можливість розрахунку вартості захисту на довільну кількість років, що дозволяє більш точно врахувати вартість експлуатації в загальній вартості системи захисту; складний алгоритм оптимізації, що дає максимально ефективний варіант системи захисту навіть при великих вибірках контрзаходів; збереження та експорт результатів розрахунків в формати файлів Excel і txt; клієнт-серверна архітектура, що дозволяє працювати в багатокористувацькому режимі.

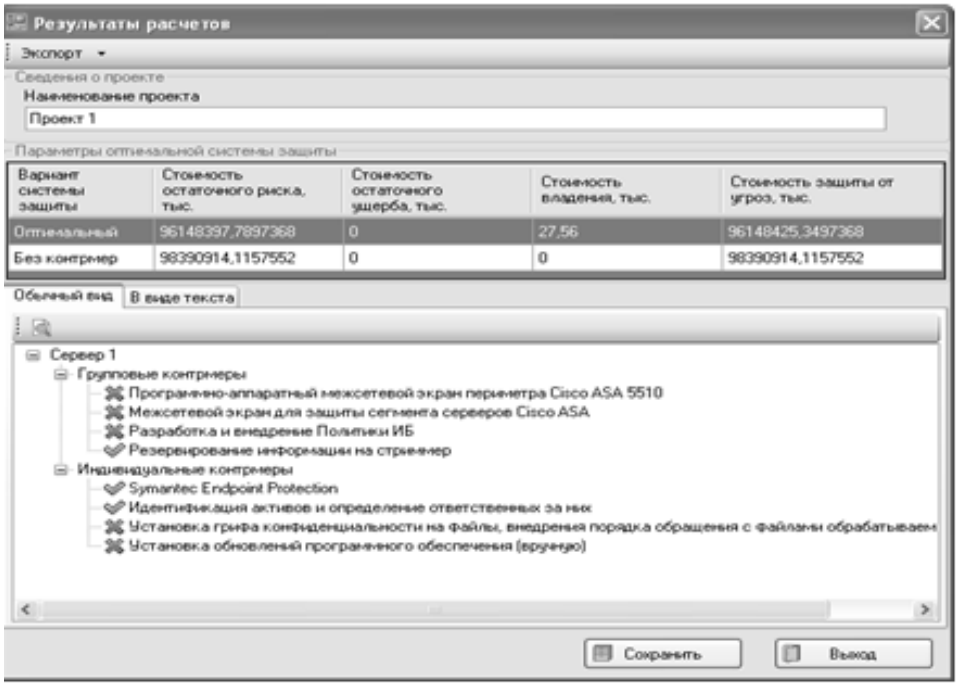

Рис. 4. Автоматична система BCM-Analyser

Висновок. Отже, на ринку інформаційних технологій існує велика кількість програмних засобів впровадження яких забезпечує гідне керування інформаційними потоками та виявлення інформаційно – технологічних ризиків пов'язані з діяльністю організації для якої застосовуються ті чи інші методи. Усі програми можна розділити на дві великі групи: програми, що застосовуються якісну та кількісну оцінку ризиків. Є комплекси, що об'єднують ці два підходи наприклад Digital Security Office, РискМенеджер, Oracle Crystal Ball, @Risk, але для більшої надійності потрібно чітко оцінити потреби організації виходячи із сфери їх діяльності та здійснювати вибір необхідного програмного забезпечення.

ЛІТЕРАТУРА

1. Proteus® Enterprise. // infogov.co.uk: сайт компанії «InfoGov» 2012.URL: http:// infogov.co.uk.
2. Программные продукты для анализа рисков. // iso27000.ru: русскоязычный информационный портал, посвященный вопросам управления информационной безопасностью 2012.URL: http:// iso27000.ru.
3. BCM-Analyser. // iradd.ru: сайт компанії «IRADD» 2012.URL: http:// iradd.ru.ru.

Надійшла: 19.10.2012 р.

Рецензент: д.т.н., професор Юдін О.К.

УДК 004.056.5(045)

Пархоменко І.І., Воскобойніков А.О.

ЗАХИСТ WEB-РЕСУРСІВ ВІД АТАК ТИПУ COMMAND EXECUTION

В статті розглянуто принцип забезпечення захисту web-сайту від атак command execution, а саме від SQL-ін'єкцій та XSS, доведено небезпечність даних атак та наведені можливі наслідки, до яких вони можуть призвести. Запропоновано варіант захисту від command execution через фільтрацію вхідних даних.

Ключові слова: атака виконання коду, SQL-ін'єкція, міжсайтовий скриптинг, безпека веб-сайту

Беручи до уваги інтенсивність розвитку мережі Інтернет, проблема захисту web-ресурсів від атак command execution є досить актуальною на сьогоднішній день. Для демонстрації цього достатньо здійснити пошук в Google за запитом «inurl:.php?id=», що дасть нам список ресурсів, в яких передача параметрів на сайт здійснюється відкрито через масив GET, тобто таких ресурсів, які є потенційно вразливими для атаки SQL-injection та XSS - різновиду атак command execution.

Надалі, вже на перших сторінках результатів пошуку можна знайти вразливий web-ресурсів. Для перевірки на вразливість достатньо в якості параметра передати web-серверу одинарну кавичку і отримавши у відповідь помилку представлену на рис. 1.

```
Query failed You have an error in your SQL syntax;check the manual that
Corresponds to your MySQL server version for the right syntax to use
Near '\' ORDER BYlastname' at line 1 SELECT * FROM person_old
WHERE id=5\' ORDER BY lastname
```

Рис.1 Повідомлення від сервера про помилку в синтаксисі SQL

- можна вважати, що даний ресурс рано чи пізно стане жертвою крекера чи хакера.

Постає питання, чим загрожує sql – injection?[4] Щоб відповісти на це питання – достатньо в якості параметра передати на вразливий ресурс sql запит. Власне, знаючи sql та розуміючи логіку її обробки в php зловмисник може досягнути будь-яких результатів – від простого розкриття паролів користувачів і до видалення всієї бази даних.

Наразі було показано лише один варіант розвитку подій з багатьох можливих, але цього цілком достатньо, щоб зацікавитись вирішенням проблеми захисту web-ресурсів від атак типу command-execution.

Отже, далі я спробую описати причину виникнення подібних вразливостей, та можливі способи захисту від них [6]. Результатом цієї роботи є рекомендації по захисту web-ресурсів від атак подібного роду та готова функція фільтрації даних, що повинна спростити забезпечення такого захисту.

Для того щоб, розробити захист, нам спочатку варто розібратись як працюють такі атаки. Почнемо з вже розглянутої нами атаки SQL-injection. [4,5]

Так виглядає звичайний запит до БД SQL в php:

```
$result = mysql_query('
                SELECT *
                FROM users
                        WHERE user_id = '.$user.'
                        AND
                        password ="'.$pswd.'"
                ,$db);
```

Рис.2 PHP запит до БД SQL

Варто звернути увагу, що в якості параметрів для запиту передаються змінні $user_id та $pswd. В залежності від потреб розробників, ці змінні можуть отримуватись як з форми, тобто їх може відправляти користувач, так і з інших джерел, наприклад, з іншого запиту до БД. Тобто, якщо id користувача дорівнює 1, а значення пароля дорівнює pswd, то після підставлення даних ми отримаємо наступний запит:

```
$result = mysql_query('
                SELECT *
                FROM users
                        WHERE user_id = 1
                AND
                        password ="pswd"
                ,$db);
```

Рис. 3 PHP запит до БД SQL з підставленими значеннями

Тепер уявимо, що «користувач» знає тонкощі sql та php і на сервер замість 1 відправив ще й знак подвійного тире «1’ --», який в мові запитів означає коментар, і блокує все, що знаходиться після нього. В результаті, попередній запит буде виглядати наступним чином:

```
$result = mysql_query(‘
                        SELECT *
                        FROM users
                                WHERE user_id = 1
                        ,$db);
```

Рис.4 Скомпроментований PHP запит до БД SQL

Отже, ввівши лище 1’-- хакер зміг пропустити перевірку пароля. Хоч це і дуже спрощений варіант атаки, проте він гарно демонструє як вона реалізується.

А трохи модифікувавши запит до наступного:

```
$result = mysql_query(‘
                        SELECT *
                        FROM users
                                WHERE user_id = null
                        UNION ALL
                        SELECT 1,2, group_concat(table_name),4,5
                        FROM information_schema.tables
                                WHERE table_schema=databse()’
                        ,$db);
```

Рис. 5 Приклад SQL інєкції

хакер зможе легко дізнатись імена таблиць бази даних. Власне, подальші маніпуляції з БД не є важкими. Наразі ми ознайомились з механізмом роботи SQL-injection, тепер перейдемо до наступного типу атак – Cross Site Scripting або скорочено – XSS. На відмінну від попередньої атаки, ця спрямована на користувачів, тобто на клієнтів web-серверу, а не на сам web-сервер.

Така атака можлива, коли зловмисник може вставити в код сторінки свій код для виконання. Відбувається це коли інформація, що виводиться на екран (наприклад, ім’я користувача) береться з БД, куди заноситься попередньо самим же користучем. Якщо користувач зможе занести в БД наступний код [3]:

```
<iframe src=“test1.php” style=“display:none;” ></iframe>
```

Рис. 6 HTML код підключення віддаленого файлу

Таким чином за допомогою html-тегу <iframe> хакер зміг підключити до сторінки вразливого сайту власний файл з шкідливим кодом. Даний файл буде завантажуватись кожного разу при відкритті сторінки.

Вміст файлу test1.php може бути будь-яким і залежити від фантазії зловмисника. Саме просте, що він може робити це - зчитувати активну сесію користувача та записувати її собі на сервері в файл.

Отже, ми розглянули основні способи реалізації атак sql-injection та xss. Аналізуючи наведену інформацію, можна зробити висновок, що загальною проблемою є відсутність достатньої фільтрації вхідних даних.

Тепер ми знаємо, захистом є проста фільтрація даних, проте треба розібратись, що саме варто фільтрувати.

Ось основні символи, які є небезпечними по відношенню до sql-injection [1]:

- Одинарні та подвійні кавички. Їх небезпеку я вже згадував;
- Знак рівності =;
- Крапка з комою “;”.

– Символи коментарів “--” та “/* */”.

По відношенню до xss, основними даними, що потребують фільтрації є [1]:

– Кутові дужки “<” та ”>”.
– Ключові слова такі як “script” та “javascript”.
– Символи лапок “ та ‘, слешів / та \, решітки #, відсотків %, амперсанта & та знаку рівності =.

В php є декілька функцій, які забезпечують фільтрацію, проте вони не завжди є зручними у використанні [2]. Надалі я наведу код можливої фільтрації даних на основі регулярних виразів:

```
function dataFilter($input , $type = 'alfa' , $sql = 'false'){
switch ($type) {
        case 'alfa':
                $input = preg_replace('/[^A-Za-z\']/',",$input);
                break;
        case 'num':
                $input = preg_replace('/[^0-9]/',",$input);
                break;
        case 'alfa_num':
                $input = preg_replace('/[^A-Za-z0-9\']/',",$input);
                break;
        case 'alfa_num_ext':
                $input = preg_replace('/(\-{2})|[^A-Za-z0-9!@$_\-^*#\(\)\-\-\'\?]/',",$input);
                break;
        case 'text':
                $input = preg_replace('/(\-{2})|[^A-Za-z0-9!@#$_^*\(\)\?\-\-\.,\' ]/',",$input);
                break;
        case 'key_op':
                $input = preg_replace('/(insert)||(update)||(select)||(delete)||
                                (or)||(and)||(char\((.)+\))||(script)||(javascript)||(src)/i',",$input);
                break;
                default:
                        $input = preg_replace($type,",$input);
                        break;}
```

Рис.7 Функція фільтрації

Логіка функції побудована на ідеї пропускання лише дозволених даних та блокуванні всіх символів, що не є явно дозволеними в даному випадку. Функція приймає данні, які необхідно відфільтрувати і в залежності від обраного шаблону забезпечує необхідний рівень фільтрації. Шаблони забезпечують існування інформації п’яти рівнів критичності. Даний скрипт є простим та легким для розуміння, що робить можливим його подальше масштабування. У випадку ж потреби, користувач зі знанням регулярних виразів зможе легко модифікувати функцію під свої вимоги та завдання. Отже, в даній статі було досліджено атаки виконання коду та можливість застосування функції фільтрації вхідних даних на основі регулярних виразів, як спосіб протидії такій атаці. Було показано актуальність захисту від подібного роду атак, в особливості від SQL-ін’єкцій та XSS та наведено приклади їх реалізації, а також приклади наслідків, до яких може призвести успішне виконання цих атак.

ЛІТЕРАТУРА

1. Фленов М. Е. - PHP глазами хакера . – СПБ.: БХВ-Петербург, 2005. – 305с.
2. http://php.net/ - Документація PHP
3. Матросов А., Сергеев А., Чаунин М. HTML 4.0 – Санкт-Петербург: БХВ-Петербург, 1999. – 672с.
4. Фролов А.В. Базы данных в Интернете: Практическое руководство по созданию Web-приложений с базами данных/ А.В. Фролов, Г.В. Фролов. 2-е изд., перераб. – М.: Русская Редакция, 2000. – 448.

5. Веплинг Л. Томсон. Л. Разработка Web-приложений с помощью PHP и MySQL, 2-е издание. – Издательский дом «Вильямс», 2003 – 800с.
6. Ноблес Р., Греди К., Эффективный Web-сайт: Учебное пособие – М: Издательство ТРИУМФ, 2004 – 560с.

Надійшла: 9.11.2012 р.

Рецензент: д.т.н., професор Юдін О.К.

УДК 621.396:004.056.523

Чевардін В.Є., Ізофатов Д.О.

АНАЛІЗ МЕХАНІЗМІВ ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ТА АВТЕНТИЧНОСТІ ПОВІДОМЛЕНЬ В МЕРЕЖАХ MANET

Забезпечення цілісності та автентичності повідомлень в сучасних мережах MANET є актуальною науково-технічною проблемою. Способом її вирішення є проведення постійного аналізу загроз безпеки MANET, а також розробка й вдосконалення механізмів забезпечення цілісності та автентичності інформації в мережі. Проведено аналіз сучасних атак на мережі MANET та відповідних ним загроз цілісності та автентичності інформації. Розглянуті недоліки та переваги протоколів безпечної маршрутизації. В результаті цього визначені перспективні напрямки вдосконалення та подальшого розвитку систем безпеки та протоколів безпечної маршрутизації в мережах MANET.

Ключові слова: MANET, ad hoc мережа, механізми забезпечення безпеки в MANET, захист від атак на MANET, протоколи безпечної маршрутизації.

1. Формулювання задачі

Сучасна військово-політична ситуація в світі, досвід останніх конфліктів показують, що вирішальним фактором у сучасній війні є інформаційна перевага. Для ефективного управління військами в сучасному військовому конфлікті необхідна мобільна, надійна та живуча інформаційно-телекомунікаційна мережа. Забезпечити зростаючі вимоги мереж військового призначення неможливо без використання децентралізованих радіомереж. Прикладом таких мереж є MANET або ad hoc мережі [1–5]. Їх особливістю є використання однотипних засобів зв'язку (низьких за вартістю, низьких за енергоживленням, невеликих в розмірі та автономних), які забезпечують прийом, передачу інформаційних пакетів та їх ретрансляцію. Одними з найбільших загроз мереж MANET є загрози цілісності та спостереженості як відкритої так і службової інформації [6]. Характерною рисою таких мереж є відсутність центрів управління мережею, авторизованих центрів генерації криптографічних ключів та видачі сертифікатів відкритих ключів, які в свою чергу необхідні для механізмів забезпечення цілісності та спостереженості в сучасних автоматизованих системах. Механізмами забезпечення цілісності та спостереженості, що використовуються сьогодні в ad hoc мережах є алгоритми генерації та верифікації MAC-кодів (Message Authentication Code), безключові геш-функції SHA-256, SHA-384, SHA-512, RIPEMD-160, MDx, алгоритми автентифікації користувачів на основі рукопотискання, алгоритми автентифікації на основі електронного цифрового підпису DSA, ECDSA. Стійкість та надійність механізмів цілісності та спостереженості залежить від наступних показників: стійкість алгоритмів гешування до колізій та інших атак на геш-алгоритми, стійкість до зламу алгоритмів електронного цифрового підпису, стійкість алгоритмів рукопотискання, стійкість алгоритмів генерації та розповсюдження криптографічних ключів.

Питанням забезпечення безпеки інформації в ad hoc мережах присвячено багато робіт [1–5, 7–15]. Багато підходів лягло в основу протоколів безпечної маршрутизації: SAODV, TAODV, ARAN, SAR, SRP, SEAD, SLSP, CONFIDANT та інші, які мають як переваги так і певні недоліки [1, 2, 8]. Слід зауважити, що в умовах розвитку сучасних інформаційно-телекомунікаційних технологій неможливо забезпечити безпеку мереж MANET без використання потужних алгоритмів автентифікації, генерації та розповсюдження криптографічних ключів. Відсутність сьогодні нормативної бази з побудови системи захисту інформації в сучасних ad hoc мережах створює ряд проблем в подальшому розвитку мереж MANET в нашій державі.