

Захист інформації та кібернетична безпека

УДК 623.4.05

DOI: 10.30748/soi.2020.160.17

А.О. Гапон¹, В.М. Федорченко¹, А.О. Поляков²

¹ Харківський національний університет радіоелектроніки, Харків

² Харківський національний економічний університет ім. С. Кузнеця, Харків

ПІДХОДИ ДО ПОБУДОВИ МОДЕЛІ ЗАГРОЗ ДЛЯ АНАЛІЗУ БЕЗПЕКИ ВІДКРИТОГО ПРОГРАМНОГО КОДА

Гарантування безпеки програмного продукту з відкритим вихідним кодом є актуальною проблемою, бо навіть у проектах з закритим вихідним кодом можуть бути присутні *open source* бібліотеки, що робить можливим появу вразливості у них. Серед методів, що використовують для виявлення вразливостей, варто виділити моделювання загроз, бо цей метод дозволяє вже на ранніх етапах розробки програмного коду прийняти заходи, що знизять витрати на ліквідацію вразливостей та спростять їх усунення і зміни до архітектури додатку. Підбір відповідного підходу при побудові моделі загроз залежить від специфіки проекту, ресурсів, а також кваліфікації адміністраторів.

Ключові слова: модель загроз, відкритий вихідний код, *STRIDE*, *OCTAVE*, *TRIKE*, *PASTA*, *VAST*.

Вступ

Програмне забезпечення з відкритим вихідним кодом (*open source*) активно розробляється відповідними спільнотами і стає загальнодоступним. Програмне забезпечення з відкритим кодом як концепція почала розвиватися з ідей прозорості, підвищення якості, високої надійності, гнучкості, низької вартості та припинення прив'язки до постачальника. Але досягнення цього можливо тільки за рахунок збільшення тривалості періоду розробки [1].

Безпечне програмне забезпечення є ключовою вимогою для організацій, які планують впровадити програмне забезпечення з відкритим вихідним кодом як частину свого програмного стека.

Організації, які впроваджують неперевірене на безпеку програмне забезпечення з відкритим вихідним кодом в свою систему або використовують для розробки відкриті фреймворки, більш уразливі для атак, тому важливо виявляти уразливості в програмному забезпеченні з відкритим вихідним кодом до його реліза або використання фреймворків. Слід також враховувати профіль безпеки інформаційної системи, критично відстежувати безпеку програмного забезпечення, виявляючи уразливості в вихідному коді на ранній стадії розробки, а також уразливості проектування і архітектури ПО та його інфраструктури шляхом розробки моделі загроз.

Підприємства та інші організації зберігають величезні обсяги конфіденційних і критичних даних в своїх комп'ютерних системах, і це робить безпеку

системи життєво важливою. Якщо такі підприємства або організації впроваджують в свій програмний стек відкритий вихідний код, зловмисники можуть скомпрометувати конфіденційну і критичну інформацію, використовуючи деякі уразливості в програмному забезпеченні. Тому, для отримання максимального ефекту від програмного забезпечення з відкритим вихідним кодом, код повинен бути належним чином проаналізовано, щоб виявити будь-яку вразливість, виявити і виправити її до розгортання програмного продукту. Таким чином, підприємства та інші користувачі програмного забезпечення з відкритим вихідним кодом можуть забезпечити безпеку своєї важливої і критичної інформації і тим самим знизити ризик втрат і збитків [2].

Первинним етапом для оцінки безпеки коду ПО є модель загроз. Існує ряд методів розробки моделей загроз, які найчастіше спрямовані на інформаційну систему в цілому, також розглядають питання криптографічної моделі загроз, а також мережеву модель загроз, що спрямована на інфраструктуру. Моделі, які спрямовані на аналіз безпеки коду, в явному вигляді відсутні.

Таким чином, метою роботи є адаптація існуючих макетів до моделі загроз для аналізу безпеки відкритого програмного коду. Розроблена модель загроз може використовуватися при побудові надійної стратегії безпеки для програми.

Виклад основного матеріалу

Модель загроз

Моделювання загроз застосовується до широкого спектру систем організації, включаючи бізнес-процеси, інформаційні системи, мережеву інфраструктуру, розподілені підсистеми, додатки і сервіси, програмний код і т.д.

Процес моделювання загроз може виконуватися на будь-якій стадії розробки, переважно на ранній стадії, щоб результати могли допомогти при розробці проекту і скоротити витрати.

Таким чином, під моделюванням загрози будемо розуміти метод, який використовується для розробки моделі шляхом ітеративної оцінки вразливостей в додатку, також така модель допомагає виявляти, повідомляти і розуміти загрози та заходи щодо їх зниження в контексті захисту критичних ресурсів.

Адекватні моделі загроз інформаційній безпеці дозволяють виявити існуючі загрози, розробити ефективні контрзаходи, підвищивши тим самим рівень інформаційної безпеки, і оптимізувати витрати на захист, сфокусувавши її на актуальних загрозах.

Моделювання загрози використовується, щоб допомогти адміністратору безпеки (DevOpsSec) для прийняття рішень по мінімізації ризиків. Аналіз ризику загрози ділиться на два типи – кількісні і якісні.

Модель загроз включає в себе:

- опис (специфікація) / архітектура / модель критичних ресурсів інформаційної системи;
- список рекомендацій, які можуть бути перевірені або оскаржені в майбутньому при зміні ландшафту загроз;
- список потенційних загроз для системи;
- список дій, які необхідно вжити для перекриття кожної загрози;
- стратегію тестування моделі загроз і перевірки успішності вжитих заходів [3].

У якості відправної точки необхідно визначити область застосування моделі загроз. Для цього потрібно проаналізувати додаток (програмний код), з використанням наступних підходів:

- аналіз архітектурних схем;
- переходи потоку даних;
- класифікація даних.

Слід враховувати, що коли програмний код змінюється (виправляються дефекти або впроваджується новий функціонал), то це впливає на безпеку всієї програми. При цьому необхідно розуміти, що такі дії не завжди очевидні.

Моделювання загроз безпеки дозволяє створити профіль загрози системи, вивчивши її очима потенційних злоумисників. За допомогою таких методів, як ідентифікація точки входу, межі привілеїв і дерева загроз, стає можливим з'ясувати стратегії для зниження потенційних загроз для програмного коду.

Існує п'ять аспектів моделювання загроз безпеці:

Визначити загрози. Перше, що потрібно зробити, – це визначити цікаві ресурси. Спочатку моделюють систему або за допомогою діаграм потоків даних (DFD), або з діаграмами розгортання UML. З цих діаграм можна визначити точки входу в систему, такі як: джерела даних, інтерфейси прикладного програмування (API), веб-служби та сам призначений для користувача інтерфейс. Оскільки злоумисник отримує доступ до системи через точки входу, він є відправною точкою для розуміння потенційних загроз. Щоб допомогти ідентифікувати загрози безпеки, ви повинні додати “кордони привілеїв”. Кордони привілеїв розділяють процеси, об'єкти, вузли і інші елементи, які мають різні рівні довіри. Скрізь, де аспекти інформаційної системи перетинають кордони привілеїв, можуть виникнути проблеми з безпекою.

Зрозуміти загрози. Щоб зрозуміти потенційні загрози в точці входу, потрібно визначити будь-які критично важливі для безпеки дії та уявити, що може зробити злоумисник, щоб атакувати або неправильно використати систему. Слід виходити з того, що злоумисник може використовувати ресурс системи (ПО) для зміни контролю над системою, вилучення обмеженої інформації, маніпулювання інформацією в системі, збою системи або неможливості її використання, або отримання додаткових прав. Таким чином, можна визначити шанси того, що злоумисник отримає доступ до ресурсу, не пройшовши аудит, не пропустивши ніяких перевірок контролю доступу та не скориставшись аккаунтом іншого користувача.

Класифікувати загрози. Для класифікації загроз безпеки варто розглянути підходи до побудови моделі загроз, наприклад STRIDE. Класифікація загрози [4] є першим кроком до ефективного перекриття загрози або зниження її критичності.

Визначити стратегії перекриття загроз. Щоб визначити, як знизити критичність загрози, формується діаграма, яка називається деревом загроз. У корені дерева знаходиться сама загроза, а його дочірні елементи (або листя) є умовами, які повинні бути справедливими, щоб противник відтворив загрозу. Умови, в свою чергу, можуть мати підумови. Кожен шлях через дерево загроз, який не закінчується стратегією пом'якшення, є вразливістю системи.

Тестова стратегія. Модель загроз стає планом для тестування на проникнення. Тестування на проникнення досліджує загрози, безпосередньо атакуючи систему, поінформованим або не поінформованим способом. Поінформовані тести на проникнення – це, по суті, тести білого ящика, які відображають наявність доступу до внутрішнього пристрою системи, в той час як не інформовані тести за своєю природою є тестуванням чорного ящика, тобто без доступу до самого коду програми.

Підходи до побудови моделей безпеки

Розглянемо підходи, такі як STRIDE, TRIKE, OCTAVE, PASTA, VAST.

STRIDE

При проектуванні системи аналізують загрози з точки зору зловмисника. Моделювання загроз, або аналіз архітектурних ризиків, являє собою контроль безпеки для виявлення і зниження ризиків. Модель загроз STRIDE розподіляє загрози за категоріями, щоб потенційні загрози могли оцінюватися з точки зору зловмисників.

Піддроблена особистість (Spoofing identity).

Прикладом піддроблення ідентифікаційних даних є незаконний доступ, а потім використання аутентифікаційної інформації іншого користувача, такої як ім'я користувача і пароль. Приклад: фішингова атака, щоб обдурити користувача і відправити облікові дані на фальшивий сайт.

Незаконна зміна (Tampering with data). Підробка даних, яка включає в себе зловмисну зміну даних. Приклади включають несанкціоновані зміни, внесені в постійні дані, наприклад такі, що зберігаються в базі даних, і їх зміна під час передачі між двома комп'ютерами по відкритій мережі, такої як Internet. Приклад: цілісність повідомлення скомпрометована для зміни параметрів або значень.

Відмова (Repudiation). Прикладом відмови є виконання неприпустимою операції в системі, в якій відсутня можливість відстежувати заборонені операції. Наприклад, якщо користувач, який виконав дію в системі, вніс зміни в базу даних або програмний код, то його дії повинні бути занесені в log журнали аутентифікації і змін бази даних. Таким чином, система підтверджує причетність користувача до змін.

Розкриття інформації (Information disclosure). Загрози розкриття інформації включають в себе розкриття інформації особам, які не повинні мати доступу до неї. Наприклад, можливість користувачів роботи з файлом, до якого їм не надано доступ, або здатність зловмисника отримувати дані при їх передачі по каналу зв'язку. Приклад: незашифрований трафік по протоколах TCP / HTTP.

Відмова в обслуговуванні (Denial of service). Атаки типу "відмова в обслуговуванні" (DoS) відмовляють в обслуговуванні чинним користувачам, наприклад, роблячи веб-сервер тимчасово недоступним або непридатним для використання. Необхідно захищати систему та інфраструктуру різних типів DoS-загроз, атаки можуть проводитися як на мережеві служби DNS, routing, програмні сервера (web-сервера, бази даних), а також сам web-додаток.

Несанкціоноване підвищення привілеїв (Elevation of privilege). В даному типі загроз непривільований користувач отримує привілейований

доступ і, таким чином, має достатній доступ для злому або руйнування всієї системи. Загрози підвищення привілеїв включають в себе ті ситуації, в яких зловмисник ефективно проник в усі захисні системи і стає частиною самої довіреної системи, і це дійсно небезпечна ситуація. Приклад: зловмисник змінює членство в групі [5].

Найпростіший спосіб застосувати модель STRIDE до додатка – розглянути, як кожна із загроз в моделі впливає на кожен компонент системи і кожне з його з'єднань або зв'язків з іншими компонентами програми.

TRIKE

Моделювання загроз Trike є процес моделювання загроз з відкритим вихідним кодом, спрямований на задоволення процесу аудиту безпеки з точки зору управління кібер ризиками.

Основою методології є "модель вимог", яка гарантує, що призначений рівень ризику для кожного активу є "прийнятним" для різних зацікавлених сторін.

Документ з вимогами до додатка повинен бути розбитий по ресурсам, з якими будуть взаємодіяти різні типи користувачів: інформаційна система і користувачі в сукупності зі своїми правами доступу.

На етапі інтеграції коду його слід перевірити на відповідність зі специфікацією продукту, щоб переконатися, що він не виходить за рамки технічного завдання. По можливості слід використовувати набори тестів з безпеки, щоб автоматично підтверджувати покриття кодом вимог специфікації з метою виявлення непокритого коду. Якщо в ході реалізації будуть виявлені нові вектори атак або якщо написаний код відхиляється від специфікації, документ специфікації повинен бути оновлений, щоб забезпечити правильність моделі загроз.

Модель реалізації аналізується для створення моделі загроз. Загрозам присвоюються відповідні значення ризику, за якими адміністратор безпеки створює вектори атак. Потім призначаються засоби і методи для зниження значущості потенційних загроз, необхідні для усунення пріоритетних загроз і пов'язаних з ними ризиків. У висновку, адміністратори розробляють модель ризику на основі завершеної моделі загроз за допомогою активів, ролей, дій і схильності загрозам [6].

OCTAVE

Octave підтримує профілювання виконання коду на рівні функцій. Кожен виклик функції (підтримуються вбудовані модулі, оператори, функції в файлах, призначені для користувача функції в коді Octave і анонімних функцій) записується під час виконання коду. Після цього ці дані можуть допомогти в аналізі поведінки коду і, зокрема, допоможуть знайти "гарячі точки" в коді з точки зору безпеки.

Методологія моделювання загроз OCTAVE

орієнтована на оцінку організаційних (нетехнічних) ризиків, які можуть виникнути в результаті порушення даних ресурсів. Використовуючи цю методологію моделювання загроз, інформаційні ресурси організації ідентифікуються, а набори даних, які вони містять, отримують атрибути в залежності від типу даних, що зберігаються.

У OCTAVE відсутня масштабованість – оскільки технологічні системи автоматично додають користувачів, додатки і функціональні можливості, внаслідок цього ручний процес може швидко стати некерованим. Даний метод найбільш корисний при створенні корпоративної культури з урахуванням ризиків. Він легко налаштовується відповідно до конкретної політики/профілю безпеки організації і області ризику [7].

Хоча моделювання загроз OCTAVE забезпечує надійне, орієнтоване на ресурси уявлення і розуміння організаційних ризиків, документація може стати об'ємною.

PASTA

Дана методологія об'єднує вплив на бізнес, властивий додатку ризик, межі довіри між компонентами програми, корельовані загрози і шаблони атак, які використовують виявлені недоліки в ході моделювання загроз. До PASTA більшість моделей загроз додатків не враховували фактичні загрози.

Визначити бізнес-контекст додатку. На даному етапі враховується характерний для додатка профіль ризику і враховуються інші фактори впливу на бізнес на ранніх етапах SDLC або для даного Sprint в рамках Scrum.

Перелік технологій. Призначений для декомпозиції технологічного стека, який підтримує компоненти програми, що реалізують бізнес-цілі, визначені на попередньому етапі.

Декомпозиція додатку. Орієнтована на розуміння потоків даних між компонентами і службами додатку в моделі загроз.

Аналіз загроз. Перевірка властивих для даної системи загроз з урахуванням галузевих аналітичних даних про загрози, що зачіпають роботу сервісів, обробку та зберігання даних, конфігурації і моделі розгортання.

Ідентифікація уразливості. Визначення вразливостей і слабких місць в архітектурі і коді програми і зіставленні їх, щоб переконатися, що вони підтверджують відомості про погрози з попереднього етапу.

Моделювання атаки. Фокусується на емуляції атак, які можуть використовувати виявлені вразливості на попередньому етапі. Це також допомагає визначити життєздатність загроз по шаблонах атак.

Аналіз залишкового ризику. Концентрується навколо виправлення вразливостей в коді або архітектурі, які можуть сприяти загрозам і базовим моделям

атак. Це може гарантувати прийняття деяких ризиків замовниками або менеджерами по розробці [8].

PASTA надає підхід до моделювання загроз, орієнтований на ризик, який заснований на фактичних даних. Експерти з безпеки співвідносять реальні загрози з поверхнею атаки компонентів програми та виявляють ризики. Також проводять тести експлуатації, які визначають потенційні загрози моделі, щоб перевірити, чи є вони ймовірними.

Співвідношення життєздатності зі стійким впливом дозволяє цій методології позиціонувати себе як високоефективний підхід до моделювання загроз, орієнтований на ризики.

Методологія моделювання загроз PASTA є семиетапним процесом узгодження бізнес-цілей і технічних вимог, а також аналізом ризиків, який не залежить від платформи.

PASTA поєднує орієнтований на зловмисника погляд на потенційні загрози з додатком і його інфраструктурі, виходячи з чого стає можливим розробити стратегію зниження ризиків.

Дана методологія найкраще підходить для організацій, які хочуть узгодити моделювання загроз зі стратегічними цілями, оскільки вона включає аналіз впливу на бізнес як невід'ємну частину процесу.

VAST

Методологія VAST з'явилася для усунення обмежень і недоліків інших методологій побудови моделей загроз. Принцип методології VAST полягає в важливості масштабування процесу моделювання загроз по всій інфраструктурі і SDLC (Secure Development Lifecycle), а також досягнення плавної інтеграції в методологію гнучкої розробки програмного забезпечення. Мета VAST – надати рекомендації зацікавленим сторонам, включаючи розробників і фахівців з безпеки.

Методологія VAST включає три необхідні компоненти для підтримки масштабується рішення: автоматизація, інтеграція і співпраця.

Автоматизація. Автоматичне моделювання загроз скорочує необхідний час від годин до хвилин. Це дозволяє продовжувати процес моделювання загроз – загрози можуть оцінюватися в процесі проектування, реалізації та після розгортання на регулярній основі, що дозволяє масштабувати моделювання загроз, щоб охопити всю інформаційну систему, гарантуючи, що загрози ідентифікуються, оцінюються і розставляються пріоритети на всьому протязі процесу розробки.

Інтеграція. Процес моделювання загроз повинен інтегруватися з інструментами, використовуваними в SDLC, щоб забезпечити узгоджені результати для оцінки. Ці інструменти можуть включати інструменти, призначені для підтримки Agile-середовища для розробки програмного забезпечення, в якій особлива увага приділяється адаптивному

плануванню і постійному вдосконаленню.

Завдяки Agile в великих проєктах SDLC декомпонується на короткострокові цілі, які виконуються протягом двох тижнів. Для того, щоб методології моделювання загроз враховували Agile підхід.

Співпраця. Система моделювання загроз в масштабах проєкту вимагає участі ключових зацікавлених сторін, включаючи розробників програмного забезпечення, системних архітекторів і адміністраторів з безпеки DevSecOps.

Масштабоване моделювання загроз вимагає, щоб ці зацікавлені сторони співпрацювали – використовуючи комбіноване уявлення різних наборів навичок і функціональних знань для оцінки загроз та визначення пріоритетів у зниженні рівня ризиків. Без співпраці неможливо досягти загальноорганізаційного уявлення загроз. З іншого боку, спільна робота допомагає команді масштабувати діяльність з моделювання загроз, щоб охопити всі етапи SDLC і реагувати на нові загрози з більш глибоким розумінням ризиків, пов'язаних з проєктом в цілому.

Інтеграція з Agile, а також іншими виробничими інструментами формує основу для спільного, комплексного процесу моделювання загроз. Використання VAST забезпечує цілісне уявлення всієї поверхні атаки, дозволяючи проєкту мінімізувати загальний ризик.

Рекомендації для формування моделі загроз, що враховує вразливості в програмному коді

Для забезпечення безпеки програмного коду при розробці програми слід правильно оцінити можливі загрози і сформувати модель загроз, розробка якої відповідно до техніки формування моделі загроз складається з декількох етапів.

Визначте цілі безпеки (Identify Security Objectives). При розгортанні додатків з відкритим вихідним кодом в системі найкраще визначити цілі і вимоги безпеки ще до того, як інтегрувати додаток в систему. Це обмеження, які впливають на конфіденційність, цілісність і доступність даних і інших додатків системи. Визначення цілей безпеки допомагає визначити, де зосередити свої зусилля, а також виявити потенційних зловмисників в додатку. Визначення цілей безпеки виконується багаторазово шляхом вивчення вимог програми та умов використання.

У моделюванні ризиків загроз ідентифікація цілей безпеки є першим кроком, який необхідно зробити, щоб розробити безпеку додатку. Як тільки цілі безпеки визначені, вони можуть використовуватися як основа для подальших кроків. Однак це не означає, що ці цілі повинні залишатися незмінними; вони можуть змінюватися з часом і з новими ситуаціями, які можуть виникнути в процесі. Будь-які зміни в цілях безпеки впливають на інші дії безпеки. В результаті необхідно час від часу переглядати мо-

дель загроз при зміні профілю безпеки або програмного коду.

Дослідити додаток (Survey the Application). Після того, як цілі безпеки були визначені, наступним кроком буде огляд додатку. Дослідження додатку виконується шляхом аналізу структури додатка з метою визначення компонентів, потоків даних і кордонів довіри системи. Іншими словами, цей крок допомагає ідентифікувати ключові функціональні можливості, характеристики та клієнтів додатку, що, в свою чергу, допоможе виявити відповідні загрози, які будуть використовуватися на наступних етапах.

Деякі моменти, які використовуються для створення огляду програми:

Створити сценарій наскрізного розгортання: розробити схему, яка зображує компоненти, структури, підсистеми, характеристики розгортання, аутентифікацію, авторизацію і механізми комунікації компонентів у додатку.

Визначення ролей допомагає визначити як те, що має статися, так і те, що не повинно статися. Наприклад, визначити, хто може читати дані, хто може оновлювати дані, хто може видаляти дані.

Визначити ключові сценарії використання значить визначити важливі функції програми. Не варто перераховувати всі варіанти використання програми, натомість перерахувати основні.

Визначити технології: перерахувати технології та ключові функції програмного забезпечення. Деякі приклади включають операційну систему, мову розробки і так далі.

Ідентифікація технологій допомагає зосередитися на специфічних для технології загрози. Визначити механізми захисту програми: визначити будь-які ключові моменти з наступного: аутентифікація, авторизація, помилки під час введення, криптографія тощо [9].

Декомпозиція програми. (Decompose the Application). На цьому кроці варто розбити додаток, щоб визначити межі довіри, потоки даних, точки входу і точки виходу з впливом безпеки, яке необхідно оцінити. Простіше виявляти загрози та виявляти уразливості, коли модулі програми стають більш визначеними.

Визначення межі довіри додатків допомагає сфокусувати аналіз на певних галузях, що становлять інтерес. Межі довіри – це ті точки, де рівні контролю доступу змінюються для доступу до ресурсів або точок входу в додатках, де дані, що проходять через цю точку, не є повністю довіреними.

Визначити потоки даних: відстеження потоків даних через додаток, починаючи від входу до виходу. Це прояснює, як додаток взаємодіє із зовнішньою системою, а також прояснює, як взаємодіють внутрішні компоненти.

Визначити точки входу: зловмисники використовують точку входу програми для зламу системи. Цей пункт призначений для клієнтів.

Визначити, де знаходяться ці точки входу і тип даних, які вони приймають.

Визначити точки виходу: це точки, в яких додаток відправляє дані клієнта або в зовнішні системи [10].

Виявлення загроз (Identify Threats). На цьому етапі потрібно ідентифікувати загрози та атаки, які можуть поставити під загрозу безпеку додатку. Ці загрози є потенційними загрозами додатків. Як правило, важко ідентифікувати невідомі загрози, тому важливо зосередитися на відомих загрозах. Для виконання цього процесу можна використовувати такі підходи:

Почати з загальних загроз та атак: варто почати з підготовки списку поширених загроз і атак, а потім застосувати список загроз в своєму додатку. Існує ймовірність того, що деякі загрози просто усуваються, оскільки вони не застосовуються в додатку.

Використовувати підхід, заснований на питаннях: це допомагає у виявленні відповідних загроз і атак. Для цього підходу можна використовувати модель STRIDE, щоб задавати питання, пов'язані із застосуванням програми.

Дослідити рівень додатки за рівнем, шар за шаром і функцію за функцією при виявленні загроз. При виконанні ідентифікації загроз варто сконцентруватися на тих областях, де помилки безпеки найчастіше допускаються. Зверніть увагу, що загрози, виявлені на цьому етапі, не обов'язково вказують на уразливості.

Виявлення вразливостей (Identify Vulnerabilities). На цьому кроці розглядається безпека додатка, чітко виявляючи уразливості, виконуючи ту ж процедуру, що і при виявленні загроз на попередньому кроці. Однак представлені на цей раз приклади питань повинні бути розроблені таким чином, щоб допомогти ідентифікувати уразливості, а не загрози. Хороший підхід до виявлення вразливостей полягає в тому, щоб дослідити прикладний рівень за шаром, переглядаючи всі групи вразливостей на кожному рівні. Деякі категорії вразливостей, які визначаються у процесі виявлення:

- аутентифікації;
- авторизації;
- введення і перевірки даних;
- управління конфігурацією;
- конфіденційних даних;
- криптографії;
- маніпулювання параметрами;
- управління винятками;
- аудиту та ведення журналів [11].

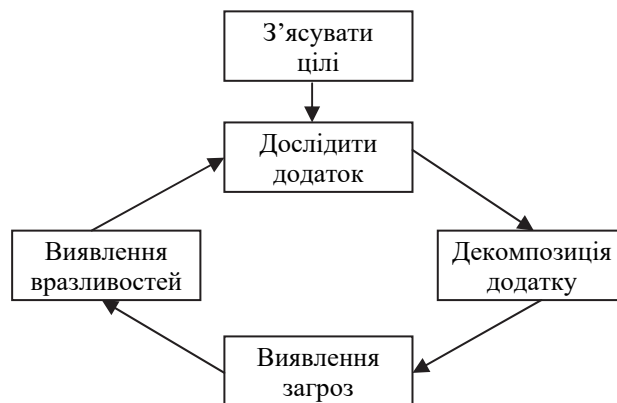


Рис. 1. Етапи моделювання загроз

Стратегії зниження рівня ризиків

Після того, як були визначені загрози і вразливості для програми та встановлені ризики та їх пріоритети, наступним етапом є розробка впорядкованої і економічно ефективної стратегії пом'якшення.

В першу чергу, слід враховувати вплив стратегії на всі складові ризиків. Тобто стратегія повинна бути застосовна до всіх ризиків додатку, а не тільки до деяких його частин. З точки зору бізнес-контексту обмежуючим фактором для стратегії пом'якшення наслідків повинен бути рівень, який організація може собі дозволити, інтегрувати. Стратегія також повинна бути в змозі продемонструвати належне зниження ризиків [12].

Висновки

Деякі методології моделювання загроз, такі як OCTAVE, зосереджені на практиці аналізу систем на наявність потенційних загроз. Інші, такі як STRIDE або PASTA, фокусуються на точці зору розробника або зловмисника. VAST враховує Agile підхід з орієнтацією на зібране додаток і процес розробки.

Рішення завдання аналізу безпеки відкритого програмного коду передбачає першим етапом розробки моделі загроз. При побудові моделі загроз необхідно враховувати аспекти моделювання загрози безпеки, такі як: визначення, поняття, класифікація загроз, визначення стратегії перекриття загроз або зниження рівня ризиків, а також тестової стратегії побудованої моделі. Існуючі підходи класифікують загрози, враховуючи весь спектр потенційних загроз, а не зосереджені на окремих областях, наприклад, відкритому програмному коді.

На основі проаналізованих підходів до побудови моделей безпеки для реалізації моделі варто використовувати рекомендації для формування моделі загроз, що враховують уразливості в програмному коді.

Список літератури

1. Opensource. Opensource Initiative [Electronic resource]. – URL: <https://opensource.org/licenses/alphabetical> (дата звернення: 25.12.2019).
2. Wheeler David A. Secure Programming HOWTO [Electronic resource] / David A. Wheeler. – URL: <https://dwheeler.com/secure-programs/Secure-Programs-HOWTO/open-source-security.html> (дата звернення: 27.12.2019).
3. OWASP. Application Threat Modeling [Electronic resource]. – URL: https://owasp.org/www-community/Application_Threat_Modeling (дата звернення: 04.01.2020).
4. Agile Modeling. Security Threat Models: An Agile Introduction [Electronic resource]. – URL: <http://www.agilemodeling.com/artifacts/securityThreatModel.htm> (дата звернення: 4.01.2020).
5. Guzman Aaron. IoT Penetration Testing Cookbook: Identify Vulnerabilities and Secure your Smart Devices / Aaron Guzman, Aditya Gupta. – Packt Publishing, 2017. – p. 34-35.
6. Shostack Adam. Threat Modeling: Designing for Security / Adam Shostack. – Wiley, 2014. – 624 p.
7. Eaton John W. Octave [Electronic resource] / John W. Eaton. – URL: <https://octave.org/doc/interpreter/> (дата звернення: 10.01.2020).
8. Versprite. Application Threat Modeling Helping Clients Learn & Build Risk-Based Threat Models [Electronic resource]. – URL: <https://versprite.com/security-offerings/appsec/application-threat-modeling/> (дата звернення: 10.01.2020).
9. OWASP. Application Threat Modeling [Electronic resource]. – URL: https://www.owasp.org/index.php/Threat_Risk_Modeling (дата звернення 11.01.2020).
10. MSDN [Electronic resource]. – URL: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach> (дата звернення 10.01.2020).
11. MSDN Threat Modeling Web Applications [Electronic resource]. – URL: [https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648006\(v=pandp.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648006(v=pandp.10)?redirectedfrom=MSDN) (дата звернення 10.01.2020).
12. Threatmodeler. Getting Started with Threat Modeling: How to Identify Your Mitigation Strategy [Electronic resource]. – URL: <https://threatmodeler.com/getting-started-with-threat-modeling-how-to-identify-your-mitigation-strategy/> (дата звернення 10.01.2020).

References

1. The official site of Opensource (2019), *Opensource Initiative*, available at: www.Opensource.org/licenses/alphabetical (accessed 25.12.2019).
2. Wheeler, David A. (2015), *Secure Programming HOWTO*, available at: www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/open-source-security.html (accessed 27.12.2019).
3. The official site of OWASP (2020), *Application Threat Modeling*, available at: www.owasp.org/www-community/Application_Threat_Modeling (accessed 04.01.2020).
4. The official site of Agile Modeling (2018), *Security Threat Models: An Agile Introduction*, available at: www.agilemodeling.com/artifacts/securityThreatModel.htm (accessed 4.01.2020).
5. Guzman, Aaron and Gupta, Aditya (2017), *IoT Penetration Testing Cookbook: Identify Vulnerabilities and Secure your Smart Devices*, Packt Publishing, pp. 34-35.
6. Shostack, Adam (2014), *Threat Modeling: Designing for Security*, Wiley, 624 p.
7. Eaton, John W. (2020), *Octave*, available at: www.octave.org/doc/interpreter/ (accessed 10.01.2020).
8. The official site of Versprite (2018), *Application Threat Modeling Helping Clients Learn & Build Risk-Based Threat Models*, available at: www.versprite.com/security-offerings/appsec/application-threat-modeling/ (accessed 10.01.2020).
9. The official site of OWASP (2020), *Application Threat Modeling*, available at: www.owasp.org/index.php/Threat_Risk_Modeling (accessed 11.01.2020).
10. The official site of Microsoft (2019), *MSDN*, available at: www.docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach (accessed 10.01.2020).
11. The official site of Microsoft (2010), *MSDN Threat Modeling Web Applications*, available at: [www.docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648006\(v=pandp.10\)?redirectedfrom=MSDN](http://www.docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648006(v=pandp.10)?redirectedfrom=MSDN) (accessed 10.01.2020).
12. The official site of Threatmodeler (2018), *Getting Started with Threat Modeling: How to Identify Your Mitigation Strategy*, available at: www.threatmodeler.com/getting-started-with-threat-modeling-how-to-identify-your-mitigation-strategy/ (accessed 10.01.2020).

Надійшла до редколегії 15.01.2020

Схвалена до друку 11.02.2020

Відомості про авторів:

Гапон Андрій Олександрович
аспірант
Харківського національного
університету радіоелектроніки,
Харків, Україна
<https://orcid.org/0000-0003-2560-7426>

Information about the authors:

Andrii Hapon
Doctoral Student
of Kharkiv National
University of Radio Electronics,
Kharkiv, Ukraine
<https://orcid.org/0000-0003-2560-7426>

Федорченко Володимир Миколайович

кандидат технічних наук
доцент кафедри Харківського національного
університету радіоелектроніки,
Харків, Україна
<http://orcid.org/0000-0001-7359-1460>

Volodymyr Fedorchenko

Candidate of Technical Sciences
Senior Lecturer of Kharkiv National
University of Radio Electronics,
Kharkiv, Ukraine
<http://orcid.org/0000-0001-7359-1460>

Поляков Андрій Олександрович

кандидат технічних наук
доцент кафедри Харківського національного
економічного університету ім. С. Кузнеця,
Харків, Україна
<https://orcid.org/0000-0003-1805-9011>

Andrii Polyakov

Candidate of Technical Sciences
Senior Lecturer of Simon Kuznets
Kharkiv National University of Economics,
Kharkiv, Ukraine
<https://orcid.org/0000-0003-1805-9011>

ПОДХОДЫ К ПОСТРОЕНИЮ МОДЕЛИ УГРОЗ ДЛЯ АНАЛИЗА БЕЗОПАСНОСТИ ОТКРЫТОГО ПРОГРАММНОГО КОДА

А.А. Гапон, В.Н. Федорченко, А.А. Поляков

Обеспечение безопасности программного продукта с открытым исходным кодом является актуальной проблемой, потому что даже в проектах с закрытым исходным кодом могут присутствовать open source библиотеки, что делает возможным появление уязвимости в них. Среди методов, используемых для выявления уязвимостей, стоит выделить моделирование угроз, так как этот метод позволяет уже на ранних этапах разработки программного кода принять меры, снизить расходы на ликвидацию уязвимостей. Подбор соответствующего подхода при построении модели угроз зависит от специфики проекта, ресурсов, а также квалификации администраторов.

Ключевые слова: модель угроз, открытый исходный код, STRIDE, OCTAVE, TRIKE, PASTA, VAST.

APPROACHES TO THE CONSTRUCTION OF THE THREAT MODEL FOR ANALYSIS OF SECURITY OF THE OPEN SOFTWARE CODE

A. Hapon, V. Fedorchenko, A. Polyakov

Ensuring the security of an open source software product is an urgent problem, because even in closed source projects open source libraries may be present, which makes vulnerabilities possible in them. Among the methods used to identify vulnerabilities, it is worth highlighting threat modeling, since this method allows you to take measures at the early stages of developing program code and reduce the cost of eliminating vulnerabilities. The selection of an appropriate approach when building a threat model depends on the specifics of the project, resources, and also the qualifications of administrators. There are various threat modeling methodologies that provide the foundation for a complex security process for developing software code. Although each of these threat modeling methodologies has its own strengths and can be used to identify, evaluate, and prioritize the elimination of potential threats, they are lacking in taking into account the features of the analysis of program code and the features of the development process. As processes become more complex, it becomes increasingly difficult to implement effective threat modeling. Security is an important aspect and an integral part of all stages of software development. Both in open source software and closed source software, reliability depends on certain aspects of the design and quality of the software development. The quality and reliability of open source software can be assessed by viewing the software source code once it becomes publicly available.

The analysis of the models of construction of threats taking into account the specifics of the modern development process using the agile methodology and scrum process allows to conclude about the low efficiency and complexity of their use. When building an open source security threat model, they are not effective enough at the development stage because they are not specialized in the field but are generic approaches. Therefore, from our point of view, it is preferable to use recommendations to form a threat model that take into account the vulnerabilities in the code.

Keywords: threat model, open source, STRIDE, OCTAVE, TRIKE, PASTA, VAST.