200 РОКІВ
ОСВІТНІХ ТРАДИЦІЙ

ТЕЗИ
70-ої наукової конференції професорів, викладачів, наукових працівників, аспірантів та студентів університету

Том 2

ПОЛТАВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ ІМЕНІ ЮРІЯ КОНДРАТЮКА

Міністерство освіти і науки України
Північно-Східний науковий центр НАН України та МОН України
Полтавський національний технічний університет
імені Юрія Кондратюка

Тези

70-ої наукової конференції професорів, викладачів, наукових працівників, аспірантів та студентів університету

Том 2

23 квітня – 18 травня 2018 р.

Полтава 2018

УДК 043.2
ББК 448лО

Тези 70-ої ювілейної наукової конференції професорів, викладачів, наукових працівників, аспірантів та студентів університету. Том 2. (Полтава, 23 квітня – 18 травня 2018 р.) – Полтава: ПолтНТУ, 2018. – 380 с.

У збірнику тез висвітлені результати наукових досліджень професорів, викладачів, наукових працівників, аспірантів та студентів університету.

можуть заважати зміни коду, а не сприяти йому. Ситуація не така вже й рідкісна, а тому й далеко не остання в списку неприємностей.

Але найгірше те, що навіть тести, успішно виконані й перевиконані після внесення усіляких змін, не дають ніякої гарантії, що в коді немає помилок. І справа тут навіть не в підгонці тестів під код, а в людському факторі. Іноді дещо змінивши значення вхідних параметрів в рамках тестів програміст може, сам того не бажаючи, пропустити помилку в тестованому коді або, гірше того, внести її туди. Але, від помилок, пов'язаних з людським фактором, не застрахований ніхто і ніколи.

Вище я згадував, що є спеціальні інструменти, які допомагають програмісту писати unit-тести, нижче наведено основні із них.

Назви більшості популярних фреймворків для unit-тестування утворені від назви мови, на якій пишеться програмний код, і слова "unit". Наприклад, для Java це JUnit, для .NET'овскіх мов - NUnit, для Delphi - DUnit, для C ++ - CPPUnit, для PHP - PHPUnit, для Python - PyUnit, для ActionScript - AsUnit ... Ну і так далі. А якщо не знаходиться фреймворк для unit-тестування для тієї мови, на якій пишете, швидше за все, ця мова або ще не придумали, або вже ніде не використовують, або це Асемблер.

Що ж, як бачите, в самій ідеї unit-тестування немає нічого складного. Та й в практиці, мабуть, теж. Потрібно тільки знайти час та сили на вивчення тестіровочного фреймворка для використовуваного вами мови програмування і для того, щоб писати самі тести.

Література

1. KV.by [Електронний ресурс].– Режим доступу: https://www.kv.by/archive/ index2009421107.htm.

2. Про Тестинг - Тестирование Программного Обеспечения [Електронний ресурс].– Режим доступу: http://www.protesting.ru/testing/levels/component.html.

3. Хабрахабр [Електронний ресурс].– Режим доступу: https://habrahabr.ru /post/169381/.

УДК 005.8

О.О. Бородіна, старший викладач
кафедра комп'ютерних та інформаційних технологій і систем
В. М. Фіней, студент 302-ТН групи
Полтавський національний технічний
університет імені Юрія Кондратюка

ПРИЗНАЧЕННЯ ТЕСТУВАННЯ БЕЗПЕКИ. ВИДИ УРАЗЛИВОСТЕЙ

Тестування програмного забезпечення – оцінка розроблюваного програмного забезпечення / продукту, з метою перевірити його можливості та відповідність очікуваним результатам. Тестування програмного забезпечення є невід'ємною частиною циклу розробки програмного забезпечення [1].

Комп'ютерні системи дуже часто є мішенню незаконного проникнення. Під проникненням розуміється широкий діапазон дій: спроби хакерів проникнути в систему з спортивного інтересу, помста розгніваних службовців, злом шахраями для незаконної наживи.

Тестування безпеки перевіряє фактичну реакцію захисних механізмів, вбудованих в систему, на проникнення. В ході тестування безпеки випробувач грає роль зломщика. Йому дозволено все:

– спроби дізнатися пароль за допомогою зовнішніх засобів;

– атака системи за допомогою спеціальних утиліт, які аналізують захист;

– придушення системи (в надії, що вона відмовиться обслуговувати інших клієнтів);

– цілеспрямоване введення помилок в надії проникнути в систему в ході відновлення;

– перегляд несекретних даних в надії знайти ключ для входу в систему.

Звичайно, при необмеженому часі й ресурсах хороше тестування безпеки зламає будь-яку систему. Завдання проектувальника систем – зробити ціну проникнення вищою, ніж ціна одержуваної в результаті інформації.

Основним завданням системного тестування є перевірка як функціональних, так й не функціональних вимог в системі в цілому. При цьому виявляються дефекти, такі як неправильне використання ресурсів системи, непередбачені комбінації даних, несумісність з оточенням, непередбачені сценарії використання, відсутня або неправильна функціональність, незручність використання та ін. Для мінімізації ризиків, пов'язаних з особливостями поведінки в системи в тому чи іншому середовищі, під час тестування рекомендується використовувати оточення максимально наближене до того, на яке буде встановлено продукт після видачі.

В даний час найбільш поширеними видами уразливості в безпеці програмного забезпечення є [2]:

XSS (Cross-Site Scripting) – вид уразливості програмного забезпечення (Web додатків), при якій, на генерованої сервером сторінці, виконуються шкідливі скрипти, з метою атаки клієнта. Самі по собі XSS атаки можуть бути дуже різноманітними. Зловмисники можуть спробувати вкрасти куки, перенаправити на сайт, де відбудеться більш серйозна атака, завантажити в пам'ять будь-який шкідливий об'єкт та ін., всього навсього розмістивши шкідливий скрипт на сайті.

XSRF / CSRF (Request Forgery) – вид уразливості, що дозволяє використовувати недоліки HTTP протоколу, при цьому зловмисники працюють за такою схемою: посилання на шкідливий сайт встановлюється на сторінці, що користується довірою у користувача, при переході по шкідливому посиланні виконується скрипт, який зберігає особисті дані користувача (паролі, платіжні дані та ін..), або відправляє СПАМ повідомлення від особи користувача, або змінює доступ до облікового запису користувача, для отримання повного контролю над нею.

Code injections (SQL, PHP, ASP і т.д.) – вид уразливості, при якому стає можливо здійснити запуск виконуваного коду з метою отримання доступу до системних ресурсів, несанкціонованого доступу до даних або виведення системи з ладу.

Server-Side Includes (SSI) Injection – вид уразливості, що використовує вставку серверних команд в HTML код або запуск їх безпосередньо з сервера.

Authorization Bypass – вид уразливості, при якому можливо отримати несанкціонований доступ до облікового запису або документам іншого користувача. Користувач А може отримати доступ до документів користувача Б. Припустимо, є реалізація, де при перегляді свого профілю, що містить конфіденційну інформацію, в URL сторінки передається userID, а даному випадку є сенс спробувати підставити замість свого userID номер іншого користувача. І якщо ви побачите його дані, значить ви знайшли дефект.

Прикладів вразливостей й атак існує величезна кількість. Навіть провівши повний цикл тестування безпеки, не можна бути на 100% впевненим, що система по-справжньому безпечна. Але можна бути впевненим в тому, що відсоток несанкціонований проникнень, крадіжок інформації, втрат даних буде на кілька порядків нижче, ніж у тих хто не проводив тестування безпеки [3].

Література

1. Блог Web программиста [Електронний ресурс] – Режим доступу:: http://juice-health.ru/program/software-testing/495-software-testing-methods

2. Getbug [Електронний ресурс] – Режим доступу: http://getbug.ru/testirovanie-bezopasnosti-saytov-i-prilozheniy/

3. Тестирование безопасности [Електронний ресурс] – Режим доступу: https://poisk-ru.ru/s11582t2.html

СЕКЦІЯ КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В СИСТЕМ