ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТАЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технологія розробки системи перевірки захищеності Web-служб»

зі спеціальності *125 Кібербезпека*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Марія НАСОНОВА
(підпис)

Виконав: здобувач(ка) вищої освіти групи БСДМ-62
НАСОНОВА Марія
(прізвище, ім'я)

Керівник д.т.н.,професор, КАЗМІРЧУК Світлана
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2024

ЗМІСТ

ВСТУП

З давніх часів одним із найціннішим ресурсом була інформація. З розвитком людини та світу інформація набула більшої цінності. Разом із розвитком інформаційних технологій, розвивається сфера захисту інформації. Адже інформація, в наш час, стала дуже важливим ресурсом, і в будь-яких сферах діяльності постає питання її належного захисту. В даний час обмін та зберігання інформації виходить на цифровий рівень.

Обробка інформації сьогодні переважно здійснюється на ПЕОМ, повсякденна діяльність все більше пов'язує ПЕОМ та мережеві технології. Прикладом такої взаємодії є WEB-служби. WEB-служби дозволяють зберігати та представляти інформацію користувачам у сприятливому для них вигляді. Простежується неуклінне збільшення їх популярності серед різних установ, та проектів. [1]

Зловмисники мають множину способів проводити атаки на WEB-служби та ПЕОМ користувачів з метою отримання необхідних їм інформаційних ресурсів. На сьогоднішній день кількість таких атак стає все більшою. Виконуються атаки на WEB-сервіси великих компаній та проектів, таких, як SONY, eBay та інші. [1]

З поступовим запровадженням мережевих технологій Інтернет в урядових закладах, постає питання забезпечення захисту інформації. В Україні існує ряд нормативних документів регулюючих такі питання:

- Закон України «Про інформацію» від 02.10.1992 № 2657-XII
- Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» від 05.07.1994 № 80/94-ВР
- Закон України «Про державну таємницю» від 21.01.1994 № 3855-XII

У даній роботі проводиться аналіз безпеки протоколів взаємодії WEB-служб та розробка системи аналізу захищеності WEB-служб. Дана система охоплює систему знань про WEB-служби та засіб аналізу їх захищеності. [1] [2] [3] [10]

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ARPA – Advanced Research Project Agency;
ПЕОМ – Персональна Електронна Обчислювальна Машина;
TCP – Transmission Control Protocol;
IP – Internet Protocol;
RPC – Remote Procedure Call;
WWW –World Wide Web;
XML – Extensible Markup Language;
SOA – Service-Oriented Architecture;
SOAP – Simple Object Access Protocol;
API – Application Programming Interface;
UDDI – Universal Description Discovery & Integration;
WSDL – WEB Services Description Language;

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО WEB СЛУЖБИ

1.1. Введення у поняття WEB-служб

Інтернет був створений в 1969 році. Під керівництвом U.S. Department of Defenses Advanced Research Project Agency (ARPA), з задуму на папері Інтернет перероіс у маленьку мережу (ARPANET), яка була призначена для обміну інформацією між ПЕОМ дослідників у Сполучених Штатах. ARPANET одразу привернув увагу спільноти, та набув нечуваного успіху. Кількість з'єднаних користувачів (хостів), а саме Інститутів та урядових дослідних центрів поступово збільшувалась, в результаті чого була створена комерційна версія ARPANET. У 1977 році Бобом Каном та Вінтом Серфом був створений загальний стек протоколів TCP/IP для взаємодії ПЕОМ в Інтернет. ARPANET з того часу еволюціонував у Інтернет, що став таким, яким ми знаємо його сьогодні.

В Україні, все почалося у 1990 році, але національний домен верхнього рівня.ЗUA було зареєстровано 1 грудня 1992 року. Поступово кількість користувачі та провайдерів зростала. Після 2006 року Інтернет почав стрімко розвиватися та розповсюджуватися. На сьогоднішній день більша половина населення України користується послугами Інтернет.

Незважаючи на всі проблеми, недоліки, існуючі обмеження, мережа Інтернет все більше використовується в державних установах. Використання даних технологій передбачає собою передачу інформації та взаємодію ПЕОМ. Для урядових установ необхідно забезпечити належні умови передачі інформації. У кожній установі існує орган що регламентує умови передачі інформації, в свою чергу усі вони підпорядковуються та співпрацюють зі Службою безпеки України. [3] [10]

Проблеми інформатизації органів державної влади, державних організацій та підприємств були викладені в Указі Президента України № 663 від 17 червня 1997 року та Національній програмі інформатизації України. В Україні також були запроваджені такі нормативні документи: Закон України "Про Основні засади розвитку інформаційного суспільства в Україні" та Розпорядження Кабінету Міністрів "Про схвалення Концепції розвитку телекомунікацій в Україні". Ці документи сприяли швидшому розвитку мережі Інтернет в Україні.

Паралельно із технологією Інтернет, розвивалась не менш важлива технологія World Wide Web.

World Wide Web (WWW) являє собою клієнт-серверну систему надання інформації. Простіше кажучи, WEB – це засіб отримання інформації в мережі Інтернет. WEB працює за допомогою протоколу передачі даних HTTP, котрий визначає спосіб форматування та відправки повідомлень. Протокол HTTP регламентує дії WEB-браузерів та WEB-серверів.

WEB-служба це програмний інтерфейс, який описує набір операцій, котрі можуть бути викликані віддалено по мережі за допомогою стандартизованих XML повідомлень. Для опису операцій або даних, що викликаються, використовуються

протоколи, що базуються на мові XML. Група WEB-служб, котрі взаємодіють один з одним вище описаним чином, представляють собою додаток WEB-служби типу SOA. Така архітектура заснована на моделі RPC (Remote Procedure Call). PRC повністю взаємодіє за допомогою протоколу SOAP. [1] [2]

В основі WEB-служб лежать Internet-стандарти. Ці стандарти визначають протоколи взаємодії та передачі даних. Стандарти WEB-служб розробляються спільно такими компаніями, як IBM, Microsoft, Ariba і деякими іншими, та затверджуються комітетом World Wide WEB Consortium (W3C).

Ключовим фактором у взаємодії WEB-служб та користувача є API (Application Programming Interface). Саме за допомогою цих API відбувається взаємодія користувача з послугами WEB-служби. API дають змогу не вникаючи у подробиці виконання, користуватися послугами WEB-служби.

Розглянувши WEB-службу, як суцільний механізм можна визначити кілька його компонентів:

- WEB-сервер на якому зберігається WEB-служба як додаток;
- WEB-служба як набір операцій (послуг);
- API з котрим взаємодіє користувач;
- Протокол передачі повідомлень SOAP. [5]

Всі ці компоненти, на сьогоднішній день мають широкий перелік потенційних вразливостей. Тож питання безпеки WEB-служб, особливо тих котрі взаємодіють з інформацією з обмеженим доступом є дуже актуальним. В цьому напрямку ведеться активний розвиток, а саме створюються різні концепції безпеки, специфікації та стандарти.

1.2 Схема взаємодії WEB-служб

Розглянемо WEB-служби з точки зору механізму, який має визначену архітектуру, порядок взаємодії та компоненти. WEB-служби вирішують проблему сумісності, даючи змогу службам працювати у різних оточеннях.

Архітектура WEB-служб складається з трьох компонентів : користувач служби, постачальник служби та реєстр служб.

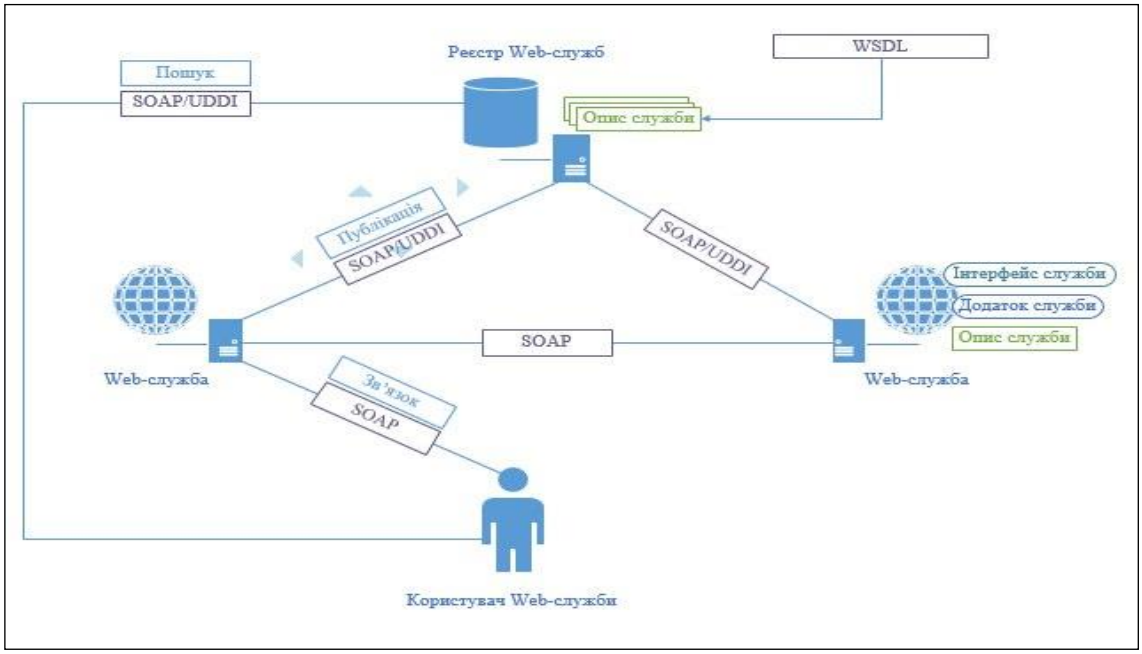


Рисунок 1.1. Взаємодія між компонентами SOA

Частіше за все у взаємодії WEB-служб присутні наступні поняття:

Відкритість взаємодії WEB-служб являє собою слабку захищеність або взагалі незахищеність процедури обміну даними між WEB-службами.

Сумісність WEB-служб пояснюється використанням WEB-службами стандарту XML, а отже і однотипного форматування документів. [4]

Розрізняють три архітектурних компонента SOA:

користувач служби: програма, програмний модуль або служба, котра здійснює пошук і виклик необхідної служби з реєстру служб за описом служби, а також використовує службу, надану постачальником служби, відповідно з інтерфейсом служби;

постачальник служби: програма, програмний модуль або служба, котра здійснює реалізацію служби у вигляді WEB-служби, прийом і виконання запитів користувачів служби, а також публікацію служби в реєстрі служб;

реєстр служб: бібліотека служб, що надає користувачам служби ресурси пошуку і виклику необхідної служби, а також приймаюча запити постачальників служб на публікацію служб.

Кожен компонент може грати або тільки одну роль (бути, наприклад, тільки користувачем служби), або одночасно відразу кілька ролей (наприклад, бути постачальником одних служб і користувачем інших).

Зауважимо, що в даному описі компонентів SOA та взаємодії між ними слід розрізняти терміни "служба" і "WEB-служба". Під службою розуміється послуга, під WEB-службою - програмна реалізація послуги.

У ході взаємодії один з одним компоненти SOA виконують такі основні операції:

публікація: для того, щоб служба була доступною (викликається) користувачам служби, необхідно зробити її інтерфейс відомим їм;

пошук: користувач служби повинен мати можливість знайти в реєстрі служб необхідну службу, що задовольняє заданим критеріям;

зв'язування і виклик: після отримання опису служби, користувач служби повинен мати можливість викликати і використовувати службу відповідно з описом служби.

Розглядаючи взаємодію компонентів SOA необхідно відзначити відмінність наступних двох компонентів:

опис служби: визначає формат запиту і відгуку при взаємодії користувача служби та постачальника служби, а також необхідну якість служби;

служба: власне служба, котра може бути викликана і використана користувачем служби відповідно до опублікованими інтерфейсом служби.

Ці компоненти взаємодіють між собою наступним чином. Постачальник служби створює WEB-службу та описує інтерфейс для її виклику. Після чого постачальник служби публікує опис служби в реєстрі служб. З метою класифікації служб, реєстр служб використовує інформацію, яка включена до опису служб. Користувач служби робить запит до служби, примушуючи реєстр служб розпочати пошук служби. Реєстр служб виконує пошук служби та формує відповідь включаючи у неї опис служби, його місцезнаходження та як порядок виконання виклику. Користувач служби зв’язується зі службою шляхом її виклику.

Експерти розділяють три фази в ході роботи WEB-служби: публікація, пошук, зв’язок.

Під час фази публікації постачальник вирішує запропонувати службу. Коли додаток розроблений та доступний як WEB-служба, організація описує інтерфейс додатку таким чином, щоб користувачі могли зрозуміти як взаємодіяти з ним. Такий опис може існувати у вигляді документу WSDL, котрий з легкістю розпізнається інструментами для розробки WEB-служб. Після опису служби, він має стати доступним для організацій, які мають інтерес у його використання. [7]

Під час фази пошуку користувач може робити запит до реєстру UDDI на наявність можливих постачальників, а також дізнатися про організацію що пропонує службу, та інтерфейс служби.

Остання фаза в циклу розробки являється зв’язок. Після того, як користувач обрав службу, він розпочинає роботу з інтерфейсом.

Після ознайомлення з процесом взаємодії доречним є розгляд алгоритму створення WEB-служби. Алгоритм створення передбачає наступні кроки:

Створення додатку котрий реалізує функціонал WEB-служби;

Створення опису інтерфейсу WEB-служби (частіше за все на мові WSDL (WEB Services Description Language)). Опис інтерфейсу визнає набір послуг, котрі підтримуються даною WEB-службою, а також кількість та тип параметрів у цих операціях. Посилання на інтерфейс WEB-служби можуть публікуватися в реєстрах служб, таких, як UDDI. В деяких випадках, немає необхідності створювати в явному вигляді файл з описом інтерфейсу;

Публікація опису інтерфейсу в реєстрі служб. (необов’язковий крок, виконуваний з метою передбачення доступу до служби у зовнішньому (public) реєстрі служб);

Розгортання WEB-служби на сервері (WEB service development).

Два логічних компонента беруть участь у виклику WEB-служби:

- клієнтський додаток, що ініціює виклик;
- серверна частина, що приймає запит клієнта та безпосередньо виконуюча виклик.

На кроці розгортання необхідно забезпечити можливість здійснення виклику WEB-служби на стороні серверу. Необхідні для розгортання дії залежать від реалізації WEB-служби, платформи, на котрій він працює, протоколу його виклику, що забезпечують транспорт та середовище роботи WEB-служби;

Створення опису реалізації WEB-служби.

Створення WSDL-документу, що вміщує інформацію про реалізацію служби. Цей документ може публікуватися в реєстрах служб, таких, як UDDI, та повинен вміщувати посилання на інтерфейс WEB-служби, котру він реалізує. Одному і тому самому інтерфейсу можуть відповідати декілька різних реалізацій (та, відповідно, їх описів); [7] [8]

Публікація опису реалізації в реєстрі служб. Цей крок є необов'язковим, та виконується у випадку необхідності доступу до служби у зовнішньому реєстрі служб;

Створення клієнтського додатку для доступу до WEB-служби. Як правило, на основі опису реалізації WEB-служби створюється так званий proxy-клас, котрий використовується клієнтським додатком та вміщує у себе усі подробиці виклику WEB-служби (наприклад, використання API-реалізації протоколу SOAP);

Розглянемо WEB-служби більш детально, розклавши на складові та рівні.

Рисунок 1.2. Стек технологій WEB-служб

Стек технологій WEB-служб принципово розбивається на наступні дві складові:

- технології, що забезпечують функціонал WEB-служб;
- технології, що забезпечують якість послуг WEB-служб.

Ці складові в свою чергу утворюються кількома рівнями:

- технології, що забезпечують функціонал WEB-служб:
- транспортний рівень (transport layer);
- комунікаційний рівень (service communication layer);
- рівень описів WEB-служб (service description layer);
- сервісний рівень (service layer);
- рівень процесів (process layer);
- рівень реєстрів WEB-служб (service registry layer).

Технології, що забезпечують якість послуг WEB-служб:

- рівень політик (policy layer);
- рівень безпеки (security layer);
- рівень транзакцій (transaction layer);
- рівень управління (management layer).

В цілях розуміння призначення рівнів, надається короткий опис кожного з них у таблиці 1.1.

Таблиця 1.1 Рівні стека технологій WEB-служб

Найменування рівня	Призначення рівня	Технології, що реалізують рівень
Функціональність		
Транспортний рівень	Описує засоби обміну даними між WEB-службами	Стандартні: HTTP, JMS Перспективні: WS-ReliableMessaging, BEEP
Комунікаційний рівень	Описує засоби формалізації механізмів використання транспортних протоколів	Стандартні: SOAP Перспективні: REST

Продовження табл.1.1

Рівень опису WEB-служби	Описує засоби формалізації інтерфейсів WEB-служб з метою забезпечення їх функціонування незалежно	Стандартні: XML, WSDL Перспективні: ebXML

	від програмно-апаратної платформи реалізації або мови програмування	
Сервісний рівень	Описує програмне забезпечення, яке викликається за допомогою WSDL-описів WEB-служб.	
Рівень процесів	Описує можливості організації WEB-служб для безперервної роботи. При цьому визначаються правила, що задають послідовність взаємодії WEB-служб з метою задоволення вимог.	Стандартні: у наш час відсутні Перспективні: BPE4WS
Рівень реєстрів WEB-служб	Описує можливості організації WEB-служб в ієрархічні бібліотеки, що дозволяють публікацію, пошук і виклик WEB-служб з їх описів WSDL	Стандартні: UDDI Перспективні: WS-Inspections
Якість WEB-служби		
Рівень політик	Описує правила та умови, згідно з якими WEB-служби можуть бути використані. Оскільки дані правила і умови відносяться як до функціонального аспекту WEB-служб, так і до аспекту забезпечення якості послуг на Рис. 1, даний шар є загальним для обох аспектів	Стандартні: у наш час відсутні. Перспективні: WS-Policy, WS-PolicyAssertions и WS-PolicyAttachment
Рівень безпеки	Описує можливості забезпечення безпеки WEB-служб і безпеки їх функціонування (авторизація, аутентифікація і поділ доступу)	Стандартні: WS-Security Перспективні: WS-Secure Conversation, WS-Federation, WS-Authorization, WS-Trust и WS-Privacy

Продовження табл.1.1

Рівень транзакцій	Описує властивість транзакційності розподілених систем на основі WEB-служб для забезпечення надійності їх функціонування	Стандартні: у наш час відсутні Ті, що розвиваються: WS-Transaction и WS-Coordination

Рівень управління	Описує можливості управління WEB-службами характеристиками їх функціонування	

Представлений вище стек технологій WEB-служб вводить ієрархію в множину технологій відповідно до їх функціонального призначення. При цьому в таблиці вказані лише найбільш широко вживані і усталені технології.

Стандартними названі технології, що отримали офіційний статус стандартів міжнародних консорціумів з розробки IT-стандартів (W3C, OASIS або WS-I). Привівши схему взаємодії та алгоритм створення WEB-служби вважаю за логічне привести механізм взаємодії клієнта та сервера у SOA.

Механізм взаємодії клієнта та сервера.

Клієнтський додаток створює примірник об'єкта SOAPClient.

SOAPClient читає файли опису WEB-служби (WSDL та WEB Services Meta Language – WSML). Ці файли можуть зберігатися і у клієнта.

Клієнтський додаток SOAPClient викликає WEB-службу. SOAPClient формує пакет запиту (SOAPEnvelope) та надсилає на сервер. Можливо використання будь-якого транспортного протоколу, але як правило використовується HTTP.

Пакет приймає серверний додаток Listener (може являти собою ISAPI додаток або ASP сторінку), створює об'єкт SOAPServer та передає йому пакет запиту.

SOAPServer читає опис WEB-служби, завантажує опис та пакет запиту. SOAPServer викликає метод об'єкта/додатку, котрий реалізує служба. Результат виконання методу або опис помилки конвертується об'єктом SOAPServer у пакет відповіді та відправляються клієнту. Об'єкт SOAPClient проводить розбір прийнятого пакету та повертають клієнтському додатку результати роботи служби або опис помилки. [1] [2] [5]

1.3 Протоколи та стандарти взаємодії WEB-служб

Створення, взаємодія та робота WEB-служб реалізується за допомогою стандартів та протоколів.

Протокол – сукупність правил, відповідно до яких взаємодіють об'єкти в мережі Інтернет з метою обміну інформацією.

Extensible Markup Language – XML

XML був створений у XML Working Group (спершу відомою як SGML Editorial Review Board), сформованою під патронатом World Wide WEB Consortium (W3C) у 1996 році. До цього багато компаній розробляли свої власні закриті стандарти і формати. А зараз нам для роботи потрібно знати XML (eXtensible Markup Language), який передається по протоколу HTTP.

XML була задумана як гнучка і в той же час формальна метамова для використання в Інтернеті. Перевагами XML є змога формалізації документів одним способом, щоб зробити їх легко оброблюваними для машин (комп'ютерів).

Метамова (metalanguage) - це мова, яка призначена для опису інших мов. Наприклад, можна сказати, що словник англійської мови в сукупності з англійською граматикою утворюють метамова, що описує англійську мову.

Що стосується мови XML, то її призначення - описувати мову розмітки. У мові розмітки (markup language) для структурування даних використовуються теги. Мова розмітки відрізняється гнучкістю і легко підлаштовується під потреби користувача. Головне - знати і дотримуватись основних правил. Початок документа XML, його перший рядок повинні містити обов'язкову конструкцію, яка вказує на версію XML, принцип кодування та які бібліотеки підключаються для цього. [4]

Мова гіпертекстової розмітки (Hypertext Markup Language, HTML), яка являється найбільш поширеною на сьогоднішній день мовою розмітки, початково була написана на SGML, але мала би змогу бути написанаю і на XML. Необіхдно зробити нотатку в тому, що XML і HTML не замінюють один одного. Можна лише перетворити код з одного формату в інший. З цього робимо висновок, що XML в HTML виконується за допомогою онлайн-конвертерів. Мова XML призначена для зберігання і відправки даних, а HTML служить для їх відображення на веб-сторінці.

XML є незалежною від платформ, а отже постачальник та користувач, який робить запит можуть спілкуватися один з одним за допомогою XML, не зважаючи на те, які платформи вони використовують. Повідомлення XML можуть бути передані по мережі Інтернет використовуючи протокол HTTP.

У додатках XML зазвичай використовуються наступні типи даних і допоміжні функції:

- сам файл XML, який має строго певну структуру;
- визначення типу документа (Document Type Definition, DTD), де визначається структура файлу XML (необов'язковий елемент);
- таблиці стилів, що містять інформацію про те, як дані повинні бути відформатовані при виведенні (необов'язковий елемент);
- процесор XML і різні службові функції для маніпулювання даними та переформатування даних.

Різниця між WEB-службами та іншими технологіями, з якими розробникам доводилося зустрічатись (наприклад, DCOM, іменовані канали - namedpipes, RMI) в тому, що WEB-служби, які засновані на відкритих стандартах легко опанувати і ці стандарти широко підтримуються на платформах Unix та Windows.

Simple Object Access Protocol – SOAP.

SOAP протокол обміну інформацією в децентралізованому, розподіленому середовищі. Протокол SOAP створений в 1998 році командою розробників під керівництвом Дейва Вінера (Dave Winer), яка працювала в корпорації Microsoft і фірмі Userland, але потім переданий в консорціум W3C.

Остання версія стандарту на сьогоднішній день - SOAP 1.2. У версії 1.1 SOAP розшифровувався як Simple Object Access Protocol - простий протокол доступу до

об'єктів. Ця назва відображала його первісне призначення - звертатися до методів віддалених об'єктів. Зараз призначення SOAP змінилося, тому різні розробники пропонували свої варіанти розшифровки. Тому в версії 1.2 абревіатуру вирішили ніяк не розшифровувати.

Протокол SOAP не розрізняє виклик процедури і відповідь на нього, а просто визначає формат послання (message) у вигляді документа XML. Послання може містити виклик процедури, відповідь на нього, запит на виконання якихось інших дій або просто текст. Специфікацію SOAP не цікавить вміст послання, вона задає тільки його оформлення. Цей протокол базується на XML і розширює деякі протоколи прикладного рівня - HTTP, FTP, SMTP і т.д. Як правило найчастіше використовується HTTP. Замість використання HTTP для запиту HTML-сторінки, яка буде показана в браузері, SOAP відправляє за допомогою HTTP-запиту XML-повідомлення і отримує результат у HTTP-відгуку. Для правильної обробки XML-повідомлення процес-«слухач» HTTP (напр. Apache або Microsoft IIS) повинен надати SOAP-процесор, або, іншими словами, повинен мати можливість обробляти XML.

SOAP є найголовнішою частиною технології Web-сервісів. Він здійснює перенесення даних по мережі з одного місця в інше.

SOAP забезпечує доставку даних веб-сервісів. Він дозволяє відправнику і одержувачу XML-документів підтримувати загальний протокол передачі даних, що забезпечує ефективність мережевої зв'язку.

SOAP - це базова однонаправлена модель з'єднання, що забезпечує узгоджену передачу повідомлення від відправника до одержувача, потенційно допускає наявність посередників, які можуть обробляти частину повідомлення або додавати до нього додаткові елементи. Специфікація SOAP містить угоди по перетворенню односпрямованого обміну повідомленнями відповідно до принципу «Запит / відповідь», а також визначає як здійснювати передачу всього XML-документа. [4] [5]

SOAP-повідомлення являє собою XML-документ; повідомлення складається з трьох основних елементів: конверт (SOAP Envelope), заголовок (SOAP Header) і тіло (SOAP Body).

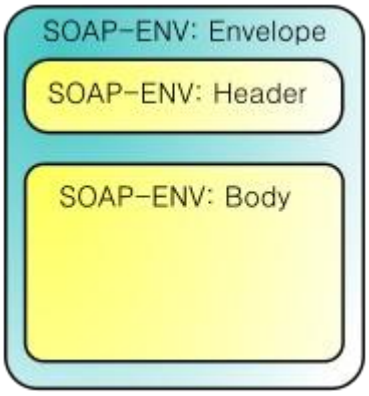


Рисунок 1.3. Структура повідомлення SOAP

Приклад повідомлення SOAP:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:t="www.example.com">
```

```
<SOAP-ENV:Header>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
      <t:CurrentDate>
            <Year>2011</Year>
            <Month>February</Month>
            <Day>12</Day>
            <Time>18:02:00</Time>
      </t:CurrentDate>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Пакет (SOAP Envelope)

Є «вищим» елементом SOAP повідомлення. Описується за допомогою елемента Envelope з обов'язковим простором імен soap-envelope для версії 1.2 і soap для версії 1.1.

В елементі Envelope можуть бути атрибути xmlns, що визначають простори імен, та інші атрибути, забезпечені префіксами. Зазвичай простори імен дочірніх елементів мають назви, відповідні до назв своїх елементів.

Envelope може мати необов'язковий дочірній елемент Header. Якщо цей елемент присутній, то він повинен бути першим прямим дочірнім елементом пакета.

Наступний дочірній елемент пакета повинен мати ім'я Body. Це обов'язковий елемент і він повинен бути другим прямим дочірнім елементом пакета, якщо є заголовок, або першим - якщо заголовка немає.

Версія 1.1 дозволяла після тіла повідомлення записувати довільні елементи, забезпечені префіксами. Версія 1.2 це забороняє.

Елементи Header і Body можуть містити елементи з різних просторів імен.

Пакет змінюється від версії до версії. SOAP-процесори, сумісні з версією 1.1, при отриманні повідомлення, що містить пакет з простором імен версії 1.2, будуть генерувати повідомлення про помилку. Аналогічно для SOAP-процесорів, сумісних з версією 1.2. Помилка - VersionMismatch.

Заголовок SOAP (SOAP Header)

Перший прямий дочірній елемент пакета. Є не обов'язковим. Заголовок крім атрибутів xmlns може містити 0 або більше стандартних атрибутів:

- encodingStyle
- actor (або role для версії 1.2)
- mustUnderstand
- relay

Атрибут encodingStyle

У SOAP-повідомленнях можуть передаватися дані різних типів (числа, дати, масиви, рядки тощо). Визначення цих типів даних виконується в схемах XML. Типи, визначені в схемі, заносяться в простір імен, ідентифікатор якого служить

значенням атрибута encodingStyle. Атрибут encodingStyle може з'явитися в будь-якому елементі SOAP-повідомлення, але версія SOAP 1.2 забороняє його появу в кореневому елементі пакета. Зазначений атрибутом encodingStyle простір імен буде відомо у тому елементі, в якому записаний атрибут, і у всіх вкладених у нього елементах. [4] [5]

Стандартний простір імен, в якому розташовані імена типів даних SOAP 1.1, називається encoding. У версії 1.2 - soap-encoding. Ідентифікатор того чи іншого простору імен, в якому визначені типи даних, зазвичай отримує префікс enc або SOAP-ENC.

Атрибут actor

Тип даних URI. Задає адресу конкретного SOAP-сервера, якому призначено повідомлення. SOAP-повідомлення може пройти через кілька SOAP-серверів або через кілька додатків на одному сервері. Ці додатки виконують попередню обробку блоків заголовка посилання і передають його один одному. Всі ці сервери і / або програми називаються SOAP-вузлами (SOAP nodes). Специфікація SOAP не визначає правила проходження повідомлення по ланцюжку серверів. Для цього розробляються інші протоколи, наприклад, Microsoft WS-Routing.

Атрибут actor задає кінцеву точку призначення, SOAP-вузол - той, який розташований в кінці ланцюжка і буде обробляти заголовок повністю. Значення next атрибуту actor показує, що обробляти заголовок буде перший же сервер, що отримав його. Атрибут actor може зустрічатися в окремих блоках заголовка, вказуючи вузол-обробник цього блоку. Після обробки блок видаляється з SOAP-повідомлення.

У версії 1.2 атрибут actor замінений атрибутом role, тому що в цій версії SOAP кожен вузол грає одну або кілька ролей. Специфікація поки визначає три ролі SOAP-вузла:

- Роль ultimateReceiver грає кінцевий, вий вузол, який буде обробляти заголовок.
- Роль next грає проміжний або цільовий вузол. Такий вузол може грати і інші, додаткові ролі.
- Роль none не повинен відігравати жоден SOAP-вузол.

Розподілені додатки, виходячи зі своїх потреб, можуть додати до цих ролей інші ролі, наприклад, ввести проміжний сервер, перевіряючий цифровий підпис і визначити для нього цю роль рядком URI.

Значенням атрибута role може бути будь-який рядок URI, що показує роль вузла, якому призначений даний блок заголовка. Значенням за замовчуванням для цього атрибута служить порожнє значення, тобто, просто пара лапок.

Значення атрибута role показує, що блок повинен бути оброблений вузлом, що відіграє певну роль, відповідну значення рядка.

Атрибут mustUnderstand

Тип даних - boolean. За умовчанням 0. Якщо значення дорівнює 1, то SOAP-вузол при обробці елемента обов'язково повинен враховувати його синтаксис,

визначений у схемі документа, або зовсім не обробляти повідомлення. Це підвищує точність обробки повідомлення.

У версії SOAP 1.2 замість цифр потрібно писати true або false.

Атрибут relay

Тип даних - boolean. Показує, що заголовний блок, адресований SOAP-посереднику, повинен бути переданий далі, якщо він не був оброблений. Якщо заголовний блок оброблений, правила обробки SOAP вимагають, щоб він був видалений. Типово, необроблений заголовний блок, призначений ролі, яку виконує SOAP-посередник, повинен бути вилучений перед відправкою повідомлення.

Усі прямі дочірні елементи заголовка називаються блоками заголовка (у версії 1.1. - Статтями). Блоки заголовка використовуються для розширення повідомлень децентралізованим способом, шляхом додавання аутентифікації, адміністрування транзакцій та інші. Їх імена обов'язково повинні позначатися префіксами. У блоках заголовка можуть бути атрибути role, actor і mustUnderstand. Дія цих атрибутів відноситься тільки до даного блоку. Це дозволяє обробляти окремі блоки заголовка проміжними SOAP-вузлами, чия роль збігається з роллю, зазначеної атрибутом role. Нижче дано приклад такого блоку: [5]

```
<env:Header>
<T: Transaction xmlns: t = " transaction"
env: role = ultimateReceiver"
env: mustUnderstand = "true">
</ T: Transaction>
</ Env: Header>
```

Елементи, вкладені в блоки заголовка, вже не називаються блоками. Вони не можуть містити атрибути role, actor і mustUnderstand.

Тіло SOAP (SOAP Body)

Елемент Body обов'язково записується відразу за елементом Header, якщо він є в повідомленні, або першим в SOAP-повідомленні, якщо заголовок відсутній. В елемент Body можна вкласти довільні елементи, специфікація ніяк не визначає їх структуру. Визначений лише один стандартний елемент, який може бути в тілі повідомлення - Fault, що містить повідомлення про помилку.

SOAP-повідомлення з вкладеннями

Існують ситуації, коли клієнт і сервер повинні обмінюватися даними у форматі, відмінному від текстового. Це можуть бути дані у форматах додатків, мультимедійних даних, тощо. З точки зору обміну даними нетекстові дані розглядаються як дані в двійкових кодах.

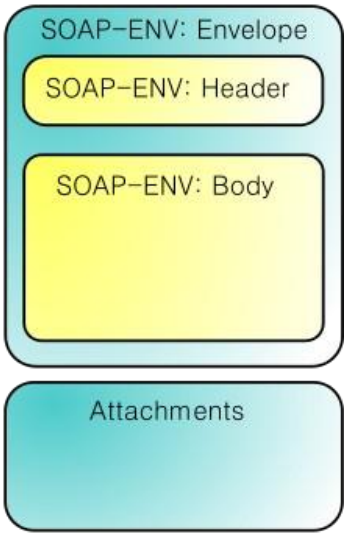


Рисунок 1.4. Структура повідомлення SOAP з вкладенням

Двійкові дані включаються до повідомлення у вигляді вкладення. У 2000 році консорціумом W3C розробив специфікацію «SOAP-повідомлення з вкладеннями» (SOAP with Attachments) (http://www.w3.org/TR/SOAP-attachments/), яка спирається на версію SOAP 1.1. У специфікації описані правила включення SOAP-повідомлення в MIME-повідомлення типу multipart / related і правила пересилки його протоколом HTTP.

Цей протокол визначає пересилку SOAP-повідомлення всередині MIME-повідомлення, що складається з декількох частин. Перша частина MIME-повідомлення, частина SOAP - містить XML: пакет SOAP з вкладеними в нього заголовком і тілом повідомлення. Інші частини вкладення містять дані в будь-якому форматі, двійковому або текстовому. Кожна частина випереджається MIME-заголовком, що описує формат даних частини і містить ідентифікатор частини (Content-ID). За цим ідентифікатором тіло SOAP-повідомлення може посилатися на вкладення (href).

WEB Services Description Language – WSDL.

WSDL розшифровується як мова опису веб-сервісів на основі XML. Це стандартний формат для опису WEB-служби. WSDL був розроблений спільно Microsoft і IBM. Щоб надати змогу користуватися WEB-службою, спочатку треба її описати, представити детальну інформацію щодо використання, та розмістити його у правильну категорію, подібно до довідника. The WEB Services Description Language був створений з метою виконання даних вимоги. Він створює стандартний формат описання деталей WEB-служб, та використовується для опису деталей WEB-служб, таких як інтерфейси, зв'язки тощо. З цим стандартизованим шляхом описання деталей служби, клієнти можуть знайти та використати службу, навіть не маючи попередніх знань про неї. [1] [2] [4] [7]

Також WSDL часто використовується в поєднанні з SOAP і XML-схемою для надання веб-сервісів через Інтернет. Клієнтська програма, підключається до WEB-служби, може прочитати WSDL, щоб визначити, які функції доступні на сервері. Всі використовувані спеціальні типи даних вбудовуються в файл WSDL в формі XML-схеми. Потім клієнт може використовувати SOAP для фактичного виклику

однієї з функцій, перерахованих в WSDL. Ariba, IBM і Microsoft були залучені в розвиток WSDL1.1.

Universal Description, Discovery and Integration – UDDI.

Протокол універсального опису, виявлення та інтеграції (UDDI) є центральним елементом групи пов'язаних стандартів, що складають стек веб-служб. UDDI специфікація визначає стандартний метод публікації та виявлення мережевих програмних компонентів сервісно-орієнтованої архітектури (SOA). Його розробкою керує консорціум постачальників програмного забезпечення для підприємств та клієнтів OASIS.

Функціональним призначенням UDDI реєстру є представлення даних та метаданих про WEB-сервіси. Реєстр для використання у загальнодоступній мережі або в межах внутрішньої інфраструктури організації, пропонує каталог, заснований на стандартах механізму класифікації і управління WEB-службами, щоб вони могли бути виявлені та використані іншими програми. Як частина узагальненої стратегії опосередкування між додатками UDDI пропонує кілька переваг для IT-менеджерів як під час проектування так і під час час роботи, включаючи збільшення повторного використання коду та вдосконалення інфраструктури управління: [8]

- Публікація інформації про веб-служби та правила класифікації, характерні для організації.
- Пошук веб-сервісів (всередині організації або між організаціями мережі), які відповідають заданим критеріям.
- Визначення протоколів безпеки та транспорту, що підтримуються певними Інтернет послугами та параметри, необхідні для виклику служби.
- Забезпечення засобів для ізоляції додатків (а також забезпечення інтелектуальних та безвідмовних процесів маршрутизації) від збоїв або змін у викликаних послугах.

UDDI описує реєстр веб-служб та програмних інтерфейсів для публікації, отримання та управління інформацією про послуги, описані в них.

Насправді, сам UDDI це і є веб-сервіс. Він визначає послуги які підтримують опис та відкриття підприємств, організацій та інших провайдерів веб-послуг та технічні інтерфейси, які можуть бути використані для доступу до цих служб та управління ними. Він базується за кількома встановленими галузевими стандартами, включаючи HTTP, XML, XML схеми XSD, SOAP, WSDL.

Основна інформаційна модель, що використовується UDDI реєстром, визначається декількома XML схемами. XML був обраний, оскільки він пропонує нейтральний для платформи перегляд даних і дозволяє описувати природним чином ієрархічні взаємозв'язки. XSD було обрано через підтримку розширених типів даних й здатність легко описувати та перевіряти інформацію на основі інформаційних моделей, представлених у схемах. UDDI XSDS визначає кілька

основних типів інформації, які повинні знати користувачі та програми, щоб використовувати певну веб-службу. Разом вони утворюють базову інформаційну модель і рамки взаємодії UDDI реєстрів. [4] [8]

Модель складається з:

- Опис ділової функції послуги (називається businessService).
- Інформація про організацію, яка видала послугу (businessEntity).
- Технічні деталі служби (bindingTemplate), включаючи посилання на програмний інтерфейс служби або API.
- Різні інші атрибути або метадані, такі як систематика, транспорт, цифрові значення.

Інфраструктура послуг добре підходить для вирішення. UDDI використовує WSDL для описання інфраструктури WEB-служби та використовує SOAP як основний протокол. Специфікація UDDI включає дві категорії інтерфейсів API для доступу до служби UDDI з додатків: API використовують для пошуку інформації в реєстрі; Publisher API- дозволяють реєструвати служби в реєстрі.

Висновок до Розділу 1

Технологія WEB-служб все більше запроваджується у наше повсякденне життя. У тому числі й в урядові установи. Розглянувши більш детально дану технологію можна помітити ряд переваг, що вона надає.

Однією з головних переваг даної технології є малі витрати на запровадження. Обмежений бюджет вже не буде такою проблемою як раніше.

Не можна не зазначити й гнучкість даної технології. Дана гнучкість стосується даних що передаються, середовища роботи WEB-служб та задач, котрі вони виконують.

РОЗДІЛ 2
ЗАГРОЗИ ТА МЕХАНІЗМИ ЗАХИСТУ ПРОЦЕДУРИ ВЗАЄМОДІЇ WEB-СЛУЖБ

2.1 Загальні поняття безпеки WEB-служб.

WEB-служби є відкритими та сумісними за своєю будовою. Відвертість з якою WEB-служби взаємодіють між собою може створити слабкі місця в безпеці. Зловмисники можуть отримати доступ до даних всередині передачі та виправляти конфіденційні дані, або навіть красти важливу інформацію. Несанкціонований доступ до даних, навмисна зміна значень та/або підтвердження невірної або хибної інформації або переривання може спричинити загрози безпеці інформаційній системі.

Безпека WEB-служби це цеглина, що використовується у взаємодії з іншими WEB-службами та протоколами для забезпечення широкого вибору моделей безпеки та шифрувальних технологій.

Багато експертів обговорювали проблеми безпеки WEB-служб XML, але більшість з них направлені на технології що базуються на перевірених механізмах таких як XML Encryption, XML signature. Більшість установ використовують WEB-служби, навіть не розуміючи повного обсягу ризиків. Такі механізми безпеки, як ідентифікація, шифрування, санкціонування, конфіденційність та цілісність інформації грають визначну роль у безпеці WEB-служби. [3] [10] [4]

Ідентифікація користувачів надає можливість контролю клієнтів.

Щоб упевнитися у цілісності повідомлень та захищеності важливих даних, у WEB-службах використовується шифрування.

Аутентифікація використовується для визначення сторін котрі встановили зв'язок з WEB-службою. Користувач служби повинен спочатку пройти аутентифікацію у постачальника служби перед пересилкою інформації та навпаки.

Авторизація використовується для визначення, прав аутентифікованого користувача, а саме до якої інформації він має доступ, які операції він може виконувати.

Цілісність даних надає упевненість, що повідомлення не було змінене неавторизованим процесом або користувачем після його створення. Для цього впроваджуються XML encryption та XML signature. Цифрові підписи дають змогу упевнитися адресату, що зміст не був змінений та відправник повідомлення не був підроблений.

Конфіденційність упевнює користувачів що дані захищені від неауторизованих користувачів. Конфіденційність є головним пріоритетом коли розмова іде про транзакції у WEB-службах. Для всестороннього захисту WEB-служб усі зазначені механізми безпеки повинні бути поєднаними та виконаними належним чином. Додатковим рівнем безпеки є рівень додатків котрий захищає кожну частину повідомлення під час передачі.

Для захисту на рівні додатків визначаються чотири поняття: Public Key Cryptography (PKC), SOAP розширення безпеки, цифрова безпека, XML digital signature.

Public Key Cryptography використовується для аутентифікації та шифрування даних, а також створення та обміну цифровими підписами, даючи тим самим змогу до безпечного обміну інформацією. PKC упевнює у цілісності, конфіденційності даних та аутентифікації користувачів.

Розширення безпеки SOAP це заголовки які є необов'язковою частиною повідомлення SOAP у документі XML. SOAP має змогу працювати на різних платформах, використовуючи HTTP та XML для передачі та отримання повідомлень. SOAP не вимагає гарного зв'язку між клієнтом та сервером. Для визначення типів документів SOAP використовує W3C XML Schema.

SOAP обмінюється інформацією використовуючи повідомлення у форматі пакетів SOAP. Вміщуючи усю інформацію яка пересилається кожен пакет розділяється на дві частини: заголовок та тіло. Повідомлення SOAP не обов'язково має заголовок, але усі повідомлення повинні мати тіло, котре обов'язкове. Тіло повідомлення не може бути зміненим на відміну від заголовку. [5]

Цифрова безпека упевнює цілісність компонентів операційної системи, драйверів та програмованих об'єктів.

Безпека XML орієнтується на захист повідомлень. [4]

Через взаємодію WEB-служб з різними системами, пристроями, платформами важко виявити неавторизовану активність. Отже не має цілковитого захисту, але механізми безпеки затримують зловмисників.

2.2 Класифікація атак на протоколи взаємодії

В даній роботі приводиться широкий перелік атак націлених на WEB-служби. Як було зазначено раніше WEB-служби складають єдину систему з кількох компонентів, а отже атаки також мають різні цілі та шляхи здійснення.

Нижче приведено класифікацію атак, котрі розглядаються у даній роботі. Дана класифікація приведена за властивостями захищеності інформації, котрі визначені у ДСТСЗІ СБ України НД ТЗІ 3.7-001-99.

У даному документі визначені наступні властивості захищеності інформації:

- Конфіденційність
- Цілісність
- Доступність
- Спостереженість

Класифікація:

Атаки націлені на властивість "доступність"

- BPEL Instantiation Flooding
- BPEL Indirect Flooding

- BPEL State Deviation
- Coercive Parsing
- Oversized XML attack
- Reference Redirect
- RecursiveCryptography
- SOAP ArrayAttack
- SOAP Parameter DOS
- WS-Addressing spoofing
- XML Document Size Attack
- XML Signature – Transformation DOS
- XML ExternalEntity DOS
- XML EntityExpansion
- XML Entity Reference Attack
- XML Flooding
- XML Signature – KeyRetrieval DOS
- XMLInjection
- XMLSignature – KeyRetrievalXSA (CrossSiteAttack)
- XML Signature – XLST Code Execution

Атаки націлені на властивість “спостережність”

- XpathInjection
- ReplayAttack
- SOAPAction Spoofing

Атаки націлені на властивість “цілісність”

- Metadata spoofing
- XML SignatureWrapping
- Active WS-MITM

Атаки націлені на властивість “конфіденційність”

- Attack Obfuscation
- WSDL Disclosure

2.3 Загрози та захист протоколів взаємодії WEB-служб

Нижче наведений перелік атак, передумови для їх виконання та контрзаходи відомих атак на сьогоднішній день. Дані атаки виконуються на WEB-служби, що класифіковані за властивостями захищеності інформації.

Атаки за властивістю “доступність”

BPEL Instantiation Flooding

Кожне, засноване на BPEL визначення процесу роботи, включає у себе хоча б одну дію, яка створює новий примірник процесу з кожним отриманим повідомленням. Такий примірник процесу негайно починає виконуватись, слідуючи інструкціям, що надаються у визначенні процесу. Процес виконання буде зупинений кожен раз, як буде досягатися дія приймання, та поновлюватиметься після прийому очікуваного повідомлення. Після досягнення кінцевої точки виконання зупиняється та примірник процесу видаляється.

Двигун BPEL самотужки виконує усі процеси, окрім дії приймання. Для кожного вхідного повідомлення SOAP, двигун BPEL створюватиме окремий примірник процесу та розпочинатиме його виконання, виконуючи кожен до тих пір, поки вони не дійдуть до кінцевої точки виконання. В результаті чого двигун BPEL може бути перенавантажений обробляючи повідомлення, що зведе виконання його функцій до нуля. [5]

Також треба розуміти, що двигун BPEL не є єдиною ціллю даної атаки. Кожен створений примірник процесу виконується як законний, включаючи усі вихідні запити WEB-служб до зовнішніх учасників спілкування. Таким чином відкривається шлях до перенавантаження учасників спілкування.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби;

- Зловмисник може дістатися до кінцевої точки з його місця знаходження.

Контрзаходи

Захистом від такої атаки є вимога ідентифікації та відхилення семантично невірно оформлених запитів. Проте, це завдання не є тривіальним через те, що визначити чи є запит злоякісним можна тільки після його обробки.

BPEL Indirect Flooding

Дана атака використовує ту саму методологію яка представлена вище, але відрізняється її ціль. Головною ідеєю є використання двигуна BPEL, як посередника для атаки на систему що знаходиться поза двигуном BPEL. Наприклад, процес BPEL багаторазово викликає WEB-службу котра надається обраною системою, створюючи облікові записи користувачів з кількома деталями. [9] [13]

Якщо багаторазово посилати повідомлення, двигун BPEL буде підданий великому навантаженню від самого себе, але також це віддзеркалить дане навантаження на обрану систему. Тож, якщо система не настільки потужна як двигун BPEL, вона втратить свою придатність.

Цей метод не може бути реалізований у системах, що використовують WS-Security. Зв'язок між двигуном BPEL та системою здійснюється тільки по довіреному та повністю законному шляху.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби;
- Зловмисник знає метадані, такі, як файли WSDL;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від такої атаки є вимога ідентифікації та відміни атакуючого повідомлення. Треба розуміти, що кінцева ціль атаки не двигун BPEL, а система за ним.

BPEL State Deviation

BPEL процеси повинні бути викликаними зовнішніми учасниками спілкування, двигун BPEL забезпечує отримання кожного вхідного повідомлення кінцевою точкою WEB-служби.

Підтипи атак:

BPEL Correlation Invalidation

Зловмисник може атакувати кінцеву точку WEB-служби, використовуючи повідомлення з вірним форматом але з некоректною командою записаною в середині. Обробка даного повідомлення не займе багато часу, повідомлення буде

відкинуто двигуном BPEL. Надіславши 1000 таких повідомлень процесор BPEL буде завантажений на декілька годин.

BPEL State Invalidation

Атака головною метою котрої є виклик операції що не підтримується у створеному примірнику процесу при обробці повідомлення. Це повідомлення буде відкинуто двигуном BPEL але викликає великі затрати ресурсів, зменшенням якості або навіть відмовою.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає метадані, такі як файли WSDL;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від таких атак є відхилення невірно сформованих повідомлень, та повідомлень з невірними твердженнями. Потрібно зазначити, що визначення таких атак суто відрізняється для різних типів повідомлень.

Coercive Parsing

Одна з найпростіше виконуваних атак. Атака націлена на виснаження системних ресурсів WEB-служби. Зловмисник відправляє звичайне повідомлення SOAP з неймовірною кількістю відкритих тегів у тілі. Ця атака являється однією з найбільш руйнівних. [5] [9] [13]

WEB-служби, що використовують синтаксичний аналізатор DOM сприятливі до таких атак. Синтаксичний аналізатор DOM створює переоформлену копію всередині пам’яті.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби. WSDL не вимагається у випадку націленості атаки на синтаксичний аналізатор XML.
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від таких атак є використання вимогливої системи аналізу. Кожен WSDL повинен вміщувати детальний опис кожного елемента, атрибута, типа даних.

Oversized XML attack

У звичному повідомлені SOAP, довжина тегу XML складає кілька символів. Оголошення простору імен може досягати довжини кількасот символів, але зазвичай не викликає проблем у синтаксичного аналізатора XML. Відмову синтаксичного аналізатора XML можна викликати шляхом переповнення його пам'яті. Надіславши повідомлення SOAP з надто довгим тегом зловмисник може викликати відмову аналізатора. Не обмеженими елементами являються: [4] [5]

- ім'я
- ім'я атрибуту
- простір імен
- кількість атрибутів

Підтипи атак:

XML Extra Long Names

Атака дуже легка до виконання. Все, що потрібно зробити зловмиснику, це використати дуже довге ім'я елементу, атрибуту або простору імен. Результатом є переповнення буферу синтаксичного аналізатору XML для імен елементів, атрибутів та просторів імен, відмова служби.

XMLNamespacePrefixAttack

Зловмисник розташовує багато атрибутів в елемент, що спричиняє переповнення буфера синтаксичного аналізатора XML.

XMLOversizedAttributeContent

Переповнення буфера спричинюється використанням дуже довгої послідовності, як значення атрибуту.

XMLOversizedAttributeCount

Переповнення буфера спричинюється використанням великої кількості атрибутів у елементі.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Після того,як будь-які обмеження довжини компонентів всередині тегів XML у стандарті XML були відкинуті, розробник має особисто виставляти обмеження. Шляхом ручного визначення максимальної довжини кожного елементу, атрибуту та значення атрибуту.

Reference Redirect

Використовуючи XML signature чи XML encryption для захисту звичайного повідомлення SOAP ви маєте широкий вибір можливостей у підписуванні або

зашифровуванні даних. Можливо навіть підписувати чи зашифровувати дані поза межами повідомлення. Потрібно тільки зробити посилання на файл, що зберігає інформацію котра повинна бути підписаною чи зашифрованою. Отримуючи таке повідомлення службі необхідно отримати даний файл для обробки повідомлення. Зловмисник може використати це для здійснення атаки. Використавши надто великий файл, наприклад в 1гБ можна привести службу у неактивний стан на час обробки такого повідомлення. [4] [5]

Підтипи атак:

Signature redirect

Атака, у котрій зловмисник використовує посилання на підписаний елемент.

Encryption redirect

Атака, у котрій зловмисник використовує посилання на зашифрований елемент.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє захищений заголовок та зашифрований елемент або підписаний елемент. Якщо WEB-служба не обробляє зашифровані повідомлення, він просто не обробить повідомлення і атака буде марною;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захиститися від такої атаки дуже легко заборонивши посилання на зовнішні джерела у повідомленнях. Якщо ж без посилань не обійтися не так легко знайти безпечне рішення. Вирішенням даної ситуації може бути використання білого списку посилань.

Recursive Cryptography

WEB-служби надають велику множину варіантів щодо захисту коли іде мова про використання механізмів захисту, таких як цифрові підписи та криптографія. Різні частини повідомлення SOAP можуть бути зашифровані або підписані з використанням різних ключів. Це може відкрити шлях зловмиснику до перенавантаження служби.

Підтипи атак:

Chained Cryptographic Keys

Атака у котрій перенавантажується служба шляхом створення нескінченної послідовності зашифрованих ключів.

Nested Encrypted Blocks

Атака у котрій перенавантажується служба шляхом створення нескінченної послідовності зашифрованих елементів.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє захищений заголовок та зашифрований елемент або підписаний елемент. Якщо WEB-служба не обробляє зашифровані повідомлення, він просто не обробить повідомлення і атака буде марною;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом є використання розробником “Strict WS-Security PolicyEnforcement”. Це передбачає використання тільки повідомлень SOAP. Слідуючи вимогам безпеки, зазвичай WS – SecurityPolicy визначає тільки мінімальні вимоги до повідомлення SOAP. Використовуючи “Strict WS-Security PolicyEnforcement” замість мінімальних вимог до безпеки реалізуються максимальні. Повідомлення SOAP котрі не відповідають вимогам “Strict WS-Security PolicyEnforcement” не обробляються синтаксичним аналізатором XML. [4] [5]

SOAP ArrayAttack

Повідомлення SOAP славляться своєю гнучкістю. В повідомленнях SOAP підтримуються масиви. Перед тим як використовувати масиви SOAP необхідно визначити їх розмір. По замовчанню розмір масиву необмежений. Уявімо що зловмисник зробить масив з 1,000,000,000 елементів. При обробці повідомлення синтаксичним аналізатором, WEB-служба зарезервує місце у пам’яті RAM для 1,000,000,000 елементів. Використавши це зловмисник зазвичай виснажує пам'ять атакованої системи. [5]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби. WSDL не вимагається у випадку націленості атаки на синтаксичний аналізатор XML.
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від такої атаки є використання вимогливої системи аналізу. В багатьох випадках максимальне число елементів уже відоме. Якщо ж максимальне число елементів не може бути передбачено, існує інше рішення. Воно передбачає перевірку на кількість зазначених елементів у атрибуті “soapenv_arrayType” та кількість насправді існуючих елементів масиву. Якщо елементи не співпадають, повідомлення не оброблюється. Цей механізм повинен запроваджуватися розробником WEB-служби.

SOAP Parameter DOS

Зазвичай кожен запит SOAP вміщує деякий набір параметрів котрі призначені логіці додатку. Якщо логіка додатку не передбачує перевірку типів параметрів, може бути виконане класичне переповнення буферу. Простим прикладом є рядок значень, діапазон котрих значно перевищений. [5]

Зазначте, що усі додатки котрі передбачають користувацький ввід, можуть бути вразливими до такої атаки.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає які параметри очікуються логікою додатку;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захист від таких атак є вимоглива система аналізу вхідних даних, неважливо з яких джерел надходить інформація. Додатки WEB-служби не виключення. Від тоді, як запити до WEB-служби обробляються у декілька кроків, при використанні вимогливої системи аналізу атака може бути виявлена достатньо рано. Також, використання вимогливої системи аналізу має свою перевагу у тому, що викривши атаку наступні кроки такі, як підтвердження підпису або розшифрування не виконуються. Це заощаджує системні ресурси. [9] [13]

WS-Addressing spoofing

Стандарт адресації WEB-служби передбачає додавання інформації маршрутизації до заголовку SOAP, дозволяючи асинхронне спілкування.

Підтипи атак:

WS-Address spoofing – Generic

Зловмисник надсилає повідомлення SOAP, що вміщує інформацію WS-Address серверу WEB-служби. Елемент <ReplyTo> не вміщує адресу зловмисника, а адресу клієнта WEB-служби котру обрав зловмисник. В такому випадку з’являється небажаний трафік що призводить до відмови служби.

WS-Address spoofing – BPEL Rollback

Зловмисник надсилає повідомлення що вміщує елемент <ReplyTo> з підробленою адресою. Обробляючи повідомлення SOAP двигун BPEL намагається

викликати адресу що підроблена. Всі спроби будуть марними, а повернені запити, оброблятимуться двигуном результатом буде перенавантаження та відмова.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє захищений заголовок та зашифрований елемент або підписаний елемент. Якщо WEB-служба не обробляє зашифровані повідомлення, він просто не обробить повідомлення і атака буде марною;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від таких атак може бути попереднє підтвердження адреси, перед подальшим виконанням будь-яких дій. Особливо важливо коли повідомлення обробляється двигуном BPEL. У наш час не існує стандартизованого шляху виконання даної аналізу.

XMLInjection

Під час виконання даної атаки зловмисник намагається ввести різні XMLTags в повідомленні SOAP намагаючись змінити структуру XML. Успішне виконання даної атаки може привести до виконання забороненої операції. Залежно від виконаної операції, різні захищені об’єкти можуть бути вразливими. Приклади: [4] [5]

-зміна даних оплати приведе до зазіхання на цілісність даних;

-неавторизований вхід адміністратора приведе до отримання контролю.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає метадані WEB-служби такі як WSDL;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Кожен WSDL повинен вміщувати детальний опис використаних елементів, атрибутів та типів даних. Використання вимогливої системи аналізу може надати достатній захист, але її використання передбачає великі затрати ресурсів. [7]

Іншим варіантом є використання синтаксичного аналізатора SAX. Але переходячи на нову структуру, що використовує синтаксичний аналізатор DOM, сервер стає вразливим. Це є ще однією причиною використання вимогливої системи аналізу.

XML Document Size Attack

Дана атака дуже легка до виконання. Вона націлена на обмеження придатності обраної WEB-служби.

Все починається з надсилання надвеликого повідомлення SOAP до обраної WEB-служби. Синтаксичний аналізатор намагаючись обробити документ зіштовхується з недостачею пам’яті.

Підтипи атак:

Oversized SOAP Header

Випадок атаки, коли контент вміщується у заголовку повідомлення SOAP

Oversized SOAP Body

Випадок атаки, коли контент вміщується у тілі повідомлення SOAP

Oversized SOAP Envelope

Випадок атаки, коли контент вміщується у пакеті повідомлення SOAP, але за межами заголовка та тіла SOAP

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Найпростішим шляхом до захисту від таких атак, є перевірка розміру документа перед потраплянням його до синтаксичного аналізатора. Розмір документа повинен залежати від типу WEB-служби.

Найкращим захистом від даних атак є використання вимогливої системи аналізу. Кожен документ WSDL повинен вміщувати детальний опис використаних елементів, атрибутів та типів даних.

Використання вимогливої системи аналізу передбачає затрату ресурсів. Як би там не було її використання надає максимальний захист від таких атак.

XML Signature – Transformation DOS

Під час процесу створення підпису у повідомлені SOAP виконується кілька кроків. Один з таких кроків є перетворення даних котрі очікуються як елемент <Reference>. Специфікація XML Signature не визначає обмежень числу

перетворень даних, ні типів перетворень. Це може бути використано зловмисником для перенавантаження служби. [4] [5]

Підтипи атак:

XML Signature – C14 DOS

Атака у котрій здійснюється багаторазове зашифровування даних з використанням алгоритму C14. Алгоритм C14 є дуже вимогливим до ресурсів;

XML Signature – XSLT DOS

Атака у котрій здійснюється багаторазове перетворення даних з використанням мови Extendible Stylesheet Language Transformations;

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє підписаний елемент. Якщо WEB-служба не обробляє підписані елементи, він просто не обробить елемент <signedinfo> і атака буде марною; [9] [13]
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом є наявність у додатку вимогливої системи до дозволених перетворень. Загалом для захисту від атаки XML Signature – C14 DOS потрібно визначити ліміт числа перетворень.

Для захисту від атаки XML Signature – XSLT DOS перетворення XSLT повинні бути заборонені. Якщо ж вони необхідні, переконайтеся у перевірці підписів даних перетворень, щоб підписи були тільки з довірених джерел. Також можливе використання білого списку дозволених перетворень.

XML ExternalEntity DOS

Стандарт XML дозволяє використання документів DTD. DTD визначають легальні блоки документа XML. Однією з рис документів DTD є можливість використання сутностей. Такі сутності можуть використовувати ярлики до рядків спеціальних значень. Звичайним прикладом зумовлених сутностей є ті, що використовуються у середині HTML. При бажанні використати “<” або “>” за межами HTMLTags, вони повинні бути замінені наступним шляхом: [4]

Символ “>” матиме сутність “&gt”;

Символ “<” матиме сутність “&lt”;

Сутності, що не є зумовленими можуть бути оголошеними внутрішньо або зовнішньо:

Внутрішнє оголошення – сутність визначена у тому самому документі;

Зовнішнє оголошення – сутність визначена у документі за межами даного. Надається тільки посилання.

Зовнішні сутності, що знаходяться у повідомленні SOAP, можуть викликати відмову служби представивши злоякісний контент під час процесу синтаксичного аналізу.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом є використання стандарту SOAP 1.2. Якщо ви не впевнені у правильному функціонуванні SOAP 1.2, перед синтаксичним аналізом робіть ручну перевірку на наявність відкриваємого DTDTag.

XML EntityExpansion

Стандарт XML дозволяє використання документів DTD. DTD визначають легальні блоки документа XML. Однією з рис документів DTD є можливість використання сутностей. Такі сутності можуть використовувати ярлики до рядків спеціальних значень. Звичайним прикладом зумовлених сутностей є ті, що використовуються у середині HTML. При бажанні використати “<” або “>” за межами HTMLTags, вони повинні бути замінені наступним шляхом:

Символ “>” матиме сутність “&gt”;

Символ “<” матиме сутність “&lt”;

Сутності, що не є зумовленими можуть бути оголошеними внутрішньо або зовнішньо:

Внутрішнє оголошення – сутність визначена у тому самому документі;

Зовнішнє оголошення – сутність визначена у документі за межами даного. Надається тільки посилання.

Сутності, що знаходяться у повідомленні SOAP, можуть використовуватися у атаках, котрі направляють усі системні ресурси на створення великих структур пам’яті.

Підтипи атак:

XML Generic Entity Expansion

Атака у котрій здійснюється використання великого числа посилань на документи великого обсягу.

XML Recursive Entity Expansion

XML Bomb котра виконується за допомогою експоненціального збільшення документа.

XML Remote Entity Expansion

Атака у котрій зловмисник визначає сутність котра посилається на наступну сутність і так далі. Обробник не може зупинитися поки обробить усі сутності.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;

- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом є використання стандарту SOAP 1.2. Якщо ви не впевнені у правильному функціонуванні SOAP 1.2, перед синтаксичним аналізом робіть ручну перевірку на наявність відкриває мого DTDTag.

XML Entity Reference Attack

Стандарт XML дозволяє використання документів DTD. DTD визначають легальні блоки документа XML. Однією з рис документів DTD є можливість використання сутностей. Такі сутності можуть використовувати ярлики до рядків спеціальних значень. Звичайним прикладом зумовлених сутностей є ті, що використовуються у середині HTML. При бажанні використати “<” або “>” за межами HTMLTags, вони повинні бути замінені наступним шляхом:

Символ “>” матиме сутність “&gt”;

Символ “<” матиме сутність “&lt”;

Сутності, що не є зумовленими можуть бути оголошеними внутрішньо або зовнішньо:

Внутрішнє оголошення – сутність визначена у тому самому документі;

Зовнішнє оголошення – сутність визначена у документі за межами даного. Надається тільки посилання.

Сутності, що знаходяться у повідомленні SOAP, можуть використовуватися у атаках, котрі викривають дані, що знаходяться на обраній службі.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;

- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі

Контрзаходи

Захистом є використання стандарту SOAP 1.2. Якщо ви не впевнені у правильному функціонуванні SOAP 1.2, перед синтаксичним аналізом робіть ручну перевірку на наявність відкриває мого DTDTag.

XML Flooding

Атака виконується шляхом надсилання великої кількості законних повідомлень SOAP. Цю атаку можна порівняти з класичною атакою відмови

служби, котра виконується шляхом надсилання великої кількості законних запитів HTTP.

Підтипи атак:

Single XML Flooding атака у котрій усі запити надсилаються від зловмисника з його місця знаходження. [4] [9] [13]

Distributed XML Flooding така у котрій багато різних користувачів WEB-служби надсилають запити, частіше за все вони контролюються зловмисником.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Розглядаючи захист від даного типу атак важливо передбачати обидва:

Single XML Flooding – передбачається шляхом обмеження кількості можливих запитів що надсилаються з однієї адреси за визначений проміжок часу. [4]

Distributed XML Flooding – не має рішення на сьогоднішній день

XML Signature – KeyRetrieval DOS

Використання XML Signature передбачає використання ключів що передаються між сторонами обміну. Дані ключі зазвичай наперед відомі обом сторонам, але також передбачений варіант обміну ними під час пересилки повідомлення. Спосіб представлення ключа визначений у захищеному заголовку SOAPв середині елемента <KeyInfo>. Існують різні шляхи представлення ключа. Одним з варіантів є використання <RetrevialMethod>як дочірню елемента <KeyInfo>. <RetrevialMethod> вміщує посилання на ключ куди, що знаходиться десь у документі. Це відкриває шлях до спричинення відмови служби шляхом посилання на себе та створення нескінченної петлі. [4] [5]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє захищений заголовок та підписаний елемент. Якщо WEB-служба не обробляє підписані елементи, вона просто не обробить елемент <Signedinfo> та <KeyInfo> і атака буде марною; [9] [13]

- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від даної атаки може бути відмова від використання елемента <RetrievalMethod>. Це передбачає ручну перевірку на наявність його у повідомленні SOAP. При його наявності повідомлення не обробляється. Якщо є необхідність, слід обмежити його підтримку у роботі служби.

XMLSignature – KeyRetrievalXSA (CrossSiteAttack)

Використання XML Signature передбачає використання ключів що передаються між сторонами обміну. Дані ключі зазвичай наперед відомі обом сторонам, але також передбачений варіант обміну ними під час пересилки повідомлення. Спосіб представлення ключа визначений у захищеному заголовку SOAP в середині елемента <KeyInfo>. Існують різні шляхи представлення ключа. Одним з варіантів є використання URI посилання на ключ. Такі посилання можуть викликати проблеми, особливо якщо представлені дані повертаються зловмиснику котрий надіслав запит. [4] [5]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;

- Зловмисник знає що WEB-служба обробляє захищений заголовок та підписаний елемент. Якщо WEB-служба не обробляє підписані елементи, він просто не обробить підписаний елемент і атака буде марною; [9] [13]

- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Заборонивши зовнішні посилання при представленні ключів, можна передбачити такі атаки. Якщо їх використання обов’язкове, рішенням є обмеження підтримки використовуючи білі списки.

XML Signature – XLST Code Execution

Під час процесу створення підпису у повідомлені SOAP виконується кілька кроків. Один з таких кроків є перетворення даних котрі очікуються як елемент <Reference>. Зазвичай операція перетворення виконується тільки для обраного набору даних. Специфікація XMLSignature не визначає обмежень числу перетворень даних та типів перетворень. Один з типів перетворень дозволений специфікацією XMLSignature є перетворення XSLT. XSLT (Extendible Stylesheet Language Transformations) незалежна потужна мова перетворення документів XML.

Таким чином зловмисник може зробити перетворення котрі задіють додатковий код. [4] [5]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє підписаний елемент. Якщо WEB-служба не обробляє підписані елементи, він просто не обробить підписаний елемент і атака буде марною; [9] [13]
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом від таких атак є заборона перетворень XLST. Якщо їх використання обов'язкове, упевнитися у виконанні підпису з цими перетвореннями тільки з довірених джерел.

Іншим рішенням є використання білого списку перетворень. При розбіжностях перетворення не обробляється.

Атаки за властивістю "цілісність"

Active WS-MITM

Active WS-MITM (Active WEB Service – Man in the middle) атака під час котрої зловмисник змінює дані що пересилаються між WEB-службою та клієнтом. Зловмисник змінює повідомлення SOAP під час передачі, тим самим зазіхаючи на властивість "цілісність". [5] [9] [13]

В основному WEB-служби засновані на технологіях TCP/IP. Саме це і відкрило дорогу даному типу атак.

З'явився раніше невідомий кут атак на WEB-служби. Запит WEB-служби проходить крізь довільну кількість посередніх WEB-служб на шляху до точки призначення. Якщо зловмисник контролює хоча б одну посередню WEB-службу, цього достатньо для заміни запиту SOAP.

Підтипи атак:

Malicious Morphing атака у котрій зловмисник перехоплює повідомлення SOAP, змінює дані та пересилає його адресату.

Routing Detour атака у котрій зловмисник перехоплює повідомлення SOAP та змінює заголовок SOAP додаючи нові дані в інформацію маршрутизації.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник має доступ до посередньої WEB-служби, котра приймає участь у передачі повідомлень між обраним клієнтом WEB-служби та сервером.

Контрзаходи

Захистом є шифрування важливих даних. Підписання важливих частин повідомлення SOAP.

Для запобігання атаки Routing Detour повинні бути виконані особливі дії. Уся інформація щодо маршрутизації повинна підписуватися і перевірятися. Схема маршрутизації відома усім посереднім вузлам.

Metadata spoofing

Перед тим як клієнт WEB-служби зможе взаємодіяти з іншими службами він повинен отримати інформацію як викликати обрану службу. До цієї інформації належать адреса WEB-служби, формат повідомлень, необхідні параметри та вимоги безпеки. Усі дані зберігаються у документі котрий передається викликаною службою. Зазвичай це документ WSDL або WS-Security-Policy. Зловмисник замінює документ на свій злоякісний. В свою чергу документ розповсюджується клієнтам WEB-служби. [7]

Підтипи атак:

WSDL Spoofing атака у котрій зловмисник замінює контент документа WSDL.

WS Security Policy Spoofing атака у котрій зловмисник замінює контент документа WSDL. Зазвичай це відображається на вимогах безпеки. В результаті спілкування між WEB-службами відбувається не зашифрованим, зловмисник отримує змогу передивлятись повідомлення. [7] [9] [13]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник має доступ до файлу котрий він хоче замінювати;
- Зловмисник має змогу розповсюджувати повідомлення між клієнтами WEB-служби, котрі намагаються її викликати.

Контрзаходи

Захист може бути перевіркою документа клієнтом, котрий його отримує. У наш час немає стандартизованого методу аналізу аутентифікації. Розробники WEB-служби повинні приводити свої рішення такі, як підпис документів.

XML SignatureWrapping [4]

WEB-служби надають неймовірну гнучкість, коли мова йде про впровадження рис цілісності. Зазвичай для забезпечення цілісності повідомлення SOAP, важливі частини повідомлення підписуються.

Уявімо, що клієнт WEB-служби посилає підписане повідомлення адресованій WEB-служби. За ідеальних умов будь-які злоякісні зміни у повідомленні виявляються WEB-службою, якщо звісно зловмисник не має змоги зламати алгоритм шифрування. Виконуючи таку атаку зловмисник змінює підписану частину без порушення цілісності підпису.

Підтипи атак:

XML SignatureWrapping – SimpleContext

Атака у котрій підписані дані знаходяться у тілі повідомлення;

XML SignatureWrapping – Optional Element

Атака у котрій підписані дані знаходяться у заголовку повідомлення;

XML SignatureWrapping – Optional element in Security Header

Атака у котрій підписані дані знаходяться у SecurityHeader.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає що WEB-служба обробляє захищений заголовок та підписаний елемент. Якщо WEB-служба не обробляє підписані елементи, він просто не обробить підписаний елемент і атака буде марною; [9] [13]
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Для захисту від таких атак потрібний комплексний підхід у питанні безпеки, та прорахування кожної із них.

Атаки за властивістю “конфіденційність”

Attack Obfuscation

Не являє собою атаку, але визначає техніки для приховування атаки від компонентів що націлені на її викриття. Наприклад, при виконанні атаки DOS такої як CoerciveParsing, вимоглива система аналізу без зусиль її зупинить. Але, беручи до уваги той факт що перевірка виконується раніше за розшифрування, якщо така атака буде зашифрованою, вона може бути успішною. [9] [13]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.
- Зловмисник повинен бути упевненим що дана техніка працює на обраній WEB-службі.

Контрзаходи

Важко знайти рішення даній загрозі. Воно завжди залежить від атаки, що схована даною технікою.

Навіть якщо перевірка виконується до розшифрування, вимоглива система аналізу також повинна розповсюджуватися на розшифровані дані.

Кожне з використовуваних джерел повинно виконувати розшифрування та перевірку крок за кроком.

Passive WS-MITM

Атака під час котрої зловмисник читає дані, що пересилаються між клієнтом WEB-служби та WEB-службою одержувачем. Отримуючи доступ до даних котрі не призначені зловмиснику він зазіхає на критерій безпеки “конфіденційність”.

В основному WEB-служби засновані на технологіях TCP/IP. Саме це і відкрило дорогу даному типу атак.

З’явився раніше невідомий кут атак на WEB-служби. Запит WEB-служби проходить крізь довільну кількість посередніх WEB-служб на шляху до точки призначення. Якщо зловмисник контролює хоча б одну посередню WEB-службу, цього достатньо для читання запиту SOAP. [5]

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник має доступ до посередньої WEB-служби котра приймає участь у передачі повідомлень між обраним клієнтом WEB-служби та сервером.

Контрзаходи

Захистом може бути використання шифрування важливих даних. В такому випадку будь-який сенс такої атаки втрачається. Зловмисник дізнається тільки те, що повідомлення було відправлене.

WSDL Disclosure

Існують приховані WEB-служби про котрі відомо тільки людям або працівникам котрі працюють в області їх застосування. Більшості суспільства, а так і потенційних зловмисників не знають про ці WEB-служби, що значно зменшує ризик атак. Зазвичай область застосування таких WEB-служб передбачає виконання ними критично важливих операцій, що робить їх дуже привабливими для зловмисників.

Нажаль безпекою більшості з таких WEB-служби є їхня прихованість від суспільства. Вони обмінюються інформацією без будь якого захисту.

Атаки даного типу націлені на знаходження даних WEB-служб шляхом викриття файлу WSDL даної WEB-служби.

Підтипи атак:

WSDL Google Hacking

Атака у котрій використовується пошукова функція Google для пошуку файлів з розширенням .wsdl. У результаті пошуку міститься адреса URL до даного WSDL файлу, що викриває його місце розташування.

WSDL Enumeration

Атака у котрій для виконання зловмисник повинен знати кінцеву точку WEB-служби, як до неї дістатися та іншу інформацію з файлу WSDL. Зловмисник робить запити різними методами, аналізуючи відповідь він може визначати чи використовуються обраний метод. Знайшовши підходящий він отримує доступ до усіх необхідних параметрі, шляхом аналізу помилок.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захист не повинен покладатися на скритність файлу WSDL. Захист WEB-служби повинен складатися з усіх його можливих методів. Після правильного налаштування жодних проблем з безпекою не повинно виникати. [7]

Даний метод можна порівняти з криптографічним алгоритмом, ефективність котрого не повинна не залежить від його прихованості.

Атаки за властивістю “спостережність”

XpathInjection

Мова XPath котра використовується для запиту обраних частин документа XML. Її можна порівняти до мови SQL котра використовується для запиту баз даних.

У деяких випадках параметри у тілі SOAP використовуються для вводу у запит xpath. Зловмисник може змінювати запит XPath як він забажає. У випадку успішної атаки зловмисник отримає змогу прочитати зовнішній документ XML котрий викликається. [4] [5]

Атаки такого типу вважаються більш небезпечними ніж SQLInjection, враховуючи що документи XML не мають жодного контролю доступу.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник знає метадані такі як, .WSDL WEB-служби;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захистом є перевірка кожного вводу користувача в запит XPath. Продумавши кожний варіант, заборонивши усі можливі символи.

ReplayAttack

Зловмисник відтворює процес входу в систему, далі намагається його відтворити у іншій системі.

Перед виконанням даної атаки зловмисник повинен отримати доступ до повідомлення SOAP, котре вміщує у себе відомості про вхід. Є декілька способів досягнення цього:

- Зловмисник контролює посередній вузол між відправником та одержувачем;

- Зловмисник має доступ до комп’ютера жертви;
- Зловмисник використовує технології перехоплення повідомлень засновані на TCP/IP.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник має змогу перехопити повідомлення SOAP котрі пересилаються між клієнтом WEB-служби та одержувачем;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Захист від такої атаки передбачає собою створення випадкової інформації для кожного процесу входу в систему. Якщо кожна зі сторін упевниться у унікальності інформації для кожного разу входу в систему дані атаки будуть неможливими.

SOAPAction Spoofing

Кожен запит до WEB-служби вміщує операцію котра пізніше виконується логікою служби. Ця інформація знаходиться у першому дочірньому елементі тіла SOAP. Якщо для передачі повідомлення SOAP використовується HTTP у стандарті SOAP дозволяється додати елемент у заголовок HTTP так званий SOAPAction. Цей елемент містить назву виконуваної операції. Це виконується з метою інформування WEB-служби про операцію що вміщується у тілі SOAP, звільнюючи від виконання синтаксичного аналізу. Така оптимізація може спричинити до виконання атак зловмисником. [5]

Підтипи атак:

SOAPAction Spoofing – MITM Attack

Атака під час котрої зловмисник перехоплює повідомлення, замінює операцію що вкладена.

SAOPAction Spoofing – Bypass Attack

Атака під час котрої використовується дозволена операція у тілі повідомлення, а після аналізу виконується операція з елемента котра є злоякісною.

Передумови для атаки:

Для виконання такої атаки зловмисник повинен мати наступні дані:

- Зловмисник повинен знати кінцеву точку WEB-служби, у іншому випадку він не має можливості дістатися WEB-служби;
- Зловмисник повинен мати доступ до файлу .WSDL;
- Зловмисник може дістатися до кінцевої точки з його місця знаходження. Ця передумова важлива у випадку доступності WEB-служби тільки для користувачів визначеної мережі.

Контрзаходи

Якщо не має необхідності у SOAPAction, він повинен бути відключеним. Якщо ж необхідно, операція у середині SOAPAction та тіло SOAP повинні обов’язково перевірятися перед виконанням операції. Будь-які розбіжності повинні розцінюватися як атака. [5] [9] [13]

Висновок до Розділу 2

Даний розділ показує нам комплексність поняття безпеки WEB-служби. Розробник при створенні повинен виконати усі контрзаходи зазначені вище. Але навіть після цього не можна до кінця бути впевненим у безпеці WEB-служби.

Архітектура SOA передбачає собою взаємодію кількох WEB-служби, якщо ж одна з них не захищена, усі хто з нею взаємодіють наражаються на небезпеку. Привівши класифікацію атак по властивостям захищеності інформації, що приведені СБУ, були систематизовано представлені дані, котрі допомагають зрозуміти якого характеру є виділена атака, та які контрзаходи треба виконати для її запобігання.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Визначення функціоналу системи аналізу захищеності WEB-служб

Першочергово необхідно визначитися з областю, яку охоплює система. Предметною областю даної системи являється WEB-служби. Так, як дана область надто велика, вона була зменшена за рахунок більш точного визначення теми, а саме до механізму взаємодії, використовуваних протоколів та вразливостей.

В результаті дана система увібрала до себе ієрархічно впорядковані знання, котрі вміщують інформацію щодо розвитку технології WEB-служб, схеми взаємодії, алгоритму створення, класифікацію атак та контрзаходи. Додатково, до цієї інформаційно-довідкової системи був розроблений засіб для перевірки захищеності WEB-служб.

Зібравши знання, систематизувавши їх та розробивши засіб постала необхідність розробити єдину систему. Рішенням даного питання було створення WEB-сайту з розміщеними на його сторінках відомостями по перерахованим питанням. На головній сторінці розміщений засіб перевірки захищеності.

Поставлена мета була досягнута розробкою засобу котрий перевіряє захищеність WEB-служб за допомогою набору атак, котрі є найпопулярнішими. Даний засіб може легко використовуватися. Єдине що користувач повинен мати – це IP-адреса обраної для перевірки WEB-служби. Ввівши адресу у форму, яка розміщена на головній сторінці системи, та натиснувши кнопку «запуск» запускається засіб перевірки. Після цього відкривається сторінка завантаження. Виконавши перевірку засіб формує звіт, який буде представлений користувачу у вигляді HTML файлу у вікні браузеру.

У звіті буде представлена інформація щодо виявлених вразливостей. Дана інформація включає у себе кількість виявлених вразливостей, навпроти кожного типу атак буде написана їх кількість. Поряд з кількістю вразливостей буде представлене посилання на сторінку інформаційно-довідкової системи, на котрій розміщена інформація про даний тип атак. Натиснувши на тип атаки, ви перейдете до початку переліку вразливостей по даній атаці. У цьому переліку буде представлена адреса файлу, в якому виявлена вразливість та URL адреса по котрій дана вразливість може бути використана.

Окрім функції перевірки захищеності WEB-служб дана система вміщує систематизовану інформацію щодо WEB-служб. Після виконання перевірки, або при іншій потребі ви маєте змогу відшукати інформацію стосовно функціонування або вразливостей WEB-служб у формі пошук, котра реалізована на головній сторінці. Етапами сумісного процесу використання засобу користувачем та роботи засобу є:

- визначення IP-адреси користувачем;
- введення IP-адреси у форму та запуск користувачем;
- послідовне виконання перевірок засобом;

- формування звіту та представлення звіту користувачу;
- перехід до обраних даних по посиланням користувачем, або завершення роботи з засобом.

Отже, дана система може використовуватись для перевірки та подальшого аналізу захищеності WEB-служб. Вона буде відповідати наступним вимогам:

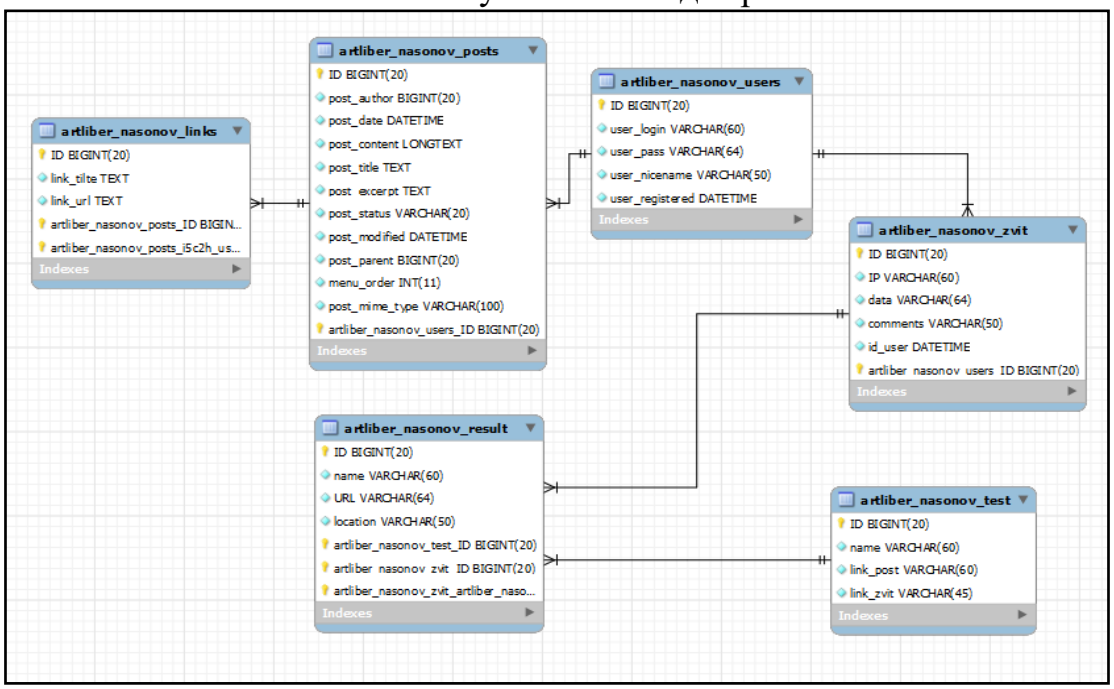
- актуальність даних, що надаються;
- повнота даних, що надаються;
- ергономічність та інтуїтивна зрозумілість інтерфейсу;
- портативність (у випадку, коли система буде не доступною у мережі Інтернет).

3.2 Фізична реалізація бази даних системи аналізу захищеності WEB-служб

Першим кроком проектування даної системи є визначення її призначення. Тобто чітке окреслення області що вона охоплює. Після чого визначаються вимоги до системи в ході її експлуатації. Далі проводиться детальне проектування системи, наприклад проектування об’єктів даних, проектування екранних форм, тощо. Наступним кроком обирається модель життєвого циклу. І останнім є створення структури сайту.

Завданням даної системи є перевірка WEB-додатку на захищеність, представлення інформації користувачу, та збереження інформації системи та результатів проведених перевірок.

Рисунок 3.1 ER-діаграма



Тож вважаю за доречне привести пояснення стосовно рядків таблиць:
artliber_nasonpv_posts:

- ID – порядковий номер рядка
- post_author – номер користувача який виклав новину(1- адміністратор)
- post_date – час релізу новини
- post_content – інформація яка відображається на сторінці
- post_title – заголовок посту
- post_modified – час останнього редагування

artliber_nasonov_users:

- user_login – логін користувача
- user_pass – пароль користувача(зашифрований)
- user_nicname – нікнейм користувача
- user_registered – час та дата реєстрації

artliber_nasonov_links:

- link_title – назва вразливості
- link_url – посилання на інтернет джерело

artliber_nasonov_zvit:

- IP – адреса перевіреної WEB-служби
- data – дата аналізу WEB-служби
- comments – коментарі до звіту
- Id_user – номер користувача, який виконав перевірку

artliber_nasonov_result

- name – ім’я звіту
- URL – посилання представлені в звіті
- location – ім’я файлів в яких виявлені вразливості

artliber_nasonov_test

- name – ім’я виконуваної аналізу
- link_posts – посилання до сторінки і інформацією
- link_zvit – посилання до звіту

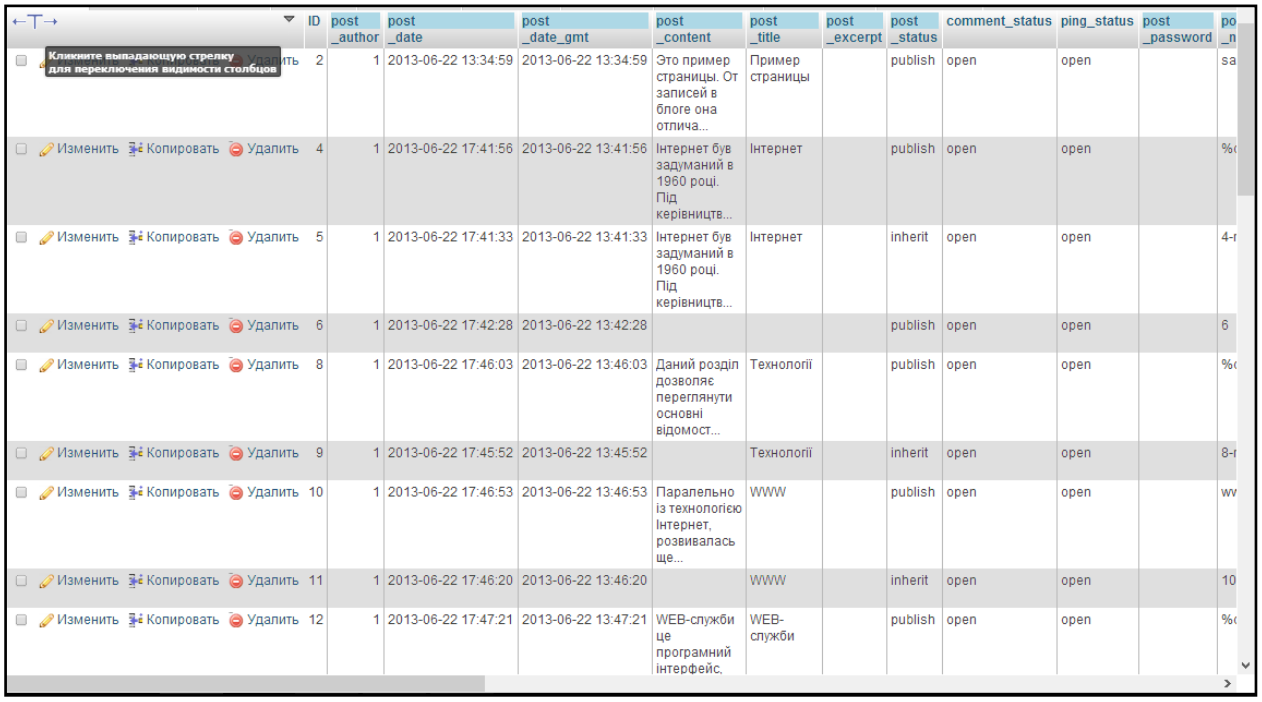
Для зручного використання даної системи користувачами, інформація повинна бути представлена систематизовано та логічно. Перейдемо до реалізації бази даних. Код створення таблиць бази даних представлений у ДОДАТКУ А.

В даній роботи використовується база даних MySQL 5.5.30. Доступ до неї можна отримати через http://localhost/Tools/phpmyadmin або через гіперпосилання у XAMPP.

Для зручного керування базою використовується phpMyAdmin. У інтерфейсі ви можете побачити список створених баз даних. У даній роботі використовується artliber_nasonov.

Натиснувши на будь-яку з них можна дізнатися які дані їй підпорядковуються. Головною таблицею бази даних інформаційно-довідкової системи я вважаю artliber_nasonov_posts. Дана таблиця зберігає інформацію про документи, що розміщені на сторінках системи.

Рисунок 3.2 Сторінки бази даних



ID	post _author	post _date	post _date_gmt	post _content	post _title	post _excerpt	post _status	comment_status	ping_status	post _password	po _n
Изменить Копировать Удалить 2	1	2013-06-22 13:34:59	2013-06-22 13:34:59	Это пример страницы. От записей в блоге она отлича...	Пример страницы		publish	open	open		sa
Изменить Копировать Удалить 4	1	2013-06-22 17:41:56	2013-06-22 13:41:56	Інтернет був задуманий в 1960 році. Під керівництв...	Інтернет		publish	open	open		%(
Изменить Копировать Удалить 5	1	2013-06-22 17:41:33	2013-06-22 13:41:33	Інтернет був задуманий в 1960 році. Під керівництв...	Інтернет		inherit	open	open		4-r
Изменить Копировать Удалить 6	1	2013-06-22 17:42:28	2013-06-22 13:42:28				publish	open	open		6
Изменить Копировать Удалить 8	1	2013-06-22 17:46:03	2013-06-22 13:46:03	Даний розділ дозволяє переглянути основні відомост...	Технології		publish	open	open		%(
Изменить Копировать Удалить 9	1	2013-06-22 17:45:52	2013-06-22 13:45:52		Технології		inherit	open	open		8-r
Изменить Копировать Удалить 10	1	2013-06-22 17:46:53	2013-06-22 13:46:53	Паралельно із технологією Інтернет, розвивалась ще...	WWW		publish	open	open		ww
Изменить Копировать Удалить 11	1	2013-06-22 17:46:20	2013-06-22 13:46:20		WWW		inherit	open	open		10
Изменить Копировать Удалить 12	1	2013-06-22 17:47:21	2013-06-22 13:47:21	WEB-служби це програмний інтерфейс,	WEB-служби		publish	open	open		%(

Кликните выпадающую стрелку для переключения видимости столбцов

3.3 Реалізація системи аналізу захищеності WEB-служб

3.3.1 Реалізація інформаційно-довідкової підсистеми

Створивши базу даних, перейдемо до практичної реалізації. Взявши до уваги всі вимоги до даної системи, на сонові створеної бази даних був розроблений ергономічний інтерфейс.

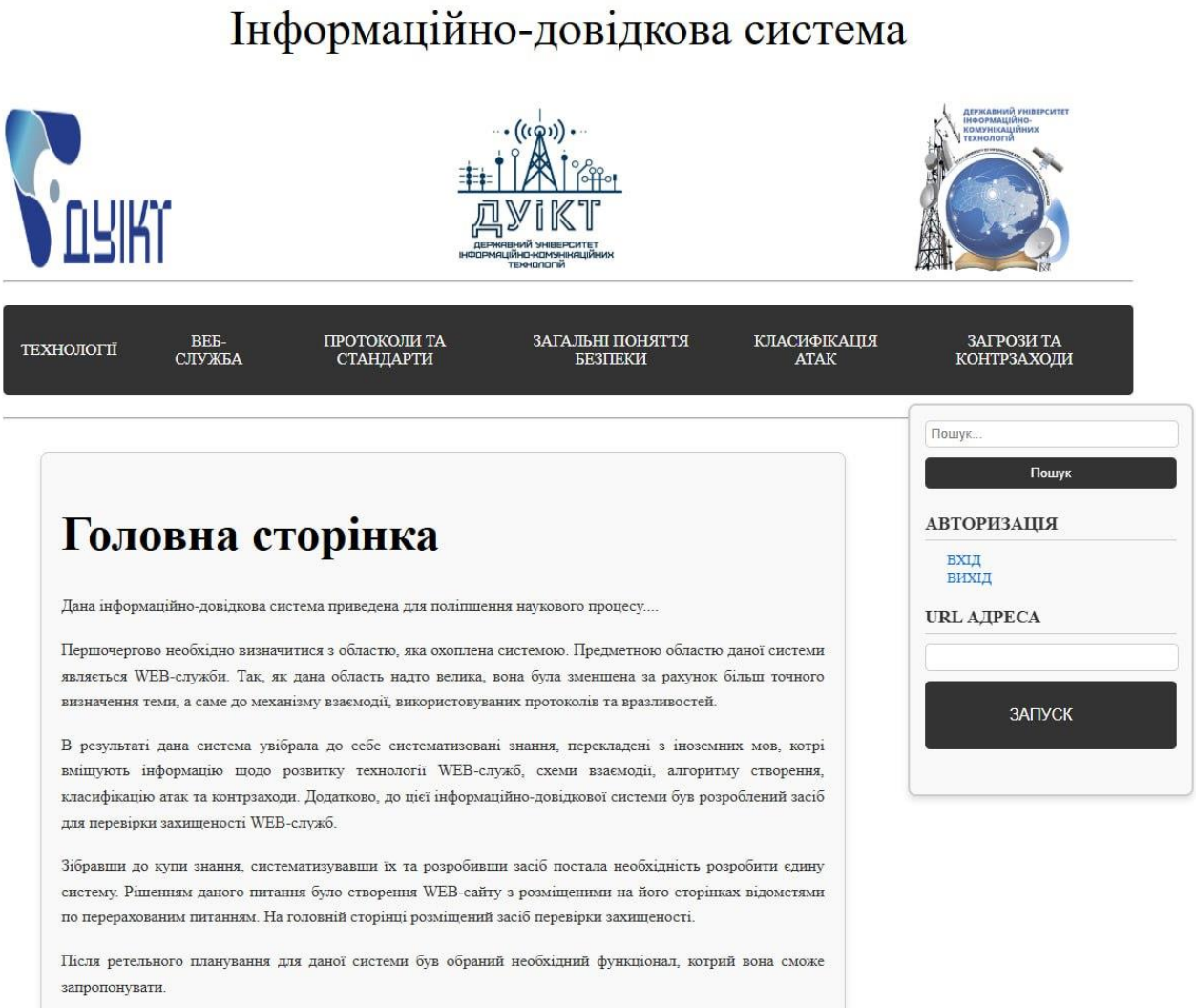


Рисунок 3.3 Головна сторінка інформаційно-довідкової системи

До даної системи було включено відомості про:

- технології котрі стали основою створення WEB-служб;
- відомості про схему взаємодії, алгоритм створення WEB-служб;
- стандарти та протоколи взаємодії WEB-служб;
- загальні поняття безпеки WEB-служб;
- класифікація атак що можуть бути виконані на WEB-служб;

- перелік загроз, передумови та контрзаходи.

Зручність інтерфейсу обумовлена кілька рівневим меню. Цільовим матеріалом даної системи є перелік загроз та контрзаходи до них.

Рисунок 3.4 Приклад сторінки інформаційно-довідкової системи

Тобто користувач має змогу продивитися будь-які дані представлені у системі знайшовши їх у головному меню, або використавши пошук, котрий знаходиться на головній сторінці. Даною системою може користуватися одночасно декілька користувачів.

Також передбачена система редагування даних. Доступ до неї надано лише викладачу. Увійшовши, викладач отримує доступ до будь-якого редагування інформації. Звичайний користувач має доступ лише до перегляду інформації. Дана система без великих зусиль може бути розгорненою на кафедрі Інституту, використовуючи сервер, що реалізований на кафедрі. За своїми особливостями створене рішення можна віднести до WEB-служби, технології WEB 3.0. Вирішальною рисою, є доступ до редагування інформаціїї лише привілейованим користувачам.

3.3.2 Реалізація підсистеми аналізу захищеності WEB-служб

У розділі 3.2.1 був виділений функціонал та вимоги до системи, що розроблюється. У розділі 3.2.2 був створений інтерфейс та структура інформаційно-довідкової складової даної системи.

Після розробки інформаційно-довідкової складової у вигляді WEB-сайту, необхідно створити засіб перевірки захищеності WEB-служб. Даний засіб розміщується на головній сторінці, так як, являється головною складовою даної системи.

Для зручного користування даним засобом, він був розроблений у вигляді однієї форми для запису IP-адреси та кнопки «Запуск». Дана кнопка реалізована на мові PHP, за допомогою скрипта PHP виконується запуск засобу, що реалізований на мові Python.

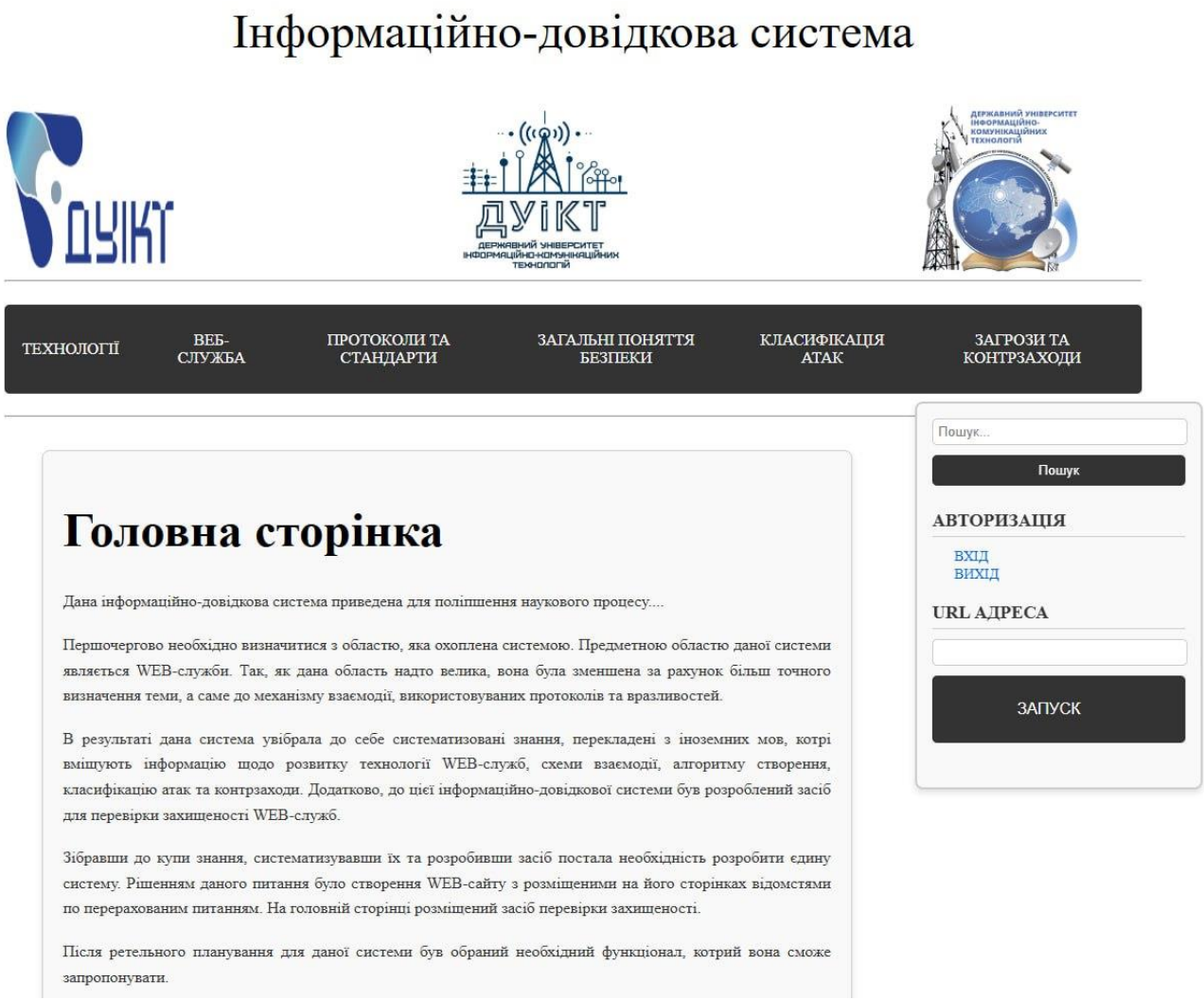


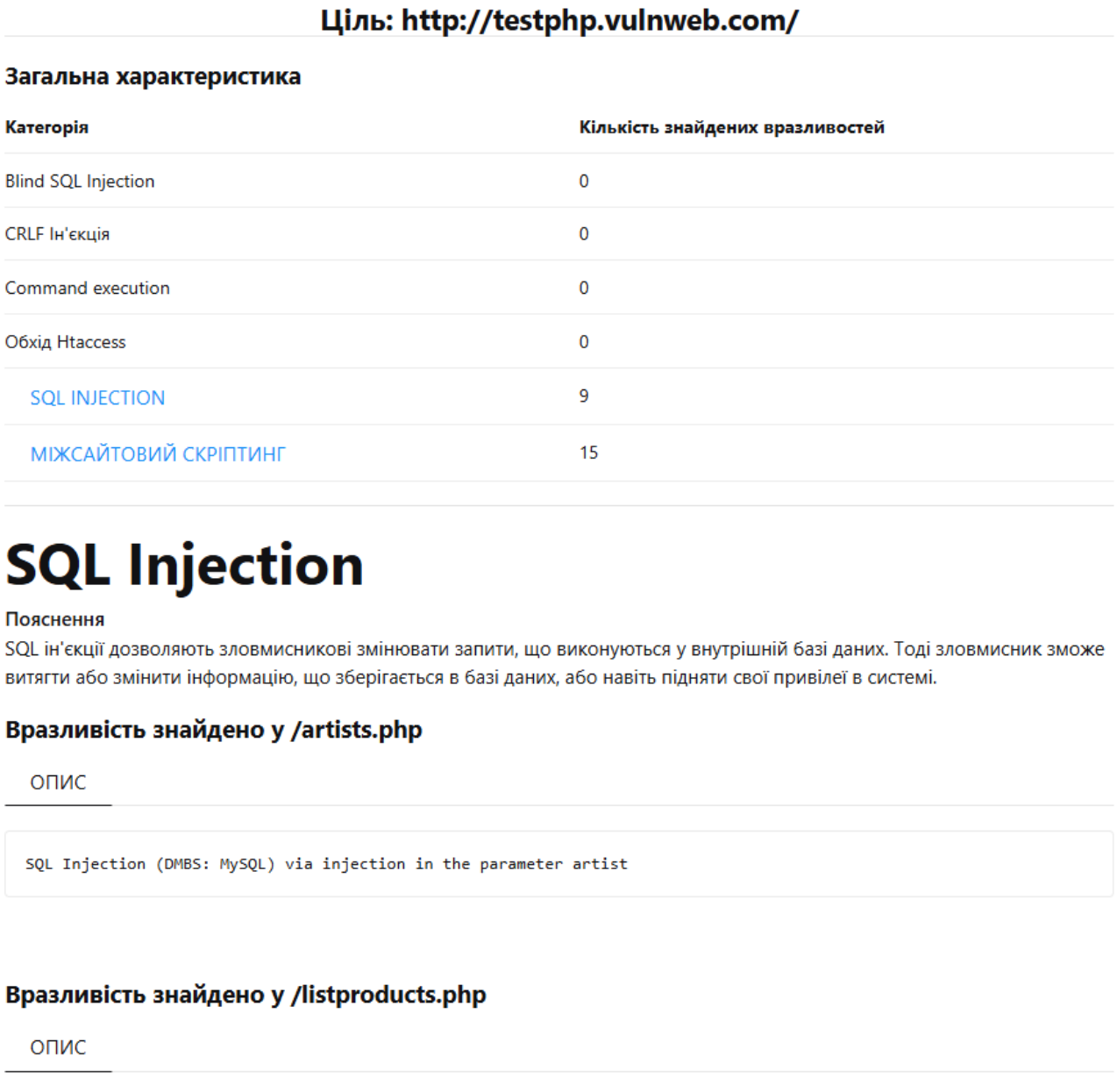
Рисунок 3.5 Головна сторінка інформаційно-довідкової системи

Скрипт Python виконує виклик функції, ціль котрої завантаження «корисного навантаження» для окремого типу перевірки. «Корисне навантаження» завантажується окремо для кожного типу перевірки, так як, воно для усіх різне.

Далі виконується запуск функції атаки, з двома варіантами використовуваних запитів. Даними запитами є GET та POST. Необхідність використання обох запитів обумовлюється виконанням функції атаки різних звернень до WEB-додатку. Так, для звернення до однієї і тої самої форми різними типами запитів може дати різний результат. Також, важливою характеристикою є кількість даних, що передаються.

Рисунок 3.6 Сторінка завантаження

Після запуску функції по одному з типів запитів, запускається перевірка по одному з визначених типів атак. Після виконання відправки запиту по створеній адресі, WEB-додаток, що перевіряється, повертає відповідь на даний запит. Дана відповідь обробляється у блоці перевірки відповідей. Залежно від отриманої відповіді, вона або зараховується як успішно проведена, або не враховується. Останнім етапом роботи засобу, є формування звіту по виявленим вразливостям.



Ціль: http://testphp.vulnweb.com/

Загальна характеристика

Категорія	Кількість знайдених вразливостей
Blind SQL Injection	0
CRLF Ін'єкція	0
Command execution	0
Обхід Htaccess	0
SQL INJECTION	9
МІЖСАЙТОВИЙ СКРІПТИНГ	15

SQL Injection

Пояснення

SQL ін'єкції дозволяють зловмисникові змінювати запити, що виконуються у внутрішній базі даних. Тоді зловмисник зможе витягти або змінити інформацію, що зберігається в базі даних, або навіть підняти свої привілеї в системі.

Вразливість знайдено у /artists.php

ОПИС

```
SQL Injection (DMBS: MySQL) via injection in the parameter artist
```

Вразливість знайдено у /listproducts.php

ОПИС

Рисунок 3.7 Сторінка представленого звіту

Звіт представляється користувачу у вигляді WEB-сторінки. Він відчиняється у вікні браузера після виконання перевірок. Можемо побачити типи виконаних перевірок, кількість знайдених вразливостей, посилання на сторінки інформаційно-довідкової системи, перелік знайдених вразливостей з URL по котрим вони можуть бути реалізовані. Натиснувши на перевірку у колонці тип, посилання перенесе вас на початок переліку знайдених вразливостей по даному типу атак.

Висновки до розділу 3

У даному розділі розроблена Інформаційно-довідкова система, котра може бути використана для покращення процесу навчання та аналізу захищеності WEB-служб. Дана система поєднує у собі дві складові:

- систему знань про WEB-служби
- засіб перевірки захищеності WEB-служб

До неї зібрані загальні відомості про технології, що використовуються у WEB-службах. Також, приведений перелік атак на WEB-служби, їх передумови та контрзаходів. Але, головною складовою є засіб перевірки захищеності WEB-служб.

ВИСНОВОК

В ході даної роботи вирішені наступні задачі:

- виконано аналіз технологій на котрих засновані WEB-служби;
- виконано аналіз протоколів взаємодії WEB-служб;
- виконано аналіз існуючих загроз WEB-служб;
- систематизовано знання про різновиди, передумови та контрзаходи до атак на WEB-служби;
- розроблено Інформаційно-довідкову систему.
- розроблено засіб для перевірки захищеності WEB-служб.

На підставі виконаного аналізу можна зробити висновки, що безпека WEB-служби є комплексним поняттям. Захищати WEB-службу потрібно не лише з точки зору протоколів взаємодії, а також й інших компонентів.

В ході даної роботи ознайомившись з нормативними документами з питань запровадження технології Інтернет, можна дійти до висновку у скорому запровадженні технології Інтернет у всіх можливих сферах життєдіяльності.

Ознайомившись з нормативним документом СБУ було про класифіковано атаки за властивостями захищеності даних.

Проаналізувавши протоколи та стандарти за допомогою котрих розробляються та взаємодіють WEB-служби можна зрозуміти потужність та гнучкість даної технології.

Після розгляду та систематизації існуючи знань про атаки, передумови та контрзаходи, стало очевидним що більшість атак є так званими DOS атаками.

На базі даного матеріалу була розроблена Інформаційно-довідкова система для поліпшення навчального процесу, з ергономічним інтерфейсом у вигляді сайту.

Поставлена мета досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IJCSNS International Journal of Computer Science and 154 Network Security, VOL.6 No.1B
2. Welcome to WS-Attacks.org! http://clawslab.nds.rub.de /index.php/Main_Page
3. Безпека WEB-сервісів http://www.iso.ru/print /rus/document5972.phtml
4. XML http://bourabai.kz/xml/1.htm
5. SOAP Version 1.2 http://www.w3.org/TR/soap/
6. Фастівський Є.Г. Сервіс-орієнтовані технології інтеграції інформації 2011р. http://khpi-iip.mipk.kharkiv.edu/library/sotii/lectures/Lecture5.pdf
7. WSDL http://citforum.ck.ua/internet/webservice/wsdl_1/
8. UDDI http://www.uddi.org/pubs/uddi-tech-wp.pdf
9. Category:Attack https://www.owasp.org/index.php/Category:Attack
10. WEB Services Security v1.0 https://www.oasis-open.org/standards#wssv1.0
11. НД ТЗІ 3.7-001-99 http://info-stand.com/downloads/nd-tzi/3-7_001_99/NDTZI_3.7_001_99.pdf.
12. Category:Attack https://www.owasp.org/index.php/Category:Attack Security of WEB Service-based Applications Evangelos Markopoulos June 2012
13. WEB SERVICES VULNARABILITIES by Nishchal Bhalla, Sahba Kazerooni February 15, 2007.

ДОДАТОК А

Створення таблиць бази даних

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8 */;

CREATE DATABASE IF NOT EXISTS `artliber_nasonov` DEFAULT
CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `artliber_nasonov`;

CREATE TABLE IF NOT EXISTS `artliber_nasonov_posts` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `post_author` bigint(20) unsigned NOT NULL DEFAULT '0',
  `post_date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `post_date_gmt` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
  `post_content` longtext NOT NULL,
  `post_title` text NOT NULL,
  `post_excerpt` text NOT NULL,
  `post_status` varchar(20) NOT NULL DEFAULT 'publish',
  `comment_status` varchar(20) NOT NULL DEFAULT 'open',
  `ping_status` varchar(20) NOT NULL DEFAULT 'open',
  `post_password` varchar(20) NOT NULL DEFAULT '',
  `post_name` varchar(200) NOT NULL DEFAULT '',
  `to_ping` text NOT NULL,
  `pinged` text NOT NULL,
  `post_modified` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
  `post_modified_gmt` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
  `post_content_filtered` longtext NOT NULL,
`post_parent` bigint(20) unsigned NOT NULL DEFAULT '0',
`guid` varchar(255) NOT NULL DEFAULT '',
  `menu_order` int(11) NOT NULL DEFAULT '0',
  `post_type` varchar(20) NOT NULL DEFAULT 'post',
  `post_mime_type` varchar(100) NOT NULL DEFAULT '',
  `comment_count` bigint(20) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  FOREING KEY (`artliber_nasonov_users_ID`);

CREATE TABLE IF NOT EXISTS `artliber_nasonov_links`
`ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
`link_title` text NOT NULL,
`link_url` text NOT NULL,
PRIMARY KEY (`ID`),
```

```
    FOREING KEY (`artliber_nasonov_posts_ID`),
    FOREING KEY (`artliber_nasonov_users_ID`);

    CREATE TABLE IF NOT EXISTS `artliber_nasonov_users`
    `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
    `user_login` varchar(60) NOT NULL DEFAULT '',
    `user_pass` varchar(64) NOT NULL DEFAULT '',
    `user_nicename` varchar(64) NOT NULL DEFAULT '',
    `user_registered`  datetime  NOT  NULL  DEFAULT  '0000-00-00
00:00:00',
    `user_registered_gmt`  datetime  NOT  NULL  DEFAULT  '0000-00-00
00:00:00',
    PRIMARY KEY (`ID`);

    CREATE TABLE IF NOT EXISTS `artliber_nasonov_zvit`
    `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
    `IP` varchar(60) NOT NULL DEFAULT '',
    `data` varchar(64) NOT NULL DEFAULT '',
    `comments` varchar(50) NOT NULL DEFAULT '',
    `id_user` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
    `id_user_gmt` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
    PRIMARY KEY (`ID`),
    FOREING KEY (`artliber_nasonov_users_ID`);

    CREATE TABLE IF NOT EXISTS `artliber_nasonov_result`
    `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
    `name` varchar(60) NOT NULL DEFAULT '',
    `URL` varchar(64) NOT NULL DEFAULT '',
    `location` varchar(50) NOT NULL DEFAULT '',
    PRIMARY KEY (`ID`),
    FOREING KEY (`artliber_nasonov_zvit_ID`),
    FOREING              KEY              (`artliber_nasonov_zvit_ID
artliber_nasonov_users_ID`),
    FOREING KEY (`artliber_nasonov_test_ID`);

    CREATE TABLE IF NOT EXISTS `artliber_nasonov_test`
    `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
    `name` varchar(60) NOT NULL DEFAULT '',
    `link_post` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
    `link_post_gmt` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
    `link_zvit` varchar(60) NOT NULL DEFAULT '',
    PRIMARY KEY (`ID`);
```

ДОДАТОК Б

Фрагмент коду, в якому ведеться облік помилок

```
    def logVulnerability(self,
                                category=None,
                                level=0,
                                request=None,
                                parameter="",
                                info=""):

            peer = None
            ts = ""

          vulnerability                                                    =
self.__xmlDoc.createElement("Vulnerability")

            stLevel = None
            if level == 1:
                stLevel = "Low"
            elif level == 2:
                stLevel = "Moderate"
            else:
                stLevel = "Important"

            levelNode = self.__xmlDoc.createElement("Severity")

levelNode.appendChild(self.__xmlDoc.createTextNode(stLevel))
            vulnerability.appendChild(levelNode)

            tsNode = self.__xmlDoc.createElement("DetectionDate")

#tsNode.appendChild(self.__xmlDoc.createTextNode(ts.isoformat()))
            vulnerability.appendChild(tsNode)

            ##
            urlDetailNode = self.__xmlDoc.createElement("URLDetail")
            vulnerability.appendChild(urlDetailNode)

            urlNode = self.__xmlDoc.createElement("URL")

urlNode.appendChild(self.__xmlDoc.createTextNode(request.url))
            urlDetailNode.appendChild(urlNode)

            if peer is not None:
                peerNode = self.__xmlDoc.createElement("Peer")
                if isPeerAddrPort(peer):
                    addrNode = self.__xmlDoc.createElement("Addr")

addrNode.appendChild(self.__xmlDoc.createTextNode(peer[0]))
                    peerNode.appendChild(addrNode)
```

```
                    portNode = self.__xmlDoc.createElement("Port")

portNode.appendChild(self.__xmlDoc.createTextNode(str(peer[1])))
                    peerNode.appendChild(portNode)
                else:
                    addrNode = self.__xmlDoc.createElement("Addr")

addrNode.appendChild(self.__xmlDoc.createTextNode(str(peer)))
                    peerNode.appendChild(addrNode)
                urlDetailNode.appendChild(peerNode)

            parameterNode = self.__xmlDoc.createElement("Parameter")

parameterNode.appendChild(self.__xmlDoc.createTextNode(parameter))
            urlDetailNode.appendChild(parameterNode)

            ##

            infoNode = self.__xmlDoc.createElement("Info")
            info = info.replace("\n", "<br />")
            infoNode.appendChild(self.__xmlDoc.createTextNode(info))
            urlDetailNode.appendChild(infoNode)

            self.__addToVulnerabilityList(category, vulnerability)
```