НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп'ютерного практикума

ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ РЕАЛІЗАЦІЇ ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ, ЯКІ ВИКОРИСТОВУЮТЬ КРИПТОГРАФІЧНІ МЕХАНІЗМИ ЗАХИСТУ ІНФОРМАЦІЇ

Виконали студентки
групи ФІ-32мн
Зацаренко А. Ю.
Футурська О. В.

Перевірила:
Селюх П. В.

Київ — 2023

ЗВІТ

1.1 Мета роботи

Отримання практичних навичок побудови гібридних криптосистем.

1.2 Завдання на лабораторну роботу

Дослідити основні задачі, що виникають при програмній реалізації для серверу в системі клієнт-сервер (серверна частина). Запропонувати методи вирішення задачі контролю доступу до ключової інформації, що зберігається в оперативній пам'яті ЕОМ для різних (обраних) операційних систем. Запропонувати методи вирішення задачі контролю правильності функціонування програми криптографічної обробки інформації. Порівняти з точки зору вирішення цих задач інтерфейси Crypto API, PKCS 11.

1.3 Необхідні теоретичні відомості

У даному розділі буде наведено основні визначення та описи, що стосуються системи клієнт-сервер.

1.3.1 Модель «клієнт-сервер»

Модель клієнт-сервер описує, як сервер надає одному або декільком клієнтам доступ до ресурсів і послуг. Прикладами серверів є поштові, веб- та файлові сервери. Клієнтські пристрої, включаючи настільні комп'ютери, ноутбуки, планшети та мобільні пристрої, мають доступ до ресурсів на кожному з цих серверів. Клієнти та сервери часто мають зв'язок типу "один до багатьох що означає, що один сервер може надавати ресурси кільком клієнтам одночасно.

Коли клієнт запитує з'єднання з сервером, сервер має можливість прийняти або відхилити запит. Якщо з'єднання прийнято, сервер встановлює і підтримує з'єднання з клієнтом за певним протоколом.

Наприклад, щоб надіслати повідомлення, поштовий клієнт може вимагати SMTP-з'єднання з поштовим сервером. Потім SMTP-програма поштового сервера запросить автентифікаційну інформацію, таку як адреса електронної пошти та пароль. Сервер доставить електронний лист вказаному одержувачу, якщо облікові дані збігаються з обліковим записом на поштовому сервері.

Хоча до інтернет-серверів можуть підключатися кілька клієнтів одночасно, кожна фізична система може обробляти лише певний обсяг трафіку. Як наслідок, відомі онлайн-сервіси використовують метод, відомий як розподілені обчислення, щоб розосередити клієнтів між численними фізичними серверами. Здебільшого не має значення, до якого саме комп'ютера підключаються користувачі, оскільки всі сервери надають однакові послуги.

Клієнт-серверна мережа може бути реалізована на одній комп'ютерній системі, хоча найчастіше вона використовується в багатьох місцях. Завдяки цій технології кілька комп'ютерів або людей можуть спілкуватися та обмінюватися інформацією. Парадигма клієнт-сервер дозволяє користувачам отримувати доступ до спільної бази даних або додатків у міру того, як фірми зростають, а люди працюють разом на величезних відстанях. Це також вірно, коли люди використовують Інтернет для доступу до своїх банківських рахунків або оплати рахунків. Користувачі вводять свої запити на сервер банку, а сервер потім передає їм інформацію.

Централізований дизайн клієнт-серверної моделі спрощує захист даних за допомогою обмежень доступу, встановлених політикою безпеки, що є суттєвою перевагою. Також не має значення, чи працюють клієнти і сервер під управлінням однієї операційної системи, оскільки обмін даними відбувається за допомогою клієнт-серверних протоколів, що не залежать від платформи.

Важливим недоліком клієнт-серверного підходу є те, що якщо занадто багато клієнтів одночасно запитують дані з сервера, сервер може бути перевантажений. Крім того, що це може призвести до перевантаження

мережі, надмірна кількість запитів може спричинити відмову в обслуговуванні.

1.3.2 Архітектури моделі

Існує безліч архітектур клієнт-сервер, кожна з яких має свої переваги та недоліки. Нижче наведено деякі з найпоширеніших архітектур клієнт-сервер:

1) Однорівнева архітектура: Однорівнева архітектура – це базова програма, яка працює на одному комп'ютері без доступу до мережі. Запити користувача не контролюють жодних мережевих протоколів; отже, код є простим, а мережа позбавлена надлишкового трафіку.

2) Дворівнева архітектура: Дворівнева архітектура складається з клієнта, сервера та протоколу, що з'єднує ці два рівні. Код графічного інтерфейсу користувача знаходиться на клієнтському хості, в той час як логіка домену знаходиться на серверному хості. Графічний інтерфейс користувача клієнт-сервер написаний на мовах програмування високого рівня, таких як C++ та Java. Щоб краще зрозуміти цю концепцію, розглянемо клієнта як споживача, який відвідує ресторан швидкого харчування і робить замовлення безпосередньо кухарю (серверу). Потім кухар готує і подає клієнту страву. Однак процес може стати менш ефективним, якщо є велика кількість клієнтів або якщо споживач бажає більш складну страву.

3) Трирівнева архітектура: Рівень презентації – це рівень користувацького інтерфейсу, рівень додатків - це сервісний рівень, який здійснює детальну обробку, а рівень даних – це сервер бази даних, який зберігає дані. Щоб краще зрозуміти цю архітектуру, уявіть собі клієнта як відвідувача ресторану, який робить замовлення офіціанту (серверу додатків). Потім офіціант зв'язується з кулінарним персоналом (сервер бази даних), щоб отримати необхідні інгредієнти і приготувати страву. Це підвищить ефективність системи обслуговування в ресторані, оскільки додаткові офіціанти та кухарі зможуть обслуговувати більшу кількість

клієнтів.

4) N-рівнева архітектура: N-рівнева архітектура розділяє додаток на логічні рівні, які розділяють обов'язки і керують залежностями, і фізичні рівні, які працюють на окремих машинах, покращують масштабованість і збільшують мережеву затримку. N-рівнева архітектура може бути закритою, коли рівень може взаємодіяти лише з рівнем, розташованим нижче, або відкритою, коли рівень може взаємодіяти з усіма рівнями, розташованими нижче. Щоб краще зрозуміти це, уявіть собі клієнта як відвідувача ресторану з чітко встановленою ієрархією персоналу. Кожен співробітник виконує певну функцію, наприклад, адміністратор (рівень презентації), сервер (рівень додатків) і кулінарний персонал (рівень даних). Хостес вітає клієнта і проводить його до столика, офіціант приймає замовлення і передає його працівникам кухні, а працівники кухні доставляють їжу на стіл. Це дозволяє зробити процес більш ефективним і результативним, оскільки кожен співробітник виконує певну функцію і може зосередитися на своїх обов'язках.

Microsoft MySQL Server є відомим прикладом трирівневої архітектури з трьома основними компонентами: протокольним рівнем, реляційним механізмом і механізмом зберігання даних. Клієнт SQL Server повинен бути розгорнутий на всіх клієнтських пристроях, які безпосередньо взаємодіють з SQL Server. Процес виконання клієнт-сервер від Microsoft допомагає в адмініструванні більшості графічних наборів інструкцій в операційній системі Windows.

1.3.3 Клієнт-серверна мережа

Мережа клієнт-сервер – це комунікаційна архітектура, в якій клієнти отримують ресурси та послуги від виділеного хоста через локальну мережу (LAN) або глобальну мережу (WAN), таку як Інтернет.

Трафік між клієнтом і сервером (трафік з півночі на південь) і трафік між сервером і клієнтом (трафік зі сходу на захід) - це два типи мережевого

трафіку. Електронна пошта, обмін даними, друк і всесвітня павутина є популярними мережевими послугами. Ключовою перевагою мережі клієнт-сервер є централізоване адміністрування додатків і даних.

Мережа клієнт-сервер полегшує передачу даних, захищаючи їх при цьому. Використання мережі є розумним рішенням для підприємств, які шукають швидшу та безпечнішу передачу даних.

Клієнт-серверні мережі – це комп'ютерні мережі, в яких використовується виділений комп'ютер для зберігання даних, управління/надання ресурсів і контролю доступу користувачів (сервер). Сервер з'єднує всі інші комп'ютери в мережі, виконуючи роль концентратора. Комп'ютер, який підключається до сервера, називається клієнтом. Як правило, клієнт-серверним мережам надається перевага перед одноранговими мережами, в яких відсутній центральний сервер для управління мережею.

Клієнтське обладнання – це, як правило, комп'ютер або інший мобільний пристрій зі встановленими мережевими програмами. Користувач на протилежному боці комп'ютера використовує Інтернет, щоб надіслати запит до сервера. Сервер, або центр обробки даних, який знаходиться на стороні сервера, містить величезну кількість даних у файлах, базах даних і програмах.

Клієнт-серверна мережа працює за принципом вулиці з двостороннім рухом, на якій клієнт одночасно надсилає запити, а сервер реагує на них оновленнями та відповідними відповідями. Оскільки клієнт-серверна мережа має кілька клієнтів і серверів, мережевий трафік може бути значним. Після завершення операції сервер відключає клієнта від мережі, щоб заощадити пропускну здатність. В результаті, ефективність використання пропускної здатності клієнтом і сервером визначає швидкість, з якою надаються результати. В Інтернеті і в локальній мережі (LAN), наприклад, в корпорації або організації, може використовуватися архітектура клієнт-сервер.

Клієнти зазвичай з'єднуються з серверами через стек протоколів

TCP/IP. TCP створює і підтримує з'єднання, поки прикладні програми на обох кінцях не закінчать обмін повідомленнями, оскільки це протокол, орієнтований на з'єднання. Він вирішує, як розділити прикладні дані на пакети, що доставляються мережею, передає і приймає пакети з мережевого рівня, управляє потоком і повторно передає втрачені або спотворені пакети, а також підтверджує всі пакети, що надходять. TCP включає елементи рівня 4, транспортного рівня, і частини рівня 5, сеансового рівня, в комунікаційній архітектурі взаємодії відкритих систем (OSI).

IP, з іншого боку, є протоколом без з'єднання, що означає відсутність постійного з'єднання між кінцевими точками зв'язку. Кожен пакет даних, що проходить через Інтернет, розглядається як окрема одиниця даних без зв'язку з іншими одиницями даних. (TCP відповідає за розміщення пакетів у правильному порядку). IP знаходиться на третьому рівні комунікаційної моделі взаємодії відкритих систем (OSI), який є мережевим рівнем.

Клієнт-серверні мережі складаються з різних елементів, включаючи сервер, клієнтів, балансувальники навантаження та мережеві протоколи. Сервери можуть бути використані для надання функціональності користувачам, таким як сервери баз даних, веб-сервери та інші. Клієнти поділяються на тонкі, товсті та гібридні, залежно від того, наскільки самостійно вони можуть обробляти дані. Балансувальники навантаження відповідають за розподіл запитів між серверами для ефективного використання ресурсів. Мережеві протоколи, такі як TCP/IP, використовуються для комунікації між клієнтами та серверами. TCP/IP використовує шаблон запит-відповідь та розподіляє дані на пакети для передачі по мережі. Протокол IP є безз'єднувальним, що означає, що кожен пакет даних має незалежну природу.

1.3.4 Переваги та недоліки

Архітектурна концепція клієнт-сервер має декілька переваг. По-перше, централізація даних на одному сервері спрощує захист та контроль доступу

користувачів. Також дозволяє легко вирішувати проблеми усіх вузлів мережі з одного місця. По-друге, клієнт-серверна мережа має масштабованість, тобто може бути розширена за потреби, без перебоїв у роботі. Простота управління та доступність до даних також є перевагами цієї архітектури. З огляду на безпеку даних, централізована модель клієнт-серверної мережі забезпечує належний захист та можливість легко відновити дані з резервної копії.

Але мережа клієнт-сервер також має деякі недоліки. Перевантаження мережевого трафіку може спричинити проблеми з підключенням до мережі та доступом до інформації, особливо у великих компаніях. Вартість налаштування та обслуговування сервера може бути висока, що обмежує доступність для окремих користувачів. Надійність клієнт-серверної мережі також може бути проблемою через централізовану структуру. Також вимагається спеціаліст для обслуговування сервера, що підвищує складність в обслуговуванні. Нарешті, не всі ресурси на сервері доступні для незалежного використання клієнтами.

1.3.5 Типи серверів

Деякі з найпоширеніших типів серверів, що використовуються в індустрії інформаційних технологій.

❄ Веб-сервер: Ці сервери встановлюють зв'язок між вашим комп'ютером і будь-якими збереженими даними з інтернет-сайтів. Інформація для Інтернету зберігається на веб-серверах і отримується за допомогою коду "HTTP"перед передачею у ваш веб-браузер. Один з найпопулярніших типів серверів - цей.

❄ Віртуальна машина (ВМ): Віртуальні машини зберігають і з'єднують дані лише у віртуальному просторі, як вказує їхня назва. Гіпервізор, який зазвичай називають монітором віртуальних машин (VMM), - це програмне забезпечення, яке дозволяє IT-командам працювати з сотнями віртуальних машин на одному реальному обладнанні. Оскільки це

найекономічніший тип серверів, цей метод віртуалізації серверів зазвичай використовується для передачі та зберігання даних.

❄ Проксі-сервер: Проксі-сервери працюють як міст між хост-сервером і клієнтом. Після проходження через сервер проксі-сервера, проксі передає дані з веб-сайту на IP-адресу вашого комп'ютера. Оскільки інформація запитується, а потім передається від джерела до проксі-сервера, а не безпосередньо від клієнта до іншого користувача, цей метод забезпечує додатковий ступінь захисту. Численні шкідливі дії в Інтернеті можуть бути заблоковані проксі-сервером.

❄ Сервер додатків: За допомогою віртуальних серверних з'єднань ці сервери пов'язують клієнтів з програмним забезпеченням. Це дозволяє користувачам отримувати доступ до програм без завантаження даних на власні пристрої. Сервери додатків є найкращим вибором для компаній, оскільки вони можуть ефективно розміщувати великі обсяги даних додатків для багатьох користувачів одночасно.

❄ Сервер протоколу передачі файлів (FTP): Для передачі файлів з одного комп'ютера на інший використовуються FTP-сервери. Завантажені файли витягуються на ваш пристрій з сервера, тоді як завантажені файли надходять з вашого комп'ютера на сервер. Для безпечного з'єднання комп'ютерів і передачі даних цей процес називається протоколом передачі файлів.

❄ Сервер баз даних: Сервери баз даних діють як великі сховища даних, до яких підприємства можуть мати доступ і використовувати їх для роботи різноманітних додатків. Для роботи сервера баз даних не потрібно створювати будь-яку базу даних.

❄ Поштовий сервер: Поштовий сервер зберігає і доставляє повідомлення користувачам через платформи, які надають послуги електронної пошти. Користувачам не потрібно запускати будь-яке програмне забезпечення на власних пристроях, щоб отримати доступ до своєї електронної пошти, оскільки поштові сервери налаштовані на постійне підключення до мережі.

❄ Файловий сервер: Файли даних декількох користувачів зберігаються на файловому сервері. Вони дають змогу швидше записувати файли на комп'ютери та швидше отримувати дані. Коли багатьом користувачам потрібен доступ до файлів, які легше і безпечніше зберігати на сервері, ніж на персональному комп'ютері, підприємства часто використовують цей базовий тип серверів.

❄ Сервер системи доменних імен (DNS): Ці сервери перетворюють читабельні комп'ютерні доменні імена в IP-адреси, записані в комп'ютерному коді. Сервер DNS використовує введену користувачем пошукову інформацію, щоб знайти запитувану адресу та надіслати її на клієнтський пристрій.

❄ Сервер спільної роботи: Сервер спільної роботи полегшує з'єднання, коли роботу потрібно розділити між кількома користувачами. Ви можете обмінюватися та зберігати файли, програми та інші значні обсяги даних за допомогою цих серверів.

❄ Ігровий сервер: Щоб об'єднати користувачів з усього світу, великі ігрові мережі потребують серверів. На цих серверах розміщуються багатокористувацькі інтернет-ігри.

❄ Сервер управління та моніторингу: Сервери для адміністрування та моніторингу виконують різноманітні завдання. Перші приймають запити користувачів, записують їх і відстежують цифрові транзакції. Інші лише спостерігають за активністю користувачів і не беруть у ній жодної динамічної участі. Мережеві менеджери, які перевіряють стан мережі, щоб знайти небезпеки або недоліки в системі, можуть використовувати сервери моніторингу, щоб відповідати на свої запити.

❄ Сервер друку: Для друку в мережі сервер друку встановлює віддалені з'єднання з комп'ютерами, розташованими поблизу. Завдяки цим серверам компанії мають можливість використовувати один принтер для підтримки цілого відділу. Деякі принтери, встановлені в офісному приміщенні, навіть мають власний вбудований сервер, готовий до підключення до мережі.

Клієнти - це комп'ютерне обладнання або серверне програмне забезпечення, яке робить запити на ресурси та послуги, що надаються сервером. Клієнтів часто називають "запитувачами послуг". Розрізняють три категорії клієнтських обчислень: товсті, тонкі та гібридні клієнти.

– Товстий клієнт: Клієнт, який пропонує широкі функціональні можливості, виконує більшу частину обробки даних самостійно і лише трохи залежить від сервера.

– Тонкий клієнт: Сервер додатків виконує більшу частину необхідної обробки даних для сервера тонкого клієнта, який є легким комп'ютером, що значною мірою покладається на ресурси хост-комп'ютера.

– Гібридний клієнт: Гібридний клієнт поєднує в собі елементи тонкого і товстого клієнта. Він може виконувати локальну обробку, але повинен покладатися на сервер для збереження постійних даних.

Пристрій або комп'ютерна програма, яка слугує центром для інших компонентів або програм, називається сервером. Сервер - це будь-яка комп'ютеризована система, до якої клієнт може отримати доступ або використовувати для спільного використання ресурсів і розподілу завдань. До типових серверів належать такі:

✓ Сервер додатків
✓ Обчислювальний сервер
✓ Сервер баз даних
✓ Веб-сервер

1.4 Обговорення можливої реалізації

У даному розділі поговоримо про програмну реалізацію криптосистем.

1.4.1 Основні задачі, що виникають

Основні завдання, які виникають при реалізації програмного забезпечення для сервера в системі клієнт-сервер:

❄ Обробка запитів: Розпізнавання та обробка вхідних запитів від

клієнтів. Валідація та розбір клієнтських запитів для забезпечення коректності та безпеки.

❄ Паралельність і масштабованість: Управління паралельними з'єднаннями для одночасної обробки декількох клієнтських запитів. Реалізація механізмів горизонтального та вертикального масштабування для обробки збільшених навантажень. Впровадження механізмів контролю доступу для обмеження несанкціонованого доступу до ресурсів сервера. Аутентифікація та авторизація клієнтів на основі їх облікових даних.

❄ Зберігання та пошук даних: Взаємодія з базами даних або іншими системами зберігання даних для зберігання та отримання необхідної інформації. Впровадження ефективних механізмів пошуку даних у відповідь на запити клієнтів.

❄ Безпека: Впровадження заходів безпеки для захисту від поширених вразливостей, таких як ін'єкційні атаки, міжсайтовий скриптинг (XSS) та міжсайтова підробка запитів (CSRF). Захист каналів зв'язку за допомогою протоколів шифрування (наприклад, TLS/SSL).

❄ Обробка помилок і ведення журналів: реалізація надійних механізмів обробки помилок для ефективного вирішення непередбачуваних ситуацій. Журналізація критичних подій та помилок для моніторингу, аналізу та налагодження.

❄ Кешування: використання механізмів кешування для зберігання та отримання даних, до яких часто звертаються, що підвищує продуктивність. Впровадження політик виселення кешу для ефективного управління використанням пам'яті.

❄ Управління сеансами: Керування клієнтськими сесіями для збереження інформації про стан під час виконання декількох запитів. Реалізація механізмів завершення сеансу, таймауту та безпечної обробки сеансів.

❄ Балансування навантаження: Розподіл вхідних клієнтських запитів між декількома серверами для забезпечення оптимального використання ресурсів. Реалізація стратегій балансування навантаження для ефективної

обробки різних навантажень.

❄ Мережевий зв'язок: Ефективна обробка мережевого зв'язку між сервером і клієнтами. Реалізація протоколів для обміну даними, таких як HTTP, WebSocket або користувацькі протоколи.

❄ Управління конфігурацією: Дозволяє динамічно конфігурувати параметри сервера без необхідності внесення змін до коду. Реалізація конфігураційних файлів або інтерфейсів управління для легкого налаштування параметрів.

❄ Масштабована архітектура: Проектування архітектури сервера як модульної та масштабованої, що полегшує майбутні вдосконалення. Розгляд архітектури мікросервісів для великомасштабних додатків.

❄ Відповідність та нормативно-правові акти: Забезпечення відповідності законодавчим та нормативним вимогам, що стосуються сфери застосування. Впровадження заходів безпеки для захисту конфіденційних даних відповідно до нормативних вимог.

❄ Моніторинг та аналітика: Впровадження інструментів моніторингу для відстеження продуктивності сервера, виявлення вузьких місць та аналізу моделей використання. Використання аналітики для прийняття обґрунтованих рішень та оптимізації ресурсів.

❄ Резервне копіювання та аварійне відновлення: Впровадження регулярних процедур резервного копіювання для запобігання втрати даних у разі збоїв у роботі серверів. Розробка та тестування планів аварійного відновлення для швидкого відновлення системи.

Вирішення цих завдань забезпечує надійність, безпеку та масштабованість сервера в системі клієнт-сервер, що сприяє загальному успіху програми.

1.4.2 Методи вирішення задачі контролю доступу до ключової інформації

Ось кілька загальних підходів:

1) Шифрування пам'яті. Застосування шифрування для даних, які зберігаються в оперативній пам'яті, може захистити їх від несанкціонованого доступу. Наприклад, BitLocker для Windows пропонує інструменти для шифрування жорстких дисків на серверах. Цей медот допоможе захистити дані, включаючи ті, що тимчасово зберігаються в оперативній пам'яті. Це може застосовуватися до фізичних серверів і віртуальних машин. Аналогічно до BitLocker у Windows, LUKS (Linux Unified Key Setup) надає засоби шифрування для розділів диска або цілих пристроїв, включаючи області пам'яті, де зберігається конфіденційна інформація.

2) Управління доступом. Використання систем управління доступом (Access Control Lists) для контролю доступу до пам'яті дозволяє визначити, які користувачі або групи користувачів можуть отримати доступ до конкретних даних в пам'яті. Операційна система Windows має систему управління доступом (NTFS permissions), яка дозволяє встановлювати права доступу до файлів і папок. Це можна використовувати для обмеження доступу до конфіденційної інформації, що зберігається в пам'яті. Для Linux можна викоритовувати декілька систем обмеження доступу, які дозволяють визначити політики безпеки для окремих програм або сервісів. AppArmor та SELinux дозволяють контролювати те, які ресурси можуть бути використані програмами, включаючи доступ до пам'яті.

3) Моніторинг та журналювання. Системи моніторингу можуть слідкувати за доступом до даних в пам'яті, що дозволяє виявляти аномальні або несанкціоновані спроби доступу. Журналювання подій може бути використане для аналізу випадків порушень безпеки та реагування на них. Defender Credential Guard – цей інструментв Windows Server 2016 і вище захищає аутентифікаційні дані, які можуть бути використані для доступу до системи. Він використовує віртуальну машину для ізоляції та захисту цих даних від несанкціонованого доступу. В Linux можна налаштувати систему журналювання (logging) для моніторингу спроб доступу до пам'яті та інших ресурсів. Це допомагає виявляти аномалії і вразливості в системі.

4) Фізична безпека. Захист фізичного доступу до серверів, де зберігається ключова інформація, є так само важливим, як і цифровий контроль доступу. Забезпечення фізичної безпеки серверних приміщень може включати системи контролю доступу, наприклад, за допомогою доступу через картки або біометричні системи, відеоспостереження та інші заходи.

5) Використання безпечних протоколів комунікації. При передачі ключової інформації через мережу важливо використовувати безпечні протоколи для захисту даних під час транспортування. При передачі чутливої інформації між серверами використовуйте безпечні протоколи, такі як SSH для віддаленого доступу та TLS/SSL для захисту даних під час транспортування.

6) Регулярні оновлення та аудит безпеки. Важливо систематично оновлювати операційні системи та застосунки для заповнення потенційних уразливостей. Регулярний аудит безпеки також допоможе виявити можливі проблеми та заохочувати вдосконалення системи безпеки. Операційна система Windows також має різноманітні функції безпеки, такі як Windows Defender Antivirus, Windows Firewall, а також можливість налаштування аудиту подій для моніторингу спроб доступу.

Кожен з цих методів може бути використаний окремо або в поєднанні з іншими для максимального захисту інформації в оперативній пам'яті ЕОМ. Ці методи можуть бути ефективно використані як в системах Windows, так і в Linux для максимального захисту ключової інформації, що зберігається в оперативній пам'яті серверів. Комбінування цих методів створить більш повне і надійне забезпечення контролю доступу до даних.

Для забезпечення безпеки і коректності криптографічних операцій на сервері у системі клієнт-сервер варто використовувати декілька стратегій та методів. Серед них: вибір правильних алгоритмів, їх регулярне оновлення, коректна реалізація алгоритмів з використанням стандартів і бібліотек, контроль вводу/виводу даних та захист від атак на них. Також важливими є тестування та верифікація криптографічних функцій, щоб виявити

потенційні проблеми та атаки, захист від атак на побічні канали, наприклад, використання технік постійної часової складності або випадкової реакції. Заводити журнали та робити аудит є також важливим для виявлення проблем та операцій з криптографічною інформацією. Необхідно також дотримуватися стандартів і рекомендацій з інформаційної безпеки. Всі ці методи допомагають забезпечити безпечну роботу системи обробки криптографічної інформації на сервері та знизити ризик вразливостей та атак.

1.4.3 PKCS#11

Наведемо приклад застосування стандарту PKCS#11.

LSM-PKCS11 - це пакет, призначений для підтримки впровадження модулів безпеки. Цілями таких реалізацій є PKI (Інфраструктури відкритих ключів) для внутрішньокорпоративних та мережевих PKI (інфраструктури відкритих ключів) для внутрішньофірмових та мережевих додатків, які потребують нетривіального рівня безпеки, але не мають достатнього бюджету для придбання справжніх (сертифікованих) HSM, вартість яких починається від кількох тисяч доларів.

Архітектура LSM-PKCS11 – це архітектура стандартного клієнт-серверного додатку, в якому клієнтом є будь-який користувацький процес, що використовує бібліотеку PKCS#11, а сервером є сервіс фонового режиму LSM, запущений десь у мережі (для HSM, Hardware Security Modules), або у найпростішому випадку на тій самій машині, що і клієнт (для SSM, програмних модулів безпеки). Канал зв'язку між клієнтом і сервером – це стандартне TCP-з'єднання (Transmission Control Protocol, TCP, протокол управління передачею), з використанням:

✓ механізму стиснення для зменшення пропускної здатності мережі;

✓ сеансового рівня, що реалізує сеансовий протокол PKCS#11;

✓ рівня представлення, розробленим для підтримки сеансового протоколу PKCS#11.

Кінцевими точками TCP-з'єднання є два потоки, один з яких є користувацьким процесом, що використовує бібліотеку PKCS#11, а інший - сервіс фонового режимуа LSM.

Запити PKCS#11 (видані користувацьким процесом) відображаються через TCP-з'єднання на відповідний потік сервіс фонового режиму, а LSM-PKCS11, який виконує необхідні функції, відображає відповіді назад до початкового користувацького потоку. Внутрішня структура сервіс фонового режиму LSM-PKCS11 суворо дотримується стандарту PKCS#11, щоб полегшити інтерфейс бібліотеки PKCS#11 до фактичної криптографічної інформації та операцій.У цьому сенсі мережеві відображення інтерфейсних даних, що передаються через TCP-з'єднання, є свого роду системно-незалежним відображенням функцій PKCS#11.

Щодо порівняння з CryptoAPI, маємо наступне:

– Стандартизація: PKCS#11 є стандартом, що дозволяє більшу переносимість між різними платформами.

– Гнучкість: PKCS#11 може бути більш гнучким у виборі пристроїв безпеки та підтримуваних алгоритмів.

– Контроль доступу: PKCS#11 надає стандартизовані механізми для контролю доступу до криптографічних об'єктів, але це може вимагати конфігурації на рівні модуля безпеки.

Що стосується контролю правильності функціонування, обидва стандарти можуть забезпечувати відлагодження та виявлення помилок, проте враховуючи інтеграцію CryptoAPI з Windows, він може бути більш безпосереднім і залежним від функціоналу ОС.

ВИСНОВКИ

У даній лабораторній роботі було детально досліджено систему клієнт-сервер, а саме: опис її архітектури, типи серверів у ній, переваги та недоліки. Крім цього, було обговорено можливість реалізації сервера в системі та ризики, що вона несе за собою. На останок було наведено модель мережі з використанням PKCS#11, що існує в системі клієнт-сервіс.