МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Методичні вказівки до виконання практичних робіт з курсу **"Безпека програм та даних"**

Тернопіль - 2018

Шевчук Р.П., Дарморост І.А. // Методичні вказівки до виконання практичних робіт з курсу „Безпека програм та даних”, для студентів спеціальності 121«Інженерія програмного забезпечення». – Тернопіль, 2018. – 32 с.

Анотація. Особливо важливим аспектом підготовки бакалаврів зі спеціальності 121 «Інженерія програмного забезпечення» є одержання практичних навиків щодо захисту програм та даних від сучасних загроз кіберпростору. Цикл практичних робіт поданий у методичних вказівках дозволить студентам одержати ґрунтовні навички щодо особливостей проектування, програмування та налаштування контролів захисту для програмних систем та даних, що у них зберігаються. Методичні вказівки складаються із шести робіт, кожна з яких містить необхідний методичний матеріал, завдання для виконання та контрольні запитання. Розглянуті практичні роботи мають традиційну структуру: студент вивчає теорію, одержує допуск, виконує роботу, оформляє звіт, відповідає на контрольні запитання.

Укладачі: **Шевчук Руслан Петрович**, к.т.н., доцент кафедри комп’ютерних наук ТНЕУ
Дарморост Ірина Анатолівна, викладач кафедри комп’ютерних наук

Рецензенти: доцент кафедри комп’ютерної інженерії, к.т.н. **Якименко І.З.**

доцент кафедри програмної інженерії Тернопільського національного технічного університету ім. І. Пулюя, к.т.н., доцент **Кінах Я.І.**

Відповідальний за випуск: **Пукас Андрій Васильович**, к.т.н., доцент, завідувач кафедри комп’ютерних наук ТНЕУ

Затверджено на засіданні кафедри комп’ютерних наук ТНЕУ.
Протокол № 14 від «19» червня 2018 р.

Рекомендовано до друку науково-методичною радою факультету комп’ютерних інформаційних технологій ТНЕУ. Протокол № 7 від 22 червня 2018 р.

ЗМІСТ

ПРАКТИЧНА РОБОТА № 1

Тема роботи: Розмежування повноважень користувачів на основі парольної аутентифікації.
Мета роботи: Розробка програми розмежування повноважень користувачів на основі парольної аутентифікації

Теоретичні відомості

Ідентифікація та аутентифікація користувачів являються базовими механізмами забезпечення інформаційної безпеки.

Ідентифікація та аутентифікація застосовуються для обмеження доступу випадкових і незаконних суб'єктів (користувачі, процеси) інформаційних систем до її об'єктів (апаратні, програмні та інформаційні ресурси).

Загальний алгоритм роботи таких систем полягає в тому, щоб отримати від суб'єкта (наприклад, користувача) інформацію, що засвідчує його особу, перевірити її справжність і потім надати (або не надати) цьому користувачеві можливість роботи з системою.

Ідентифікація та аутентифікація застосовуються для обмеження доступу випадкових і незаконних суб'єктів (користувачі, процеси) інформаційних систем до її об'єктів (апаратні, програмні та інформаційні ресурси).

Загальний алгоритм роботи таких систем полягає в тому, щоб отримати від суб'єкта (наприклад, користувача) інформацію, що засвідчує його особу, перевірити її справжність і потім надати (або не надати) цьому користувачеві можливість роботи з системою.

Ідентифікація - присвоєння суб'єктам і об'єктам доступу особистого ідентифікатора і порівняння його з заданим. Ідентифікація виконується при спробі входу в будь-яку систему (наприклад, в операційну систему або в електронний поштовий сервіс).

Аутентифікація (встановлення автентичності) - перевірка приналежності суб'єкту доступу пред'явленого ним ідентифікатора і підтвердження його автентичності. Іншими словами, аутентифікація полягає в перевірці: чи суб'єкт, що підключається до системи є тим, за кого він себе видає. Виходить, що кожен раз, коли ви вставляєте ключ у замок, вводите пароль або ставите палець на сенсор, ви проходите аутентифікацію.

Аутентифікація здійснюється на основі того чи іншого секретного елемента (аутентифікатора), який є у розпорядженні як суб’єкта, так і інформаційної системи. Звичайно, інформаційна система має в розпорядженні не сам секретний елемент, а деяку інформацію про нього, на основі якої приймається рішення про адекватність суб’єкта ідентифікатору. Наприклад, перед початком інтерактивного сеансу роботи більшість операційних систем запитують у користувача його ім’я та пароль. Введене ім’я є ідентифікатором користувача, а його пароль - аутентифікатором.

Операційна система зазвичай зберігає не сам пароль, а його хеш-суму, що забезпечує складність відновлення пароля.

В інформаційних технологіях використовуються такі **методи аутентифікації**:

- **однобічна аутентифікація**, коли клієнт системи для доступу до інформації доводить свою аутентичність;
- **двобічна аутентифікація**, коли, крім клієнта, свою аутентичність повинна підтверджувати і система (наприклад, банк);
- **трибічна аутентифікація**, коли використовується так звана нотаріальна служба аутентифікації для підтвердження достовірності кожного з партнерів в обміні інформацією.

Методи аутентифікації також умовно можна поділити на **однофакторні та двофакторні.**

Однофакторні методи діляться на:

- **логічні** (паролі, ключові фрази, які вводяться з клавіатури комп'ютера чи клавіатури спеціалізованого пристрою);
- **ідентифікаційні** (носієм ключової інформації є фізичні об'єкти: дискета, магнітна карта, смарт-карта, штрих-кодова карта тощо. Недоліки: для зчитування інформації з фізичного об'єкта (носія) необхідний спеціальний пристрій; носій можна загубити, випадково пошкодити, його можуть викрасти або зробити копію).
- **біометричні** (в їх основі – аналіз унікальних характеристик людини, наприклад: відбитки пальців, малюнок райдужної оболонки ока, голос, обличчя. Недоліки: біометричні методи дорогі і складні в обслуговуванні; чутливі до зміни параметрів носія інформації; володіють низькою достовірністю; призначені тільки для аутентифікації людей, а не програм або інших ресурсів).

Головна перевага **парольної аутентифікації** – простота й звичність. Паролі давно вбудовані в операційні системи й інші сервіси. При правильному використанні паролі можуть забезпечити прийнятний рівень безпеки. Проте, по сукупності характеристик їх варто визнати найслабшим засобом перевірки дійсності.

Щоб пароль був запам'ятовуваним, його найчастіше роблять простим (ім'я подруги, назва спортивної команди й т.п.). Однак простий пароль неважко вгадати, особливо якщо знати пристрасті даного користувача.

Іноді паролі із самого початку не зберігаються в таємниці, тому що мають стандартні значення, зазначені в документації, і далеко не завжди після установки системи виробляється їхня зміна.

Введення пароля можна підглянути. Іноді для підглядання використовуються навіть оптичні прилади.

Паролі нерідко повідомляють колегам, щоб ті могли, наприклад, підмінити на якийсь час власника пароля. Теоретично в подібних випадках більш правильно залучити засоби керувань доступом, але на практиці так ніхто не робить: а таємниця, яку знають двоє, це вже не таємниця.

Пароль можна вгадати "методом грубої сили", використовуючи, скажемо, словник. Якщо файл паролів зашифрований, але доступний для читання, його можна скачати до себе на комп'ютер і спробувати підібрати пароль, запрограмувавши повний перебір (передбачається, що алгоритм шифрування відомий).

Проте, важливі заходи дозволяють значно підвищити надійність **парольного захисту:**

- накладення технічних обмежень (пароль повинен бути не занадто коротким, він повинен містити букви, цифри, знаки пунктуації й т.п.);
- керування терміном дії паролів: їхня періодична зміна;
- обмеження доступу до файлу паролів;
- обмеження числа невдалих спроб входу в систему, це утруднить застосування "методу грубої сили"):
- навчання користувачів;
- використання програмних генераторів паролів.

Перераховані заходи доцільно застосовувати завжди, навіть якщо поряд з паролями використовуються інші методи аутентифікації.

Розглянуті вище паролі можна назвати **багаторазовими:** їхнє розкриття дозволяє зловмисникові діяти від імені легального користувача.

Завдання для виконання

Розробити програму систему із використанням фреймворку для парольної ідентифікації \ аутентифікації із наступним функціоналом:

1. Наявність облікових записів адміністратора (користувача з фіксованим іменем ADMIN) і звичайного користувача.

2. У режимі адміністратора програма повинна підтримувати наступні функції (при правильному введенні пароля):

- зміна пароля адміністратора (при правильному введенні старого пароля);

- перегляд списку імен зареєстрованих користувачів і установлених для них параметрів (блокування облікового запису, включення обмежень на обрані паролі) - всього списку цілком в одному вікні або по одному елементу списку з можливістю переміщення до його початку або кінця;

- додавання унікального імені нового користувача до списку з порожнім паролем (рядком нульової довжини);

- блокування можливості роботи користувача з заданим ім'ям;

- включення або відключення обмежень на обрані користувачем паролі (відповідно до індивідуального завдання, що визначається номером варіанту);

- завершення роботи з програмою.

3. У режимі звичайного користувача програма повинна підтримувати тільки функції зміни пароля користувача (при правильному введенні старого паролю) і завершення роботи, а всі інші функції повинні бути заблоковані.

4. Після свого запуску програма повинна запитувати у користувача в спеціальному вікні входу введення його імені та пароля. Приведенні пароля його символи завжди повинні на екрані замінюватися символом '*'.

5. При відсутності введеного в вікні входу імені користувача в списку зареєстрованих адміністратором користувачів програма повинна видавати відповідне повідомлення і надавати користувачеві можливість повторного введення імені чи завершення роботи з програмою.

6. При неправильному введенні пароля програма повинна видавати відповідне повідомлення і надавати користувачеві можливість повторного введення. При триразовому введенні неправильного пароля робота програми повинна завершуватися.

7. При первинному введенні пароля (обов'язковому при першому вході адміністратора або користувача з зареєстрованим раніше адміністратором ім'ям) і при подальшій заміні пароля програма повинна просити користувача підтвердити введений пароль шляхом його повторного введення.

8. Якщо вибраний користувачем пароль не відповідає необхідним обмеженням (при встановленні відповідного параметра облікового запису користувача), то програма повинна видавати відповідне повідомлення і надавати користувачеві можливість введення іншого пароля, завершення роботи з програмою (при першому вході даного користувача) або відмови від зміни пароля.

9. Інформація про зареєстрованих користувачів, їх паролів, відсутність блокування їх роботи з програмою, а також включення або відключення обмежень на обрані паролі повинна зберігатися у спеціальному файлі. При першому запуску програми цей файл повинен створюватися автоматично і містити інформацію лише про адміністратора, що має порожній пароль.

10. Інтерфейс з програмою повинен бути організований на основі меню, обов'язкової частиною якого має бути підменю «Довідка» з командою «Про програму». При виборі цієї команди повинна видаватися інформація про автора програми і виданому індивідуальному завданні. Інтерфейс користувача програми може також включати панель керування з дублюючими команди меню графічними кнопками і рядок стану.

11. Обмеження на обрані користувачами програми паролі залежать від вказаного викладачем номера індивідуального варіанту.

Індивідуальні варіанти завдань

1. Довжина не менше мінімальної довжини, що встановлюється адміністратором і зберігається в обліковому записі користувача.
2. Наявність малих і великих літер.
3. Наявність букв і цифр.
4. Наявність букв і знаків пунктуації.
5. Наявність цифр і розділових знаків.
6. Наявність букв і знаків арифметичних операцій.
7. Наявність цифр і знаків арифметичних операцій.

8. Наявність латинських букв і символів кирилиці.
9. Наявність букв, цифр і розділових знаків.
10. Наявність латинських букв, символів кирилиці і цифр.
11.Наявність латинських букв, символів кирилиці і розділових знаків.
12.Наявність малих і великих літер, а також цифр.
13.Наявність малих і великих літер, а також знаків пунктуації.
14.Наявність малих і великих літер, а також знаків арифметичних операцій.
15.Наявність латинських букв, символів кирилиці і знаків арифметичних операцій.
16.Наявність букв, цифр і знаків арифметичних операцій.
17.Наявність букв, знаків пунктуації та знаків арифметичних операцій.
18.Наявність цифр, розділових знаків і знаків арифметичних операцій.
19.Відсутність повторюваних символів.
20.Чергування букв, цифр і знову букв.
21.Чергування букв, знаків пунктуації та знову букв.
22.Чергування цифр, букв і знову цифр.
23. Відсутність підряд розташованих однакових символів.
24.Чергування цифр, розділових знаків і знову цифр.
25.Чергування цифр, знаків арифметичних операцій і знову цифр.
26.Неспівпадіння з ім'ям користувача.
27.Наявність малих і великих латинських букв, цифр і символів Кирилиці.
28.Чергування букв, знаків пунктуації та цифр.
29.Чергування цифр, букв, знаків пунктуації.
30. Відсутність підряд розташованих однакових цифр.

Зміст звіту

1. Назва та зміст практичної роботи.
2. Опис індивідуального завдання.
3. Текст програми основних модулів ПЗ.
4. Висновки.

Контрольні запитання

1. Що розуміється під ідентифікацією користувача?
2. Що розуміється під аутентифікацією користувачів?
3. Чи застосовується механізм ідентифікації до процесів?
4. Перечислити можливі ідентифікатори при реалізації механізму ідентифікації.
5. Перечислите можливі ідентифікатори при аутентифікації.
6. У чому особливості динамічної аутентифікації?
7. Опишіть механізм аутентифікації користувача.
8. Перечислити види аутентифікації за рівнем інформаційної безпеки.
9. Охарактеризуйте основні методи аутентифікації.
10. Сфери застосування токенів та одноразових паролів.

ПРАКТИЧНА РОБОТА № 2

Тема роботи: Логування дій користувачів у програмних системах.
Мета роботи: Засвоїти методику та отримати практичні навики розробки процедур логування дій користувачів на прикладі підсистем ідентифікації та аутентифікації користувачів із важкооборотними однонапрямленими хеш-функціями.

Теоретичні відомості

З метою запобігання внутрішніх витоків інформації, контролю робочого часу і забезпечення інформаційного захисту застосовуються різні засоби моніторингу та спостереження за користувачем . Серед таких засобів можна виділити кейлоггер, або клавіатурний шпигун (запис даних, що вводяться з клавіатури), контроль за використанням зовнішніх носіїв інформації і блокування їх використання (зокрема - блокування USB-портів), запис екрану монітора.

Використання політик доступу до конфіденційної інформації і інформування персоналу про те, що всі дії користувача на комп'ютері логуються за допомогою програм стеження за користувачем є найбільш ефективним засобом захисту інформації від витоку та запобігання інсайдерських інцидентів.

Логування дій користувача здійснюється за допомогою спеціальних програм, перехоплюючих дії користувача, що ведуть детальний лог-файл, який містить скріншоти або запис екрану монітора, відстежує і дозволяє блокувати запуск і встановлення додатків, набраний текст, дії з файлами та зовнішніми носіями інформації.

Відомо, що парольна ідентифікація, не захищає ПЗ від програмних та апаратних засобів сканування клавіатури та ліній передачі, а отже може призвести до витоку конфіденційної інформації. Тому, як правило, в сучасних системах пароль не передається в явній формі по лініях передачі, а натомість в якості паролю використовують якесь його відображення. У якості такого відображення використовуються важкооборотні однонапрямлені функції, застосування яких гарантує неможливість розкриття пароля за його відображенням за розумний час.

В такому випадку **процедура ідентифікації** описується **таким алгоритмом** (див.рис.1):

1) Користувач вводить свій ідентифікатор.
2) Засоби ідентифікації переглядають список зареєстрованих ідентифікаторів. Якщо ідентифікатор на зареєстрований, -- то виводиться повідомлення, що такий користувач в системі не зареєстрований, і далі перехід на крок 1, або ж завершення роботи програми входу в систему. Якщо ж ідентифікатор зареєстрований, -- то перехід на крок 3.

3) ПЗ генерує випадкове число x, та обчислює значення важкооборотної однонапрямленої функції y , яка використовується в системі для відображення паролю користувача.
4) Число x передається користувачу.
5) Користувач обчислює значення важкооборотної однонапрямленої функції y’ та передає його в систему.
6) Система порівнює значення y та y’. Якщо вони співпадають, то ПЗ дозволяє вхід користувача в систему. В інакшому випадку видається повідомлення про помилку вводу паролю, перехід на крок 3, або ж завершення роботи програми входу в систему.

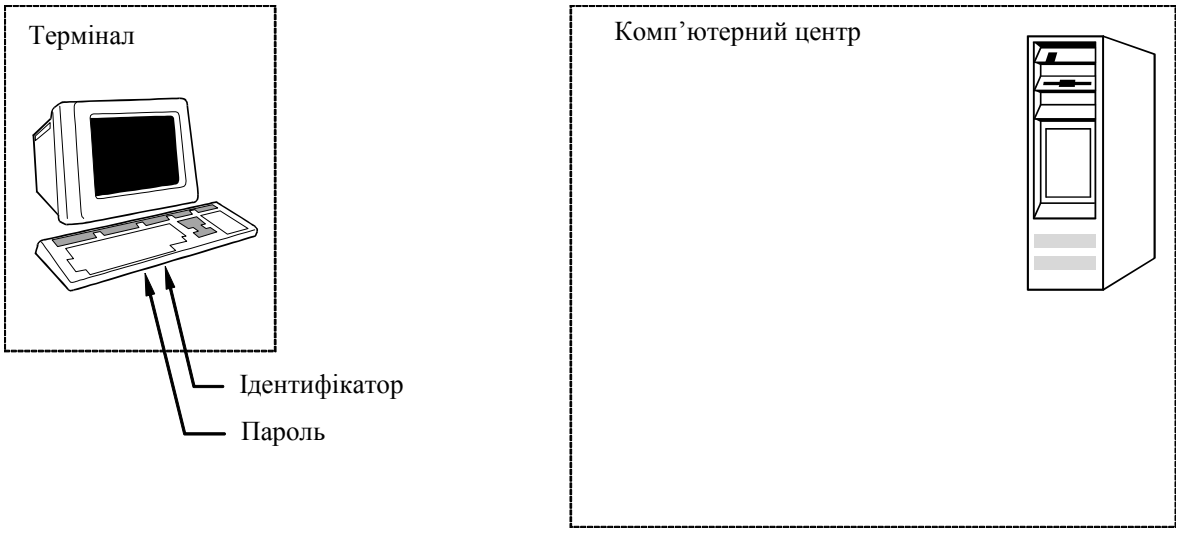

Рисунок 1 - Ідентифікація користувача за допомогою важкооборотної функції відображення паролю

Зрозуміло, що стійкість такої системи до інтерполяції використовуваної функції визначається важкооборотністю функції y та частотою генерації і розподілу ключів в системі.

Аутентифікація полягає в періодичній (стохастичній) перевірці достовірності ідентифікації користувача. Така процедура проводиться для повторної перевірки користувача. Аутентифікація може здійснюватись як апаратними, так і програмними методами за якимись особистими ознаками, чи персональними відомостями користувача.

При апаратній реалізації користувач може бути аутентифікованим за певними фізичними ознаками: вага тіла, колір очей, відбитки пальців, геометрія долоні, код ДНК і т.п. Окрім того, можуть використовуватись додаткові особисті пристрої: наручні браслети, ключі і т.д. Даний вид аутентифікації характеризується вищим рівнем надійності, проте є складнішим та дорожчим у використанні, тому він використовується на підприємствах, де необхідно забезпечити високий рівень захисту інформації.

Дешевший варіант аутентифікації користувачів полягає у створенні програмних засобів. Резидентна програма періодично з певним кроком часу задає випадковим чином запитання із заздалегідь створеного файлу, або ж

випадкові три- , чотири-розрядні десяткові числа користувачу, порівнює відповіді з наперед зареєстрованими, або ж обчисленими відповідями, і на основі цього надає, або забороняє роботу користувача. У випадку правильної відповіді за користувачем залишаються його права, а у випадку неправильної відповіді – користувач втрачає права доступу і повинен заново ввійти в систему. Стійкість даного виду аутентифікації забезпечується конфіденційністю інформації.

Процедура аутентифікації користувачів може бути реалізована як з постійним, так і з адаптивним періодом повтору. Постійний період повтору використовується в тих системах, в яких частота появи користувачів та інтенсивність їх роботи є приблизно рівномірною. При виборі періоду процедури аутентифікації слід керуватися такими міркуваннями: при досить великому періоду збільшується імовірність НСД, а при досить малому – зменшується ефективність роботи користувачів, оскільки вони постійно відволікаються від виконання основних своїх обов`язків. В системах, до яких ставляться вимоги підвищеної захищеності можуть застосовуватися засоби аутентифікації з адаптивним періодом повтору. Період повтору в таких системах визначається як інтенсивністю роботи користувачів, так і спробами НСД.

Окрім того, після встановлення достовірності ідентифікації користувача виконується логування - реєстрація в часі всіх дій користувача в **Операційному журналі системи**. В такому журналі окрім записів санкціонованого використання тих, чи інших ресурсів системи, можуть накопичуватися дані про спроби несанкціонованого доступу користувачів з автоматичною сигналізацією адміністратору системи для прийняття організаційних заходів з метою виявлення порушників. При створенні операційного журналу слід пам`ятати, що різні типи користувачів мають доступ до різних типів ресурсів.

Іншою важливою складовою частиною ПЗ є програма реєстрації. Програма реєстрації призначена для реєстрації, або видалення користувачів в Реєстраційному журналі системи із наданням їм певних прав доступу. Право реєстрації, або видалення належить лише одному користувачу – адміністратору системи. Імена користувачів, їх паролі та права доступу зберігаються в явному вигляді у файлі. Розробник повинен оцінити розмір реєстраційного журналу, виходячи із заданих параметрів. Така інформація буде корисною для оцінки трудомісткості процедур сортування та фільтрування. Окрім того, на основі отриманих даних розробник може давати рекомендації адміністратору системи стосовно регулярності керування ключами.

Завдання для виконання

1. Ознайомитись з викладеним вище матеріалом.
2. Отримати індивідуальне завдання у викладача.
3. Визначити функції системи.

4. Відповідно до індивідуального завдання удосконалити розроблену в лабораторній роботі № 1 програмну систему.
5. Провести розрахунки параметрів системи.
6. Реалізувати задані функції програмної системи.
7. Оцінити розмір реєстраційного та операційного журналу.

Індивідуальні завдання

Удосконалити розроблену в лабораторній роботі № 1 програмну систему з метою покращення функції ідентифікації та аутентифікації користувачів, котра б логувала події, а саме фіксувала вхід (вихід) користувачів в Реєстраційному журналі, а також всі вчинені користувачами дії в Операційному журналі. Необхідні параметри програми вибираються відповідно до варіанту із поданої нижче таблиці.

№ п/п	К-ть користувачів	К-ть рівнів доступу	К-ть реалізованих функцій ПЗ	Функція криптування паролю	К-ть запитань	Період повтору процедури, хв	К-ть запитань в ітерації
	Ідентифікація				Аутентифікація		
1.	5	2	>5	exp(-a*x)	15	2	3
2.	6	3	>5	lg(a*x)	10	3	4
3.	7	4	>5	a*sin(x)	20	4	2
4.	8	2	>5	a*ln(x)	15	5	3
5.	9	3	>5	a/x	15	6	4
6.	10	4	>5	lg(a/x)	10	5	2
7.	11	2	>5	x/sin(a)	20	4	3
8.	12	3	>5	a*sin(1/x)	15	2	4
9.	14	4	>5	tg(a*x)	10	3	2
10.	16	2	>5	a*ln(2+x)	20	6	3

Контрольні запитання

1. Що таке логування подій?
2. Які види логування відомі?
3. Що таке парольна ідентифікація?
4. Які недоліки класичного способу парольної ідентифікації?
5. Що таке реєстраційний журнал?
6. Для чого використовують реєстраційні журнали?
7. Як оцінити необхідний об`єм реєстраційного журналу?
8. Що таке аутентифікація користувачів?
9. Які види аутентифікації відомі?
10. Назвіть основні способи аутентифікації користувачів за допомогою програмних засобів.

11. Назвіть основні способи аутентифікації користувачів за допомогою апаратних засобів.
12. Що таке операційний журнал?
13. Для чого використовують операційні журнали?
14. Як оцінити необхідний об`єм операційного журналу?

Зміст звіту

1. Назва та зміст практичної роботи.
2. Опис індивідуального завдання.
3. Структурна схема та екранні форми основних функцій удосконаленої програмної системи.
4. Розрахунок розміру реєстраційного та операційного журналу та їх екранні форми.
5. Текст програми основних модулів ПЗ.
6. Висновки.

ПРАКТИЧНА РОБОТА № 3

Тема роботи: Методи захисту програмного забезпечення.
Мета роботи: Одержати практичні навики реалізації алгоритмів захисту програмного забезпечення для найпоширеніших моделей розповсюдження.

Теоретичні відомості

Всі методи захисту ПЗ можна розділити на апаратні та програмні. До програмних відносяться методи, у яких не зачіпаються фізичні характеристики носіїв інформації, спеціальне обладнання і т.п. До апаратних відносяться методи, що використовують спеціальне обладнання (наприклад, електронні ключі, які підключаються до портів комп'ютера) або фізичні особливості носіїв інформації (компакт-дисків), щоб ідентифікувати оригінальну версію програми і захистити продукт від нелегального використання. Розроблена значна кількість апаратних засобів різного призначення, проте найбільшого поширення отримують наступні:

- спеціальні регістри для зберігання реквізитів захисту: паролів, ідентифікаційних кодів, або рівнів грифів секретності;
- пристрої вимірювання індивідуальних характеристик людини (голосу, відбитків) з метою його ідентифікації;
- схеми переривання передачі інформації в лінії зв'язку з метою періодичної перевірки адреси видачі даних;
- пристрої для шифрування інформації (криптографічні методи).

Основними програмними методами захисту ПЗ, які спрямовані на протидію статичним способам зняття захисту від копіювання, є:

- криптографічні;
- методи прив'язки до ідентифікатора;
- методи, що базуються на роботі з переходами і стеком;
- маніпуляція з кодом програми.

Сьогодні існує три найпоширеніші **моделі розповсюдження програмного забезпечення (ПЗ)**:

- безкоштовне програмне забезпечення;
- умовно безкоштовне програмне забезпечення;
- комерційне програмне забезпечення.

Під моделлю безкоштовного ПЗ (Freeware) розуміється відсутність будь-якої оплати за користування ПЗ. Дуже часто за таким принципом розповсюджуються невеликі утиліти, які, на думку авторів, можуть виявитися корисними широкому кругу користувачів, але не матимуть попиту, якщо призначити плату за їх використання. У більшості випадків безкоштовне ПЗ розробляється одним програмістом, ентузіастом своєї справи несхильним до зайвої комерціалізації сучасних інформаційних технологій. Часто декілька добровольців організовують команду, що розробляє і підтримує достатньо складну програмну систему.

Наявність можливості оцінити програму до покупки ("try before you buy") є однією з головних рис умовно безкоштовних продуктів (share). У більшості випадків автори умовно безкоштовного ПЗ надають користувачам незареєстровані версії своїх розробок, які можна використовувати в незміненій формі. Умовно безкоштовне програмне забезпечення розробляється з метою одержання якогось прибутку чи вигоди. Однак, перед одержанням прибутку від свого ПЗ, потенційному користувачеві обов'язково надається можливість апробувати програмний продукт у дії протягом деякого періоду часу з невеликими функціональними обмеженнями. Після закінчення тестового періоду потенційний покупець повинен ухвалити рішення про придбання програми. Якщо вирішено відмовитися від покупки, то треба припинити використовувати програму і видалити її з комп'ютера. Інакше, необхідно сплатити ліцензію на програму чи надати авторові чи виконати іншу запропоновану автором умову. Після цього автор надає можливість отримання повно функціональної версії ПЗ.

Сьогодні умовно безкоштовні продукти дуже популярні. Багато відомих розробників ПЗ беруть на озброєння концепцію "Try before you buy", щоб зацікавити покупців. По суті, обмежені ознайомлювальні версії комерційних продуктів є всього лише одній з модифікацій ідеї Shareware. Навіть корпорація Microsoft безкоштовно поширює 120-денні версії Windows 2003 Server і Visual Studio .NET. Правда, невелика відмінність полягає в тому, що для перетворення 120-денної версії на повноцінну доведеться отримати диск з новою версією і виконати процедуру оновлення, а для "класичних" умовно безкоштовних програм перехід до повної версії відбувається відразу ж після введення правильного реєстраційного коду.

Іноді автори умовно безкоштовного ПЗ вибирається один з наступних методів розповсюдження:

- Cardware - кожен користувач програми, який хоче її зареєструватися, повинен надіслати авторові програми поштову листівку з виглядом місцевості, де він проживає;
- Mailwam - сучасніший варіант Cardware, у якому авторові програми надсилається електронний лист. Як правило, у відповідь автор присилає реєстраційний код, що дає можливість працювати з програмою;
- Donationware - коли автор не вимагає ніякої оплати, але пропонує всім, кому сподобалася програма, пожертвувати довільну суму, щоб підтримати розробку;
- Gifrware - майже те ж саме, що і Donationware, але автор готовий приймати не тільки грошові пожертвування, але і інші подарунки;
- Beerware - подяка за програму приймається у вигляді пива;

Комерційне програмне забезпечення створюється з метою одержання прибутку у вигляді матеріальної винагороди. Комерційне ПЗ більше всього схоже на звичайний товар, який люди звикли купувати в магазинах. Перш за все, для програмного забезпечення, що поширюється як комерційне,

застосовується принцип "гроші - вперед", тобто користувач отримує програму тільки після повного внесення оплати за неї. Дуже багато програм, поширюється у такий спосіб. У більшості випадків при покупці комерційного ПЗ користувач отримує коробку, в якій містяться носії інформації (наприклад DVD або компакт-диски), документація, реєстраційна картка та інше, на розсуд продавця.

Останніми роками XX століття, разом із бурхливим розвитком Інтернет технологій набула поширення модель розповсюдження програмного забезпечення, що демонструє рекламу (Adware), яку також можна віднести до комерційної. Основна ідея полягає в тому, що розробник отримує плату за використання програми не від кінцевого споживача, якому програма дісталась безкоштовно, а від рекламодавців. При цьому користувач комерційної програми вимушений дивитися картинки, що підкачуються з Інтернету. Зрозуміло, що цей підхід актуальний тільки для ПЗ, робота якого прямо пов'язана з доступом в Інтернет. Проте з часом ефективність реклами такого роду значно знизилася, і знайти охочих платити за неї справжніми грошима стало набагато важче. Однак все ще продовжують існувати спонсоровані програмні продукти (як правило, інформаційній спрямованості), розробка яких ведеться на гроші рекламодавців в обмін на розміщення їх інформації в програмі.

Розробники комерційних програм в рекламних цілях випускають обмежені ознайомлювальні версії своїх продуктів. Такі версії зазвичай не дозволяють плідно працювати, але створюють правдиве враження про функціональність програми. Можна виділити декілька основних типів обмежених комерційних програмних продуктів:

- Demoware - це коли в програмі присутні функціональні обмеження. Наприклад, можна обробляти файли не більші заданого розміру, не можна виконувати збереження і т.д. Такі програми іноді називають Crippleware - "урізане" програмне забезпечення;
- Trialware - наявність обмежень за часом використання ПЗ. Обмеження можуть виражатися у вигляді тривалості періоду часу, впродовж якого можна користуватися програмою (наприклад 30 днів з моменту інсталяції) або у вигляді фіксованої дати закінчення тестового періоду. Може обмежуватися число запусків програми або число процесів обробки;
- Nagware - користувачу регулярно нагадується про те, що дана версія програми не є повноцінною комерційною версією. Таке нагадування може виглядати як діалогове вікно, що з'являється при запуску програми і з деякою періодичністю під час роботи ПЗ, додаткові написи, що виводяться на принтер або екран, і т.д.

Захист комерційних версій ПЗ зазвичай зводиться до вбудовування фрагмента, що містить перевірку ключа, а для злому сучасних програм найчастіше використовують їхній динамічний аналіз і за допомогою різних наладчиків визначають місця перевірки ключа. Як правило, досить легко

виявити місце звірення уведеного ключа з «правильним» значенням і, модифікувавши код захищеної програми, домогтися її працездатності.

В більшості випадків ключі для розблокування програмного забезпечення шифруються для того щоб зловмисник не зміг зламати систему.

Одним є найпростіших класів шифрування є шифри простої заміни.

Найпоширенішими методами цього класу є **«Шифр Цезаря»** та **«Шифр Віженера».**

Шифр Цезаря (Caezar, 100-44 рр. до н.е.) реалізує кодування фрази шляхом «зрушення» усіх букв фрази на певне число kk (у оригінальному шифрі Цезаря це число kk дорівнювало 3). Якщо буква шифрованої фрази має в алфавіті позицію jj, то вона в «шифровці» замінюватиметься буквою, що знаходиться в алфавіті на позиції jj + kk.

Нехай kk = 3 і фразою для шифрування буде «i remember that september». Використовуватимемо латинські букви із стандартним проходженням букв в алфавіті. Результати шифрування вказаної вище фрази показані нижче в таблиці:

1	i		r	E	m	e	m	b	e	R		t	h	A	t	
2	9	0	18	5	13	5	13	2	5	18	0	20	8	1	20	0
3	12	3	21	8	16	8	16	5	8	21	3	23	11	4	23	3
4	12	3	21	8	16	8	16	5	8	21	3	23	11	4	23	3
5	l	c	u	H	p	h	p	e	h	U	c	w	k	d	w	C

1	s	e	p	T	e	m	b	e	R
2	19	5	16	20	5	13	2	5	18
3	22	8	19	23	8	16	5	8	21
4	22	8	19	23	8	16	5	8	21
5	v	h	s	W	h	p	e	h	u

Пояснення до таблиці:

1-й рядок - фраза для шифрування;

2-й рядок - номери букв фрази для шифрування в латинському алфавіті;

3-й рядок - номери букв фрази для шифрування, збільшені на 3;

4-й рядок - результат «ділення по модулю 27» чисел 3-го рядка;

5-й рядок - зашифрована фраза.

Шифр Віженера реалізує кодування фрази шляхом «індивідуального зрушення» букв, причому величина зрушень визначається номерами (положенням) букв в ключовому слові (фразі). Візьмемо, наприклад, ключове слово «leonid» (латиниця) і кодовану фразу «I remember that September». Букви ключового слова мають наступні номери в латинському алфавіті: 12,5,15,14,9,4. Шифрування по Віженеру полягає в «зрушенні» першої букви кодованої фрази на 12 позицій, тобто в заміні букви «i» (9-а позиція) на букву

«u», що знаходиться в 9+12=21-й позиції, в заміні пропуску « » (другої букви кодованої фрази, 0-а позиція) на букву «e», що знаходиться в 0+5=5-й позиції і так далі. При «вичерпанні» букв ключового слова, останнє використовується знову і знову до тих пір, поки не будуть закодовані всі букви кодованої фрази. Результати шифрування вказаної вище фрази показані нижче в таблиці:

1	i		r	E	m	E	m	B	e	r		t	h	A	t	
2	9	0	18	5	13	5	13	2	5	18	0	20	8	1	20	0
3	l	e	o	N	i	D	l	E	o	n	i	d	l	E	o	n
4	12	5	15	14	9	4	12	5	15	14	9	4	12	5	15	14
5	21	5	33	19	22	9	25	7	20	32	9	24	20	6	35	14
6	21	5	6	19	22	9	25	7	20	5	9	24	20	6	8	14
7	U	e	f	S	v	I	y	G	t	E	i	x	t	F	h	n

1	s	e	p	T	e	m	b	E	r
2	19	5	16	20	5	13	2	5	18
3	i	d	l	E	o	N	i	D	l
4	9	4	12	5	15	14	9	4	12
5	28	9	28	25	20	27	11	9	30
6	1	9	1	25	20	0	11	9	3
7	a	i	a	Y	t		k	I	c

Пояснення до таблиці.
1-й рядок - фраза для шифрування;
2-й рядок - номери букв фрази для шифрування в латинському алфавіті;
3-й рядок - ключове слово з довжиною рівній довжині фрази;
4-й рядок - номери букв ключового слова в алфавіті;
5-й рядок - сума номерів 2-го і 4-го рядків у відповідних стовпцях;
6-й рядок - результат «ділення по модулю 27» чисел 5-го рядка;
7-й рядок - зашифрована фраза.

Таким чином, «шифровка» матиме вигляд «uefsviygteixtfhnaiayt kic».
Декодування «шифровки» проводимо «аналогічно» кодуванню фрази.

Завдання для виконання.

1. Розробити програмний продукт (або удосконалити ПЗ розроблене в попередніх лабораторних роботах), що виконує мінімум 10 функцій (для прикладу відкриття файлу, збереження файлу, довідка, друк, перегляд параметрів файлу, пошук та інші) .
2. Реалізувати модель розповсюдження програмного забезпечення згідно варіанту.

3. Реалізувати шифрування ключа, який розблоковує функції системи згідно варіанту.

Індивідуальні завдання

№	Тип ПЗ	Обмеження	Шифр ключа
1	Demoware	Розмір файлів для відкриття не більший 100 КБ	Цезаря
2	Demoware	Відкриття файлів тільки формату BMP	Віженера
3	Demoware	Відкриття файлів тільки текстових форматів	Цезаря
4	Demoware	Відсутня функція збереження файлів	Віженера
5	Demoware	Файли відкриваються тільки з однієї директорії	Віженера
6	Demoware	Недоступна функція друку	Цезаря
7	Demoware	Недоступна функція перегляду параметрів файлу	Цезаря
8	Trialware	Блокування виконання програми після 30 днів після її встановлення	Віженера
9	Trialware	Блокування виконання програми після 10 запусків після її встановлення	Цезаря
10	Trialware	Блокування виконання програми після її відкриття 10 раз	Віженера
11	Trialware	Блокування виконання програми після 30 днів після її встановлення або 20 запусків	Віженера
12	Trialware	Блокування функцій програми після 10 днів після її встановлення	Цезаря
13	Trialware	Блокування функцій програми в останній день поточного року	Цезаря
14	Trialware	Блокування функцій програми в кінці поточно місяця	Віженера
15	Nagware	Діалогове вікно нагадування про реєстрацію програми через кожні 5 хвилин роботи з нею	Цезаря
16	Nagware	Діалогове вікно нагадування про реєстрацію програми після кожного запуску програми	Віженера
17	Nagware	Діалогове вікно нагадування про реєстрацію програми після кожного завершення програми	Віженера
18	Nagware	Діалогове вікно нагадування про реєстрацію програми після виконання операції відкриття файлу	Цезаря
19	Nagware	При відкритті програми здійснювати друк на принтері форми, що нагадує про необхідність реєстрації програми	Цезаря
20	Nagware	При відкритті програми запуск браузера на сторінці розробника ПЗ	Віженера

Зміст звіту

1. Тема та мета роботи.
2. Завдання на виконання.
3. Опис функцій розробленого програмного забезпечення.
4. Програмний код, що реалізовує індивідуальне завдання.
5. Тестовий приклад, що відображає результат виконання практичної роботи.
6. Висновок.

Контрольні запитання

1. Методи захисту ПЗ.
2. Моделі захисту програмного забезпечення?
3. Безкоштовне програмне забезпечення?
4. Умовно безкоштовне програмне забезпечення?
5. Комерційне програмне забезпечення?
6. Методи розповсюдження безкоштовного ПЗ?
7. Що ви розумієте під поняттям Cardware?
8. Що ви розумієте під поняттям Mailwam?
9. Що ви розумієте під поняттям Donationware?
10. Що ви розумієте під поняттям Gifrware?
11. Що ви розумієте під поняттям Beerware?
12. Модель розповсюдження програмного забезпечення, що демонструє рекламу
13. Типи обмежених комерційних програмних продуктів.
14. Що ви розумієте під поняттям Demoware?
15. Що ви розумієте під поняттям Trialware?
16. Що ви розумієте під поняттям Nagware?
17. Шифр Віженара
18. Шифр Цезаря

ПРАКТИЧНА РОБОТА № 4

Тема роботи: Захист веб-ресурсів від ботів та спаму за допомогою механізму CAPTCHA.
Мета роботи: отримати практичні навики реалізації механізму CAPTCHA.

Теоретичні відомості

CAPTCHA (англ. «completely automated public turing test to tell computers and humans apart» повністю автоматизований публічний тест Тюринга для розрізнення комп'ютерів і людей) - торгова марка Університет Карнегі - Меллона, комп'ютерний тест типу виклик-відповідь, який використовується для того, щоб визначити, хто використовує систему - людина чи комп'ютер

У найпоширенішому варіанті CAPTCHA від користувача потрібно ввести символи, зображені, як правило, в спотвореному вигляді на пропонованому малюнку, іноді з додаванням шуму або напівпрозорості. Рідше застосовуються CAPTCHA, що базується на розпізнаванні мови (в основному як альтернатива для людей з порушеннями зору), або на інших варіантах завдань штучного інтелекту.

CAPTCHA найчастіше використовується при необхідності запобігти використанню інтернет-сервісів ботами, зокрема, для запобігання автоматичній реєстрації, викачуванню файлів, масовим розсилкам тощо.

Для чого потрібний тест CAPTCHA?

У сучасному світі людині все частіше доводиться зіштовхуватися з роботами. Роботи керують трубопроводами, підтримують технологічний процес, виконують складні обчислення. Існує безліч роботів і в мережі Internet - тут їх, як правило, скорочено називають ботами. Internet боти стежать за порядком у чатах, займаються розсиланням листів, індексую WEB сторінки для прискорення пошуку. Однак не всі боти займаються корисною діяльністю. Існує безліч ботів написаних для деструктивних цілей. На просторах мережі часто зустрічаються:

- боти для захватів чат - кімнат в IRC;
- боти для збору адресів e-mail і розсилання за ними спаму (небажаної кореспонденції, як правило рекламного характеру);
- флуд-боти, що займаються “засміченням” форумів і гостьових книг;
- боти, що проводять масову реєстрацію облікових записів на поштових й інших сервісах;
- боти, що займаються автоматичним підбором паролів;
- боти, що роблять накручення лічильників відвідуваності сайтів;
- боти для розсилання SMS через WEB інтерфейси;
- боти для участі в голосуваннях;
- й інші.

Рішенням проблеми ботів може стати тест CAPTCHA. Цей тест найчастіше застосовується для захисту WEB форм від автоматичного

заповнення. Іноді дуже важливо чи знати заповнена форма людиною або ж її заповнив бот.

Людину від бота, звичайно, можна відрізнити за рядом непрямих ознак наприклад по швидкості заповнення форми (комп'ютер робить це набагато швидше людини) або по IP адресу (з одного IP адресу надходить цілий ряд запитів на заповнення однієї й тієї ж форми) та ін. Такі перевірки робляться досить простими методами.

Але за простотою таких перевірок криється й простота їхнього обходу з боку бота. Тому такі непрямі перевірки застосовуються лише як допоміжний засіб, а як основний засіб виступає саме CAPTCHA.

Варіанти реалізації CAPTCHA.

На сьогоднішній день існує безліч реалізацій цього тесту для мережі Internet від примітивних до найскладніших, які тільки здатна видумати людина.

Перелічимо деякі з них:

1. Найпоширеніша реалізація CAPTCHA - це буквено-цифровий тест. Суть його полягає у наступному: при заповненні форми людині показується картинка з деяким випадковим набором букв та цифр, а поруч із картинкою текстове поле в яке необхідно ввести символи зображені на картинці. Якщо введені в текстовому полі символи й символи на картинці збігаються тоді заповнена форма приймається до обробки, а якщо ні, то генерується нова картинка й людини просять увести ще раз символи з картинки. Це просте завдання для людини, але для бота воно вже досить важке. Як правило зображення на картинці зашумлюють для ускладнення спроб автоматичного розпізнавання. Приклад цієї реалізації можна побачити на рисунку 1.

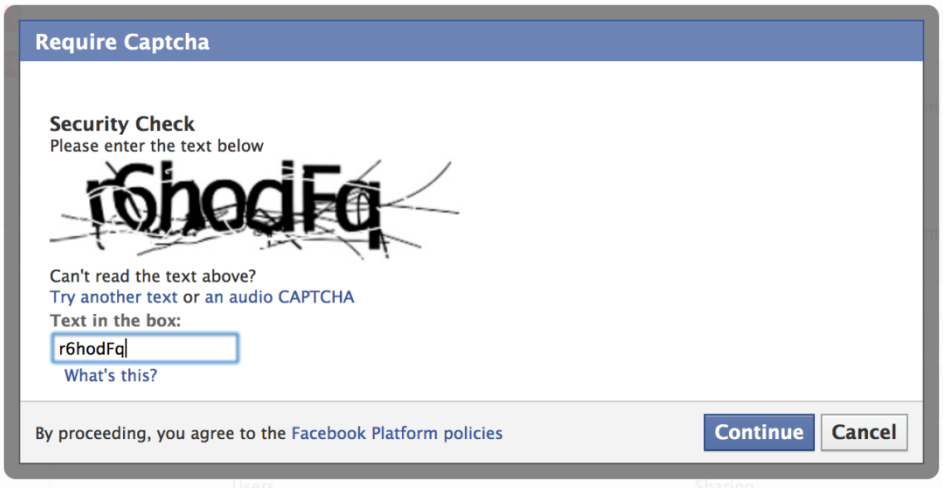

Рисунок 1 - Варіант реалізації CAPTCHA № 1

2. Іноді застосовують спрощену для користувача реалізацію описаного в першому пункті тесту, коли користувачеві показують усе той же випадковий набір символів і пропонують кілька варіантів відповіді. У цьому варіанті реалізації тесту вже неможливо зробити помилку при наборі тексту й

тому після першої або другої неправильної відповіді IP адреса блокується на кілька годин. Приклад цієї реалізації можна побачити на рисунку 2.

Рисунок 2 - Варіант реалізації CAPTCHA № 2

3. Існує також комбінація першого й другого варіанта тесту. Людині виводиться у вигляді картинки текст, що спонукує його до деякої дії (наприклад “натисніть на хвіст кішці”). А поруч із текстом розташовується картинка з декількома об'єктами (у нашому прикладі ними можуть бути кішка, собака, равлик). Координати натискання по картинці відслідковуються й перевіряються. Причому питання й супутня йому картинка із предметами щоразу різні. Приклад цієї реалізації можна побачити на рисунку 3.

Рисунок 3 - Варіант реалізації CAPTCHA № 3

4. Людині показують картинку з якимось об'єктом (наприклад портрет Шевченка або Франка) і просять увести його назву (у нашому випадку прізвище письменника). Як правило картинки перекручені випадковими

дефектами, щоб бота не навчили розпізнавати всі картинки тесту. Приклад цієї реалізації можна побачити на рисунку 4.

Рис. 4. Варіант реалізації CAPTCHA № 4

5. Людині показують кілька картинок (частіше всього 3 або 4), на яких зображено декілька предметів, і просять ввести назву того предмета, що є на всіх картинках. Приклад цієї реалізації можна побачити на рисунку 5.

Рисунок 5 - Варіант реалізації CAPTCHA № 5

6. Математична CAPTCHA вона є різновидом буквено-цифрової, але замість букв і цифр зображених на малюнку потрібно підрахувати та ввести результат математичної дії.

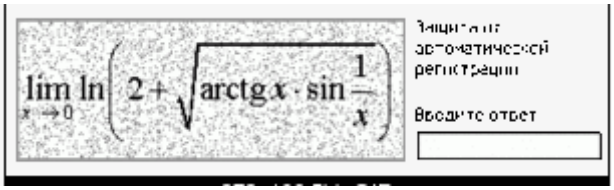

Рисунок 6 - Варіант реалізації CAPTCHA № 6

Є й інші варіанти реалізації CAPTCHA, проте вони досить складні у програмній реалізації і тому використовуються дуже рідко.

Завдання для виконання

Удосконалити ПЗ розроблене в попередніх практичних роботах в частині захисту однієї із розроблених форм за допомогою механізму CAPTCHA відповідно до індивідуального завдання.

Індивідуальне завдання

1. Для оцінки «задовільно» запрограмувати механізм CAPTCHA у вигляді запиту відповіді на звичайне зображення, яке в довільному порядку обирається з бази даних ПЗ.
2. Для оцінки «добре» запрограмувати механізм CAPTCHA у вигляді запиту відповіді на звичайне зображення із накладенням шумів, яке в довільному порядку обирається з бази даних ПЗ. Розмістити на окремій формі reCaptcha Google.
3. Для оцінки «відмінно» запрограмувати механізм CAPTCHA у вигляді запиту відповіді на зображення із накладенням фону та шумів, яке в довільному порядку обирається з бази даних ПЗ. Розмістити на окремій формі reCaptcha Google.

Зміст звіту

4. Тема та мета роботи.
5. Завдання на виконання.
6. Опис функцій розробленого програмного забезпечення.
7. Програмний код, що реалізовує індивідуальне завдання.
8. Тестовий приклад, що відображає результат виконання практичної роботи.
9. Висновок

Контрольні запитання

1. Історія створення CAPTCHA.
2. Механізми створення CAPTCHA.
3. Механізми «обходу» CAPTCHA
4. Як CAPTCHA допомагаю боротись із спамом.
5. Переваги використання CAPTCHA.
6. Варіанти реалізації CAPTCHA.
7. Особливості використання reCaptcha Google.
8. Особливості передбачення результатів CAPTCHA.
9. Переваги використання CAPTCHA.
10. Відомі CAPTCHA-сервіси.

ПРАКТИЧНА РОБОТА № 5

Тема роботи: Аналіз захищеності веб-ресурсів
Мета роботи: отримати практичні навики аналізу захищеності веб-ресурсів, формування аналітичного звіту та рекомендацій щодо усунення виявлених вразливостей.

Теоретичні відомості

Атаки на веб-ресурси є найбільш популярним способом проникнення в внутрішню інфраструктуру організацій та основною причиною всіх витоків даних.

Більше половини сучасних веб-ресурсів містять критично небезпечні вразливості, які дозволяють зловмисникам проводити різні атаки, в тому числі спрямовані на відмову в обслуговуванні та крадіжці персональних даних.

Звести до мінімуму ризик успішної атаки можна, якщо регулярно проводити аналіз захищеності веб - ресурсів.

Оцінка захищеності веб - ресурсів може виконуватися як з використанням "чорного ящика", так і шляхом аналізу вихідних кодів. Другий спосіб більш ефективний, але й більш складним та затратним.

В процесі аналізу захищеності фахівці використовують загальноприйняті методики аналізу та оцінки безпеки додатків, зокрема OWASP TOP 10, Web Application Security Consortium, Thread Classification і Common Vulnerability Scoring System.

Як правильно для отримання повної картини аналізуються всі компоненти веб-ресурсів: дизайн, мережеве взаємодія, налаштування ОС, зовнішні джерела даних, особливості зберігання інформації, використовувані механізми авторизації та аутентифікації, серверні та клієнтські компоненти.

В загальному випадку порядок проведення аналізу захищеності веб-ресурсів наступний:

1. Визначення методу, який доцільно використовувати для аналізу веб-ресурсу ("чорний ящик", аналіз вихідних кодів, комбінація методів).
2. Проведення інструментальних перевірок та перевірки автоматичним та ручним способом. Рекомендований перелік інструментів подано на сайті https://tools.kali.org/tools-listing
3. Вибір вразливостей. На основі аналізу характеристик виявлених вразливостей, таких як складність використання, доступність методів експлуатації, можливі втрати в разі атаки та ін., вибираються ті з них, які з високим ступенем експлуатації можуть бути використані.
4. Здійснення експлуатації ряду найбільш критичних вразливостей.
5. Формування аналітичного звіту та рекомендацій щодо усунення виявлених вразливостей.

Завдання для виконання

1. Розгорнути розроблене в попередніх практичних роботах ПЗ на віртуальній машині.
2. Провести за допомогою автоматизованих засобів аналізу на вразливості веб-ресурсів перевірку розробленого ПЗ.
3. Навести приклади експлуатації виявлених вразливостей.
4. Сформувати аналітичний звіт та рекомендацій щодо усунення виявлених вразливостей.

Зміст звіту

1. Тема та мета роботи.
2. Завдання на виконання.
3. Опис процесу аналізу захищеності веб-ресурсу відповідно до пунктів завдання для виконання.
4. Висновок

Контрольні запитання

1. ТОП 10 OWASP
2. Оцінка захищеності веб-ресурсів з використанням методу "чорного ящика".
3. Оцінка захищеності веб-ресурсів шляхом аналізу вихідних кодів.
4. Web Application Security Consortium.
5. Thread Classification.
6. Common Vulnerability Scoring System.
7. Особливості експлуатації вразливостей.
8. Експлоїти та їх роль.
9. Особливості проведення аналізу захищеності веб-ресурсів.
10. Функціонал автоматизованих засобів аналізу на вразливості веб-ресурсів.

ПРАКТИЧНА РОБОТА № 6

Тема роботи: Особливості захисту даних за допомогою технології Blockchain
Мета роботи: отримати практичні навики реалізації Blockchain.

Теоретичні відомості

Blockchain - це децентралізована система зберігання даних або цифровий реєстр транзакцій, угод, контрактів, яка складається з набору записів. Головна відмінність і незаперечна перевага технології - те, що цей реєстр не зберігається в одному місці. Він розподілений серед кількох сотень і навіть тисяч комп'ютерів у всьому світі. Будь який користувач цієї мережі може мати вільний доступ до актуальної версії реєстру, що робить його прозорим абсолютно для всіх учасників [2]. Blockchain працює наступним чином: усі транзакції за допомогою складних математичних алгоритмів об'єднуються в "блоки", які потім зв'язуються криптографічно і хронологічно в "ланцюг" та містять хеш попереднього блоку. Транзакція при цьому здійснюється лише тоді, коли вважається підтвердженою

Кожен блок у блокчейні пов'язаний з попереднім і містить у собі набір записів та має свій унікальний хеш. У найпростішому вигляді база даних блокчейну представляє собою ланцюжок блоків, який може бути представлена у вигляді файлу формату JSON (рисунок 1).

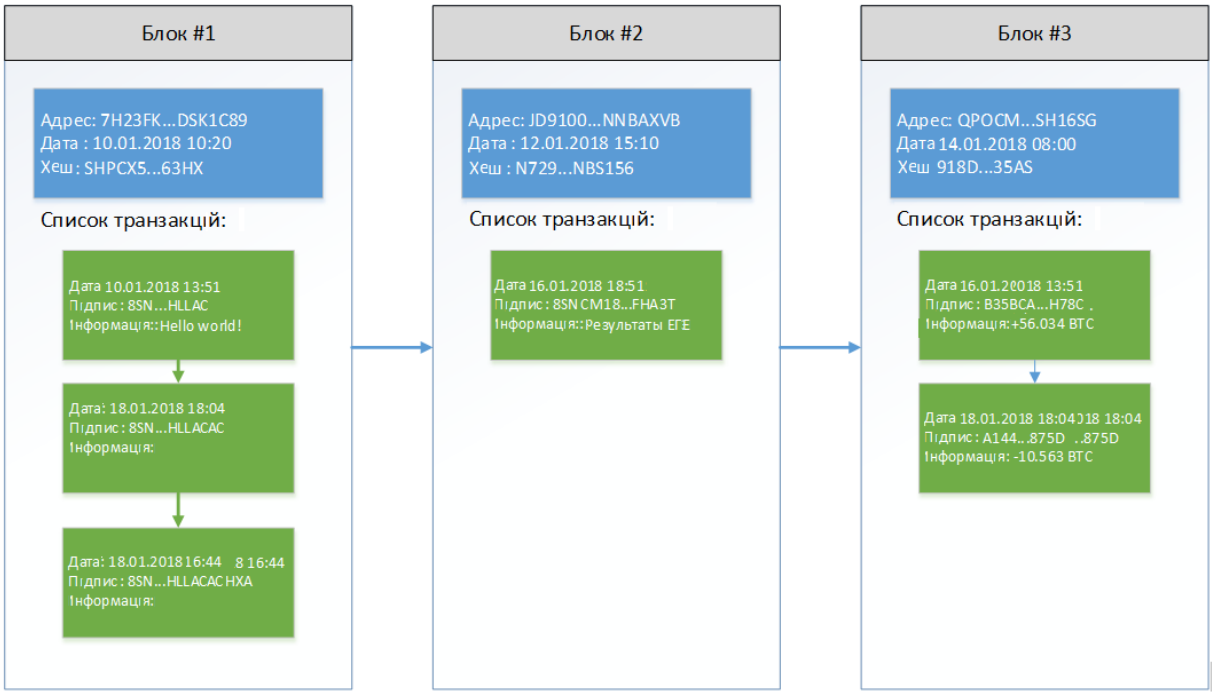

Рисунок 1 – Приклад блокчейну із трьох блоків

Для простого розуміння, що з себе представляє блок, досить уявити його у вигляді скрині з замком, коли ви щось хочете туди покласти, то вам

потрібно відімкнути замок ключем, цей ключ створюється вами при створенні блоку і називається закритий ключ.

Всі транзакції блокчейну знаходяться у мережі. Нові блоки завжди додаються в кінець ланцюжка. Якщо в результаті їх розрахунків всі вони отримують однаковий результат, то блоку присвоюється унікальна цифрова сигнатура (підпис). Як тільки реєстр буде оновлено і утворений новий блок, він вже не може бути змінений. Таким чином, підробити його неможливо. До нього можна тільки додавати нові записи. Важливо врахувати те, що реєстр оновлюється на всіх комп'ютерах в мережі одночасно. Таким чином, можна назвати одну з переваг Blockchain - неможливими є проникнення зловмисників в систему, оскільки для цього необхідно мати доступ до баз даних на всіх комп’ютерах одночасно. Процес хешування є незворотнім і навіть, якщо документ буде змінений, він отримає інший цифровий підпис, що буде сигналізувати про невідповідності в системі.

Щоб інформацію всередині транзакцій можна було підробити, кожна транзакція всередині блоку підписується електронним цифровим підписом (ЕЦП).

Для створення ЕЦП потрібно:

- Асиметричний алгоритм шифрування (наприклад, RSA);
- Хеш-функція (наприклад, SHA512);
- Інформація, яку збираємося підписувати.

Оскільки асиметричні алгоритми досить повільні в порівнянні з симетричними, то обсяг підписуються даних відіграє велику роль і якщо він великий, то зазвичай беруть хеш від підписуються даних, а не самі дані. Хеш отримують за допомогою хеш-функцій, наприклад, SHA512, яка приймає на вхід якусь інформацію і повертає хеш певної довжини.

Таким чином, ЕЦП ставиться не на сам документ, а на його хеш. Хеш-функції не є частиною алгоритму ЕЦП, тому в схемі може бути використана будь-яка надійна хеш-функція.

Сполучний хеш перераховується при кожному додаванні нової транзакції. Він отримуєтья шляхом підсумовування всіх хеш транзакцій поточного блоку і адреси попереднього блоку:

Хеш (зв’язуючий) = SHA512(block_prev_adress_hash + transaction_hash1 + transaction_hash2 + ... + transaction_hashN)

Наприклад, з рисунку 1 видно, що хеш 3-го блоку обчислювався як адреса другого блоку і два хеша двох внутрішніх транзакцій:

Хеш 3 блоку = SHA512(JD9100...NNBAXVB + B35BCA...H78C + A144...875D)

Саме зв’язуючи хеш об'єднує блоки в єдину ланцюг і найголовніше захищає блокчейн від підробки зловмисниками. Припустимо, якщо хтось

захоче "викинути" або вставити свій блок в середину ланцюжка, то блоки наступні за ним вже не пройдуть перевірку, тому що їх хеш грунтувався на адресі яку хочуть підмінити або прибрати.

Для генерації ключової пари можна скористатися різними бібліотеками, наприклад, мова C # має вбудований пакет для роботи з алгоритмами шифрування і ЕЦП.

```
// Створення нової пари ключів розміром 1024 біта
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(1024);

// приватний ключ, представлений у вигляді стрічки XML
string privateKey = rsa.ToXmlString(true);

// публічний ключ, представлений у вигляді стрічки XML
string publicKey = rsa.ToXmlString(false);
```

Отриманий закритий ключ слід зберігати в окремому файлі, публічний ключ є адресою блоку, тому його зберігати не потрібно. Таким чином для кожного користувача логін - це адреса блоку (публічний ключ), а пароль - це закритий ключ, знаючи ці два ключа можна отримати доступ до комірки блокчейна розташованої за цією адресою і використовувати інформацією, що міститься у ній.

Підпис довільних даних алгоритмом шифрування RSA, передаємо дані і приватний ключ, отримуємо підпис:

```
private static string SignData(string data, string privateKey)
{
   // Одержуємо об’єкт класу RSA через провайдер
   RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(1024);
   // Вважаємо, що у нас вже є приватний ключ і необхідно використовувати його
   rsa.FromXmlString(privateKey);
    // Перетворюємо символи стрічки послідовність байтів
   byte[] byteData = Encoding.UTF8.GetBytes(data);
   // Хешуємо наші данні з допомогою SHA512 і підписуємо вже отриманий хеш
   byte[] signedByteData = rsa.SignData(byteData, CryptoConfig.MapNameToOID("SHA512"));
   // Конвертуємо масив байтів в стрічку в кодуванні Base64
   string signedData = Convert.ToBase64String(signedByteData);
   // Повертаємо ЕЦП
   return signedData;
}
```

Завдання для виконання

Реалізувати програмну систему, яка зберігає ланцюжок блоків в окремому файлі в форматі JSON, при кожному додаванні нового блоку або транзакції - необхідно оновлювати файл.

Функції клієнта системи:

1. Реєстрація нового користувача (один блок = 1 користувач) - тобто створити новий блок, повернути користувачеві закритий ключ, а публічний використовувати як адреса блоку;
2. Авторизація - доступ до осередку блокчейна, логіном є адреса блоку, паролем - закритий ключ;
3. Після проходження авторизації - вставити транзакцію з довільною інформацією в блок (не в будь-який, а в той до якого є доступ);
4. Перегляд списку блоків і транзакцій в зрозумілому вигляді.

Структура блоку і транзакцій повинна відповідати опису в поданому в теоретичних відомостях. Також необхідно в окремому файлі зберігати закриті ключі користувачів (у цьому блокчейні закритий ключ зберігається у кожного користувача свій, нам же потрібна така функціональність для перевірки працездатності програми).

Зміст звіту

1. Тема та мета роботи.
2. Виконання завдання для виконання.
3. Опис процесу реалізації програмної системи, лістинг коду та екранних форм, які відображаються результат виконання роботи.
4. Висновок

Контрольні запитання

1. Що таке блокчейн і для чого він потрібен?
2. Що таке майнінг і для чого потрібні майнінгові ферми?
3. Характеристика технології блокчейн.
4. Переваги блокчейну.
5. Недоліки блокчейну.
6. Приклади використання блокчейну.
7. Особливості реалізації блокчейну.
8. Криптографічні алгоритми, що лежать в основі блокчейну.
9. Для чого використовується ЕЦП в блокчейні.
10. Функції асиметричних алгоритмів в блокчейні.

СПИСОК ЛІТЕРАТУРИ

1. Кузнецов О. О. Захист інформації в інформаційних системах : навч. посіб. Х. : ХНЕУ, 2015. – 510 с.
2. Поляков А. О., Євсеєв С. П., Огурцов В. В.Лабораторний практикум з навчальної дисципліни "Захист інформації в інформаційних системах" : навч.-практ. посіб. - Х. : ХНЕУ, 2012. – 208 с.
3. Гуз А.М., Довгань О.Д., Марущак А.І. Організація захисту інформації з обмеженим доступом. - К. : Наук.-вид. відділ НА СБ України, 2015. - 378 с.
4. Закон України “Про захист інформації в автоматизованих системах” від 05.07.1994.
5. Загальні положення щодо захисту інформації в комп’ютерних системах від несанкціонованого доступу. – НД ТЗІ 1.1-001-98, ДСТТСЗІ СБ України, Київ, 1998.
6. Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу. – НД ТЗІ 2.2-001-98, ДСТТСЗІ СБ України, Київ, 1998.
7. Класифікація автоматизованих систем і стандартні профілі захищеності оброблюваної інформації від несанкціонованого доступу. – НД ТЗІ 2.2-002-98, ДСТТСЗІ СБ України, Київ, 1998.
8. Згуровський М. Проблеми інформаційної безпеки в Україні, шляхи їх вирішення // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – Київ. – 2000. – С. 10 – 14.
9. Комар М.П. Нейросетевой подход к обнаружению компьютерных атак на информационные ресурсы / М.П. Комар, И.О. Палий, Т.Б. Федысив, Р.П. Шевчук // Інформатика та математичні методи в моделюванні. — 2011.— Том 1. — № 2. — С. 156—163.
10. Коркішко Т., Мельник А. Алгоримти та процесори симетричного блокового шифрування. – Львів, БаК, 2003.-163 с.
11. Ємець В., Мельник А., Попович Р. Сучасна криптографія. Основні поняття. – Львів: Бак, 2003. – 144 с.
12. Молдовян Н.А., Зима В.М. Введение в практическую криптографию. Учебное пособие. – СПб.: ВИКУ им. А.Ф.Можайского, 2011. – 186 с.
13. Галатенко В.А. Информационная безопасность: практический подход. – М.: Наука, 1998.-301 с.
14. Столлингс В. Криптография и защита сетей: принципы и практика, 2-е изд.: Пер. с. англ. – М.: Издательский дом “Вильямс”, 2011. – 672 с.
15. Таненбаум Э., Стеен М. ван. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.: ил.
16. Wattenhofer R. The Science of the Blockchain. 1st ed. Inverted Forest Publishing, 2016. 115 p
17. https://habr.com

Підписано до друку 25.06.2018 р.
Формат 84х118\32. Папір офсетний. Друк на різографі.
Умов.-друк. арк. 2,0. Зам. №22342.
Тираж 100 прим.