МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут інноваційних освітніх технологій

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Савченко А.С.

"___" _____2020 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ

"МАГІСТРА"

ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: "Захист інформації у web-серверах" .

Виконавець: студент групи УС-201Мз Охріменко Сергій Володимирович .

Керівник: к.т.н., доцент Моденов Юрій Борисович .

Нормоконтролер: . Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут інноваційних освітніх технологій

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, напрям підготовки: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за . галузями)" .

(шифр, найменування)

ЗАТВЕРДЖУЮ

Завідувач кафедри

Савченко А.С.

"06" 10 2020 р.

ЗАВДАННЯ

на виконання дипломного проекту студента

Охріменка Сергія Володимировича .

(прізвище, ім'я, по батькові)

1.**Тема проекту**: "Захист інформації у web-серверах"

Затверджена наказом ректора від 06.10.2020 р. за № 1939 /ст .

2. **Термін виконання проекту**: з 05.10.2020р. до 06.12.2020р.

3. **Вихідні данні до проекту**: огляд існуючих web-серверів, аналіз загроз для web-серверів, розробка рекомендацій щодо захисту інформації у web-серверах, тестування розроблених рекомендацій.

4. **Зміст пояснювальної записки**: Розділ 1. Огляд популярних web-серверів, Розділ 2. Дослідження загроз безпеки у сучасних web-серверах, Розділ 3. Розробка рекомендацій щодо налаштування web-серверу відповідно вимог безпеки

5. **Перелік обов'язкового ілюстративного матеріалу**: Популярність web-серверів 1995-2020 рр..

6. **Календарний план-графік:**

№ з/п	*1.1 Завдання*	*Термін виконання*	*Підпис керівника*
1	Підбір і вивчення літературних джерел.	06.10.20 – 25.10.20	
2	Обґрунтування необхідності дослідження безпеки web-серверів.	26.10.20 – 30.10.20	
3	Аналіз існуючих web-серверів.	01.11.20 – 14.11.20	
4	Аналіз загроз web-серверів.	15.11.20 – 31.11.19	
5	Розробка рекомендацій щодо налаштування web-серверів	01.11.20 – 22.11.20	
6	Написання пояснювальної записки.	23.11.20 – 20.12.20	

Студент Охріменко Сергій Володимирович

Керівник дипломної роботи Моденов Юрій Борисович

7. **Консультація з окремого(мих) розділу(ів) роботи:**

Назва розділу	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв

8. **Дата видачі завдання**: “ 6 "____10_____ 2020 р.

Керівник дипломної роботи _____ *Моденов Ю.Б.*

(підпис)

Завдання прийняв до виконання _____ Охріменко С.В.

(підпис випускника) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Захист інформації у web-серверах» містить 56 сторінок, 1 рисунок, 6 використаних джерел.

Ключові слова: WEB-СЕРВЕР, NGINX, APACHE, КІБЕРБЕЗПЕКА, ЗАГРОЗА БЕЗПЕКИ, НАЛАШТУВАННЯ.

Об’єкт дослідження – розробка рекомендацій щодо налаштування web-серверу для захисту інформації від загроз безпеки.

Мета дипломного проекту – дослідження web-серверів і загроз їх безпеці та розробка рекомендацій щодо налаштування web-серверів для захисту від загроз.

Метод дослідження – збір та аналіз необхідної інформації про web-сервери, їх можливості та загрози їх безпеки.

Встановлено, що існує багато загроз для будь-якого web-серверу, відповідно існує ризик викрадення або втрати інформації, тому кожен розробник повинен завжди пам’ятати про безпеку web-серверу, який зазвичай є першою ланкою між користувачем та кінцевим додатком.

Матеріали даного дипломного проекту рекомендується використовувати при налаштуванні будь-якого веб-сайту.

Зміст

Вступ

В наш час, більшості людей важко уявити своє оточення без інтернету. Зараз інтернет став частиною культури і повсякденного життя. Адже в ньому можна знайти інформацію на будь-який смак, крім, він може бути засобом зв'язку, джерелом розваг, засобом зберігання даних і т.д. Але, мабуть, в першу чергу інтернет це, все-таки, веб-сторінки. веб-сторінки - це те, що ми спостерігаємо в рамках своїх оглядачів. Це будь-якої текст, гіперпосилання, зображення, додатки, відео і т.п., іншими словами це контент. Він надходить з web-серверів, за допомогою спілкування і обміну інформацією веб-оглядача і серверів. На даний момент кількість веб-сайтів, перевищує мільярд і ці цифри тільки ростуть. Природно, веб-сайту не обійтися без web-сервера. Без останніх не було б ні «всесвітньої павутини», ні корпоративних мереж в тому вигляді, в якому ми маємо це зараз.

Останнім часом почастішали акти злому корпоративних веб-сайтів, що говорить про слабку захисту, а також про те, що перевантаження одного або декількох сервісів може призвести до збою працездатності web-серверів. Для web-серверів безпека є значною частиною системи. Сервери уразливі через те, що вони розраховані на обмін інформацією з користувачами. Тому хакери можуть внести зміни в код сервера або бази даних, ніж активно і займаються. Згідно з дослідженням журналу Forbes, в середньому кожен день зламують 30 000 нових веб-сайтів. Ці 30 000 сайтів зазвичай є сайтами малого бізнесу, які мимоволі поширюють шкідливе ПЗ, яке було розміщено на них в результаті злому. Кібербезпека тепер є повсякденною проблемою для компаній. Веб-сайти зламують кожен день, і деякі з цих зломів фатальні для атакованих підприємств. "Кіберзлочинність - найбільша загроза для будь-якої компанії в світі". - стверджує CEO IBM Вірджинія Рометті. Коли справа стосується розміру компанії, у кіберзлочинності немає фаворитів. Корпорації зі списку Fortune 500 і Global 2000 організації середнього та малого бізнесу є цілями атак і кіберзлочинних груп.

Для будь-якої компанії інформація є однім із найважливіших активів. Це може бути конфіденційна інформація користувачів, вихідний код програмного забезпечення або фінансова інформація компанії. Незалежно від типу інформації її

втрата може нести великі наслідки для компанії у фінансовому, правовому або юридичному вигляді.

Роблячи висновок з вище сказаного, незалежно від того у вас велика компанія або маленька, для вас є величезний ризик понести як фінансові втрати так і репутаційні. Тому метою даної роботи є визначення ризиків та розробка рекомендацій для їх зменшення на одному з найважливіших компонентів інформаційної системи - web-сервері.

Розділ 1. Огляд популярних web-серверів

Існує величезна кількість різних веб серверів. Вони можуть бути як універсальними, так і підтримувати якісь специфічні мови програмування. Для початку необхідно розуміти, що ж все таки таке веб сервер і які функції він виконує.

Веб-сервер - сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, і видає їм HTTP-відповіді, як правило, разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними.

Веб-сервером називають як програмне забезпечення, яке виконує функції web-сервера, так і безпосередньо комп'ютер, на якому це програмне забезпечення працює.

Клієнт, яким зазвичай є веб-браузер, передає web-серверу запити на отримання ресурсів, позначених URL-адресами. Ресурси - це HTML-сторінки, зображення, файли, медіа-потоки або інші дані, які необхідні клієнту. У відповідь web-сервер передає клієнту запитані дані. Цей обмін відбувається по протоколу HTTP.

Веб-сервери можуть мати різні додаткові функції, наприклад:

- автоматизація роботи веб-сторінок;
- ведення журналу звернень користувачів до ресурсів;
- автентифікація і авторизація користувачів;
- підтримка динамічно генеруються сторінок;
- підтримка HTTPS для захищених з'єднань з клієнтами.

1.1 Огляд ринку web-серверів

З кожним роком ринок web-серверів доповнюється новими представниками. Деякі з них завойовують свою нішу і свої користувачів, а деякі швидко зникають. Всі web-сервери можна поділити на дві великі групи: універсальні і спеціалізовані

До універсальних відносяться такі web-сервери, які можуть працювати с будь-якими додатками та зазвичай виступають в ролі проксі серверу між користувачами та безпосередньо web-додатками. До універсальних web-серверів можна віднести, наприклад, Nginx.

Спеціалізовані web-сервери зазвичай розробляються і працюють лише з одною мовою програмування.

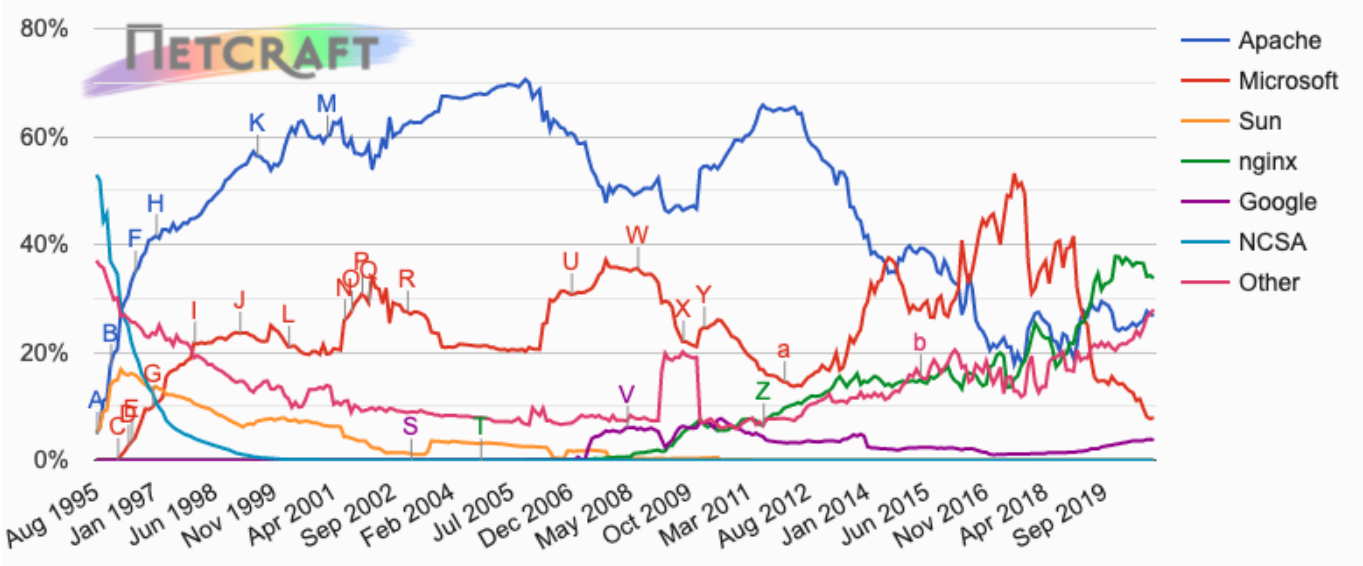

Рис. 1.1 Популярність web-серверів 1995-2020 рр.

Як можна бачити на рис. 1.1 безпосереднім лідером є Nginx (станом на жовтень 2020). За ним йде Apache, який був найпопулярнішим web-сервером у 2000-ні роки, але навіть у 2020 році його можна вважати найкращім вибором для вашого додатку, якщо він написаний на PHP.

Окрім Nginx та Apache HTTP, останнім часом набирає популярність Node JS. Завдяки широкому розповсюдженню програмістів на мові програмування Javascript, для Node JS знайшлось місце у багатьох сферах інтернету: від інтернет магазинів до сервісів потокового стрімінгу відео.

Серед популярних альтернатив Apache HTTP можна виділити lighthttpd. Завдяки його невеликій вимогливості та функціональності багато програмістів PHP віддавали перевагу саме йому.

Останнім в цій роботі буде розглянуто web-сервер для додатків написаних на Java. Завдяки універсальності цієї мови програмування сьогодні можна зустріти багато веб сайтів розроблених на цій мові програмування та web-сервері.

1.2 Історія створення та технічні можливості web-серверів

Nginx

Nginx - це web-сервер, який також можна використовувати в якості зворотного проксі, балансувальника навантаження, поштового проксі і HTTP-кеша. Програмне забезпечення було створено Ігорем Сисоєва і публічно випущено в 2004 році. Nginx - це безкоштовне програмне забезпечення з відкритим вихідним кодом, випущене відповідно до умов ліцензії BSD з двома пунктами. Велика частина web-серверів використовує NGINX, часто як балансувальник навантаження.

Ігор Сисоєв почав розробку Nginx в 2002 році. Спочатку Nginx був розроблений для вирішення проблеми C10k і для задоволення потреб декількох веб-сайтів, включаючи пошукову систему і портал Rambler, для яких до вересня 2008 року він обслуговував 500 мільйонів запитів в день.

Однойменна компанія була заснована в 2011 році для надання підтримки і платного програмного забезпечення Nginx Plus. У березні 2019 року компанія була придбана F5 Networks за 670 мільйонів доларів.

За оцінками Netcraft, на лютий 2020 року Nginx обслужив 36,48% всіх активних веб-сайтів, посівши перше місце, випередивши Apache з 24,51%, в той час як за даними W3Techs, Apache займає перше місце з 40,1% і Nginx на другому місці з 31,8%.

Згідно з опитуванням web-серверів Netcraft за листопад 2016 року, Nginx був другим за популярністю web-сервером серед всіх «активних» сайтів (18 відсотків опитаних сайтів) і серед мільйона найнавантаженіших сайтів (28 відсотків опитаних сайтів). За даними W3Techs, його використовували 38 відсотків з 1 мільйона кращих веб-сайтів, 50 відсотків з 100 000 найкращих веб-сайтів і 57 відсотків з 10 000 кращих веб-сайтів. За даними BuiltWith, він використовується на 38 відсотках з 10 000 кращих веб-сайтів, а його зростання в сегментах 10 000, 100 000 і 1 млн.

Збільшився. Дослідження використання Docker в 2018 році показало, що Nginx є найбільш поширеною технологією, розгортання в контейнерах Docker. У OpenBSD версії 5.2 (листопад 2012 року) Nginx став частиною базової системи OpenBSD, надавши альтернативу системної вилці Apache 1.3, яку він повинен був замінити, але пізніше в версії 5.6 (листопад 2014) він був видалений на користь Apache з власним httpd OpenBSD.

Nginx можна розгорнути для обслуговування динамічного HTTP-контенту в мережі за допомогою FastCGI, обробників SCGI для сценаріїв, серверів додатків WSGI або модулів Phusion Passenger, а також він може служити програмним балансувальник навантаження.

Nginx використовує асинхронний підхід, керований подіями, а не потоки для обробки запитів. Модульна керована подіями архітектура Nginx може забезпечити більш передбачувану продуктивність при високих навантаженнях.

Файл конфігурації Nginx за замовчуванням - Nginx .conf.

HTTP-проксі і можливості web-сервера:

- Здатність обробляти більше 10 000 одночасних підключень при невеликому обсязі пам'яті (~ 2,5 МБ на 10 тис. Неактивних HTTP-з'єднань keep-alive)
- Обробка статичних файлів, індексних файлів і автоіндексування
- Зворотний проксі з кешуванням
- Балансування навантаження за допомогою внутрішніх перевірок працездатності
- TLS / SSL з підтримкою зшивання SNI і OCSP через OpenSSL
- Підтримка FastCGI, SCGI, uWSGI з кешуванням
- Підтримка gRPC з березня 2018 р версія 1.13.10.
- Віртуальні сервери на основі імені і IP-адреси
- IPv6-сумісний
- WebSockets починаючи з 1.3.13, в тому числі діє як зворотний проксі і виконує балансування навантаження додатків WebSocket.

- Оновлення HTTP/1.1 (101 протокол перемикання), підтримка протоколу HTTP/2
- Перезапис і перенаправлення URL

  Можливості поштового проксі:
  - Підтримка TLS / SSL
  - STARTTLS підтримка
  - Проксі-сервер SMTP, POP3 та IMAP
  - Необхідна перевірка справжності за допомогою зовнішнього HTTP-сервера або сценарію перевірки автентичності.

Інші функції включають оновлення виконуваного файлу і конфігурації без втрати клієнтських підключень, а також модульну архітектуру з підтримкою як основних, так і сторонніх модулів.

Платний продукт Plus включає додаткові функції, такі як розширена балансування навантаження і доступ до розширеного набору показників для моніторингу продуктивності.

Apache HTTP

Apache HTTP сервер - це безкоштовне кроссплатформне web-серверне програмне забезпечення з відкритим вихідним кодом, випущене відповідно до умов Apache License 2.0. Apache розробляється і підтримується відкритим співтовариством розробників під егідою Apache Software Foundation.

Переважна більшість примірників Apache HTTP Server працюють в дистрибутиві Linux, але поточні версії також працюють в Microsoft Windows, OpenVMS і великій кількості Unix-подібних систем. Попередні версії також працювали в NetWare, OS/2 і інших операційних системах, включаючи порти на мейнфрейми.

Спочатку заснований на сервері NCSA HTTPd, розробка Apache почалася на початку 1995 року після того, як робота над кодом NCSA застопорилася. Apache зіграв ключову роль в початковому зростанні всесвітньої павутини, швидко

обігнавши NCSA HTTPd як домінуючого HTTP-сервера, і залишається найпопулярнішим у квітні 1996 року. У 2009 році він став першим програмним забезпеченням web-сервера, який обслуговує понад 100 мільйонів веб-сайти. За оцінками Netcraft, на квітень 2020 року Apache обслуговує 29,12% від мільйона найзавантаженіших веб-сайтів, а Nginx - 25,54%; за даними W3Techs, Apache обслуговує 39,5% з 10 мільйонів сайтів, а Nginx - 31,7%.

Замість реалізації єдиної архітектури Apache надає безліч модулів багатопроцесорної обробки (MPM), які дозволяють йому працювати в режимі на основі процесів, в гібридному (процес і потік) режимі або в гібридному режимі подій, щоб краще відповідати вимогам кожної конкретної інфраструктури. Тому важливим є вибір MPM і конфігурації. У тих випадках, коли необхідно йти на компроміс в продуктивності, Apache призначений для зменшення затримки і збільшення пропускної здатності в порівнянні з простою обробкою більшої кількості запитів, що забезпечує узгоджену і надійну обробку запитів в розумні терміни.

Що стосується доставки статичних сторінок, серія Apache 2.2 була визнана значно повільніше, ніж Nginx і varnish. Щоб вирішити цю проблему, розробники Apache створили Event MPM, який поєднує використання декількох процесів і декількох потоків для кожного процесу в асинхронному циклі на основі подій. Згідно Джиму Ягельскі і іншим незалежних джерел, ця архітектура, реалізована в серії Apache 2.4, працює не гірше, ніж web-сервери на основі подій. Однак деякі незалежні, але значно застарілі тести показують, що він як і раніше вдвічі повільніше, ніж Nginx , наприклад.

Apache підтримує безліч функцій, багато з яких реалізовані у вигляді скомпільованих модулів, що розширюють базову функціональність. Вони можуть варіюватися від схем автентифікації до підтримки серверних мов програмування, таких як Perl, Python, Tcl і PHP. Популярні модулі автентифікації включають mod_access, mod_auth, mod_digest і mod_auth_digest, наступника mod_digest. Приклади інших функцій включають підтримку Secure Sockets Layer і Transport Layer Security (mod_ssl), модуль проксі (mod_proxy), модуль перезапису URL

(mod_rewrite), призначені для користувача файли журналу (mod_log_config) і підтримку фільтрації (mod_include і mod_ext_filter).

Популярні методи стиснення в Apache включають зовнішній модуль розширення, mod_gzip, реалізований для зменшення розміру (ваги) веб-сторінок, що обслуговуються через HTTP. ModSecurity - це механізм виявлення і запобігання вторгнень з відкритим вихідним кодом для веб-додатків. Журнали Apache можна аналізувати через веб-браузер за допомогою безкоштовних скриптів, таких як AWStats / W3Perl або Visitors.

Віртуальний хостинг дозволяє одній установці Apache обслуговувати безліч різних веб-сайтів. Наприклад, один комп'ютер з однією установкою Apache може одночасно обслуговувати example.com, example.org, test47.test-server.example.edu і т.д.

Apache пропонує настроюються повідомлення про помилки, бази даних автентифікації на основі СУБД, узгодження вмісту і підтримує кілька графічних користувальницьких інтерфейсів (GUI).

Він підтримує автентифікацію за паролем і автентифікацію з цифровим сертифікатом. Оскільки вихідний код знаходиться у вільному доступі, будь-хто може адаптувати сервер під конкретні потреби, а також існує велика публічна бібліотека надбудов Apache.

Більш докладний список можливостей представлений нижче:

- Офлайн динамічні модулі
- Кілька режимів обробки запитів (MPM), включаючи Event-based/Async, Threaded і Prefork.
- Висока масштабованість (легко обробляє понад 10 000 одночасних підключень)
- Обробка статичних файлів, індексних файлів, автоіндексування узгодження вмісту
- .htaccess підтримка конфігурації для кожного каталогу
- Зворотний проксі з кешуванням

- Балансування навантаження за допомогою внутрішніх перевірок працездатності
- Множинні механізми балансування навантаження
- Міцні та відмово стійкість з автоматичним відновленням
- Підтримка WebSocket, FastCGI, SCGI, AJP і uWSGI з кешуванням
- Динамічна конфігурація
- TLS / SSL з підтримкою зшивання SNI і OCSP через OpenSSL або wolfSSL.
- Віртуальні сервери на основі імені і IP-адреси
- IPv6-сумісний
- Підтримка HTTP / 2
- Деталізована автентифікація і контроль доступу авторизації
- Стиснення і розпакування gzip
- Перезапис URL
- Перезапис заголовків і контенту
- Користувача ведення журналу з ротацією
- Обмеження одночасних підключень
- Обмеження швидкості обробки запитів
- Регулювання смуги пропускання
- Серверна частина включає
- Геолокація на основі IP-адреси
- Відстеження користувачів і сеансів
- WebDAV
- Вбудовані сценарії Perl, PHP і Lua
- Підтримка CGI
- Public_html веб-сторінки для кожного користувача
- Аналізатор універсальних виразів
- Перегляд статусу в реальному часі
- Підтримка XML
- Підтримка FTP (окремим модулем)

Apache Tomcat

Apache Tomcat (скорочено «Tomcat») - це реалізація з відкритим вихідним кодом Java Servlet, JavaServer Pages, Java Expression Language і WebSocket. Tomcat надає середу web-сервера HTTP на «чистому Java», в якій може працювати код Java. Tomcat був запущений в Sun Microsystems в якості еталонної реалізації Java Servlet і Java Server Pages (JSP), пізніше був переданий в Apache Software Foundation. З тих пір кілька добровольців з Sun внесли свій вклад в розробку продукту, в результаті чого в 2005 році був призначений проект Apache вищого рівня. В даний час Apache Tomcat широко використовується багатьма компаніями, оскільки він реалізує багато специфікації Java EE.

Поточна версія Apache Tomcat - 9.x і все ще знаходиться в стадії розробки. У наступній версії 10 повинна з'явитися підтримка Servlet 5.0, JSP 3.0, EL 4.0, WebSocket 2.0 і Authentication 2.0. Tomcat версій 9 і 10 підтримує Java 8 і новіше.
Причина, по якій Tomcat насправді не є повноцінним сервером додатків, полягає в тому, що він діє тільки як web-сервер і контейнер сервлетів. Він не надає повний набір функцій Java EE, але це не обов'язково є недоліком. Багатьом додатків потрібні тільки ті функції, які надає Tomcat, тому немає сенсу використовувати більш важкі інструменти. Ви можете використовувати Apache Tomcat для виробничих програм, які обробляють тисячі запитів, якщо наданих їм функцій досить. У будь-якому випадку Tomcat - це інструмент, готовий до роботи.

Tomcat - це інструмент, що не залежить від платформи, і, поки на ньому встановлена Java, установка являє собою простий процес. Вам просто потрібно завантажити бажану версію з офіційного сайту, а потім розпакувати її в свою файлову систему. Закріпити Tomcat можна, запустивши сервер за допомогою сценарію запуску в папці $CATALINA_BASE/bin.

Після запуску сервера відкрийте свій інтернет-браузер і перейдіть за URL-адресою http://localhost:8080 (якщо використовується конфігурація за замовчуванням). Якщо ви бачите сторінку, аналогічну зображеної на малюнку нижче, це означає, що Tomcat був встановлений правильно.

Розгорнути додаток на сервері дуже просто. Tomcat підтримує розгортання при запуску, що означає, що ви копіюєте стислий (.WAR) або нестислий (розгорнуте веб-додаток) в правильний каталог, який є $CATALINA_BASE/webapps/.

Популярність Tomcat з в'язана з тим, що він дуже простий в освоєнні і безкоштовний. Він пропонує базові функції, необхідні багатьом додаткам, час його запуску досить швидке, а час повторного розгортання набагато менше в порівнянні з іншими інструментами на ринку.

Зіткнувшись з проблемою, велика документація, швидше за все, допоможе вам знайти рішення. Tomcat має дуже хорошу доступну документацію, а то, чого немає в офіційній документації, ви знайдете в Інтернеті, оскільки існує безліч доступних посібників.

Також важливо відзначити, що Tomcat не є універсальним рішенням для всіх додатків Java. Вибір правильного сервера завжди залежить від потреб, що пред'являються додатком.

Компоненти Tomcat

Tomcat 4.x був випущений разом з Catalina (контейнер сервлетів), Coyote (з'єднувач HTTP) і Jasper (механізм JSP).

Catalina

Catalina - це контейнер сервлетів Tomcat. Catalina реалізує специфікації Sun Microsystems для сервлетів і JavaServer Pages (JSP). У Tomcat елемент Realm є «базу даних» імен користувачів, паролів і ролей (аналогічно групам Unix), призначених цим користувачам. Різні реалізації Realm дозволяють інтегрувати Catalina в середовища, де така аутентифікаційних інформація вже створюється і підтримується, а потім використовувати цю інформацію для реалізації безпеки, керованої контейнером, як описано в специфікації сервлетів.

Coyote

Coyote - це компонент Connector для Tomcat, який підтримує протокол HTTP 1.1 в якості web-сервера. Це дозволяє Catalina, номінально що є Java-сервлетом або контейнером JSP, також діяти як простий web-сервер, який обслуговує локальні

файли як HTTP-документи. Coyote прослуховує вхідні з'єднання з сервером через певний порт TCP і пересилає запит у Tomcat Engine для обробки запиту і відправки відповіді запитувачу клієнту. Інший коннектор Coyote, Coyote JK, прослуховує аналогічно, але замість цього перенаправляє свої запити на інший web-сервер, такий як Apache, з використанням протоколу JK. Зазвичай це забезпечує кращу продуктивність.

Jasper

Jasper - це JSP-движок Tomcat. Jasper аналізує файли JSP, щоб компілювати їх в код Java у вигляді сервлетів (це може обробляти Catalina). Під час виконання Jasper виявляє зміни в файлах JSP і перекомпілюються їх.

Починаючи з версії 5, Tomcat використовує Jasper 2, який є реалізацією специфікації Sun Microsystems JSP 2.0. Від Jasper до Jasper 2 були додані важливі функції:

- Об'єднання бібліотеки тегів JSP - кожна розмітка тега в файлі JSP обробляється класом обробника тегів. Об'єкти класу обробника тегів можуть бути об'єднані і повторно використані в усьому сервлет JSP.
- Фонова компіляція JSP - при перекомпіляції зміненого коду Java JSP старіша версія все ще доступна для запитів сервера. Старий сервлет JSP видаляється після завершення перекомпіляції нового сервлету JSP.
- Перекомпілюйте JSP при включенні змін сторінки - сторінки можна вставляти і включати в JSP під час виконання. JSP буде не тільки перекомпілювати зі змінами файлу JSP, але також з включеними змінами сторінки.

Компілятор JDT Java - Jasper 2 може використовувати компілятор Java Eclipse JDT (Java Development Tools) замість Ant і javac.

З випуском Tomcat 7 були додані три нових компонента:

Cluster

Цей компонент був доданий для управління великими додатками. Він використовується для балансування навантаження, яка може бути досягнута за

допомогою багатьох методів. Для підтримки кластеризації в даний час потрібно JDK версії 1.5 або вище.

High availability

Додана функція високої доступності, щоб легко спланувати свою оновлень системи (наприклад, нових випусків, запитів на зміну) без впливу на робоче середовище. Це робиться шляхом відправлення запитів реального трафіку на тимчасовий сервер на іншому порту, в той час як основний сервер оновлюється на основному порту. Це дуже корисно при обробці запитів користувачів в веб-додатках з високою відвідуваністю.

Web application

Він також додав призначені для користувача, а також системні вдосконалення веб-додатків, щоб додати підтримку розгортання в різних середовищах. Він також намагається керувати сеансами, а також додатками в мережі.

Tomcat створює додаткові компоненти. З Apache Tomcat можна використовувати ряд додаткових компонентів. Ці компоненти можуть бути створені користувачами, якщо вони їм знадобляться, або їх можна завантажити з одного з дзеркал.

lighttpd

lighttpd - це web-сервер з відкритим вихідним кодом, оптимізований для середовищ з критичною швидкістю, залишаючись при цьому сумісним зі стандартами, безпечним і гнучким. Спочатку він був написаний Яном Кнешке як доказ концепції проблеми c10k - як обробляти 10 000 паралельних з'єднань на одному сервері, але придбав всесвітню популярність. Його назва - це поєднання слів «світло» і «httpd».

lighttpd підтримує інтерфейси FastCGI, SCGI і CGI для зовнішніх програм, що дозволяє використовувати веб-додатки, написані на будь-якій мові програмування, з сервером. Особлива увага приділяється продуктивності PHP як особливо популярному мови. FastCGI Lighttpd можна налаштувати для правильної і

ефективної підтримки PHP з кешами опкодів (наприклад, APC). Крім того, він отримав увагу завдяки своїй популярності в спільнотах Python, Perl, Ruby і Lua. Lighttpd також підтримує WebDNA, стійку систему баз даних в пам'яті, призначену для створення веб-сайтів на основі баз даних. Це популярний web-сервер для веб-фреймворків Catalyst і Ruby on Rails. Lighttpd не підтримує ISAPI.

Можливості:

- Балансування навантаження, підтримка FastCGI, SCGI і HTTP-проксі
- Підтримка chroot
- Продуктивність механізму подій web-сервера - select (), poll () і epoll ()
- Підтримка більш ефективних схем повідомлення про події, таких як kqueue і epoll
- Умовна перезапис URL (mod_rewrite)
- TLS / SSL з підтримкою SNI через OpenSSL.
- Аутентифікація на сервері LDAP
- Статистика RRDtool
- Завантаження за правилами з можливістю скрипта, який займається обробкою тільки автентифікацію
- На стороні сервера Включає підтримку (але не на стороні сервера CGI)
- Гнучкий віртуальний хостинг
- Підтримка модулів
- Cache Meta Language (в даний час замінюється на mod_magnet) з використанням мови програмування Lua
- Мінімальна підтримка WebDAV
- Підтримка сервлетів (AJP) (у версіях 1.5.x і вище)
- Стиснення HTTP з використанням mod_compress і більш нової версії mod_deflate (1.4.42)
- Легкий (менше 1 МБ)
- Однопроцесний дизайн всього з декількома потоками. Ніякі процеси або потоки не запускалися для кожного з'єднання.

Node.js

Node.js - це кроссплатформенна внутрішнє середовище виконання JavaScript з відкритим вихідним кодом, яка виконує код JavaScript поза веб-браузера. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка і для створення сценаріїв на стороні сервера - запуску сценаріїв на стороні сервера для створення динамічного вмісту веб-сторінки перед відправкою сторінки в веб-браузер користувача. Отже, Node.js є парадигму «JavaScript всюди», що об'єднує розробку веб-додатків на одній мові програмування, а не на різних мовах для серверних і клієнтських скриптів.

Хоча .js є стандартним розширенням імені файлу для коду JavaScript, ім'я «Node.js» не відноситься до конкретного файлу в цьому контексті, а є просто назвою продукту. Node.js має архітектуру, керовану подіями, з можливістю асинхронного введення-виведення. Ці варіанти дизайну спрямовані на оптимізацію пропускної здатності і масштабованості в веб-додатках з безліччю операцій введення / виводу, а також для веб-додатків в реальному часі (наприклад, комунікаційних програм в реальному часі і браузерних ігор).

Проект розподіленої розробки Node.js раніше знаходився під управлінням Node.js Foundation, а тепер злився з JS Foundation, щоб сформувати OpenJS Foundation, якому сприяє програма спільних проектів Linux Foundation.

Корпоративні користувачі програмного забезпечення Node.js включають GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Voxer, Walmart, Yahoo!, I Amazon Web Services.

Спочатку Node.js був написаний Райаном Далем в 2009 році, приблизно через тринадцять років після появи першої серверного середовища JavaScript, Netscape LiveWire Pro Web. Початковий випуск підтримував тільки Linux і Mac OS X. Його розробкою і супроводом керував Дав, а пізніше спонсорував Joyent.

Даль піддав критиці обмежені можливості самого популярного web-сервера в 2009 році, HTTP-сервера Apache, для обробки великої кількості одночасних підключень (до 10 000 і більше) і найбільш поширений спосіб створення коду

(послідовне програмування), коли код або блокує весь процес або мається на увазі кілька стеків виконання в разі одночасних підключень.

Даль продемонстрував проект на першій європейській конференції JSConf 8 листопада 2009 року. Node.js об'єднав движок Google V8 JavaScript, цикл обробки подій і низькорівневий API вводу-виводу.

У січні 2010 року для середовища Node.js був представлений менеджер пакетів під назвою npm. Диспетчер пакетів спрощує програмістам публікацію і спільне використання вихідного коду пакетів Node.js і призначений для спрощення установки, оновлення і видалення пакетів.

У червні 2011 року Microsoft і Joyent впровадили власну версію Node.js. для Windows. Перша збірка Node.js, що підтримує Windows, була випущена в липні 2011 року.

У січні 2012 року Даль відійшов в сторону, призначивши для керівництва проектом свого колегу і творця npm Ісаака Шлютера. У січні 2014 року Шлютері оголосив, що Тімоті Дж. Фонтейн очолить проект.

У грудні 2014 року Федір Індутний запустив io.js, форк Node.js. Через внутрішнього конфлікту з приводу управління Joyent io.js був створений як відкрита альтернатива управління з окремим технічним комітетом. На відміну від Node.js, автори планували підтримувати io.js в актуальному стані з останніми випусками движка Google V8 JavaScript.

У лютому 2015 року було оголошено про намір створити нейтральний фонд Node.js. До червня 2015 року спільноти Node.js і io.js проголосували за спільну роботу в рамках фонду Node.js.

У вересні 2015 року Node.js v0.12 і io.js v3.3 були знову об'єднані в Node v4.0. Це злиття принесло функції V8 ES6 в Node.js і забезпечило довгостроковий цикл випуску підтримки. З 2016 веб-сайт io.js рекомендує розробникам повернутися на Node.js і не планувати подальші випуски io.js через злиття.

У 2019 JS Foundation і Node.js Foundation об'єдналися і утворили OpenJS Foundation.

Node.js дозволяє створювати web-сервери і мережеві інструменти з використанням JavaScript і набору «модулів», які обробляють різні основні функції. Надаються модулі для введення-виведення файлової системи, роботи в мережі (DNS, HTTP, TCP, TLS / SSL або UDP), двійкових даних (буферів), функцій криптографії, потоків даних і інших основних функцій. Модулі Node.js використовують API, призначений для спрощення написання серверних додатків.

JavaScript - єдина мова, яку Node.js підтримує спочатку, але є безліч мов компіляції в JS. В результаті застосування Node.js можна писати на CoffeeScript, Dart, TypeScript, ClojureScript та інших.

Node.js в основному використовується для створення мережевих програм, таких як web-сервери. Найбільш істотна відмінність між Node.js і PHP полягає в тому, що більшість функцій в PHP блокуються до завершення (команди виконуються тільки після завершення попередніх команд), в той час як функції Node.js не блокують (команди виконуються одночасно або навіть паралельно і використовують зворотні виклики. сигналізувати про завершення або відмову).

Node.js офіційно підтримується в Linux, macOS і Microsoft Windows 8.1 і Server 2012 (або пізнішої версії), з підтримкою рівня 2 для SmartOS і IBM AIX і експериментальної підтримкою FreeBSD. OpenBSD також працює, і версії LTS доступні для IBM і (AS / 400). Наданий вихідний код також може бути побудований на операційних системах, аналогічних офіційно підтримуваним, або може бути змінений третіми сторонами для підтримки інших, таких як NonStop OS і сервери Unix.

Архітектура платформи

Node.js забезпечує програмування на основі подій для web-серверів, що дозволяє розробляти швидкі web-сервери на JavaScript. Розробники можуть створювати масштабовані сервери без використання потоків, використовуючи спрощену модель програмування, керованого подіями, яка використовує зворотні виклики для сигналізації про завершення завдання. Node.js об'єднує простоту мови сценаріїв (JavaScript) з потужністю мережевого програмування Unix.

Node.js був побудований на основі движка Google V8 JavaScript, так як він був відкритий з вихідним кодом за ліцензією BSD. Він добре розбирається в основах Інтернету, таких як HTTP, DNS, TCP. JavaScript також був добре відомою мовою, що зробило Node.js доступним для спільноти веб-розробників.

Node.js - це середовище виконання JavaScript, яка обробляє вхідні запити в циклі, званому циклом подій

Node.js використовує під капотом libuv для обробки асинхронних подій. Libuv - це рівень абстракції для функцій мережі і файлової системи як в Windows, так і в системах на основі POSIX, таких як Linux, macOS, OSS на NonStop і Unix.

Висновки до першого розділу

В цьому розділі було розглянуто найбільш популярні web-сервери. Всі вони досить різні та сфери їх використання досить різні. Серед них можна виділити два найбільш популярні: Nginx та Apache HTTP Server. Зважаючи на популярність та універсальність Nginx його було обрано для подальшого огляду загроз безпеки та прикладах налаштування.

Розділ 2. Дослідження загроз безпеки у сучасних web-серверах

2.1 Основні терміни та визначення

У кібербезпеці існує багато термінів та понять, але без деяких із них не можливо в повній мірі розуміти, що саме буде матися на увазі в цьому розділі.

Кібербезпека — це процес застосування заходів безпеки з метою забезпечення конфіденційності, цілісності та доступності даних.

Конфіденційність — захист від несанкціонованого ознайомлення з інформацією.

Цілісність — захист інформації від несанкціонованої модифікації.

Доступність — захист (забезпечення) доступу до інформації, а також можливості її використання. Доступність забезпечується як підтриманням систем в робочому стані так і завдяки способам, які дозволяють швидко відновити втрачену чи пошкоджену інформацію.

Загроза — будь-які обставини чи події, що можуть спричинити порушення політики безпеки інформації та (або) нанесення збитку інформаційній комунікаційній системі. Тобто загроза — це будь-який потенційно можливий несприятливий вплив.

З точки зору інформаційної безпеки ризик розглядають як добуток втрат від порушення конфіденційності, цілісності, автентичності або доступності інформаційних ресурсів на імовірність такого порушення.

2.2 Типи загроз для web-серверів

Не можливо розробити рекомендації щодо безпеки будь-якого продукту без оцінки загроз цього продукту. Для web-серверу можна виділити наступні загрози:

- Розголошення конфіденційних даних
- Недолік контролю доступу

- Некоректне налаштування параметрів безпеки
- Використання компонентів з відомими уразливостями
- Недоліки журналювання і моніторингу
- Відмова в обслуговуванні
- Міжсайтовий скриптинг
- Clickjacking
- Включення файлів
- Ін'єкція команд

Розглянемо кожну з загроз більш детальніше.

Розголошення конфіденційних даних

Замість злому механізмів шифрування зловмисники крадуть ключі, проводять атаки за принципом "людина посередині" або отримують дані в незашифрованому вигляді з сервера, в процесі їх передачі або з клієнта користувача, наприклад, браузера. Подібні атаки зазвичай проводяться вручну. Раніше отримані бази даних паролів можуть бути зламані методом підбору з використанням графічних процесорів.

Протягом останніх років дана атака є найпоширенішою і небезпечною. Найчастіше зустрічається відсутність шифрування конфіденційних даних, а при наявності часто використовуються ненадійні алгоритми, протоколи, шифри, методи зберігання хешування паролів або методи створення та управління ключами.

Через уразливості часто страждають всі персональні дані (медичні записи, облікові дані, дані кредитних карт), які повинні бути захищені згідно із законом, наприклад, відповідно до Загального регламенту ЄС щодо захисту даних (GDPR) або локальними законами про недоторканність даних.

Перш за все необхідно визначити необхідний рівень захисту даних при їх передачі та зберігання. Наприклад, паролі, номери кредитних карт, медичні записи, персональні дані і комерційні таємниці вимагають додаткового захисту, особливо якщо вони підпадають під дію закону про недоторканність даних (напр., Загального

регламенту ЄС щодо захисту даних (GDPR)) або закону про захист фінансових даних (напр., Стандарту безпеки даних в сфері платіжних карт (PCI DSS)).

Недоліки контролю доступу

Експлуатація контролю доступу є основним навиком зловмисників. Інструменти SAST і DAST можуть виявити відсутність контролю доступу, але не можуть перевірити його працездатність при його наявності. Наявність контролю доступу можна виявити вручну, а його відсутність можна виявити автоматично в деяких фреймворками.

Уразливості, пов'язані з контролем доступу, досить поширені через відсутність автоматичного виявлення і ефективного функціонального тестування розробниками. Контроль доступу зазвичай не перевіряється автоматичними статичними або динамічними тестами. Тестування вручну - найкращий спосіб виявлення відсутності або неефективності контролю доступу, включаючи методи HTTP (GET, PUT і т. П.), Контролери, прямі посилання на об'єкти і т.д.

Технічні наслідки: виконання зловмисником дій з правами користувача або адміністратора; використання користувачем привілейованих функцій; створення, перегляд, оновлення або видалення будь-яких записів. Наслідки для бізнесу залежать від критичності захисту програми і даних.

Контроль доступу передбачає наявність політики, що визначає права користувачів. Обхід обмежень доступу зазвичай призводить до несанкціонованого розголошення, зміни або знищення даних, а також виконання непередбачених повноваженнями бізнес-функцій. Найбільш поширені уразливості контролю доступу включають:

- Обхід обмежень доступу шляхом зміни URL, внутрішнього стану додатка або HTML-сторінки, а також за допомогою спеціально розроблених API;
- Можливість зміни первинного ключа для доступу до записів інших користувачів, включаючи перегляд або редагування чужий облікового запису;

- Підвищення привілеїв. Виконання операцій з правами користувача, не входячи в систему, або з правами адміністратора, увійшовши в систему з правами користувача;
- Маніпуляції з метаданими, наприклад, повторне відтворення або підміна токенів контролю доступу JWT або куки-файлів, а також зміна прихованих полів для підвищення привілеїв чи некоректне анулювання JWT;
- Несанкціонований доступ до API через некоректну настройки междоменої використання ресурсів (CORS);
- Доступ нерозпізнаних користувачів до сторінок, які вимагають автентифікації, або доступ непривілейованих користувачів до привілейованих сторінок. Доступ до API з відсутнім контролем привілеїв для POST-, PUT- і DELETE-методів / запитів.

Некоректне налаштування параметрів безпеки

Зловмисники часто намагаються експлуатувати невиправлені уразливості, налаштовані за замовчуванням облікові записи, невикористовувані сторінки, незахищені файли і каталоги для отримання несанкціонованого доступу або інформації про систему.

Налаштування безпеки не буде працювати коректно на будь-якому рівні додатку, включаючи мережеві служби, платформи, веб-служби, сервер, базу даних, фреймворки, код, а також встановлені віртуальні машини, контейнери або сховища. Для пошуку вразливих налаштувань, налаштованих за умовчанням облікових записів, невикористовуваних служб, застарілих параметрів і т. П. Можна використовувати автоматизовані сканери.

Подібні уразливості дозволяють зловмисникам отримати несанкціонований доступ до системних даних або функцій, а також можуть призвести до повної компрометації системи. Наслідки для бізнесу залежать від критичності захисту програми і даних.

Додаток вразливе, якщо:

- Будь-який з компонентів програми недостатньо захищений або дозволу хмарних сервісів некоректно налаштовані;
- Включені або присутні зайві функції (наприклад, невикористовувані порти, служби, сторінки, облікові записи або привілеї);
- Облікові записи і паролі, створювані за замовчуванням, використовуються без змін;
- Обробка помилок дозволяє здійснити трасування стека або отримати занадто докладні повідомлення про помилки;
- Відключені або некоректно налаштовані останні оновлення безпеки;
- Не вибрані безпечні значення параметрів захисту серверів додатків, фреймворків (наприклад, Struts, Spring, ASP.NET), бібліотек і т. П .;
- Сервер не використовує безпечні заголовки або директиви, а також якщо вони некоректно налаштовані;
- ПО застаріло або має уразливості

Без організованою і регулярно виконуваної перевірки безпеки додатків системи більш схильні до ризику.

Використання компонентів з відомими уразливостями

Незважаючи на простоту пошуку вже готових експлойтів для більшості відомих вразливостей, деякі з них вимагають створення спеціальних засобів для їх експлуатації.

Дана уразливість є дуже поширеною. Шаблони для розробників, що містять велику кількість компонентів, можуть призвести до непорозумінь того, які компоненти реально використовуються в додатку або API.

Деякі сканери, такі як retire.js, можуть допомогти з виявленням вразливостей, але визначення складності їх експлуатації потребують додаткових зусиль.

Незважаючи на те, що не всі уразливості призводять до серйозних наслідків, причиною деяких масштабних зломів стали саме компоненти, що містять відомі

уразливості. Залежно від захищаються активів подібна загроза може виявитися на вершині вашого списку.

Додаток вразливе, якщо:

- Ви не знаєте версії всіх використовуваних (на стороні клієнта і на стороні сервера) компонентів. Сюди відносяться самі компоненти і вбудовані залежності;
- ПО містить уразливості, що не підтримується або застаріло. Сюди відносяться ОС, web-сервери, сервери додатків, СУБД, додатки, API, а також всі компоненти, середовища виконання і бібліотеки;
- Пошук вразливостей виконується нерегулярно, а також відсутня підписка на бюлетені з безпеки використовуваних компонентів;
- Своєчасно не встановлюються виправлення або оновлення для використовуваних платформ, фреймворків і залежностей. Зазвичай таке відбувається, коли наявність оновлень перевіряється раз на місяць або квартал, в результаті чого організації тижнями або місяцями не усувають виправлені вразливості;
- Розробники ПЗ не тестують сумісність оновлених або виправлених бібліотек;
- Не забезпечується безпека компонентів

Недоліки журналювання і моніторингу

Експлуатація недоліків журналювання і моніторингу лежить в основі майже всіх великих зломів. При проведенні атак зловмисники покладаються на відсутність контролю і своєчасного реагування на інциденти.

Дана уразливість включена в Топ-10 за результатами галузевих досліджень. Одним із способів визначити якість моніторингу є аналіз журналів після проведення тесту на проникнення. Для визначення можливої шкоди всі дії тестувальників повинні реєструватися відповідним чином.

Більшість атак починаються з аналізу вразливостей. Можливість проведення такого аналізу підвищує ймовірність успішної експлуатації уразливості практично

до 100%. У 2016 році виявлення факту проникнення займало в середньому 191 день - нанесений за цей час збитки міг бути величезним.

Недоліки журналювання, виявлення атак, моніторингу та реагування на інциденти виявляються постійно:

- Піддаються аудиту події, такі як вдалі і невдалі спроби входу в систему, а також важливі транзакції, не реєструються;
- Попередження і помилки не реєструються або реєструються некоректно;
- Журнали додатків і API не перевіряє на предмет підозрілої активності;
- Журнали зберігаються тільки локально;
- Порогові значення попереджень і схеми реагування на інциденти відсутні або є неефективними;
- Тестування на проникнення і сканування інструментами DAST (наприклад, OWASP ZAP) не видають попереджень;
- Додаток не може визначати, реагувати або попереджати про атаки в реальному або майже реальному часі. У системі є витік даних, якщо журнали реєстрації та попередження доступні користувачам або атакуючим

Відмова в обслуговуванні

Атака відмови в обслуговуванні (Denial of Service, або DoS) спрямована на те, щоб зробити ресурс (сайт, додаток, сервер) недоступним для тієї мети, для якої він був розроблений. Є багато способів зробити сервіс недоступним для законних користувачів, маніпулюючи мережевими пакетами, програмуванням, логічними уразливими або уразливими обробки ресурсів. Якщо служба отримує дуже велику кількість запитів, вона може перестати бути доступною для законних користувачів. Таким же чином служба може зупинитися, якщо використовується програмна уразливість або спосіб обробки службою ресурсів, які вона використовує.

Іноді зловмисник може впровадити і виконати довільний код при виконанні DoS-атаки, щоб отримати доступ до важливої інформації або виконати команди на сервері. Атаки типу «відмова в обслуговуванні» значно погіршують якість

обслуговування законних користувачів. Ці атаки викликають великі затримки відповіді, надмірні втрати і перебої в обслуговуванні, що безпосередньо впливає на доступність.

Фактори ризику можна розділити на кілька категорій. Два основних джерела ризику включають неадекватні ресурси і фактори мотивації загроз нетехнічного характеру.

Перший приклад фактору ризику - неадекватні ресурси - вимагає уваги, якщо системна архітектура не була спроектована для задоволення вимог до переповнення трафіку. Цей ризик знижує складність успішного виконання DoS-атаки і, якщо його не контролювати, може привести до появи симптомів DoS-атаки, а не реальною атаки.

Другий приклад і, можливо, найбільший фактор ризику не є технічним, а відноситься до сфери зв'язків з громадськістю або стратегічних комунікацій. Організації слід уникати дій, які можуть зробити її метою DoS-атаки, якщо тільки вигоди від цього не переважують потенційні витрати або заходи щодо пом'якшення наслідків.

Залежно від конкретного середовища також можуть існувати інші фактори ризику.

Міжсайтовий скриптинг

Атаки з використанням міжсайтових сценаріїв (більш відомий як Cross Site Scripting, або XSS) представляють собою тип впровадження, при якому шкідливі сценарії вводяться на безпечні і надійні веб-сайти. XSS-атаки відбуваються, коли зловмисник використовує веб-додаток для відправки шкідливого коду, зазвичай у вигляді сценарію на стороні браузера, іншому кінцевому користувачу. Недоліки, які дозволяють домогтися успіху цих атак, досить широко поширені і виникають скрізь, де веб-додаток використовує введення від користувача у висновку, що воно генерує без перевірки або кодування.

Зловмисник може використовувати XSS для відправки шкідливого сценарію нічого не підозрює користувачеві. Браузер кінцевого користувача не знає, що з цим

сценарієм можна довіряти, і виконає його. Оскільки він вважає, що сценарій виходить з надійного джерела, шкідливий сценарій може отримати доступ до будь-яких файлів cookie, токенів сеансу або іншої конфіденційної інформації, що зберігається браузером і використовуваної на цьому сайті.

Атаки з використанням міжсайтових сценаріїв (XSS) відбуваються, коли:

1. Дані надходять в веб-додаток через ненадійне джерело, найчастіше через веб-запит.
2. Дані включаються в динамічний вміст, яке відправляється веб-користувачу без перевірки на наявність шкідливого вмісту.

Шкідливий контент, що відправляється в веб-браузер, часто приймає форму сегмента JavaScript, але може також включати HTML, Flash або будь-який інший тип коду, який браузер може виконувати. Різноманітність атак, заснованих на XSS, практично безмежно, але зазвичай вони включають в себе передачу зловмисникові особистих даних, таких як файли cookie або іншої інформації сеансу, перенаправлення жертви на веб-контент, контрольований зловмисником, або виконання інших шкідливих операцій на машині користувача. під виглядом уразливого сайту.

Збережені атаки XSS

Збережені атаки - це атаки, при яких впроваджений сценарій постійно зберігається на цільових серверах, наприклад, в базі даних, в форумі повідомлень, журналі відвідувачів, поле коментарів і т. Д. Потім жертва витягує шкідливий сценарій з сервера, коли запитує збережений Інформація. Збережено XSS також іноді називають постійним або XSS типу I.

Відображені XSS-атаки

Відображені атаки - це атаки, при яких впроваджений скрипт відбивається від web-сервера, наприклад, в повідомленні про помилку, в результаті пошуку або в будь-якому іншому відповіді, який включає деякі або всі вхідні дані, відправлені на сервер як частина запиту. Відображені атаки доставляються жертвам іншим шляхом,

наприклад, в електронному листі або на іншому веб-сайті. Коли користувача обманом змушують клацнути шкідливе посилання, відправити спеціально створену форму або навіть просто перейти на шкідливий сайт, впроваджений код переходить на вразливий веб-сайт, що відображає атаку назад в браузер користувача. Потім браузер виконує код, тому що він прийшов з «довіреної» сервера. Відбитий XSS також іноді називають непостійним або XSS типу II.

Інші типи XSS-вразливостей

Крім збереженого і відбитого XSS, ще один тип XSS, XSS на основі DOM, був визначений Амітом Кляйном в 2005 році. OWASP рекомендує категоризацію XSS, як описано в статті OWASP: Типи міжсайтових сценаріїв, яка охоплює всі ці терміни XSS, організувавши їх в матрицю збережених і відображених XSS і серверних і клієнтських XSS, де XSS на основі DOM є підмножиною клієнтських XSS.

Clickjacking

Clickjacking, також відомий як «атака відновлення призначеного для користувача інтерфейсу», - це коли зловмисник використовує кілька прозорих або непрозорих шарів, щоб обманом змусити користувача натиснути кнопку або посилання на іншій сторінці, коли він мав намір клацнути по сторінці верхнього рівня. Таким чином, зловмисник «перехоплює» кліки, призначені для його сторінки, і перенаправляє їх на іншу сторінку, яка, швидше за все, належить іншому додатку, домену або обом.

За допомогою аналогічної техніки можна також перехопити натискання клавіш. За допомогою ретельно продуманої комбінації таблиць стилів, вікон iframe і текстових полів можна переконати користувача, що він вводить пароль для своєї електронної пошти або банківського рахунку, але замість цього вводить його в невидимий фрейм, контрольований зловмисником.

Наприклад, уявіть зловмисника, який створює веб-сайт, на якому є кнопка з написом «клацніть тут, щоб отримати безкоштовний iPhone». Однак у верхній

частині цієї веб-сторінки зловмисник завантажив iframe з вашим обліковим записом електронної пошти і збудував точно кнопку «видалити всі повідомлення» прямо над кнопкою «безкоштовний iPhone». Жертва намагається натиснути кнопку «безкоштовно iPhone», але замість цього фактично натискає невидиму кнопку «видалити всі повідомлення». По суті, зловмисник «перехопив» клік користувача, звідси і назва «clickjacking».

Одним з найгучніших прикладів clickjacking була атака на сторінку налаштувань плагіну Adobe Flash. Завантаживши цю сторінку в невидимий iframe, зловмисник може обманом змусити користувача змінити налаштування безпеки Flash, дозволивши будь Flash-анімації використовувати мікрофон і камеру комп'ютера.

Clickjacking також потрапив в новини у вигляді хробака Twitter. Ця атака clickjacking переконала користувачів натиснути на кнопку, яка змусила їх повторно твітнути розташування шкідливої сторінки і широко поширилася.

Також мали місце атаки clickjacking, в яких використовувалася функція «подобається» Facebook. Зловмисники можуть обдурити зареєстрованих користувачів Facebook, щоб вони довільно ставили лайки фан-сторінок, посилань, групам і т.д.

Є два основних способи запобігти clickjacking:

1. Відправлення правильних заголовків відповіді директиви frame-ancestors політики безпеки контенту (CSP), які інструктують браузер не дозволяти кадрування з інших доменів. (Це замінює старі заголовки HTTP X-Frame-Options.)
2. Використання захисного коду в інтерфейсі, щоб гарантувати, що поточний фрейм є вікном самого верхнього рівня

Включення файлів

Уразливість включення файлів дозволяє зловмисникові включити файл, зазвичай використовуючи механізми «динамічного включення файлів», реалізовані в цільовому додатку. Уразливість виникає через використання вводяться

користувачем даних без належної перевірки. Це може привести до висновку вмісту файлу, але в залежності від ступеня серйозності це також може привести до:

- Виконання коду на web-сервері
- Виконання коду на стороні клієнта, наприклад JavaScript, що може привести до інших атакам, таким як міжсайтовий скриптинг (XSS)
- Відмова в обслуговуванні (DoS)
- Розкриття конфіденційної інформації

Включення локального файлу

Включення локальних файлів (також відоме як LFI) - це процес включення файлів, які вже локально присутні на сервері, шляхом використання вразливих процедур включення, реалізованих в додатку. Ця вразливість виникає, наприклад, коли сторінка отримує в якості вхідних даних шлях до файлу, який повинен бути включений, і ці вхідні дані не очищаються належним чином, що дозволяє вводити символи обходу каталогу (наприклад, точка-точка-коса риска). Хоча більшість прикладів вказують на вразливі сценарії PHP, ми повинні пам'ятати, що це також характерно для інших технологій, таких як JSP, ASP і інших.

Включення віддаленого файлу

Включення віддалених файлів (також відоме як RFI) - це процес включення віддалених файлів за допомогою використання процедур уразливого включення, реалізованих в додатку. Ця вразливість виникає, наприклад, коли сторінка отримує в якості вхідних даних шлях до файлу, який повинен бути включений, і ці вхідні дані не обробляються належним чином, що дозволяє впровадити зовнішній URL-адресу. Хоча більшість прикладів вказують на вразливі сценарії PHP, ми повинні пам'ятати, що це також характерно для інших технологій, таких як JSP, ASP і інших.

Найбільш ефективне рішення для усунення вразливостей, пов'язаних з включенням файлів, - уникати передачі введених користувачем даних в будь-яку файлову систему/API фреймворку. Якщо це неможливо, додаток може вести

дозволений список файлів, які можуть бути включені на сторінку, а потім використовувати ідентифікатор (наприклад, номер індексу) для доступу до вибраного файлу. Будь-який запит, що містить неприпустимий ідентифікатор, повинен бути відхилений, таким чином, зловмисники не можуть атакувати шлях.

Ін'єкція команд

Ін'єкція команд - це атака, при якій метою є виконання довільних команд на операційній системі хоста через вразливий додаток. Атаки введення команд можливі, коли програма передає небезпечні дані, надані користувачем (форми, файли cookie, заголовки HTTP тощо) в системну оболонку. У цій атаці команди операційної системи, що надаються зловмисником, зазвичай виконуються з привілеями вразливої програми. Атаки введення команд можливі здебільшого через недостатню перевірку вводу.

Висновки до другого розділу

В цьому розділі було розглянуто загальні загрози для web-серверів. Також важливо розуміти що існує також багато інших загроз, але ризики їх наявності на експлуатації значно нижче.

Розділ 3. Розробка рекомендацій щодо налаштування web-серверу відповідно загрозам безпеки

3.1 Рекомендації налаштування для Nginx

Після установки Nginx параметри конфігурації знаходяться в файлі nginx.conf. Це основний файл конфігурації для Nginx, тому більшість перевірок безпеки буде виконуватися з використанням цього файлу. За замовчуванням ви можете знайти nginx.conf в /conf в системах Windows і в /etc/nginx або /usr/local/etc/nginx в системах Linux. Вам також може знадобитися внести деякі зміни в файли конфігурації віртуального хоста, зазвичай містяться в підкаталозі sites-available.

Відключення всіх небажаних модулів

Коли встановлюється Nginx , він автоматично включає безліч модулів. В даний час ви не можете вибирати модулі під час виконання. Щоб відключити певні модулі, вам необхідно перекомпілювати Nginx . Ми рекомендуємо вам відключити будь-які модулі, які не потрібні, оскільки це мінімізує ризик потенційних атак за рахунок обмеження дозволених операцій.

Для цього використовуйте параметр configure під час установки. У наведеному нижче прикладі ми відключаємо модуль autoindex, який генерує автоматичні списки каталогів, а потім перекомпілюя Nginx .

```
$ ./configure --without-http_autoindex_module
$ make
$ make install
```

Відключення server_tokens

За замовчуванням директива server_tokens в Nginx відображає номер версії Nginx . Він безпосередньо відображається на всіх розсилаються сторінках помилок, але також присутня у всіх HTTP-відповідях в заголовку сервера.

Це може привести до розкриття інформації - неавторизований користувач може отримати інформацію про версії Nginx , яку ви використовуєте. Ви повинні відключити директиву server_tokens в файлі конфігурації Nginx , відключивши server_tokens. У блоці конфігурації ***server*** необхідно додати наступний рядок:

```
server_tokens off;
```

Контроль ресурсів і лімітів

Щоб запобігти потенційним DoS-атаки на Nginx , ви можете встановити обмеження на розмір буфера для всіх клієнтів. Ви можете зробити це в файлі конфігурації Nginx за допомогою наступних директив:

1. client_body_buffer_size - використовуйте цю директиву, щоб вказати розмір буфера тіла запиту клієнта. Значення за замовчуванням - 8k або 16k, але рекомендується встановити його як мінімум 1k: client_body_buffer_size 1k.
2. client_header_buffer_size - використовуйте цю директиву, щоб вказати розмір буфера заголовка для заголовка запиту клієнта. Розмір буфера 1 Кбайт підходить для більшості запитів.
3. client_max_body_size - використовуйте цю директиву, щоб вказати максимально допустимий розмір тіла для клієнтського запиту. Директиви 1k повинно бути достатньо, але вам необхідно збільшити її, якщо ви отримуєте файли, завантажені через метод POST.
4. large_client_header_buffers - використовуйте цю директиву, щоб вказати максимальну кількість і розмір буферів, які будуть використовуватися для читання великих заголовків клієнтських запитів. Директива large_client_header_buffers 2 1k встановлює максимальну кількість буферів

рівним 2, кожен з максимальним розміром 1k. Ця директива приймає URI даних розміром 2 КБ.

Відключення небажаних методів HTTP

Ми пропонуємо вам відключити будь-які методи HTTP, які не використовуватимуться і які не потрібно реалізовувати на web-сервері. Якщо додати таку умову в блок розташування файлу конфігурації віртуального хоста Nginx , сервер буде вирішувати тільки методи GET, HEAD і POST і буде фільтрувати такі методи, як DELETE і TRACE.

```
location / {
        limit_except GET HEAD POST { deny all; }
}
```

Інший підхід - додати таку умову в секцію сервера (або блок сервера). Його можна вважати більш універсальним, але ви повинні бути обережні з операторами if в контексті розташування.

```
if ($request_method !~ ^(GET|HEAD|POST)$ ) {
        return 415;
}
```

Встановлення ModSecurity

ModSecurity - це модуль з відкритим вихідним кодом, який працює як брандмауер веб-додатків. Його функції включають фільтрацію, маскування ідентифікатора сервера і запобігання атак з нульовим байтом. Модуль також дозволяє виконувати моніторинг трафіку в реальному часі.

Після установки необхідних пакетів попередніх умов наступним кроком буде компіляція ModSecurity як динамічного модуля NGINX. У новій модульній архітектурі ModSecurity 3.0 libmodsecurity є основним компонентом, який включає в себе всі правила і функції. Другий головний компонент в архітектурі - це коннектор, який пов'язує libmodsecurity з web-сервером, з яким він працює. Існують окремі з'єднувачі для NGINX, Apache HTTP Server і IIS. Ми розглянемо коннектор NGINX в наступному розділі.

Нам потрібні такі пакети для компіляції вихідного коду ModSecurity і Nginx Connector в Ubuntu:

```
$ sudo apt install -y apt-utils autoconf automake build-essential git libcurl4-openssl-
dev libgeoip-dev liblmdb-dev libpcre++-dev libtool libxml2-dev libyajl-dev pkgconf
wget zlib1g-dev
```

Завантажте та скомпілюйте ModSecurity або libmodsecurity. Клонуйте репозиторій ModSecurity на github:

```
$ git clone --depth 1 -b v3/master --single-branch https://github.com/SpiderLabs/ModSecurity
```

Перейдіть в каталог і скомпілюйте вихідний код:

```
$ cd ModSecurity
$ git submodule init
$ git submodule update
$ ./build.sh
$ ./configure
$ make
$ sudo make install
```

Остання команда make install скопіює файли ModSecurity в /usr/local/modsecurity/. Завантажте та скомпілюйте коннектор ModSecurity для Nginx як динамічний модуль для Nginx . Клонуйте репозиторій коннектора ModSecurity:

```
$ git clone --depth 1 https://github.com/SpiderLabs/ModSecurity-nginx.git
$ wget http://nginx.org/download/nginx-1.19.4.tar.gz
$ tar zxvf nginx-1.19.4.tar.gz
```

Скомпілюйте модуль і скопіюйте objs/ngx_http_modsecurity_module.so в /etc/Nginx /modules:

```
$ cd nginx-1.19.4
$ ./configure --with-compat --add-dynamic-module=../ModSecurity-nginx
$ make modules
$ sudo cp objs/ngx_http_modsecurity_module.so /etc/nginx/modules/
```

Відредагуйте /etc/Nginx /Nginx .conf і завантажте ngx_http_modsecurity_module.so в контексті верхнього рівня:

```
$ sudo vim /etc/nginx/nginx.conf

user  www-data;
worker_processes  auto;
pid       /run/nginx.pid;

## Nginx ModSecurity Connector
load_module modules/ngx_http_modsecurity_module.so;

events {
   worker_connections  1024;
}

http {
   include /etc/nginx/conf.d/*.conf;
}
```

Створіть нову папку /etc/Nginx /modsec і помістіть сюди всі файли конфігурації ModSecurity:

```
$ sudo mkdir /etc/nginx/modsec
```

Завантажте рекомендований файл конфігурації ModSecurity і перейменуйте його в modsecurity.conf

```
$ cd /etc/nginx/modsec
$ sudo wget
https://raw.githubusercontent.com/SpiderLabs/ModSecurity/v3/master/modsecurity.conf-recommended
$ sudo mv modsecurity.conf-recommended modsecurity.conf
```

Відредагуйте modsecurity.conf і змініть значення за замовчуванням для SecRuleEngine DetectionOnly на SecRuleEngine On.

```
$ sudo vim modsecurity.conf
# SecRuleEngine DetectionOnly
SecRuleEngine On
```

Створіть новий файл конфігурації ModSecurity /etc/Nginx /modsec/main.conf:

```
$ sudo vim /etc/nginx/modsec/main.conf
Include /etc/nginx/modsec/modsecurity.conf
```

У головному файлі конфігурації Nginx, контексті сервера, включите modsecurity для конкретного сервера:

```
$ sudo vim /etc/nginx/nginx.conf
server {
   # ...
   modsecurity on;
   modsecurity_rules_file /etc/nginx/modsec/main.conf;
}
```

Скопіюйте unicode.mapping в /etc/Nginx /modsec/unicode.mapping, він знаходиться всередині вихідного коду ModSecurity:

```
$ sudo cp ModSecurity/unicode.mapping /etc/nginx/modsec/
```

Правила ModSecurity зараз порожні, і нам потрібно хоча б включити набір основних правил OWASP (CRS) для захисту деяких загальних атак (SQLi, XSS, LFI і т. Д.), ботів і сканерів. Завантажте основний набір правил OWASP:

```
$ wget https://github.com/coreruleset/coreruleset/archive/v3.2.0.tar.gz
$ tar zxvf v3.2.0.tar.gz
```

Перемістіть папку coreruleset-3.2.0 OWASP CRS в /usr/local:

```
$ sudo mv coreruleset-3.2.0 /usr/local
```

Право власності змінюється після переміщення; краще змінити власника на root:

```
$ sudo chown root:root -R /usr/local/coreruleset-3.2.0
```

Створіть новий crs-setup.conf як копію crs-setup.conf.example:

```
$ cd /usr/local/coreruleset-3.2.0
$ sudo cp crs-setup.conf.example crs-setup.conf
```

Увімкніть crs-setup.conf і правило / * в основний файл конфігурації ModSecurity /etc/Nginx /modsec/main.conf:

```
$ sudo vim /etc/nginx/modsec/main.conf
# Include the recommended configuration
Include /etc/nginx/modsec/modsecurity.conf

# OWASP CRS v3.2.0 rules
Include /usr/local/coreruleset-3.2.0/crs-setup.conf

# This will include all the rules, need filter later
Include /usr/local/coreruleset-3.2.0/rules/*.conf
```

Переконайтеся, що все гаразд, і перезавантажте Nginx. Тепер правила OWASP включені в ModSecurity:

```
$ sudo service nginx configtest
$ sudo service nginx reload
```

Налаштування журналів доступу і помилок

Журнали доступу і помилок Nginx включені за замовчуванням і знаходяться в logs/error.log і logs/access.log відповідно. Якщо ви хочете змінити розташування, ви можете використовувати директиву error_log в файлі конфігурації Nginx . Ви також можете використовувати цю директиву, щоб вказати журнали, які будуть записуватися відповідно до їх рівнем серйозності. Наприклад, рівень критичності змусить Nginx реєструвати критичні проблеми і всі проблеми, які мають більш високий рівень важливості, ніж критичний. Щоб встановити рівень критичності, встановіть директиву error_log наступним чином:

```
error_log logs/error.log crit;
```

Ви також можете змінити директиву access_log в файлі конфігурації Nginx , щоб вказати нестандартне розташування для журналів доступу. Нарешті, ви можете використовувати директиву log_format для настройки формату реєстрованих повідомлень, як описано в документації Nginx .

Відстеження журналів доступу і помилок

Якщо ви постійно відстежуєте і керуєте файлами журналу Nginx , ви зможете краще розуміти запити, надіслані на ваш web-сервер, а також помічати будь-які виявлені помилки. Це допоможе вам виявити будь-які спроби атак, а також визначити, що ви можете зробити для оптимізації продуктивності сервера. Ви можете використовувати інструменти управління журналами, такі як logrotate, для повороту і стиснення старих журналів і звільнення місця на диску. Також модуль ngx_http_stub_status_module надає доступ до базової інформації про статус.

Включення заголовків безпеки

Щоб додатково зміцнити свій web-сервер Nginx , ви можете додати кілька різних заголовків HTTP. Ось кілька варіантів, які ми рекомендуємо.

X-Frame-Options

Заголовок HTTP-відповіді X-Frame-Options використовується, щоб вказати, чи слід дозволити браузеру відображати сторінку в <frame> або <iframe>. Це може запобігти атакам типу "clickjacking". Тому ми рекомендуємо вам включити цю опцію для вашого сервера Nginx . Параметри налаштувань даного заголовка:

- SAMEORIGIN - Frame / iframe контенту дозволений тільки з одного і того ж сайту.
- DENY - забороняє будь-якому домену вбудовувати ваш контент за допомогою frame/iframe.
- ALLOW-FROM - дозволяє контент тільки за певним URI.

Заголовок X-Frame-Options включається в файлі конфігурації Nginx в розділі server:

```
add_header X-Frame-Options "SAMEORIGIN";
```

Strict-Transport-Security

HTTP Strict Transport Security (HSTS) - це метод, який використовується веб-сайтами для оголошення того, що до них слід звертатися тільки через безпечне з'єднання (HTTPS). Якщо веб-сайт декларує політику HSTS, браузер повинен відхиляти всі HTTP-з'єднання і забороняти користувачам приймати небезпечні сертифікати SSL. Щоб додати заголовок HSTS на ваш сервер Nginx , ви можете додати наступну директиву в розділ вашого сервера:

```
add_header Strict-Transport-Security "max-age=31536000; includeSubdomains;
preload";
```

Content Security Policy

Запобігайте атаки XSS, clickjacking і впровадження коду, реалізувавши заголовок Content Security Policy (CSP) в HTTP-відповіді вашої веб-сторінки. CSP вказує браузеру завантажити дозволений контент для завантаження на веб-сайт.

Всі браузери не підтримують CSP, тому вам необхідно перевірити його перед впровадженням. Є три способи отримати заголовки CSP:

1. Content-Security-Policy - Level 2 / 1.0
2. X-Content-Security-Policy - Не рекомендується
3. X-Webkit-CSP - Не рекомендується

Існує кілька параметрів для реалізації CSP, і ви можете звернутися до OWASP за ідеєю. Однак давайте розглянемо два найбільш часто використовуваних параметра.

- default-src - завантажувати все з певного джерела
- script-src - завантажувати тільки скрипти з певного джерела

Додайте наступне в блок server в файлі Nginx .conf:

```
add_header Content-Security-Policy "default-src 'self';";
```

X-XSS-Protection

Тема HTTP X-XSS-Protection підтримується IE і Safari і не потрібно для сучасних браузерів, якщо у вас є сувора політика безпеки контенту. Однак, щоб допомогти запобігти XSS в разі старих браузерів (які ще не підтримують CSP), ви можете додати заголовок X-XSS Protection в розділ свого сервера:

```
add_header X-XSS-Protection "1; mode=block";
```

HTTP Strict Transport Security

Тема HSTS (HTTP Strict Transport Security), щоб гарантувати, що всі повідомлення з браузера відправляються через HTTPS (HTTP Secure). Це запобігає перехід по HTTPS-запитам і перенаправляє HTTP-запити на HTTPS.

Перед реалізацією цього заголовка ви повинні переконатися, що всі сторінки вашого веб-сайту доступні по HTTPS, інакше вони будуть заблоковані. Доступні для налаштування параметрів:

- max-age - тривалість (в секундах), щоб повідомити браузеру, що запити доступні тільки через HTTPS.
- includeSubDomains - конфігурація дійсна і для поддомена.
- preload – використовується для того, щоб ваш домен був включений в список попереднього завантаження HSTS

Щоб налаштувати HSTS в Nginx, додайте наступний запис в Nginx .conf в директиві сервера:

```
add_header Strict-Transport-Security 'max-age=31536000; includeSubDomains;
preload';
```

X-Content-Type-Options

Щоб запобігти загрозі безпеки типів MIME, додайте цей заголовок у відповідь HTTP своєї веб-сторінки. Наявність цього заголовка вказує браузеру розглядати типи файлів як певні і забороняти аналіз вмісту. Вам потрібно додати тільки один параметр - «nosniff»:

```
add_header X-Content-Type-Options nosniff;
```

HTTP Public Key Pinning

Зведіть до мінімуму ризик атак «зловмисник посередині» (MITM), закріпивши сертифікат. Це можливо з заголовком HPKP (закріплення відкритого ключа HTTP).

Ви можете закріпити відкритий ключ кореневого сертифіката або негайний сертифікат. На момент написання HPKP в даний час працює в Firefox і Chrome і підтримує алгоритм хешування SHA-256.

Є чотири можливих конфігурації параметрів:

- report-uri = "url" - Повідомте за вказаною URL-адресою, якщо перевірка пінінга не вдалася. Це параметр необов'язковий.
- pin-sha256 = "sha256key" - тут необхідно вказувати пінінгі
- max-age = - Браузер запам'ятовує час в секундах, що сайт доступний тільки за допомогою одного з запінених сертифікатів.
- IncludeSubDomains - чи можна застосувати до поддомену.

Давайте подивимося на приклад заголовка HPKP з facebook.com:

```
public-key-pins-report-only:max-age=500; pin-
sha256="WoiWRyIOVNa9ihaBciRSC7XHjliYS9VwUGOIud4PB18="; pin-
sha256="r/mIkG3eEpVdm+u/ko/cwxzOMo1bk4TyHIlByibiA5E="; pin-
sha256="q4PO2G2cbkZhZ82+JgmRUyGMoAeozA+BSXVXQWB8XWQ="; report-
uri=http://reports.fb.com/hpkp/
```

X-Permitted-Cross-Domain-Policies

Якщо використовуються продукти Adobe, такі як PDF, Flash і т.д., то є можливість використання цього заголовоку, щоб вказати браузеру, як обробляти запити через междоменного доступ. Реалізуючи цей заголовок, ви обмежуєте завантаження ресурсів вашого сайту з інших доменів, щоб уникнути зловживання ресурсами. Доступні параметри:

- none - ніяка політика не допускається
- master-only - дозволити тільки головну політику
- all - дозволяє все
- by-content-only - Дозволити тільки певний тип контенту. Приклад - XML
- by-ftp-only - applicable only for an FTP server

Якщо, наприклад, необхідно реалізувати master-only, необхідно додати наступне в Nginx .conf в блоці server:

```
add_header X-Permitted-Cross-Domain-Policies master-only;
```

Referrer-Policy

Є певні переваги конфіденційності та безпеки. Однак не всі параметри підтримуються всіма браузерами, тому перед впровадженням перевірте свої вимоги.

Реферрер-політика підтримує наступний синтаксис:

- no-referrer - Інформація реферера не буде передана із запитом
- no-referrer-when-downgrade - налаштування за замовчуванням, при якому реферер відправляється по тим же протоколам, наприклад HTTP на HTTP, HTTPS на HTTPS.
- unsafe-url - повний URL буде відправлений із запитом.
- same-origin - реферер буде відправлений тільки для сайту того ж походження.
- strict-origin - відправляти тільки якщо протокол HTTPS
- strict-origin-when-cross-origin - повний URL-адресу буде відправлений по суворим протоколом, наприклад HTTPS
- origin - відправляйте вихідний URL у всіх запитах

- origin-when-cross-origin - відправити повний URL з того ж джерела. Однак в інших випадках відправляйте тільки вихідний URL.

Припустимо, потрібно реалізувати передачу реферера на той же домен, то вам потрібно додати наступне:

```
add_header Referrer-Policy same-origin;
```

Expect-CT

Новий заголовок, все ще знаходиться в експериментальному статусі, призначений для вказівки браузеру перевірити з'єднання з web-серверами для забезпечення прозорості сертифіката (CT). Цей проект Google спрямований на виправлення деяких недоліків в системі сертифікатів SSL / TLS.

Наступні три змінні доступні для заголовка Expect-CT:

- max-age - в секундах - як довго браузер повинен кешувати політику.
- enforce - необов'язкова директива для забезпечення дотримання політики.
- report-uri - браузер для відправки звіту за вказаною URL-адресою, якщо дійсний сертифікат прозорості не отримана.

Приклад правила для заголовка Expect-CT:

```
add_header Expect-CT 'max-age=60, report-uri="https://mydomain.com/report"';
```

Feature-Policy

Управління функціями браузера, такими як геолокація, повноекранний режим, динамік, USB, Автовідтворення, динамік, вібрація, мікрофон, оплата, віртуальна реальність і т.д., Щоб увімкнути або вимкнути їх в веб-додатку.

Дана конфігурація дозволяє відключати функцію вібрації в мобільних браузерах:

```
add_header Feature-Policy "vibrate 'none';";
```

Або відключити геолокацію, камеру і динамік:

```
add_header Feature-Policy "geolocation 'none'; camera 'none'; speaker 'none';";
```

Налаштування SSL і комплектів шифрів

Конфігурація Nginx за замовчуванням дозволяє використовувати небезпечні старі версії протоколу TLS (згідно офіційної документації: ssl_protocols TLSv1 TLSv1.1 TLSv1.2). Це може привести до атак, таким як атака BEAST. Тому ми рекомендуємо вам не використовувати старі протоколи TLS і змінити свою конфігурацію для підтримки тільки новіших, безпечних версій TLS. Для цього додайте наступну директиву в розділ server файлу конфігурації Nginx :

```
ssl_protocols TLSv1.2 TLSv1.3;
```

Крім того, слід вказати комплекти шифрів, щоб переконатися, що не підтримуються вразливі комплекти. Щоб вибрати кращі набори шифрів, прочитайте нашу статтю про посилення захисту шифрів TLS і додайте директиву ssl_ciphers в розділ сервера для вибору шифрів (як це пропонується в статті про посилення захисту шифрів). Також рекомендується додати наступну директиву в розділ сервера:

```
ssl_prefer_server_ciphers on;
```

Ця директива дозволить приймати рішення про те, які шифри використовувати на стороні сервера, а не на стороні клієнта.

Регулярне оновлення сервера

Як і у випадку з будь-яким іншим програмним забезпеченням, ми рекомендуємо вам завжди оновлювати сервер Nginx до останньої стабільної версії. Нові оновлення часто містять виправлення вразливостей, виявлених в попередніх версіях, таких як вразливість обходу каталогів (CVE-2009-3898), яка існувала в версіях Nginx до 0.7.63 і 0.8.x до 0.8.17. Оновлення також часто включають нові функції безпеки і поліпшення. На сайті iginx.org ви можете знайти поради з безпеки в спеціальному розділі та новини про останні оновлення на головній сторінці.

3.2 Тестування розроблених рекомендацій

Тестування дозволить перевірити вищеописані рекомендації та оцінити їх ефективність. Тестування даної конфігурації буде відбуватися в кілька етапів:

1. Установка nginx і настройка тестового сайту.
2. Сканування тестового сайту і nginx «з коробки»
3. Налаштування nginx відповідно до описаних вище рекомендацій
4. Сканування тестового сайту і nginx з «безпечною» конфігурацією

В якості тестового сайту була вибрана збірка DVWA (Damn Vulnerable Web Application) для nginx. Даний проект використовується для фахівців з безпеки для перевірки своїх навичок та інструментів в правовому середовищі, а так само для допомоги веб-розробникам краще зрозуміти процеси захисту веб-додатків. Даний проект містить в собі велику кількість типових вразливостей, велика частина з яких була розглянута у другому розділі.

Для для сканування тестового сайту буде використаний популярний безкоштовний сканер вразливостей Nikto.

Nikto - це сканер веб-серверів з відкритим вихідним кодом, який виконує комплексні тести на веб-серверах для декількох елементів, включаючи понад 6700 потенційно небезпечних файлів. Це один з найбільш широко використовуваних інструментів сканування веб-сайтів на уразливості в усій галузі, а в багатьох колах він вважається галузевим стандартом.

Сканування проводилось без додаткових налаштувань сканеру, було вказано тільки безпосередньо ціль для сканування. Для web-серверу без додаткових налаштувань безпеки були отримані наступні результати:

```
- Nikto v2.1.6
---------------------------------------------------------------------
+ Target IP:          example.site
+ Target Hostname:    example.site
+ Target Port:        80
+ Start Time:         2020-11-05 17:04:40 (GMT0)
---------------------------------------------------------------------
+ Server: nginx/1.19.4
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-Content-Type-Options header is not set. This could allow
the user agent to render the content of the site in a different
fashion to the MIME type.
+ No CGI Directories found (use '-C all' to force check all
possible dirs)
```

```
+ Web Server returns a valid response with junk HTTP methods,
this may cause false positives.
+ ///etc/hosts: The server install allows reading of any system
file by adding an extra '/' to the URL.
+ /phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-3268: /config/: Directory indexing found.
+ /config/: Configuration information may be available remotely.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP
reveals potentially sensitive information via certain HTTP
requests that contain specific QUERY strings.
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script
which runs phpinfo() was found. This gives a lot of system
information.
+ OSVDB-3268: /docs/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ /.gitignore: .gitignore file found. It is possible to grasp the
directory structure.
+ /wp-
content/themes/twentyeleven/images/headers/server.php?filesrc=/et
c/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-
content/themes/twentyeleven/images/headers/server.php?filesrc=/et
c/hosts: A PHP backdoor file manager was found.
+ /wp-includes/Requests/Utility/content-
post.php?filesrc=/etc/hosts: A PHP backdoor file manager was
found.
+ /wordpress/wp-includes/Requests/Utility/content-
post.php?filesrc=/etc/hosts: A PHP backdoor file manager was
found.
+ /wp-
includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A
PHP backdoor file manager was found.
+ /wordpress/wp-
includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A
PHP backdoor file manager was found.
+ /assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file
manager was found.
+ /login.cgi?cli=aa%20aa%27cat%20/etc/hosts: Some D-Link router
remote command execution.
+ /shell?cat+/etc/hosts: A backdoor was identified.
+ 8114 requests: 4 error(s) and 22 item(s) reported on remote
host
+ End Time:           2020-11-05 17:19:52 (GMT0) (912 seconds)
---------------------------------------------------------------
+ 1 host(s) tested
```

Як можна бачити з результатів роботи програми, с тестовому додатку було виявлено велику кількість вразливостей, серед яких є навіть запуск будь-яких команд в ОС серверу.

Тепер зробимо налаштування Nginx відповідно рекомендаціям з попереднього розділу. Результати сканування з новою «безпечною» конфігурацією:

```
- Nikto v2.1.6
---------------------------------------------------------------------
+ Target IP:          example.site
+ Target Hostname:    example.site
+ Target Port:        443
+ Start Time:         2020-11-05 20:44:27 (GMT0)
---------------------------------------------------------------------
+ Server: nginx
+ No CGI Directories found (use '-C all' to force check all
possible dirs)
+ 8113 requests: 4 error(s) and 1 item(s) reported on remote host
+ End Time:           2020-11-05 21:14:39 (GMT0) (1812 seconds)
---------------------------------------------------------------------
+ 1 host(s) tested
```

Як можна бачити в результатів сканування web-серверу, всі вразливості знайдені сканером у попередній конфігурації були успішно усунені.

Висновки до третього розділу

В третьому розділі були розроблені рекомендації налаштування web-серверу Nginx та протестовані на наявність вразливостей з базовими налаштуваннями та налаштуваннями розробленими відповідно до загроз.

Висновки

В результаті виконання дипломної роботи було розроблено рекомендацій щодо налаштування web-серверу для захисту інформації від загроз безпеки. Досліджено технічні можливості web-серверів та загрози безпеки для інформації. Для кожної із виділених загроз розроблена конфігурація для web-серверу Nginx. Ефективність розроблених рекомендацій доведена на практиці за допомогою сканера вразливостей Nikto. Матеріали даного дипломного проекту рекомендується використовувати при налаштуванні будь-якого веб-сайту.

Список використаних джерел

1. September 2020 Web Server Survey [Електронний ресурс]. – 2020 - Режим доступу: https://news.netcraft.com/archives/2020/09/23/september-2020-web-server-survey.html – Title from the screen.
2. R. Bowen. Apache Cookbook: Solutions and Examples for Apache Administrators.– 2008. – 310.
3. D. DeJonghe NGINX Cookbook. – 2019 – 175.
4. Website Hacking Statistics in 2020 [Електронний ресурс]. – 2020 - Режим доступу: https://www.webarxsecurity.com/website-hacking-statistics-2018-february/ – Title from the screen.
5. NGINX ModSecurity WAF [Електронний ресурс]. – 2020 - Режим доступу: https://docs.nginx.com/nginx-waf/ – Title from the screen.
6. Node.js v13.14.0 Documentation [Електронний ресурс]. – 2020 - Режим доступу: https://nodejs.org/docs/latest-v13.x/api/ Title from the screen.