)-ендпойнти, що може бути менш зручним для мережевого обладнання;

3. Підтримка різних типів даних - Zabbix підтримує різноманітні типи даних, включаючи метрики, журнали, пастки SNMP та інші. Prometheus в основному зосереджений на метриках;

4. Автодискавері - Zabbix має розширені можливості автоматичного виявлення пристроїв і служб у мережі. Це спрощує налаштування і управління великими середовищами. Prometheus теж має механізми дискавері, але вони більш обмежені і часто вимагають додаткових налаштувань;

5. Управління конфігурацією - Zabbix пропонує зручний вебінтерфейс для управління конфігурацією і налаштування моніторингу. Prometheus конфігурується через текстові файли, що може бути менш зручним для великих інфраструктур;

6. Готові шаблони - Zabbix має велику кількість готових шаблонів для моніторингу різних систем і сервісів, що дозволяє швидко налаштувати моніторинг без необхідності створювати все з нуля. Prometheus має менш розвинуту систему шаблонів і більше покладається на користувачів для створення власних конфігурацій;

7. Гнучкі сповіщення - Zabbix пропонує розширені можливості для сповіщень, включаючи різні канали (електронна пошта, SMS, месенджери) і умови сповіщень. Prometheus використовує Alertmanager для сповіщень, але це потребує додаткової конфігурації і налаштувань;

8. Підтримка баз даних - Zabbix зберігає всі дані у реляційній базі даних (mysql, postgresql, Oracle тощо), що забезпечує надійне зберігання та можливість виконання складних запитів. Prometheus використовує власний формат зберігання даних, що може бути менш зручним для інтеграції з іншими системами.

Обидві системи мають свої сильні сторони, і вибір між ними залежить від конкретних вимог та умов вашої інфраструктури. Zabbix більше підходить для комплексного моніторингу різнорідних середовищ, тоді як Prometheus оптимальний для моніторингу контейнеризованих додатків і мікросервісів.

2.1.1 Використання Zabbix для моніторингу вебсерверу

Zabbix — це програмне забезпечення для моніторингу численних параметрів мережі, а також стану та працездатності серверів. Zabbix використовує гнучкий

механізм сповіщень, що дозволяє користувачам налаштовувати оповіщення електронною поштою практично для будь-якої події. Це дає можливість швидко реагувати на проблеми з сервером. Zabbix пропонує відмінні можливості звітності та візуалізації даних, базуючись на зібраних даних. Це робить Zabbix ідеальним інструментом для планування та масштабування.

Zabbix підтримує опитування даних (пуллер) та отримання даних (траппер). Всі звіти та статистика Zabbix, так само як і параметри налаштувань, доступні через вебінтерфейс. Вебінтерфейс забезпечує можливість оцінювати стан вашої мережі та життєздатність ваших серверів з будь-якого місця. Добре налаштований Zabbix може відігравати важливу роль у моніторингу IT-інфраструктури. Це так само важливо для малих організацій з кількома серверами, як і для великих компаній з великою кількістю серверів.

Zabbix є безкоштовним. Zabbix написаний і розповсюджується під ліцензією GPL General Public License версії 2. Це означає, що його вихідний код вільно розповсюджується та доступний широкій публіці.

Також доступна комерційна підтримка, яка надається компанією Zabbix.

У Zabbix є 4 основних інструменти, за допомогою яких можна моніторити певне робоче середовище та збирати про нього повний пакет даних для оптимізації роботи.

1. Сервер — ядро, яке зберігає всі дані системи, включаючи статистичні, оперативні та конфігурацію. Дистанційно керує мережевими сервісами, сповіщає адміністратора про наявні проблеми з обладнанням, що перебуває під наглядом;

2. Проксі — сервіс, що збирає дані про доступність і продуктивність пристроїв, який працює від імені сервера. Усі зібрані дані зберігаються в буфері та завантажуються на сервер. Потрібен для розподілу навантаження на сервер. Завдяки цьому процесу можна зменшити навантаження на процесор і жорсткий диск. Для роботи проксі zabbix окремо потрібна база даних;

3. Агент — програма (демон), яка активно моніторить і збирає статистику роботи локальних ресурсів (накопичувачі, оперативна пам'ять, процесор та ін.) І додатків;

4. Вебінтерфейс — є частиною сервера системи та потребує для роботи вебсервер. Часто запускається на тому ж фізичному вузлі, що й zabbix.

Функціонал включає загальні перевірки для найбільш поширених сервісів, включаючи СУБД, SSH, Telnet, VMware, NTP, POP, SMTP, FTP тощо. Якщо стандартних налаштувань системи недостатньо, їх можна змінити самостійно або користуватися доповненням через API.

Для опису системи моніторингу Zabbix існує два ключових поняття:

- вузли мережі — робочі пристрої та їх групи (сервери, робочі станції, комутатори), які необхідно перевіряти. Із створення та налаштування вузлів мережі зазвичай починається практична робота з Zabbix;
- елементи даних — набір самостійних метрик, за якими відбувається збір даних з вузлів мережі. Налаштування елементів даних здійснюється на вкладці «Елемент даних» або в автоматичному режимі — через підключення шаблону.

Сам Zabbix-агент здатний відображати поточний стан фізичного сервера, збираючи сукупність даних. У нього досить багато метрик. За їх допомогою можна перевірити завантаженість ядра (Processor load), час очікування ресурсів (CPU iowait time), обсяг системи підкачки (Total swap space) та багато іншого.

Система моніторингу Zabbix сама по собі є потужним інструментом, але через інтеграцію з іншими інструментами моніторингу та управління її функціональність може бути значно розширена[3].

Інтеграція з Grafana

Grafana - це потужний інструмент для візуалізації даних. Інтегруючи Zabbix з Grafana, можна використовувати дані моніторингу для створення динамічних та легкозрозумілих графіків і діаграм. Це дозволяє отримувати чітку та цінну інформацію про стан інфраструктури в режимі реального часу.

Інтеграція з Prometheus

Prometheus - це система моніторингу та сповіщення, спеціалізована на зборі та аналізі метрик. Інтегруючи Zabbix з Prometheus, можна об'єднати потужну платформу моніторингу Zabbix з гнучкими можливостями збору та аналізу даних Prometheus, використовуючи переваги обох систем.

Інтеграція з Nagios

Nagios - це відома система моніторингу з безліччю функцій. Інтегруючи Zabbix з Nagios, можна об'єднати різні плагіни та функції моніторингу Nagios з потужними аналітичними та сповіщувальними можливостями Zabbix, отримуючи максимальну вигоду від обох систем.

Інтеграція з інструментами управління

Інтеграція Zabbix з інструментами управління, такими як Ansible, Puppet або Chef, дозволяє автоматизувати налаштування та управління системами моніторингу. Це спрощує впровадження, забезпечує узгодженість налаштувань та дозволяє швидко реагувати на зміни в інфраструктурі.

Для відстеження алертів від системи моніторингу, буде використовуватися сповіщення через Telegram-бота.

Інтеграція сповіщень в Zabbix через Телеграм-бота значно покращує ефективність та зручність моніторингу інфраструктури. Telegram-бот дозволяє отримувати сповіщення миттєво, що скорочує час реакції на інциденти та проблеми. Сповіщення можна отримувати на будь-якому пристрої з доступом до інтернету, що забезпечує високу гнучкість та мобільність. Додатково, Telegram-боти дозволяють налаштовувати формат і зміст повідомлень, що допомагає уникати перевантаження інформацією та отримувати лише найбільш важливі дані. Інтерактивні команди дозволяють виконувати певні дії безпосередньо з месенджера, наприклад, підтверджувати сповіщення або запитувати додаткову інформацію. До того ж, використання Telegram не потребує додаткових витрат, що робить цю інтеграцію економічно вигідною.

Проте, така інтеграція має і свої мінуси. Відсутність доступу до інтернету унеможливлює доставку сповіщень, що може бути критичним у певних ситуаціях. Дані, що передаються через Telegram, можуть бути вразливими без належних заходів безпеки. Іноді повідомлення можуть затримуватись або не доставлятись через проблеми зі зв'язком або сервісом Telegram. Хоча Telegram-боти мають широкі можливості, вони можуть бути менш гнучкими порівняно з іншими методами сповіщення та управління.

Існують також певні вразливості та загрози при використанні цієї інтеграції. Якщо бот не налаштований належним чином або не використовує шифрування, дані можуть бути перехоплені під час передачі, що може призвести до витоку конфіденційної інформації про інфраструктуру. Зловмисники можуть намагатися обманом отримати доступ до бота або використовувати його для фішингових атак. Некоректні налаштування бота можуть дозволити неавторизованим користувачам отримувати сповіщення або взаємодіяти з ботом. Бот також може стати ціллю для DDoS атак, що призведе до затримки або втрати сповіщень. Використання сторонніх бібліотек для роботи бота може містити вразливості, які зловмисники можуть використати для атаки на систему. Тому необхідно ретельно налаштовувати систему, використовувати додаткові заходи безпеки та постійно контролювати її стан.

2.2 Моніторинг за допомогою Python-скрипту та Telegram-бота

Для моніторингу вебсервера було розроблено python-скрипт, який призначений для моніторингу використання оперативної пам'яті на сервері та інформування адміністратора через Telegram-бота у випадку перевищення встановленого порогу використання пам'яті[13]. Крім того, скрипт може:

1. Надсилати статистику про систему за запитом;
2. Виконувати команди оболонки;
3. Налаштовувати інтервал опитування та поріг використання пам'яті;
4. Створювати графіки використання пам'яті і надсилати їх через telegram.

Для його роботи використано наступні бібліотеки язика програмування python:

- telepot - використовується для взаємодії з Telegram Bot API. Ця бібліотека дозволяє створювати ботів, які можуть надсилати та отримувати повідомлення, виконувати команди, надсилати фотографії та багато іншого.
- subprocess - використовується для запуску нових процесів, взаємодії з вхідними/вихідними потоками і отримання результатів виконання команд.

- psutil - надає функції для роботи з інформацією про системні процеси і використання ресурсів (CPU, пам'ять, диски, мережа і сенсори).
- matplotlib - використовується для створення графіків і візуалізації даних. У скрипті використовується для побудови графіків використання пам'яті.

Плюси використання цього виду моніторингу

1. Простота налаштування та використання: не потребує складного налаштування. Бот легко налаштовується і може бути використаний без глибоких знань у програмуванні;
2. Миттєві сповіщення: адміністратор миттєво отримує повідомлення у telegram при перевищенні порогу використання пам'яті;
3. Гнучкість: можливість змінювати налаштування через telegram-бота (наприклад, інтервал опитування або поріг використання пам'яті);
4. Візуалізація даних: скрипт може створювати графіки використання пам'яті, що допомагає в аналізі та моніторингу;
5. Функціональність: можливість виконання команд оболонки через бот, що дозволяє адміністратору виконувати віддалене керування сервером.

Мінуси та вразливості цього методу:

1. Безпека: використання команд оболонки через Telegram-бота може бути небезпечним. Якщо бот потрапить у неправильні руки, зловмисник може виконувати шкідливі команди на сервері;
2. Залежність від Telegram: якщо Telegram недоступний або заблокований, бот не зможе надсилати сповіщення;
3. Перевантаження сервера: занадто часте опитування може призвести до перевантаження сервера, особливо якщо бот працює на тому ж сервері, який проводить моніторинг;
4. Відсутність високої деталізації: порівняно з іншими системами моніторингу, цей бот не надає детальної інформації про всі аспекти системи, наприклад, про використання ресурсів конкретними процесами у реальному часі;
5. Конфіденційність: дані про систему передаються через Telegram, що може бути проблемою з точки зору конфіденційності.

Загрози безпеки при використанні скрипту:

1. Зловмисний доступ: якщо токен бота або конфіденційна інформація буде скомпрометована, зловмисник може отримати доступ до управління ботом;

2. Відмова в обслуговуванні: неправильне налаштування або зловмисне використання може призвести до відмови в обслуговуванні сервера через надмірне навантаження;

3. Невідповідність налаштувань: неправильне налаштування порогу пам'яті або інтервалу опитування може призвести до пропуску критичних станів або надмірних сповіщень.

Урахування цих аспектів допоможе забезпечити більш надійний та безпечний моніторинг системи.

Далі приведено приклад роботи даного бота в групі Telegram, де можливо використовувати наступні команди:

1. /stats - виводить повідомлення про: час онлайн, кількість оперативної пам’яті, кількість доступної оперативної пам’яті, процент використання оперативної пам’яті, процент використання фізичної пам’яті, процеси (які навантажують систему)

2. /setmem - використовується для встановлення максимального значення для оперативної пам’яті

3. /memgraph - використовується, щоб побачити графік використання оперативної пам’яті

4. /shell - використовується для відправки команд в Power Shell

5. /setpoll - потрібна для встановлення часу оновлення даних в секундах

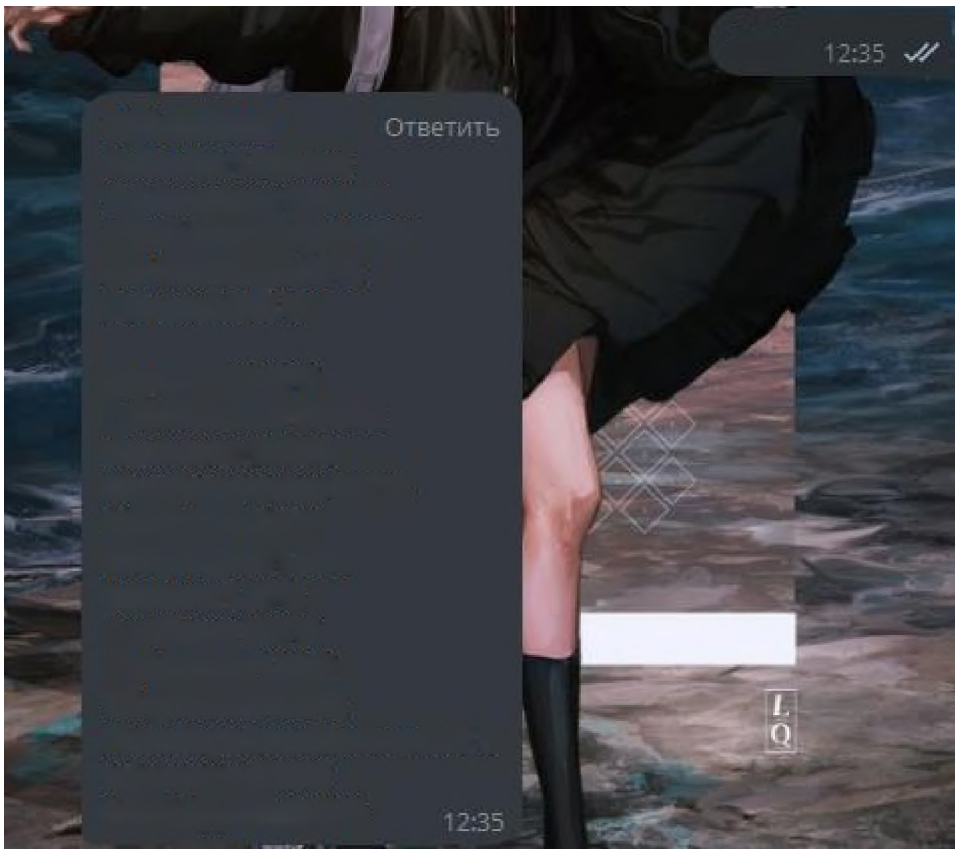

Рис. 2.2.1 - Працездатність команди /stats

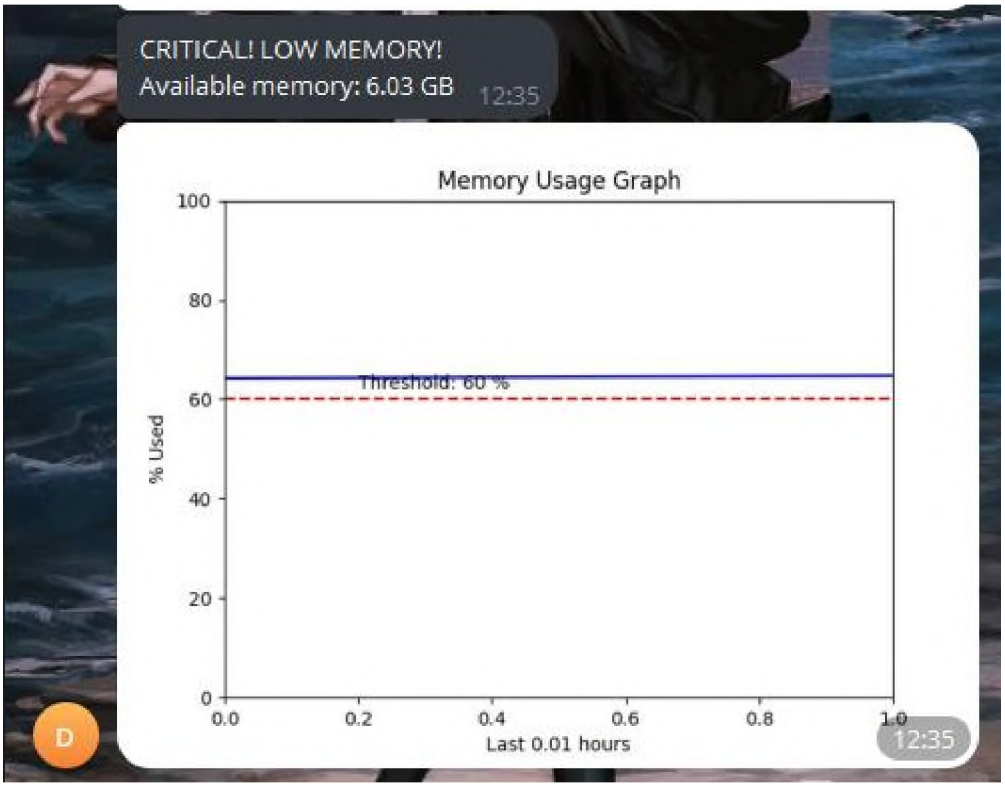

Рис. 2.2.2 - Повідомлення про підвищене використання оперативної пам’яті

2.3 Основні відомості про плату ESP32

Відладочна плата оснащена модулем ESP-WROOM-32, який містить двоядерний 32-розрядний мікропроцесор Tensilica Xtensa LX6. Цей процесор аналогічний до ESP8266, за винятком того, що включає два ядра процесора (кожне з яких може працювати окремо), має тактову частоту від 80 до 240 МГц та продуктивність до 600 DMIPS (мільйонів інструкцій за секунду)[6].

ESP32 має вбудований приймач і передавач Wi-Fi стандарту 802.11b/g/n HT40, що дозволяє не тільки підключатися до мережі Wi-Fi для взаємодії з Інтернетом, але й створювати власну мережу, до якої можуть підключатися інші пристрої напряму. Wi-Fi Direct також підтримується ESP32, що є зручною альтернативою для однорангових з'єднань, для яких не потрібна точка доступу. Легше налаштувати Wi-Fi Direct, і він має значно вищу швидкість передачі даних, ніж Bluetooth. Чіп також підтримує як Bluetooth 4.0 (BLE/Bluetooth Smart), так і Bluetooth Classic (BT), що робить його ще більш універсальним.

NodeMCU ESP32 представляє собою серію бюджетних мікроконтролерів з низьким енергоспоживанням та вбудованим ESP32 Wi-Fi і двомодовим Bluetooth. ESP32 призначений для додатків Інтернету речей (IoT) з низьким енергоспоживанням. Висока обчислювальна потужність у поєднанні з вбудованими функціями Wi-Fi, Bluetooth та режиму глибокого сну, 520 КБ SRAM, 448 КБ ROM і 4 МБ флеш-пам'яті (для зберігання програмного забезпечення та даних) роблять його підходящим для більшості портативних пристроїв Інтернету речей.

Живлення

Плата оснащена регулятором напруги LDO для підтримки стабільної напруги на рівні 3,3 В, тоді як діапазон робочих напруг Arduino ESP32 складає від 2,2 В до 3,6 В. Коли ESP32 споживає до 250 мА під час радіочастотної передачі, він може надійно подавати до 600 мА, чого має бути більше ніж достатньо. Вихід регулятора також виведений на одну зі сторін плати та позначений як 3V3. Через цей вивід можна подавати живлення на зовнішні компоненти. Вбудований порт MicroUSB забезпечує живлення відладочної плати ESP32. Можна використовувати вивід VIN

для живлення ESP32 та його периферійних пристроїв безпосередньо через зовнішнє джерело живлення 5 В.

Для роботи ESP32 потрібне джерело живлення 3,3 В та логічні рівні 3,3 В. Контакти GPIO не допускають напруги 5 В.

Розпиновка

Плата ESP-32 має 48 контактів загального призначення GPIO, з яких лише 25 доступні як контактні роз'єми з обох сторін плати ESP-32. Ці контакти можуть бути призначені для всіх видів периферійних функцій[12].

- контакти живлення: вивід VIN та вивід 3,3 В є двома контактами живлення. Якщо є керований джерело живлення 5 В, можна використовувати вивід VIN для безпосереднього живлення ESP32 та його периферійних пристроїв. Вихід бортового регулятора напруги підключений до контакту 3,3 В. Через цей вивід можна подавати живлення на зовнішні компоненти;
- контакти Arduino: апаратні контакти I2C та SPI ESP32, які можна використовувати для підключення всіх видів датчиків та периферійних пристроїв;
- контакти GPIO: плата розробки ESP32 оснащена 25 виводами GPIO, які можуть бути програмно призначені для різних функцій. Кожен GPIO з цифровою підтримкою може бути налаштований на високий імпеданс або внутрішнє підтягування чи опускання. Він також може бути встановлений на edge-trigger або level-trigger для генерації переривань процесора при налаштуванні в якості вхідних даних;
- заземлення: контакт заземлення відладочної плати ESP32;
- канали АЦП: плата має 12-розрядні АЦП SAR та 15 каналів вимірювання (аналогові контакти з підтримкою). Деякі з цих виводів можуть бути використані для створення програмованого підсилювача для вимірювання малих аналогових сигналів. ESP32 також здатний вимірювати напруги, перебуваючи у сплячому режимі;
- канали ЦАП: два 8-розрядних канали ЦАП у схемі перетворюють цифрові сигнали в реальні аналогові напруги. Цей подвійний ЦАП може керувати іншими схемами;

- сенсорні панелі: плата оснащена 9 графічними процесорами з ємнісним датчиком, які виявляють зміни ємності, спричинені прямим контактом або близькістю GPIO до пальця чи іншого об’єкта;
- контакти UART: плата розробки ESP32 містить два інтерфейси UART, UART0 та UART2, які забезпечують асинхронний зв'язок (RS232 та RS485) і IrDA зі швидкістю до 5 Мбіт/с. UART забезпечує апаратне керування сигналами CTS та RTS, а також програмне керування потоком (XON та XOFF);
- контакти SPI: контакти SPI ESP32 мають три SPI (SPI, HSPI та VSPI) у режимах slave та master. Усі SPI також можуть використовуватися для підключення до зовнішньої флеш-пам’яті/SRAM та ЖК-дисплею;
- контакти PWM: плата має 25 каналів (майже всі контакти GPIO) контактів PWM, керованих контролером широтно-імпульсної модуляції (PWM). PWM-вихід може використовуватися для керування цифровими двигунами та світлодіодами. Контролер складається з таймерів PWM та оператора PWM. Кожен таймер забезпечує синхронізацію в синхронному або незалежному режимі, і кожен оператор PWM генерує форму сигналу для одного каналу PWM;
- контакт EN: використовується для включення ESP32. Чип активується при високій напрузі. При низькій напрузі чип працює на мінімальній потужності.

Також існує аналог такої плати - її попередник - ESP8266, далі приведено причини використання плати ESP32, замість ESP8266:

ESP8266 має вбудований процесор, але через багатозадачність, пов'язану з оновленням стека Wi-Fi, більшість додатків використовують окремий мікроконтролер для взаємодії з датчиками, цифрового вводу-виводу та обробки даних. При використанні ESP32 може не знадобитися використовувати додатковий мікроконтролер, оскільки ESP32 оснащений двома 32-розрядними мікропроцесорами і працюватиме на розподільчих платах і модулях з частотою від 160 МГц до 240 МГц. Це забезпечує достатню швидкість для будь-якого додатку, якому потрібен мікроконтролер з можливістю підключення.

- ESP32 швидший, ніж ESP8266;
- ESP32 має більше GPIO з багатьма функціями;

- ESP32 підтримує аналогові вимірювання на 18 каналах (аналогові контакти) у порівнянні з одним 10-розрядним виводом АЦП на ESP8266;
- ESP32 підтримує Bluetooth, тоді як ESP8266 - ні;
- ESP32 є двоядерним, а ESP8266 - одноядерним;
- обидві плати можуть бути запрограмовані за допомогою Arduino IDE або інших підтримуваних IDE;
- обидві плати підтримують MicroPython.

2.3.1 Віддалене керування живленням вебсерверу за допомогою зовнішнього пристрою на основі плати ESP32 з розробленим програмним забезпеченням

Уявіть собі ситуацію: ви працюєте над важливим проєктом, використовуючи свій персональний комп'ютер, що є потужною графічною станцією, на якій встановлено безліч спеціалізованих програм. Під час відпустки або відрядження ви раптом отримуєте терміновий запит на внесення змін або доступ до файлів цього проєкту. Але ваш комп'ютер залишився вдома, і доступ до нього обмежений. Ви маєте з собою лише легкий ноутбук, який не може забезпечити таку ж продуктивність, як ваша графічна станція. Така ситуація може призвести до значних затримок і непередбачуваних труднощів у роботі.

Цей приклад ілюструє проблему, з якою стикаються багато професіоналів: необхідність віддаленого доступу до ресурсів потужного комп'ютера, що залишається вдома.

Насправді, зазначена проблема дуже легко вирішується: встановленням на ваш персональний комп'ютер системи віддаленого включення/вимкнення і перезавантаження. Цей підхід дозволить завжди мати під рукою всю потужність вашої домашньої машини та всі необхідні документи і проекти, що зберігаються на ній.

До початку опису своєї розробки хочу сказати, що звичайно, вона не є єдино можливим способом для віддаленого включення вимкненого комп'ютера. Наприклад, добре відома технологія Wake-on-LAN. Суть цієї технології полягає в тому, що вона дозволяє включити вимкнений комп'ютер, відправивши спеціальний пакет на його MAC-адресу.

Мережевий адаптер комп'ютера, що підтримує Wake-on-LAN, знаходиться в цей момент у режимі зниженого споживання і аналізує всі пакети, що надходять. Якщо одним із надходячих пакетів виявиться так званий magic packet, мережевий адаптер видасть сигнал на включення живлення комп'ютера.

Magic packet — це спеціальна послідовність байтів, яку для нормального проходження по локальних мережах можна вставити в пакети транспортного рівня, що не потребують встановлення з'єднання (наприклад, протокол UDP або застарілий IPX). Зазвичай для Wake-on-LAN пакети протоколів верхнього рівня розсилають широкомовно, оскільки у разі динамічного призначення адрес невідомо, яка IP-адреса відповідає якому MAC-адресу. Однак, для коректного проходження через маршрутизатор, що забороняє широкомовні пакети, можна надіслати пакет на якусь певну адресу.

З мінусів цієї технології варто зазначити, що вона:

- повинна підтримуватися апаратно, не всі материнські плати підтримують дану функцію;
- можливі проблеми з відновленням роботи при відключенні робочої станції від мережі та/або її відсутності;
- є небезпечною для апаратної частини, бо перепади напруги можуть нашкодити працездатності робочої станції.

Зовнішній девайс для віддаленого реагування на інциденти на основі плати ESP32 є декілька переваг серед інших засобів вирішення проблеми такого плану:

- це більш безпечно, ніж Wake-on-LAN, має окреме джерело живлення, що не нашкодить цьому девайсу при перепадах напруги та/або її відсутності;
- можна встановити навіть на пристрої, що не підтримують Wake-on-LAN;
- прошивка пристрою забезпечена засобами відновлення зв'язку після втрати живлення/перезавантаження;
- дозволяє примусово перезавантажити «завислий» комп'ютер — віддалено.

В даному випадку керується вебсервер, на якому розміщується сайт та оброблюються всі дані локальної мережі. Сервер знаходиться вдома, зазвичай його

керування виконується за допомогою віддаленого робочого стола, SSH-з'єднання. Однак декілька разів була ситуація, коли вебсервер потребував фізичного втручання, коли треба було віддалено його увімкнути або перезавантажити за декількох причин:

1. Тестова атака на сервер: проводилася тестова ddos-атака на робочу станцію, що призвело до надмірного навантаження на центральний процесор. Для найбільш швидкого відновлення роботи робочої станції треба було перезавантажити її;

2. Проблема з електропостачанням: наразі в країні спостерігаються відключення світла, що погано впливає на працездатність робочої станції. У випадку вимкнення вебсерверу при відключення електропостачання, він не вмикається при відновлення. За допомогою девайсу стає можливим керувати живленням робочої станції віддалено;

3. Зависання системи: декілька разів на вебсервері програмне забезпечення зазнавало збої та зависання з різних причин. В такому випадку ні віддалені команди, ні віддалене з'єднання не дає результати, через перенавантаження. Тому найшвидшим рішенням є перезавантаження робочої станції.

2.4 Висновок

В спеціальній частині описано роботу сервісу моніторингу Zabbix та порівняно з іншими популярними сервісами. Це програмне забезпечення було обрано з причини багатої кількості інтеграцій, що зробило можливим налаштувати сповіщення через Telegram. Це дозволило скоротити затримку реагування на інциденти при виникненні проблем, коли персонал підтримки не на робочому місці та віддалено проводити моніторинг стану вебсерверу не знаходячись на робочому місці.

Для додаткового моніторингу стану використано Python-скрипт на основі бібліотеки psutil. Цей скрипт дозволить призводити моніторинг стану оперативної пом'яті, проценту використання оперативної пам'яті, використання фізичної пам'яті, проценту використання фізичної пам'яті, проценту використання центрального процесора, процеси, які займають більше 0.5% використання

центрального процесора та кількість минулого часу сервера після останнього включення. Можливі побудови графіків на основі використання оперативної пам'яті на основі кількості робочих годин - це дозволить наглядно побачити характер навантаження на оперативну пам'ять та зрозуміти, чи навантаження було підвищено гармонійно системою, або різко із зовнішнього середовища. Для графіків використовується бібліотека matplotlib. Скрипт дозволяє повідомляти про надвисоке навантаження оперативної пам'яті у чат Telegram за допомогою бота. Ліміт використання пам'яті регулюється через Telegram-бота із використанням спеціальної команди. Пропонується виконання PowerShell запитів за допомогою спеціалізованих запитів до Telegram-бота, що робить реагування на інциденти більш швидким.

Після виникнення інциденту, при якому вебсервер має навантаження на оперативну пам'ять, центральний процесор та/або фізичну пам'ять біля 100%, немає можливості відправити віддалені запити до системи. Зазвичай при такому навантаженні система просто не зможе опрацювати жодного нового запиту, в такому випадку більш рентабельним способом буде перезавантажити сервер. Для цього потрібен доступ до фізичної машини 24 години на 7 днів, але проблема вирішується пристроєм на основі ESP32 під віддаленим керівництвом Telegram. Сама плата напряму регулює перемикач включення, вимкнення та перезавантаження системи. Це дозволяє віддалено реагувати на критичні інциденти, де програмно неможливо вирішити проблему. Сама плата налаштована для автономного живлення й не залежить від живлення усієї системи. Команди бот розпізнає від одного користувача, тому шанс хибних спрацювань є невеликим.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Вступ

Основними задачами економічного розділу є аналіз витрат на розробку, впровадження та підтримку системи, а також оцінка потенційних економічних вигод від її використання. Витрати на розробку включають в себе заробітну плату співробітників, витрати на обладнання та програмне забезпечення, а також інші супутні витрати. Впровадження системи потребує детального планування та координації, що також вимагає певних фінансових ресурсів. Підтримка системи включає регулярне оновлення програмного забезпечення, моніторинг та реагування на інциденти, а також навчання персоналу.

З іншого боку, економічні вигоди від впровадження системи можуть включати зниження ризику фінансових втрат від кібератак, підвищення ефективності роботи вебсервера, покращення репутації підприємства завдяки підвищенню рівня безпеки, а також можливість залучення нових клієнтів, які цінують надійність та безпеку. Оцінка економічної ефективності дозволяє виявити співвідношення витрат та вигод, що є ключовим фактором при прийнятті рішення про впровадження нової системи.

3.2 Розрахунок капітальних витрат на впровадження системи моніторингу та реагування на інциденти.

Визначення трудомісткості розробки системи моніторингу та реагування інциденти.

$$t = t_{\text{тз}} + t_{\text{в}} + t_{\text{а}} + t_{\text{вз}} + t_{\text{озб}} + t_{\text{овр}} + t_{\text{д}}, \text{годин}, \quad (3.1)$$

де $t_{\text{тз}}$ - тривалість складання технічного завдання на розробку системи моніторингу та реагування на інциденти,

$t_{\text{в}}$ - тривалість розробки концепції системи моніторингу та реагування на інциденти в організації,

$t_{\text{а}}$ - тривалість процесу аналізу даних,

$t_{\text{вз}}$ - тривалість визначення вимог до програмного та апаратного забезпечення,

$t_{озб}$ - тривалість вибору платформи та основних рішень,

$t_{овр}$ - тривалість організації виконання відновлювальних робіт та забезпечення неперервного функціонування організації,

$t_{д}$ - тривалість документального оформлення системи моніторингу та реагування на інциденти в організації.

Для визначення трудомісткості правці будуть використані наступні вхідні дані:

$t_{тз}$ = 10 годин,

$t_{в}$ = 15 годин,

$t_{а}$ = 20 годин,

$t_{вз}$ = 12 годин,

$t_{озб}$ = 8 годин,

$t_{овр}$ = 18 годин,

$t_{д}$ = 10 годин.

Тоді загальна трудомісткість розробки системи моніторингу та реагування на інциденти визначається як:

$$t = 10 + 15 + 20 + 12 + 8 + 18 + 10 = 93 \text{ години.}$$

Розрахунок витрат на розробку системи моніторингу та реагування на інциденти.

$$K_{рп} = З_{зп} + З_{мч}, \quad (3.2)$$

де $K_{рп}$ - витрати на розробку системи моніторингу та реагування на інциденти,

$З_{зп}$ - заробітна плата спеціаліста,

$З_{мч}$ - вартість витрат машинного часу.

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

$$З_{зп} = t * З_{іб}, \quad (3.3)$$

де t - загальна тривалість розробки системи моніторингу та реагування на інциденти,

$З_{іб}$ - середньогодинна заробітна плата спеціаліста.

Вартість машинного часу для розробки системи моніторингу та реагування на інциденти на ПК визначається:

$$З_{мч} = t * C_{мч}, \text{ грн}, \quad (3.4)$$

де t - трудомісткість розробки системи моніторингу та реагування на інциденти,

$C_{мч}$ - вартість 1 години машинного часу ПК, грн/година.

Вартість 1 години машинного часу ПК визначається за формулою:

$$C_{мч} = P * t_{нал} * C_е + \frac{Ф_{зал} * Н_а}{F_p} + \frac{К_{лпз} * Н_{апз}}{F_p} \quad (3.5)$$

де P - 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 ліцензійне програмне забезпечення, частки одиниці,

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

F_p - річний фонд робочого часу($F_p = 1920$).

Для розрахунків будуть використовуватися наступні вхідні дані:

P = 0,45 кВт,

$t_{нал}$ = 1 година,

$C_е$ = 4,32 грн/кВт*годину,

$Ф_{зал}$ = 20000 грн,

$Н_а$ = 0,2,

$Н_{апз}$ = 0,1,

F_p = 1920 годин,

$К_{лпз}$ = 0 грн(все програмне забезпечення у вільному доступі),

$З_{іб}$ = 300 грн/година.

$$C_{мч} = 0,45 * 1 * 4,32 + \frac{20000*0,2}{1920} + \frac{0*0,1}{1920} = 4,027 \text{ грн/годину.}$$

$$З_{мч} = 93 * 4,027 = 374,511 \text{ грн}$$

$$З_{зп} = 93 * 300 = 27900 \text{ грн}$$

$$K_{рп} = 374{,}511 + 27900 = 28\,274{,}511 \text{ грн}$$

Визначена таким чином вартість розробки системи моніторингу та реагування на інциденти $K_{рп}$ є частиною одноразових капітальних витрат разом з витратами на придбання і налагодження апаратури системи моніторингу та реагування на інциденти.

Капітальні витрати на проектування та впровадження проектного варіанта системи моніторингу та реагування на інциденти складають:

$$К = К_{пр} + К_{зпз} + К_{рп} + К_{аз} + К_{навч} + К_{н}, \quad (3.6)$$

де $К_{пр}$ - вартість розробки системи моніторингу та реагування на інциденти та залучення до цього зовнішніх консультантів, тис. грн,

$К_{зпз}$ - вартість закупівель ліцензійного основного й додаткового програмного забезпечення, тис. грн,

$К_{рп}$ - вартість розробки системи моніторингу та реагування на інциденти, тис. грн,

$К_{аз}$ - вартість закупівлі апаратного забезпечення та допоміжних матеріалів, тис. грн,

$К_{навч}$ - витрати на навчання технічних фахівців і обслуговуючого персоналу, тис. грн,

$К_{н}$ - витрати на встановлення обладнання та налагодження системи моніторингу та реагування на інциденти.

Вхідні дані для розрахунку капітальних витрат на проектування та впровадження проектного варіанта системи моніторингу та реагування на інциденти:

$К_{пр}$ = 15 тис. грн.

Вартість закупівель ліцензійного основного й додаткового програмного забезпечення буде складатися з: Zabbix - моніторинговий сервіс для серверів безкоштовний для використання в будь-якому середовищі, для розробки програмного забезпечення для плати ESP32 використовується Arduino IDE, яке є

безкоштовним в використанні, для програмування використовується вільний блокнот Notepad++, який також розповсюджується безкоштовно, Telegram - є безкоштовним месенджером.

$К_{зпз}$ = 0 тис. грн,

$К_{рп}$ = 28,27 тис. грн.

Вартість закупівлі апаратного забезпечення буде включати в себе: плату ESP32 для реалізації пристрою реагування на інциденти та управління живленням сервера - 250 грн, також докупаються додаткові модулі для плати задля виконання управління живлення IRF520 2*50 = 100 грн, телефон з підтримкою Telegram для малого бізнесу часто використовують корпоративний Samsung Galaxy A05 4/64 - 4000 грн та сім-карту для реєстрації акаунта Telegram - 100 грн.

$К_{аз}$ = 250 + 100 + 100 + 4000 = 4,45 тис. грн

$К_{навч}$ = 5 тис. грн

$К_{н}$ = 10 тис. грн

$$К = 15 + 0 + 28{,}27 + 4{,}45 + 5 + 10 = 62{,}72 \text{ тис. грн.}$$

3.3 Розрахунок експлуатаційних витрат на 1 рік

Експлуатаційні витрати - це поточні витрати на експлуатацію та обслуговування об'єкта проектування за визначений період, що виражений у грошовій формі.

Річні поточні (експлуатаційні) витрати на функціонування системи складають:

$$С = С_{в} + С_{к} + С_{ак}, \text{тис. грн.} \quad (3.7)$$

де $С_{в}$ - це витрати на Upgrade-відновлення й модернізацію системи,

$С_{к}$ - витрати на керування системою,

$С_{ак}$ - витрати, викликані активністю користувачів системи.

Витрати на керування системою можна визначити за формулою:

$$С_{к} = С_{н} + С_{а} + С_{з} + С_{єв} + С_{е} + С_{о} + С_{тос} \quad (3.8)$$

де $С_{н}$ - витрати на навчання кінцевих користувачів й персоналу,

$С_{а}$ - річний фонд амортизаційних відрахувань,

$С_з$ - річний фонд з/п інженерно-технічного персоналу,

$С_{єв}$ - єдиний внесок на загальнообов'язкове державне соціальне страхування,

$С_е$ - вартість електроенергії, яку споживає система,

$С_о$ - витрати на залучення сторонніх організацій,

$С_{тос}$ - витрати на технічне й організаційне адміністрування та сервіс системи.

Вартість електроенергії визнається за формулою:

$$С_е = P * F_p * Ц_е, \text{ грн,} \quad (3.9)$$

де Р - встановлена потужність апаратури, кВт,

F_p - річний фонд робочого часу системи,

$Ц_е$ - тариф на електроенергію, грн/кВт*годин.

Вхідні дані для розрахування витрат на електроенергію:

Р = 0,0018 + 0,005 = 0,0068 кВт (плата ESP32 та смартфон),

F_p = 8760 (цілодобова підтримка живлення),

$Ц_е$ = 4,32 грн/кВт*годину.

Розрахунок вартості електроенергії:

$$С_е = 0{,}0068 * 8760 * 4{,}32 = 257{,}33 \text{ грн}$$

Річний фонд заробітної плати інженерно-технічного персоналу - дорівнює 0 грн, з причини, що система моніторингу та реагування на інциденти - це програмне забезпечення та використання додаткового пристрою, яке обслуговується персоналом підприємства.

Розрахунок витрат на керування системою за наступними вхідними даними: вартість навчання кінцевих користувачів й персоналу визначається за рівнем вартості курсів по програмному забезпеченню та апаратному забезпеченню: курси по Zabbix можна розраховувати по 14 300 грн, та курси по ESP32 5700 грн. Тоді $С_н$ = 14 300 + 5 700 = 20 000 грн.

$С_н$ = 20000 грн,

річний фонд амортизації відрахувань можна визначити як $К_{аз}$/кількість років служби. Тоді $С_а$= 4 450 грн/5 років = 890 грн,

$С_а$ = 890 грн,

$С_з$ = 0 грн,

$С_{єв}$ = 0 грн,

$С_е$ = 257,33 грн,

$С_о$ = 30000 грн,

$С_{тос}$ = 1881,6 грн.

$$С_к = 20000 + 890 + 0 + 0 + 257{,}33 + 30000 + 1881{,}6 = 53028{,}93 \text{ грн}$$

Витрати на керування системою за рік становлять 52828,93 грн.

Далі можливо обчислити річні поточні витрати на функціонування системи. Для обчислення витрати на Upgrade-відновлення й модернізацію системи можна обчислити як: модернізація системи до Raspberry Pi 5 - 5000 грн, модуль сенсорного екрана управління - 5000 грн, додаткові модулі для легшого та більш швидкого реагування - позолоті кабелі, модулі на SMD елементах - 3170 грн,

$С_в$ = 13,17 тис. грн,

$С_к$ = 53,03 тис. грн,

для обчислення витрат, викликаних активністю користувачів системи можна взяти 28 850 грн, ця сума включає в себе з заробітну плату персоналу технічного персоналу - 10 850 грн та персоналу обслуговування системи на рік - 18 000 грн.

$С_{ак}$ = 28,85 тис. грн,

$$C = 13{,}17 + 53{,}03 + 28{,}85 = 95{,}05 \text{ тис. грн.}$$

Таким чином, річні поточні(експлуатаційні) витрати на функціонування системи моніторингу та реагування на інциденти дорівнює 95,05 тис. грн.

3.4 Загальна величина збитку та ефект від впровадження системи

Упущена вигода від простою атакованого вузла або сегмента корпоративної мережі становить:

$$U = П_п + П_в + V, \quad (3.10)$$

де $П_п$ - оплачувані витрати робочого часу та простої співробітників атакованого вузла або сегмента корпоративної мережі, грн,

$П_в$ - вартість відновлення працездатності вузла або сегмента корпоративної мережі, грн,

V - втрата від зниження обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі, грн.

Вартість відновлення працездатності вузла або сегмента корпоративної мережі можна обчислити за формулою:

$$П_п = \frac{\sum З_с}{F} * t_п \ , \quad (3.11)$$

де F - місячний фонд робочого часу(приблизно 176 годин на місяць).

$З_с$ = приблизно 5 осіб $*$ 30000 грн = 150000 грн на місяць,

$t_п$ = 2 години.

Можна визначити вартість відновлення працездатності:

$$П_п = \frac{150000}{176} * 2 = 1704{,}5 \text{ грн}$$

Витрати на відновлення працездатності вузла або сегмента корпоративної мережі включають:

$$П_в = П_{ви} + П_{пв} + П_{зч}, \quad (3.12)$$

де $П_{ви}$ - витрати на повторне уведення інформації, грн,

$П_{пв}$ - витрати на відновлення вузла або сегмента корпоративної мережі, грн,

$П_{зч}$ - вартість заміни устаткування або запасних частин = 40000 грн,

Витрати на повторне введення інформації можна визначити:

$$П_{ви} = \frac{\sum З_с}{F} * t_{ви} \quad (3.13)$$

де $З_с$ - заробітна плата співробітника = 30000 грн,

F = 176 годин,

$t_{ви}$ = 12 години.

Тоді повторне введення інформації буде дорівнювати:

$$П_{ви} = \frac{30000}{176} * 12 = 2045{,}45 \text{ грн.}$$

Витрати на відновлення вузла або сегмента корпоративної мережі, формула для визначення:

$$П_{пв} = \frac{\sum З_о}{F} * t_в. \quad (3.14)$$

де $З_о$ - заробітна плата обслуговуючого персоналу = 25000 грн(одна людина),

$t_в$ - час відновлення після атаки, персоналом, що обслуговує корпоративну мережу = 3 години.

Тоді можна визначити витрати на відновлення вузла або сегмента корпоративної мережі:

$$П_{пв} = \frac{25000}{176} * 3 = 426{,}14 \text{ грн.}$$

Тепер можна обчислити витрати на відновлення працездатності вузла або сегмента корпоративної мережі:

$$П_в = 2045{,}45 + 426{,}14 + 40000 = 42471{,}59 \text{ грн}$$

Витрати на зниження очікуваного обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі обчислити за формулою:

$$V = \frac{O}{Fr} * (t_в + t_п + t_{ви}), \quad (3.15)$$

де Fr - річний фонд часу роботи організації = 2080 годин,

О - обсяг продажів атакованого вузла або сегмента корпоративної мережі = 40 000 000 грн за рік,

$t_п$ - час простою вузла або сегмента корпоративної мережі в наслідок атаки = 3 години.

Обчислення витрат на зниження очікуваного обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі:

$$V = \frac{40\,000\,000}{2080} * (3 + 3 + 12) = 346\,153{,}85 \text{ грн.}$$

Тепер можна обчислити упущену вигода від простою атакованого вузла або сегмента корпоративної мережі:

$$U = 1704.5 + 42471{,}59 + 346\,153{,}85 = 390\,329{,}94 \text{ грн.}$$

Далі обчислення загального збитку від атак на вузол або сегмент корпоративної мережі:

$$B = \sum i \sum n\,U \quad (3.16)$$

де I - число атакованих вузлів або сегментів корпоративної мережі = 1,

N - середнє число атак на рік = 3 (середня кількість для малого бізнесу).

Тепер обчислення:

$$B = 1 * 3 * 390\,329{,}94 = 1\,170\,989{,}82 \text{ грн.}$$

Тобто загальний збиток від атаки на вузол або сегмент корпоративної мережі організації складає 1 170 989,82 грн.

Загальний ефект від впровадження системи моніторингу та реагування на інциденти визначається:

$$E = B * R - C, \tag{3.17}$$

де R - очікувана імовірність атаки на вузол або сегмент корпоративної мережі = 0,3 (середня частка для атаки на малі бізнеси).

Тоді:

$$E = 1\,170{,}99 * 0{,}3 - 95{,}05 = 256{,}25 \text{ тис. грн.}$$

3.5 Визначення та аналіз показників економічної ефективності системи

Показник сукупної вартості володіння (ТСО) використовується, якщо величину відверненого збитку від атаки на вузол або сегмент корпоративної мережі важко або неможливо визначити у вартісній формі.

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження системи моніторингу та реагування на інциденти.

$$ROSI = \frac{E}{K}, \tag{3.18}$$

де E = 256,25 тис. грн,

K = 62,72 тис. грн.

Тоді розрахунок:

$$ROSI = \frac{256{,}25}{62{,}72} = 4{,}09.$$

Організація здійснює фінансування капітальних інвестицій у систему моніторингу та реагування на інциденти за рахунок позикових коштів. Тоді проект визначається економічно доцільним, коли:

$$ROSI > (N_{\text{кр}} + N_{\text{інф}})/100, \tag{3.19}$$

де $N_{\text{кр}}$ - банківська кредитна ставка = 7% для малого бізнесу,

$N_{\text{інф}}$ - рівень річної інфляції = 5,1 %.

Тоді розрахунок:

$$4{,}09 > (7 + 5{,}1)/100,$$

$$4{,}09 > 0{,}121,$$

є вірним і проект системи моніторингу та реагування на інциденти є економічно доцільним.

Термін окупності капітальних інвестицій показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження системи моніторингу та реагування на інциденти:

$$T_o = \frac{1}{ROSI}, \text{ років.} \quad (3.20)$$

Тоді можна обчислити термін окупності капітальних інвестицій:

$$\frac{1}{4.09} = 0.25 \text{ років.}$$

3.6 Висновок

В економічному розділі дипломного проекту було проведено детальний аналіз витрат та економічної ефективності впровадження системи моніторингу та реагування на інциденти для захисту вебсерверу. В ході аналізу було розраховано капітальні витрати у розмірі 62,72 тис. грн, експлуатаційні витрати за рік у розмірі 95,05 тис. грн та вартість розробки системи в 27,28 тис. грн.

Результати розрахунків показали, що інвестиції в систему моніторингу та реагування на інциденти є економічно доцільними ROSI = 4,09 та здатні забезпечити швидку окупність $T_o = 0.25$ років за рахунок зменшення простоїв, підвищення ефективності реагування на інциденти та зниження потенційних збитків від кібератак.

Таким чином, впровадження даної системи сприятиме не лише забезпеченню безпеки інформації, але й підвищенню загальної продуктивності та стабільності роботи вебсерверу, що є критично важливим для забезпечення безперервної діяльності підприємства.

ВИСНОВКИ

У роботі було розглянуто та впроваджено систему захисту інформації для вебсервера з функцією моніторингу та реагування на інциденти. Основними цілями роботи були забезпечення безперебійної роботи вебсервера, своєчасне сповіщення про інциденти та можливість віддаленого керування сервером. Для досягнення мети роботи було використано: інструмент для моніторингу Zabbix з інтеграцією в Telegram для сповіщення про інциденти та загрози для стану вебсервера; Python-скрипт для додаткового віддаленого моніторингу за станом, а також для віддаленого керування за допомогою оболонки PowerShell; зовнішній пристрій на основі плати ESP32 з розробленим програмним забезпеченням для розширення можливостей керування мережею вебсервера, що дозволяє швидко реагувати віддалено на інциденти з фізичною машиною.

У першому розділі було розглянуто загальне використання вебсерверів Nginx та Apache, оглянуто сучасні методи захисту інформації на вебсерверах та системи моніторингу та реагування на інциденти. Було детально проаналізовано сучасні підходи до захисту інформації, включаючи використання антивірусних програм, систем виявлення та запобігання вторгнень (IDS/IPS), політики контролю доступу та шифрування даних.

У другому розділі було описано засоби для створення надійної системи захисту інформації вебсервера з функцією моніторингу та реагування на інциденти. Було розглянуто архітектуру системи, налаштування та інтеграцію Zabbix для моніторингу, розробку скриптів на Python для збору даних та відправлення сповіщень через Telegram-бота, а також впровадження зовнішнього пристрою на основі плати ESP32 для віддаленого керування живленням сервера.

У третьому розділі здійснено аналіз економічної ефективності впровадження системи моніторингу та реагування на інциденти, оцінку витрат на розробку, впровадження та підтримку, а також потенційні економічні вигоди для організацій. Було проаналізовано трудомісткість розробки системи, витрати на закупівлю апаратного забезпечення, навчання персоналу та підтримку системи. Проведений

економічний аналіз показав, що впровадження даної системи дозволить значно знизити ризики, пов'язані з кіберзагрозами, зменшити час простою серверів та підвищити загальний рівень безпеки вебсервера.

Практична цінність розробки полягає у підвищенні надійності працездатності вебсервера за допомогою системи моніторингу Zabbix та скриптів на Python, своєчасного сповіщення за допомогою інтеграції з Telegram, віддаленому керуванні мережею вебсервера зовнішнім пристроєм на основі плати ESP32 з розробленим програмним забезпеченням, а також підвищенні загального рівня безпеки сервера. Розроблена система дозволяє оперативно виявляти та реагувати на інциденти, що забезпечує безперебійне функціонування вебсервера та знижує ризик втрат даних чи доступу до конфіденційної інформації. Також перевагою цієї системи є мала ціна реалізації, що дозволяє запровадити її в системи багатого спектру. Багата інтеграція Zabbix та гнучке налаштування можливостей плати ESP32 дає можливість запровадити систему майже на всі платформи та не залежить від самої машини.

Таким чином, проведене дослідження та впроваджена система моніторингу та реагування на інциденти демонструють високу ефективність у забезпеченні захисту інформації вебсервера та можуть бути рекомендовані для використання у подібних проектах та організаціях, які прагнуть підвищити рівень кібербезпеки та надійності своїх інформаційних систем.

ПЕРЕЛІК ПОСИЛАНЬ

1. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів спеціальності 125 Кібербезпека / Упоряд: О.В. Герасіна, Д.С. Тимофеєв, О.В. Кручінін, Ю.А. Мілінчук - Дніпро: НТУ «ДП», 2020.

2. Методичні вказівки до виконання економічної частини дипломного проекту зі спеціальності 125 Кібербезпека / Упоряд.: Д.П. Пілова. - Дніпро: Національний технічний університет "Дніпровська політехніка", 2019.

3. Інтеграція Zabbix з іншими інструментами моніторингу та управління. URL: https://blog.silvery.ua/zabbix-integration-with-other-tools

4. Nginx Documentation. URL: https://nginx.org/en/

5. Apache HTTP Server (httpd): A Comprehensive Review. URL: https://www.linkedin.com/pulse/apache-http-server-httpd-comprehensive-review-mohd-asif-ansari-qk8re/

6. Hands-on review: ESP32 offers a powerful IoT-enabled MCU for novices and pros alike. URL: https://www.electronicproducts.com/hands-on-review-esp32-offers-a-powerful-iot-enabled-mcu-for-novices-and-pros-alike/

7. Overview of Web Server Security Methods and Tools. URL: https://www.sans.org/white-papers/36367/

8. Intrusion Detection and Prevention Systems (IDS/IPS). URL: https://www.cisco.com/c/en/us/products/security/intrusion-prevention-system-ips/index.html

9. Antivirus Software for Servers. URL: https://www.kaspersky.com/small-to-medium-business-security

10. Data Encryption Techniques. URL: https://www.ibm.com/security/data-encryption

11. Access Control Policies and Management. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-12.pdf

12. ESP32 Technical Reference Manual. URL: https://www.espressif.com/en/support/download/documents

13. Python Scripting for Monitoring and Automation. URL:

https://realpython.com/tutorials/scripting/

14. May 2024 Web Server Survey URL: https://www.netcraft.com/blog/may-2024-web-server-survey/

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	**Формат**	**Найменування**	**Кількість листів**	**Примітка**
1	А4	Реферат	2	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	3	
5	А4	1 Розділ	42	
6	А4	2 Розділ	20	
7	А4	3 Розділ	11	
8	А4	Висновки	2	
9	А4	Перелік посилань	2	
10	А4	Додаток А	1	
11	А4	Додаток Б	7	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	
14	А4	Додаток Д	1	

ДОДАТОК Б. Лістінг програми для плати ESP32 та Python-скрипт моніторингового Telegram-бота

Код Telegram-бота для керування платою ESP32

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

const char* ssid = "SSID_Network";
const char* password = "PASSWORD_Network";

#define BOTtoken "BOT_Token"
#define CHAT_ID "-CHAT_ID"
#ifdef ESP8266
  X509List cert(TELEGRAM_CERTIFICATE_ROOT);
#endif
WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

const int mosfetPin19 = 19;
const int mosfetPin21 = 21;

const int LED = 2;
bool LEDoff = LOW;
bool LEDon = HIGH;
const uint32_t comp_ON_period = 500;
const uint32_t comp_Restart_period = 500;
const uint32_t comp_OFF_period = 4000;
uint32_t pauseStartTime = 0;

WiFiClient wifiClient;

void setup_wifi() {
   delay(10);
   WiFi.mode(WIFI_STA);
   WiFi.begin(ssid, password);
   #ifdef ESP32
   client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
   #endif
   while (WiFi.status() != WL_CONNECTED) {
     delay(1000);
     Serial.print("Connecting to WiFi..");
   }
   Serial.println("IP address: ");
```

```
  Serial.println(WiFi.localIP());
  bot.sendMessage(CHAT_ID, "Bot Started", "");
}
void handleNewMessages(int numNewMessages) {
 Serial.println("handleNewMessages");
 Serial.println(String(numNewMessages));

 for (int i=0; i<numNewMessages; i++) {
  String chat_id = String(bot.messages[i].chat_id);
  if (chat_id != CHAT_ID){
   bot.sendMessage(chat_id, "Unauthorized user", "");
   continue;
  }

  String text = bot.messages[i].text;
  Serial.println(text);

  String from_name = bot.messages[i].from_name;

  if (text == "/offpc") {
   bot.sendMessage(chat_id, "Turning off your PC", "");
   digitalWrite(LED, LEDoff);

   pauseStartTime = millis();

   digitalWrite(mosfetPin19, LEDon);
   digitalWrite(LED, LEDon);

   while ( (millis() - pauseStartTime)<comp_OFF_period ){};

   digitalWrite(mosfetPin19, LEDoff);
   digitalWrite(LED, LEDoff);
   bot.sendMessage(chat_id, "Your PC was turned off", "");
  }

  if (text == "/onpc") {
   bot.sendMessage(chat_id, "Turning on your PC", "");

   pauseStartTime = millis();

   digitalWrite(mosfetPin19, LEDon);
   digitalWrite(LED, LEDon);

   while ( (millis() - pauseStartTime)<comp_ON_period ){};

   digitalWrite(mosfetPin19, LEDoff);
   digitalWrite(LED, LEDoff);

    bot.sendMessage(chat_id, "Your PC was turned on", "");
```

```
    }

    if (text == "/restartpc") {
     bot.sendMessage(chat_id, "Restarting your PC", "");

     pauseStartTime = millis();

     digitalWrite(mosfetPin21, LEDon);
     digitalWrite(LED, LEDon);

     while ( (millis() - pauseStartTime)<comp_Restart_period ){};

     digitalWrite(mosfetPin21, LEDoff);
     digitalWrite(LED, LEDoff);

     bot.sendMessage(chat_id, "Your PC was restarted", "");
    }
   }
}


void setup() {
  Serial.begin(115200);
  #ifdef ESP8266
    configTime(0, 0, "pool.ntp.org");
    client.setTrustAnchors(&cert);
  #endif
  pinMode(LED, OUTPUT);
  Serial.setTimeout(500);
  setup_wifi();
  pinMode(mosfetPin19, OUTPUT);
}

void loop(){
  if (WiFi.status() != WL_CONNECTED)
     {
         setup_wifi();
     }
  if (millis() > lastTimeBotRan + botRequestDelay) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

Лістінг Python-скрипту

```
from tokens import *
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import psutil
from datetime import datetime
from subprocess import Popen, PIPE, STDOUT
import operator
import collections
import time
import telepot

memorythreshold = 85
poll = 10

shellexecution = []
timelist = []
memlist = []
xaxis = []
settingmemth = []
setpolling = []
graphstart = datetime.now()

stopmarkup = {'keyboard': [['Stop']]}
hide_keyboard = {'hide_keyboard': True}

def clearall(chat_id):
    if chat_id in shellexecution:
        shellexecution.remove(chat_id)
    if chat_id in settingmemth:
        settingmemth.remove(chat_id)
    if chat_id in setpolling:
        setpolling.remove(chat_id)

def plotmemgraph(memlist, xaxis, tmperiod):
    plt.xlabel(tmperiod)
    plt.ylabel('% Used')
    plt.title('Memory Usage Graph')
    plt.text(0.1*len(xaxis), memorythreshold+2, 'Threshold: '+str(memorythreshold)+ ' %')
    memthresholdarr = []
    for xas in xaxis:
        memthresholdarr.append(memorythreshold)
    plt.plot(xaxis, memlist, 'b-', xaxis, memthresholdarr, 'r--')
    plt.axis([0, len(xaxis)-1, 0, 100])
    plt.savefig('/tmp/graph.png')
    plt.close()
    f = open('/tmp/graph.png', 'rb')
    return f
```

```
class YourBot(telepot.Bot):
    def __init__(self, *args, **kwargs):
        super(YourBot, self).__init__(*args, **kwargs)
        self._answerer = telepot.helper.Answerer(self)
        self._message_with_inline_keyboard = None

    def on_chat_message(self, msg):
        content_type, chat_type, chat_id = telepot.glance(msg)
        print("Your chat_id:" + str(chat_id))
        if chat_id in adminchatid:
            if content_type == 'text':
                if msg['text'] == '/stats' and chat_id not in shellexecution:
                    bot.sendChatAction(chat_id, 'typing')
                    memory = psutil.virtual_memory()
                    disk = psutil.disk_usage('/')
                    boottime = datetime.fromtimestamp(psutil.boot_time())
                    now = datetime.now()
                    timedif = "Online for: %.1f Hours" % (((now - boottime).total_seconds()) / 3600)
                    memtotal = "Total memory: %.2f GB " % (memory.total / 1000000000)
                    memavail = "Available memory: %.2f GB" % (memory.available / 1000000000)
                    memuseperc = "Used memory: " + str(memory.percent) + " %"
                    diskused = "Disk used: " + str(disk.percent) + " %"
                    pids = psutil.pids()
                    pidsreply = ''
                    procs = {}
                    for pid in pids:
                        p = psutil.Process(pid)
                        try:
                            pmem = p.memory_percent()
                            if pmem > 0.5:
                                if p.name() in procs:
                                    procs[p.name()] += pmem
                                else:
                                    procs[p.name()] = pmem
                        except:
                            print("Hm")
                    sortedprocs = sorted(procs.items(), key=operator.itemgetter(1), reverse=True)
                    for proc in sortedprocs:
                        pidsreply += proc[0] + " " + ("%.2f" % proc[1]) + " %\n"
                    reply = timedif + "\n" + \
                            memtotal + "\n" + \
                            memavail + "\n" + \
                            memuseperc + "\n" + \
                            diskused + "\n\n" + \
                            pidsreply
                    bot.sendMessage(chat_id, reply, disable_web_page_preview=True)
                elif msg['text'] == "Stop":
```

```
                clearall(chat_id)
                bot.sendMessage(chat_id, "All operations stopped.",
reply_markup=hide_keyboard)
            elif msg['text'] == '/setpoll' and chat_id not in setpolling:
                bot.sendChatAction(chat_id, 'typing')
                setpolling.append(chat_id)
                bot.sendMessage(chat_id, "Send me a new polling interval in seconds? (higher than
10)", reply_markup=stopmarkup)
            elif chat_id in setpolling:
                bot.sendChatAction(chat_id, 'typing')
                try:
                    global poll
                    poll = int(msg['text'])
                    if poll > 10:
                        bot.sendMessage(chat_id, "All set!")
                        clearall(chat_id)
                    else:
                        1/0
                except:
                    bot.sendMessage(chat_id, "Please send a proper numeric value higher than 10.")
            elif msg['text'] == "/shell" and chat_id not in shellexecution:
                bot.sendMessage(chat_id, "Send me a shell command to execute",
reply_markup=stopmarkup)
                shellexecution.append(chat_id)
            elif msg['text'] == "/setmem" and chat_id not in settingmemth:
                bot.sendChatAction(chat_id, 'typing')
                settingmemth.append(chat_id)
                bot.sendMessage(chat_id, "Send me a new memory threshold to monitor?",
reply_markup=stopmarkup)
            elif chat_id in settingmemth:
                bot.sendChatAction(chat_id, 'typing')
                try:
                    global memorythreshold
                    memorythreshold = int(msg['text'])
                    if memorythreshold < 100:
                        bot.sendMessage(chat_id, "All set!")
                        clearall(chat_id)
                    else:
                        1/0
                except:
                    bot.sendMessage(chat_id, "Please send a proper numeric value below 100.")

            elif chat_id in shellexecution:
                bot.sendChatAction(chat_id, 'typing')
                p = Popen(msg['text'], shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
                output = p.stdout.read()
                if output != b'':
                    bot.sendMessage(chat_id, output, disable_web_page_preview=True)
```

```
                else:
                    bot.sendMessage(chat_id, "No output.", disable_web_page_preview=True)
            elif msg['text'] == '/memgraph':
                bot.sendChatAction(chat_id, 'typing')
                tmperiod = "Last %.2f hours" % ((datetime.now() - graphstart).total_seconds() /
3600)
                bot.sendPhoto(chat_id, plotmemgraph(memlist, xaxis, tmperiod))
    TOKEN = telegrambot
    bot = YourBot(TOKEN)
    bot.message_loop()
    tr = 0
    xx = 0
    # Keep the program running.
    while 1:
      if tr == poll:
        tr = 0
        timenow = datetime.now()
        memck = psutil.virtual_memory()
        mempercent = memck.percent
        if len(memlist) > 300:
          memq = collections.deque(memlist)
          memq.append(mempercent)
          memq.popleft()
          memlist = memq
          memlist = list(memlist)
        else:
          xaxis.append(xx)
          xx += 1
          memlist.append(mempercent)
        memfree = memck.available / 1000000
        if mempercent > memorythreshold:
          memavail = "Available memory: %.2f GB" % (memck.available / 1000000000)
          graphend = datetime.now()
          tmperiod = "Last %.2f hours" % ((graphend - graphstart).total_seconds() / 3600)
          for adminid in adminchatid:
            bot.sendMessage(adminid, "CRITICAL! LOW MEMORY!\n" + memavail)
            bot.sendPhoto(adminid, plotmemgraph(memlist, xaxis, tmperiod))
      time.sleep(10)
      tr += 10
```

ДОДАТОК В. Перелік документів на оптичному носії

Пояснювальна записка.docx

Презентація.pptx

ДОДАТОК Г. Відгук керівника економічного розділу

Економічний розділ виконаний відповідно до вимог, які ставляться для кваліфікаційних робіт, та заслуговує на оцінку 92б.(«відмінно»)

Керівник розділу _____ Дар'я ПІЛОВА

(підпис) (ім'я, прізвище)

ДОДАТОК Д. Відгук керівника кваліфікаційної роботи

ВІДГУК

на кваліфікаційну роботу студента групи 125-20-2
Ткаченко Євгенія Валентиновича
на тему: «Система захисту інформації вебсерверу з функцією моніторингу та реагування на інциденти»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, викладених на 97 сторінках.

Метою кваліфікаційної роботи є забезпечення безпрервної роботи вебсервера за рахунок збільшення часу віддаленого моніторингу за його станом та розширення можливостей для віддаленого керування вебсервером.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 125 «Кібербезпека». Для досягнення поставленої мети в кваліфікаційній роботі вирішуються наступні задачі: вивчення основних алгоритмів і методів, оцінка ефективності сучасних систем, проведення експериментальних досліджень, розробка та впровадження прототипу, аналіз можливих відвернених збитків.

Практичне значення результатів кваліфікаційної роботи полягає у забезпеченні безперебійної роботи вебсервера за допомогою впровадження системи моніторингу на основі сервісу Zabbix з інтеграцією в Telegram та Python-скрипту з керуванням за допомогою Telegram та реагування на інциденти за допомогою віддалених можливостей керування зовнішнім пристроєм на основі плати ESP32 з розробленим програмним забезпеченням.

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

Кваліфікаційна робота заслуговує оцінки «».

Керівник кваліфікаційної роботи
к.т.н., доц. каф. БІТ Максим ТКАЧ