додаток для створення і заповнення анкет користувачів і генерації звітів за результатами оцінки ризиків.

У методі RiskWatch як показники для оцінки та управління ризиками використовуються прогнозовані середньорічні втрати (Annualized Loss of Expectancy, ALE) та оцінка повернення від інвестицій (Return on Investment, ROI).

Отже, можна зробити висновок, що можуть застосовувати комбінований підхід як найкращий підхід до оцінки ризиків інформаційної безпеки. Цей підхід включає в себе використання набору методів аналізу ризиків, що складається з методів, які найкраще підходять для конкретних областей конкретної організації. Наприклад, метод Дельфі добре підходить для CRAMM, оскільки він дозволяє враховувати специфіку конкретного об'єкта. Це дозволяє підвищити точність оцінки ризиків інформаційної безпеки за рахунок обліку всіх характеристик інформаційної системи. Тому розглянуті загальні методи аналізу ризиків можуть бути успішно застосовані до інформаційної безпеки, якщо їх адаптувати і вдосконалити.

Список використаних джерел:

1. Соціальна інженерія // Вікі-знання/Тернопіл. нац. техн. ун-т ім. Івана Пулюя. URL: http://wiki.tstu.edu.ua/Соціальна_інженерія.

СПОСОБИ ВИЯВЛЕННЯ SQL-ІН'ЄКЦІЙ

Васильєв Б. Г.

студент 1-го курсу магістратури
факультету кібербезпеки та інформаційних технологій
Національного університету «Одеська юридична академія»

Перші відомості про SQL-ін'єкцію з'явилися у відкритому доступі в статті «Вразливості вебтехнологій NT», опублікованій ще в 1998 році. З 2000 року дану техніку почали широко застосовувати кіберзлочинці для взлому інтернет-сайтів. Сьогодні є багато технік експлуатації SQL-ін'єкцій для різних СКБД та застосунків на їх основі. За минулі кілька років, атаки, які були спрямовані на різні Web застосунки вимагали особливої уваги з боку захисного персоналу [1]. Все це відбувається через те, що наскільки б сильним не був між мережний захист або наскільки старанно б не латали "дірки" в існуючих програмах, все одно може статися, що розробники ПЗ не досить добре захистили свої програми, і зловмисники можуть використовувати це для злому порту. Існує два найбільш частовикористовуваних методів для нападів: SQL ін'єкція і міжсайтовий скріптніг.

SQL ін'єкція належить до методики вставки мета-симолів і SQL команд в поля введення на Web сторінках, у результаті чого відбуваються зміни виконання кінцевих SQL запитів [2, 27]. Такі напади в основному спрямовані на Web-сервера. Принцип дії міжсайтового скриптингу полягає у використанні

знаходження шкідливих програм в URL , через натискання користувачем на URL посилання.

У наших прикладах ми будемо використовувати програму IDS Snort і складемо регулярні вирази, відповідно правил щодо знаходження подібних різновидів атак. Написана величезна кількість правил для знаходження таких атак. Якщо треба знайти будь-який варіант SQL атаки, то треба уникнути спроби вводу будь-якого з мета-символів SQL типу одинарної лапки ('), крапки з комою (;) або подвійного тире (–). Точно такий же параноїдальний спосіб знаходження CSS атак складається з виявлення кутових дужок (<>), що вказують на HTML-тег. Але при подібних умовах є ймовірність того, що більшість вірних виразів, стануть виявлені як вірогідність атак. Щоб уникнути цього, слід дотримуватись більш конкретних умов перевірки і при цьому не приводити до неправильної ідентифікації. Кожна з таких умов може застосовуватися як безпосередньо, так і за допомогою команди Snort сигнатур, використовуючи ключове слово pcre. Відповідні сигнатури можуть також взаємодіяти з утилітами типу grep для перевірки файлів реєстрації на Web серверах. Але такий спосіб може застосовуватися, лише якщо застосунки використовують GET запити, тому що дані про POST запитах не зберігаються в файлах реєстрації.

Значним при виборі регулярного виразу для знаходження SQL ін'єкцій є те, що може застосувати зловмисник в полі введення або поле cookie. Ми повинні аналізувати всі вхідні дані користувача як підозрілі. Як було згадано раніше, найпростіше регулярний вираз, що застосовується для знаходження SQL ін'єкцій, змушений знаходити певні мета-символи SQL типу одинарної лапки або подвійного тире. Для знаходження таких символів і їх шістнадцяткових еквівалентів може застосовуватися наступні вирази:

Регулярний вираз для визначення SQL мета-символів:

/ (\% 27) | (\ ') | (\ - \ -) (\% 23) | (#) / ix </ TD <tr>

Пояснення: для початку виявляється шістнадцятковий еквівалент одинарної лапки або безпосередньо сама лапка, а також наявність подвійного тире. Відповідні мето-символи MS SQL Сервер і Oracle, що вказують на початок коментаря (тобто все що слід за цими символами - ігнорується). Додатково, в разі застосування MySQL, треба перевірити наявність '*' або його шістнадцятирічного еквіваленту. Звернути увагу на те, що при цьому не треба перевіряти шістнадцятковий еквівалент подвійного тире, тому що він не є мета-символом HTML і не буде закодований браузером. Також, якщо зловмисник буде намагатися вручну внести зміни замість подвійного тире на його шістнадцятковий еквівалент (використовуючи проксі типу Achilles), то в нападі з використанням SQL ін'єкції відбудеться збій. Зазначений вище вираз необхідно наступним чином додати до нового правила Snort:

alert tcp $ EXTERNAL_NET any -> $ HTTP_SERVERS $ HTTP_PORTS
(msg: "SQL Injection - Paranoid";

flow: to_server, established; uricontent: ". pl";

pcre: "/ (\% 27) | (\ ') | (\ - \ -) | (% 23) | (#) / i";

classtype: Web-application-attack; sid: 9099; rev: 5;) </ TD <tr>

У нашому випадку, ключове слово uricontent має значення".pl", тому що в середовищі, в якій проводяться наші випробування, CGI сценарії написані на Perl. Залежно від певного застосування це значення може бути ".php", або ".asp", або ".jsp", та ін. За допомогою описаного вище вираження ми виявляємо присутність подвійного тире, тому що бувають ситуації, коли SQL ін'єкція можлива навіть без застосування одинарної лапки. Прикладом може служити SQL запит з пропозицією where, що містить тільки числові значення:

select value1, value2, num_value3 from database

where num_value3 = some_user_supplied_number

В цьому випадку зловмисник може здійснити додатковий SQL запит, виконавши введення наступного вираження:

3; insert values into some_other_table

І нарешті, pcre модифікатори 'i' і 'x' використовуються, відповідно, для ігнорування регістра символів і пробілів (або знаків табуляції).

Описана вище сигнатура може бути доповнена для виявлення знака "крапка з комою". Однак крапка з комою може цілком нормально бути частиною HTTP трафіку. Для зменшення ймовірності неправильної ідентифікації, така сигнатура може бути модифікована, щоб спершу виявляти символ "=". Введення даних користувачем зазвичай відбувається як POST або GET запит, в якому вхідні поля будуть виглядати наступним чином:

username = some_user_supplied_value & password =

some_user_supplied_value

Тому при спробі SQL ін'єкції, даних користувача передував би знак = або його шістнадцятковий еквівалент.

Модифіковані регулярні вирази для виявлення SQL метасимволів

/ ((\% 3D) | (=)) [^ \ n] * ((\% 27) | (\ ') | (\ - \ -) | (\% 3B) | (;)) / i < / TD <tr>

Пояснення: відповідна сигнатура спочатку виявляє символ "=" або його шістнадцятковий еквівалент (% 3D). Після цього, ігноруючи роздільники рядків і прогалини, проводитиметься пошук входжень одинарної лапки, подвійного тире або крапки з комою. Зазвичай спроби SQL-ін'єкцій зводяться до використання одинарних лапок для управління початковим запитом. У більшості прикладів, наведених в статтях присвячених цього типу нападів, використовується рядок 1'or '1' = '1. Однак від виявлення цього рядка можна легко ухилитися, використовую значення типу 1'or2> --. Таким чином, єдиною постійною частиною таких виразів залишається символьне значення, що супроводжується одинарною лапкою, з подальшим за нею словом "or".

Регулярний вираз для типових SQL ін'єкцій

/ \ W * ((\% 27) | (\ ')) ((\% 6F) | o | (\% 4F)) ((\% 72) | r | (\% 52)) / ix </ TD <tr>

пояснення: \ W * - нуль або інші алфавітно-цифрові символи або символи підкреслення

(\% 27) | \ '- одинарна лапка або її шістнадцятковий еквівалент

(\% 6F) | o | (\% 4F)) ((\% 72) | r | (\% 52) - слово 'or' в різних комбінаціях верхнього і нижнього регістрів і їх шістнадцяткові еквіваленти.

Часто, застосування SQL-ін'єкцій при атаках на різні бази даних супроводжується використанням ключового слова "union". При використанні описаних раніше регулярних виразів ми можемо лише визначити наявність одинарних лапок або інших мета-символів SQL. Але далі ми будемо модифікувати наші регулярні вирази для визначення одинарної лапки і ключового слова "union". Те ж саме можна зробити і для інших ключових слів SQL типу 'select', 'insert', 'update', 'delete' та ін.

Регулярне вираження для визначення SQL ін'єкції з використанням ключового слова "union"

/ ((\% 27) | (\ ')) union / ix

(\% 27) | (\ ') - одинарна лапка і її шістнадцятковий еквівалент

union - ключове слово «union»

Подібні вирази можуть бути написані і для інших SQL запитів типу > select, insert, update, delete, drop та ін.

У разі якщо хакер виявив, що WEB додаток вразливе до SQL-ін'єкції, він почне експлуатувати цю уразливість. Якщо потрібна база даних зберігатися на MS SQL сервері, то хакер, швидше за все, спробує виконати одну їх багатьох небезпечних збережених або розширених збережених процедур. Назви таких процедур починаються, відповідно, з 'sp' або 'xp'. Як правило, він спробував би виконати розширену процедуру 'xp_cmdshell', що дозволяє виконувати через SQL сервер команди оболонки Windows. Права доступу до яких виконатися ця процедура будуть такі ж як і у облікового запису з якої запущений SQL сервер (зазвичай Local System). Додатково, зловмисник спробує внести зміни до реєстру за допомогою використання процедур xp_regread, xp_regwrite, і т.д.

Регулярний вираз для визначення SQL-ін'єкції на MS SQL сервері

/ Exec (\ s | \ +) + (s | x) p \ w + / ix </ TD <tr>

пояснення: exec - ключове слово необхідне для запуску збереженої або розширеної процедури

(\ S | \ +) + - один або кілька пробілів або їх еквіваленти закодовані HTTP

(S | x) p - символи 'sp' або 'xp' необхідні для ідентифікації збереженою або розширеної процедури, відповідно

\ W + - один або кілька алфавітно-цифрових символів або символів підкреслення, необхідних для завершення написання імені процедури.

Отже, деякі з таких виразів просто параноїдальні і видають попередження навіть при натяках на атаку. Але існує ймовірність, що використання таких сигнатур призведе до неправильної ідентифікації виразів. Для попередження цього,

необхідно змінити тривіальні сигнатури, забезпечивши їх додатковими перевірками і зробивши більш точними. Ми рекомендуємо використовувати ці регулярні вирази в якості відправної точки при налаштуванні будь-якої системи виявлення вторгнень.

Список використаних джерел:

1. Cyber Attacks 2020! : URL: https://sectigostore.com/blog/cyber-attacks-2020-notable-2020-cyber-attacks/
2. Корнага Я., Базака Ю., Мар'єнко Є. Способи оптимізації SQL–запитів для поліпшення роботи з базою даних в високонавантажених системах. *Адаптивні системи автоматичного управління.* Київ, 2020. Том 2 № 37. С.26-30.

Науковий керівник: доцент Лобода Ю.Г.

ЗАСОБИ ТА МЕХАНІЗМИ ПРОТИДІЇ DDOS-АТАКАМ

Величко С.В.

студентка 1 курсу факультету кібербезпеки та інформаційних технологій Національного університету «Одеська юридична академія»

DDoS-атака або атака типу «відмова в обслуговуванні» (Distributed Denial of Service) спрямована на комп'ютерну систему з метою створення несприятливих умов для використання певних ресурсів або сервісів користувачами [1]. Класифікація атак відповідає базовій еталонній моделі взаємодії відкритих систем OSI [2]. На найнижчому мережевому (третьому) рівні для атаки відбуваються дії з використанням UDP-пакетів. Ідея атаки полягає у надсиланні безлічі UDP-пакетів (як правило, великого об'єму) на обрані або випадкові номери портів віддаленої комп'ютерної системи. При цьому для кожного отриманого пакета віддалена система повинна визначити відповідний додаток, переконатися у відсутності його активності та відправити відповідне повідомлення по протоколу ICMP «адресат недоступний». В результаті атакований хост виявиться перенавантаженим: у протоколі UDP механізм запобігання перенавантаженням відсутній, тому після початку атаки шкідливий трафік швидко захопить всю доступну смугу пропускання каналу хоста, перешкоджаючи при цьому трафіку від звичайного користувача.

На транспортному (четвертому) рівні атака полягає у відправці завеликої кількості SYN (*synchronize*) - запитів на встановлення підключень за протоколом TCP. При цьому зворотні пакети від обчислювальної системи з комбінацією флагів SYN+ACK (*acknowledges*) або ігноруються, або обробляються навмисно не вірно. Таким чином у системі створюється велика кількість напіввідкритих з'єднань збільшуючи при цьому чергу на підключення.

На рівні представлення даних (шостому рівні моделі взаємодії OSI) атака полягає у зловмисному використанні протоколу SSL. Для розшифрування даних