Магомедов К.О.
студент;
Тімченко М.К.
студент,
Державний університет інтелектуальних технологій та зв'язку

ДОСЛІДЖЕННЯ МЕТОДІВ ЗАХИСТУ КОНФІДЕНЦІЙНИХ ДАНИХ В ЖИТТЄВОМУ ЦИКЛІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У середині 20-го сторіччя розпочався швидкий прогрес комп'ютерних наук. До 1950-х років розробка програмного забезпечення в цілому не була достатньо комплексним процесом для того, щоб існувала потреба у створенні власних спеціалізованих методологічних принципів та прийомів, спрямованих на вирішення в першу чергу організаційних питань, пов'язаних безпосередньо з самим процесом розробки.

Однак по мірі зростання складності та масштабів ПЗ виникла необхідність у більш структурному підході до процесу створення програмного продукту. Згодом, саме програмування, як і будь-яка виробнича діяльність, почало вимагати більший рівень завчасного планування та велику кількість детально продуманих моделей розробки. Це призвело до зародження поняття «життєвого циклу програмного забезпечення».

Відповідно до [1] життєвий цикл програмного забезпечення полягає у застосуванні стандартних бізнес-практик до створення програмного продукту. Він складається із окремих етапів робіт або стадій, що проходить програмне забезпечення у заданому порядку протягом певного періоду часу, починаючи з бізнес-ідеї та закінчуючи її імплементацією в коді, реалізацією у прибуткову діяльність та подальшій підтримці.

В загальному випадку, життєвий цикл визначається моделлю й описується у формі методології. Модель, або парадигма життєвого циклу, визначає загальну організацію і, як правило, основні його фази та принципи переходу між ними. Методологія визначає комплекс робіт, їх детальний зміст, а також рольову відповідальність спеціалістів на всіх етапах вибраної моделі.

Початкові методології розробки ПЗ розглядали дії, спрямовані на забезпечення інформаційної безпеки, як окрему задачу, що виконується під час фази тестування. Основним недоліком такого постфактумного підходу була велика кількість вразливостей або помилок, що виявляються

занадто пізно в процесі. Зараз вже не викликає сумнівів той факт, що безпека є критично важливим аспектом успішного життєвого циклу будь-якої програми, а інтеграція дій із забезпечення інформаційної безпеки у методологію розробки допомагає створювати більш надійне програмне забезпечення.

Одним із найважливіших аспектів інформаційної безпеки у контексті програмного забезпечення є робота із конфіденційними даними. Як зазначено в [2], під конфіденційними даними в цьому випадку мається на увазі будь-яка інформація з обмеженим доступом, що необхідна для коректного функціонування програмного забезпечення, наприклад:

– комбінація логіна та пароля для авторизації у зовнішньому ресурсі;

– URL бази даних, яка використовується в якості джерела даних;

– ключі для авторизації запитів до прикладного програмного інтерфейсу стороннього сервісу;

– TLS-сертифікат при використанні технології mTLS;

– приватний RSA ключ для підключення за протоколом SSH.

В більшості випадків за зберігання конфіденційних даних відповідають самі розробники. Оскільки програма повинна використовувати ці дані безпосередньо під час своєї роботи, перед розробником постає питання вибору механізму та місця їх зберігання. Цей додатковий рівень відповідальності досить часто призводить до ситуацій, коли людська помилка стає причиною компрометації та витоку такої конфіденційної інформації.

Вирішення полягає у переміщенні місця розташування подібної інформації до централізованої системи сховища даних з функцією шифрування та безпечного зберігання. Такий підхід дозволить звузити периметр потенційних загроз та звільнить розробників від відповідальності за безпеку зовнішніх ресурсів.

Основною задачею імплементації такої системи захисту є виключення ризиків людського фактору – її використання повинно запобігати ситуаціям, що виникають внаслідок людської помилки. В даному випадку це досягається за рахунок автоматичного зчитування конфіденційних даних програмними засобами замість ручного додавання цих значень до конфігурацій оточення або ж безпосередньо у вихідний код.

Окрім реалізації самого сховища, необхідне також доопрацювання безпосередньо на рівні програми, що буде ним користуватись. Програма буде відповідальна за отримання власних конфігураційних та авторизаційних даних за допомогою запитів до централізованого сховища. Таке сховище повинно відповідати певним вимогам до його функціоналу, які зображено на рис. 1:

Рис. 1. Вимоги до сховища секретів конфіденціальних даних

Загальною метою моделі безпеки такої системи є забезпечення конфіденційності, цілісності, доступності, спостережності інформації, а також аутентифікація її користувачів. Це означає, що дані повинні бути захищені від підслуховування чи підробки як в рухомому, так і в нерухомому стані. Особа користувача повинна бути належним чином виявлена шляхом аутентифікації та авторизована на доступ до даних або зміни політики. Усі взаємодії повинні підлягати аудиту та однозначно і унікально простежуватися до об'єкта походження. Система повинна бути надійною по відношенню до навмисних спроб обійти будь-який її контроль доступу.

Між клієнтом та сервером сховища не повинно існувати безумовної взаємної довіри. Клієнти можуть використовувати протокол TLS для перевірки автентичності сервера та встановлення захищеного каналу зв'язку. Сам сервер, в свою чергу, повинен вимагати від клієнта із кожним запитом надавати відповідний авторизаційний токен, який також використовується для ідентифікації клієнта [3]. Клієнт, який не надає свій токен, може лише робити запити на вхід.

Дані, що не перейшли так званий «бар’єр безпеки» та знаходяться у стані спокою, повинні шифруватись самим сервером. Їх розшифрування буде відбуватись безпосередньо перед їх передачею за клієнтським запитом. Наприклад, шифрування може відбуватись за допомогою 256-бітового шифру AES (з англ. Advanced Encryption Standard – розширений стандарт шифрування) у режимі GCM (англ. Galois/Counter

Mode – лічильник з автентифікацією Галуа) з 96-бітними криптографічними одноразовими кодами (nonce). Nonce генерується випадковим чином для кожного зашифрованого об'єкта. Коли дані зчитуються з бар'єру безпеки, тег аутентифікації GCM перевіряється під час процесу розшифрування для виявлення будь-яких підробок.

Окрім захисту безпосередньо самих конфіденційних даних, постає також питання захисту головного ключа, що використовується для їх шифрування. Ризик розповсюдження головного ключа полягає в тому, що один зловмисник, який має доступ до нього, може розшифрувати всі дані, що зберігаються в централізованому сховищі.

Наявність такого ключа вимагає абсолютну довіру до його адміністратора, що в цілому порушує модель безпеки такої системи. Тому необхідно, щоб система підтримувала схеми розділення ключа шифрування на окремі порції. Однією з таких схем є схема розділення секрету Шаміра, що дозволить розділити повідомлення між n сторонами так, щоб щоб тільки будь-які k та більше сторін ($k \leq n$), могли його відтворити. При цьому $k - 1$ і менше сторін не зможуть відтворити секрет [4].

Можемо зробити висновок, що використання централізованої системи сховища даних при розробці програмного забезпечення вирішує не тільки проблему зберігання, а ще й задачі шифрування, аудиту та контролю доступу до конфіденційної інформації.

Список використаних джерел:

1. Життєвий цикл ПЗ [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ua/articles/software-development-metodologies.html

2. Secrets in Source code [Електронний ресурс]. – Режим доступу: https://apiiro.com/the-secrets-about-secrets-in-code/

3. Hashicorp Vault Tokens [Електронний ресурс]. – Режим доступу: https://www.vaultproject.io/docs/concepts/tokens

4. Інтерполяційний поліном Лагранжа [Електронний ресурс]. – Режим доступу: http://www.mathros.net.ua/interpoljacijni-formuly-lagranzha-dlja-nerivno-viddalenyh-vuzliv-interpoljacii.html