Дмитрієва О. А.,
доктор технічних наук, професор, зав. кафедри прикладної математики та інформатики Донецького національного технічного університету, м. Красноармійськ

Гарбуз В. І.,
студент Донецького національного технічного університету, м. Красноармійськ

СИСТЕМА АВТОМАТИЗАЦІЇ РЕГРЕСІЙНОГО ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Вступ.

Останнім часом в області інформаційних технологій у зв'язку з постійним зростанням конкуренції особливої актуальності набувають питання автоматизованого тестування інформаційної безпеки, як діяльності, що підвищує якість програмних продуктів [1]. На сьогоднішній день одним з необхідних видів тестування вразливостей, що застосовується в ході процесу розробки і модифікації програмних продуктів, є регресійне тестування [2], яке являє собою повторне тестування частини програми, що залежить від змін, внесених до неї.

Однак в умовах великих проектів промислового масштабу тестування в цілому і регресійне тестування зокрема є дуже дорогою і трудомісткою процедурою [3–4]. Це обумовлено необхідністю проводити регресійне тестування у разі внесення навіть незначних змін в код програми, в той час, як процес регресійного тестування може включати в себе виконання досить великої кількості тестів на скоригованій версії програми. І, незважаючи на те, що зусилля, необхідні для внесення невеликих змін, як правило, мінімальні, вони можуть вимагати чималих зусиль для перевірки якості зміненої програми. Тим не менш, надійна та ефективна розробка і супровід програмного забезпечення неможливі без регресійного тестування. Одним з очевидних виходів у ситуації, що склалася, є автоматизація процесу тестування [5].

Мета роботи полягає у розробці, обґрунтуванні і реалізації багатопотокової автоматизованої крос-платформної програмної системи регресійного тестування вразливостей інформаційної безпеки в серверних додатках,що сприяє мінімізації витрат на моніторинг розвитку програмного продукту.

За допомогою легкого масштабування тестових наборів, швидкого аналізу результатів та зручного формування звітів процес автоматизованого тестування стає більш близьким до оптимального, а витрати на ресурси і час зменшуються.

Програмна система регресійного тестування інформаційної безпеки.

Регресійне автоматизоване тестування (далі – РАТ) – це програмна система, яка призначена для автоматизації процесу пошуку і ліквідації вразливостей програмного продукту, аналізу поводження з погляду безпеки для того, щоб активно і з мінімальними зусиллями моніторити розвиток продукту. Тестування проводиться в нічний час доби, що дозволяє зафіксувати проблему до виходу програмного забезпечення. Регресійне автоматизоване тестування не відслідковує нові проблеми, що стосуються безпеки, а всього один раз тестує нову версію програмного забезпечення на відомих проблемах, які були раніше зафіксовані. Таке регресійне випробування виконується для того, щоб визначити, чи не викликало усунення дефекту в проекті змін, які стосуються безпеки і чи стало воно ефективним. Запуск пошуку вразливості та аналізу шляхів їх усунення вимагає певного рівня знань в області інформаційної безпеки, що робить дані дефекти неможливими для аналізу персоналом, який займається забезпеченням якості тестування. Також є проблема в поєднанні регресійного тестування з виконанням поточних завдань. Взявши дані проблеми до уваги і розуміючи усі складності регресійного тестування, було прийняло рішення стосовно пошуку нових шляхів автоматизації. Розроблена система озволила розв’язати цю проблему.

Регресійне автоматизоване тестування розроблена на інтерпретованій високорівневій мові програмування Python. Основною причиною вибору Python

стала його гнучкість, відсутність високих вимог до продуктивності та простота інтеграції з концептуальним кодом, попередньо призначеним для пошуку вразливостей безпеки, які, як правило, також написані на Python.

Регресійне автоматизоване тестування складається з основної програми RAT.py, яка знаходиться в корені ієрархії,та множини підсистем, що містять весь необхідний опис, який відноситься до кожного набору тестів, даних і концептуального коду. Цей підхід дозволяє легко розширити тестову базу програми і керувати даними в цілому.

Кожна частина концептуального коду вносить свої зміни і запускається динамічно для конкретного середовища тестування. Це дозволяє зменшити складність системи, даючи можливість операційним системам моніторити ефективність багатопоточності, де відбувається розпаралелювання випробувань. Додавання кожного нового тесту вимагає незначної зміни в існуючому концептуальному коді, як правило для того, щоб у PAT була можливість автоматичної переконфігурації.

Для того, щоб знизити кількість часу та зусиль, які необхідні на розробку та реалізацію, була розроблена власна RAT-мова сценаріїв, яка значно спрощує і прискорює процес автоматизації регресійного тестування безпеки. Заснована дана мова на XML-розмітці, що дозволяє з легкістю помічати та фіксувати будь-яке перетворення даних, регулярних виразів, вбудованих в Python сценаріїв, і це забезпечує більше можливостей та гнучкість у поєднанні з простотою.

PAT сценарії використовують низькорівневі мережеві сокети, які дозволяють підтримувати не тільки HTTP протокол, але і будь-який інший, що засновується на TCP/IP. Також легко підтримувати UDP, але в даний час в такій функціональності немає необхідності.

Через недостатність обсягу даних, необхідних для зберігання і доступу, PAT використовує XML файли для зберігання і управління метаданими. У порівнянні з використанням системи управління даними SQL таке зберігання дозволяє зменшити складність системи і підвищити рівень зручності обслугову-

вання, тому що не вимагає додаткового сервера бази даних або програмного забезпечення для управління.

Після запуску тестового набору PAT генерує звіт у форматі, який обирається користувачем. Це може бути HTML сторінка, текстовий файл або JenkinsJunit XML-файл.

Синтаксис PATскрипта. PAT сценарій був розроблений як засіб для швидкої розробки регресійних тестів безпеки. Його синтаксис представляє суміш XML, TCP трафіку (наприклад, HTTP запити) та спеціальної розмітки для перетворення даних запиту. Також PAT сценарій використовує механізм Smart Match-and-Fetch для того, щоб будувати динамічні запити до серверу.

Структура скрипта. Кожний PAT сценарій XML документу містить кореневий елемент <Exploit>. Також він повинен містити один або декілька елементів <Step>.Кожен елемент <Step> – це, в основному, один запит до сервера з метаданими його частин і з тим, які повинні бути зміни PATскрипту, а також про те, що дані повинні бути перехоплені з відповіді сервера до поточного запиту, щоб мати змогу повторно використовувати запити в майбутньому (наприклад, значення cookie, XSRF або сесії токенов).

Кожен елемент <Step> повинен містити один елемент <Request> - скриптовий движок буде відправляти зміст одразу до сервера через TCP сокет. Кожен елемент <Step> може містити в собі <Matches>, елементи з одним або більше <Matches> називаються суб-елементами. Кожен суб-елемент<Matches> описує регулярний вираз, згідно з яким PAT проводить перехоплення інформації від відповіді сервера на виданий запит і визначає назву цієї інформації для подальшого використання в інших запитах (рисунок 1).

```
<Matches>
            <Match name = "@@LWSSO@@">LWSSO_COOKIE_KEY=.*?;</Match>
</Matches>
```

Рисунок 1. – Зміст <Matches>

Недостатньо просто повторити серію статичних запитів до серверу для виявлення вразливостей. Великий обсяг даних повинен бути перехоплений з попередніх запитів і використовуватися повторно в майбутніх запитах. Деякі з даних запиту перед його повторним використанням також повинні бути оброблені будь-яким чином. Для швидкого вирішення даних проблем був розроблений механізм Smart-Match-and-Fetch. Він дозволяє користувачеві системи самому вказати регулярний вираз, який входить у відповідь сервера і буде порівнюватися з усіма необхідними даними.

Кожному збігу надається ім'я (назва), яке має бути унікальним і відрізнятися від поточних параметрів імен в TestSet.xml. Ім'я, розташоване між двома послідовностями знаків «@@», буде служити маркуванням PAT сценарію з фактичними перехопленими даними. Перед тим, як помістити дані в тіло наступного HTTP запиту, вони повинні бути оброблені. Для цього передбачено режим для перетворення даних з PATScripting. У даний час підтримуються такі трансформатори даних:

b64Dec / b64Enc – декодує / кодує дані, використовуючи схему кодування Base64;

HexDec / hexEnc – декодує / кодує дані з / в числове подання;

urlDec / urlEnc – декодує / кодує дані, використовуючи кодування URL;

xmlDec / xmlEnc – екранує / виділяє дані для безпечної вставки в XML / HTML документ;

lower / upper – перетворює всі букви нижнього або верхнього регістру;

MD5 / SHA1 / sha256 – обчислює хеш-функцію для даних, які повертають бінарний код.

У випадку, якщо стануть за необхідні більш складні перетворення, PAT сценарії підтримують ланцюжок програмного перетворення даних.

Висновки.

У даній роботі на основі аналізу існуючих концепцій, методів та засобів автоматизації тестування і тенденцій розвитку технологій запропоновано новий

метод розробки автоматизованих тестів і створені необхідні засоби програмної підтримки. Розроблений метод відрізняє висока ефективність розроблених автоматизованих тестів, універсальність в застосуванні і відносно низька трудомісткість процесу впровадження автоматизованого тестування. Передбачено багатопоточне тестування, можливість масштабування тестових наборів, інтегрування з тестами, розробленими на будь яких скриптових мовах з доступними інтерпретаторами, обробку та аналіз результатів тестування. Проведено порівняльний аналіз результативності регресійного тестування вразливостей інформаційної безпеки в серверних додатках та визначено найвищу ефективність при використанні алгоритмів Левінштайна. Створено систему автоматизованого тестування, що здійснює підтримку автоматизованих тестів, розроблених за допомогою нової методики і з використанням мови програмування Python для забезпечення крос-платформності. Забезпечено генерацію звітів у текстовому форматі, форматованій HTML сторінці та JUnit XML, придатному для парсингу CI (ContinuousIntegration) системами.

Список використаної літератури

1. Полаженко С. Актуальность вопросов тестирования безопасности и защищённости программных продуктов [Электронный ресурс] / С. Полаженко. – Режим доступа : http://www.software-testing.ru/library/testing/security/86-security-testing.

2. Регрессионное тестирование (regressiontesting) [Электронный ресурс]. – Режим доступа : http://www.javenue.info/post/24.

3. Abdurazik A. Using UML Collaboration Diagrams for Static Checking and Test Generation [Electronic resource] / A. Abdurazik, J. Offutt // Lecture Notesin Computer Science. – Access mode : http://link.springer.com/chapter/10.1007/3-540-40011-7_28.

4. Didkovska M. Criteria for integration testing of component-based software / M. Didkovska // Электроника и связь. – К., 2004. – № 23. – С. 90–94.

5. Offutt J. Generating tests from UML specifications [Electronic resource] / J. Offutt, A. Abdurazik // Second International Conference on the Unified Modeling Language. – FortCollins, CO, IEEE Computer Society Press, 2009. – C. 416–429.