УДК 004.056
DOI: 10.31673/2409-7292.2019.035763

В.М. Білько, І.М. Лавровський, А.Є. Баром

АНАЛІЗ МЕТОДІВ ЗАХИСТУ ВІД DDoS АТАК

Дана стаття присвячена аналізу методів захисту від DDoS атак. Проведений аналіз показав відомості про атаки DDOS та SQL. Наведені рекомендації яким чином вони відбуваються та як захиститися від них.

Ключові слова: SQL-ін'єкції, DDoS-атака, XPath-впровадження, збої, інформаційні технології, кодування.

Вступ

Атаки на сайти – вчинення протиправних дій щодо веб-сайтів спрямованих на отримання конкурентних переваг шляхом злому, зараження шкідливими кодом, блокування доступу (з надалі вимогою викупу), крадіжку конфіденційних даних, виведення з ладу програмного забезпечення. На першому етапі кібер-злочинець вивчає сайт на предмет вразливостей. Вони, в свою чергу, бувають декількох типів. Уразливості коду сайту. З'являються такі уразливості через помилки або недостатньої опрацюванні питання безпеки програмістами, що створюють CMS і розширення сайту. При наявності подібних вразливостей хакер може впровадити свій код в виконувані скрипти, запити до бази даних (SQL-ін'єкції), поштового сервера сайту (email-ін'єкції), або в сторінку, яку користувач відкриває в своєму браузері, з метою крадіжки його особистих даних, включаючи паролі (міжсайтовий скриптинг). Інтернет-сайти на популярних CMS зламують масово з метою зараження через типові уразливості. Що ж стосується DDoS-атак, то їх роблять на замовлення і тут є конкретні групи ризику. Наприклад, дуже часто атакують бізнес який залежить від інтернету і просто коштує грошей. Зловмисники починають атаку і пропонують власнику відкупитися. За статистикою найчастіше атакують:

- купонні сервіси;
- платіжні системи;
- Інформаційні агрегатори;
- Електронна комерція;
- Ігри та ігрові майданчики.

Веб-сторінки банків і електронних платіжних систем зламують з метою крадіжки грошей, сайти комерційних компаній ламають заради клієнтської бази і створення проблем конкуренту, або шантажу, вимагаючи гроші за відновлення нормальної роботи, сайти урядових органів і громадських організацій атакуються ідеологічними противниками [1].

Постановка задачі

Основним джерелом загрози для сайту є його власний код, написаний недбало, з помилками, без урахування строгих правил безпеки, а також використання застарілих, або викачаних з піратських сайтів модулів, розширень і плагінів. Захист від DDoS-атак (в крайньому разі, низької та середньої потужності) здійснюється розміщенням сайту на серверах високої пропускної здатності, також використовується аналіз і фільтрування трафіку з блокуванням IP атакуючих машин. У разі ж потужних атак часто залишається лише перечекати, поки вона припиниться. Замовники DDoS-атак рідко мають у своєму розпорядженні власні ресурси для її проведення, і змушені платити хакерам-власникам ботнетів (мереж заражених комп'ютерів, з яких і ведеться атака). Відповідно, потужна атака коштує чималих грошей, і рідко триває довше декількох днів. А щоб знизити ймовірність DDoS-атак, не варто розміщувати контент, образливий для великих груп людей або впливових структур, здатних помститися. Нарешті, варто регулярно створювати резервні копії сайту, щоб в разі серйозних проблем швидко його відновити.

Аналіз існуючих DDoS атак

Атака типу «відмова в обслуговуванні» (DoS) – це спроба принести шкоду, зробивши недоступною цільову систему, наприклад веб-сайт або додаток, для звичайних кінцевих

користувачів. Зазвичай зловмисники генерують велику кількість пакетів з запитами, які в кінцевому рахунку зупиняють нормальну роботу цільової системи. У разі атаки типу «розподілена відмова в обслуговуванні» (DDoS) зловмисник використовує для проведення атаки безліч зламаних або контрольованих джерел.

У загальному випадку DDoS-атаки можна розділити на декілька видів залежно від того, на якому рівні моделі взаємодії відкритих систем (OSI) відбувається атака. Найбільш поширені атаки на мережевому рівні (рівень 3), транспортному рівні (рівень 4), рівні уявлення (рівень 6) і рівні додатків (рівень 7).

№	Рівень	Додаток	Опис	Приклад вектору
7	Додаток	Дані	Мережевий процес на адресу додатка	HTTP-флуд, DNS-флуд
6	Представлення	Дані	Подання та шифрування даних	SSL-порушення
5	Сеанс	Дані	Сеанс зв'язку між хостами	н/д
4	Транспортний	Сегменти	Зв'язок між кінцевими пунктами і надійність	SYN-флуд
3	Мережевий	Пакети	Визначення маршруту і логічна адресація	Атаки з відображенням UDP-пакетів
2	Канальний	Кадри	Фізична адресація	н/д
1	Фізичний	Біти	Середовище передачі, сигнал і виконавчі дані	н/д

Розглядаючи методи запобігання таких атак, корисно розділити їх на дві групи: атаки рівня інфраструктури (рівні 3 і 4) і атаки рівня додатки (рівні 6 і 7). Атаки на рівнях 3 і 4 зазвичай відносять до атак рівня інфраструктури. Вони є найбільш поширеним типом DDoS-атак і включають в себе такі вектори, як SYN-флуд, і інші атаки відображення, такі як UDP-флуд. Ці атаки зазвичай дуже масивні і спрямовані на те, щоб перевантажити пропускну здатність мережі або сервери додатків. Але, на щастя, такий тип атак має певні ознаки, тому їх легше виявити. Атаки на рівнях 6 і 7 зазвичай відносять до атак рівня додатків. Хоча ці атаки менш поширені, вони є більш складними. Ці атаки, як правило, не є настільки масовими, як атаки рівня інфраструктури, але вони націлені на певні дорогі частини програми і призводять до того, що воно стає недоступним для звичайних користувачів. Як приклад можна привести потік HTTP-запитів на сторінку входу в систему, до дорогого API пошуку або навіть потоки XML-RPC Wordpress (також відомі як атаки Wordpress Pingback).

SQL-ін'єкція - це небезпечна уразливість, яка виникає через недостатню фільтрації вводяться користувачем даних, що дозволяє модифікувати запити до баз даних. Результатом експлуатації SQL-ін'єкції є отримання доступу до даних, до яких в звичайних умовах у користувача не було б доступу.

Зазвичай SQLi знаходять в веб-додатках. Але насправді, SQL-ін'єкції можуть бути схильні до будь-які програми, які використовують різні бази даних (не тільки MySQL / MariaDB).

Як приклад, розглянемо додаток, яке звертається до бази даних з наступним запитом:

```
SELECT `name`, `status`, `books` FROM `members` WHERE name = 'Demo' AND password ='111'
```

Запит схожий на природну мову (англійська), і його значення досить просто інтерпретувати:

Вибрати (SELECT) поля `name`,` status`, `books` з (FROM) таблиці` members` де (WHERE) значення поля name дорівнює величині Demo (name = 'Demo') і (AND) значення поля password дорівнює величині 111 (password = '111').

Цей запит викликає обхід таблиці, в результаті якого робиться порівняння з кожним рядком, і якщо умова name = 'Demo' AND password = '111' є для будь-якої рядки істиною, то вона потрапляє в результати. В даному випадку, результати будуть тільки якщо і введене ім'я користувача та пароль у точності збігаються з тими, які зберігаються в таблиці.

При цьому значення «Demo» і «111» додаток отримує від користувача - наприклад, у формі входу на сайт.

Припустимо, що замість Demo користувач ввів такий рядок:

```
Demo' -
```

Тоді запит до бази даних буде мати вигляд:

```
SELECT `name`, `status`, `books` FROM `members` WHERE name = 'Demo' -- ' AND password ='111'
```

Дві рисочки (-) - означають коментар до кінця рядка, тобто все, що за ними, більше не враховується. Отже, з виразу умови «зникає» частина 'AND password =' 111 '

Оскільки в коментарі залишилася закриваюча лапка, то вона також була введена з ім'ям користувача, щоб не зламати синтаксис і не викликати помилку, в результаті, фактично, до бази даних робився наступний запит:

```
SELECT `name`, `status`, `books` FROM `members` WHERE name = 'Demo'
```

У ньому була порушена логіка роботи програми, закладена розробниками. Тобто тепер пошук в таблиці проводиться тільки по імені. І якщо ім'я співпало, то рядок потрапляє в результати незалежно від введеного пароля. Це і є приклад експлуатації SQL-ін'єкції. В реальній ситуації, така помилка може бути використана на веб-сайті для входу під обліковим записом адміністратора, для якої досить знати тільки ім'я, а пароль стає непотрібним.

Крім обходу аутентифікації, SQL-ін'єкція використовується для добування інформації з баз даних, виклику відмови в обслуговуванні (DoS), експлуатацію інших вразливостей (на кшталт XSS) і т.п.

Щоразу з будь-яким додатком, де б не експлуатувалася SQL-ін'єкція, використовуються наступні три базових правила впровадження:

- балансування
- впровадження
- коментування

Балансування полягає в тому, що кількість відкривають і закривають лапок і дужок має бути однаковим, щоб не викликати помилку синтаксису. При дослідженні помилки потрібно визначити, використовуються, і якщо використовуються, то які лапки і дужки.

Впровадження полягає в доповненні запиту в залежності від інформації, яку ми хочемо отримати.

Коментування дозволяє відсікти заключну частину запиту, щоб вона не порушувала синтаксис.

Коментарі в MySQL починаються з символів:

- #
- -
- / *

Тобто замість

```
Demo' --
```

можна було б ввести

```
Demo' #
```

Зверніть увагу, що після подвійної риси обов'язково потрібен пробіл, а після # пробіл необов'язковий.

Можна продовжити змінювати логіку запиту, якщо в якості імені користувача вставити:

```
Demo' OR 1 --
```

то вийде запит

```
SELECT `name`, `status`, `books` FROM `members` WHERE name = ' Demo' OR 1 -- ' AND password ='111'
```

Приберемо закоментовану частина:

```
SELECT `name`,` status`, `books` FROM` members` WHERE name = 'Demo' OR 1
```

Ми використовуємо логічне АБО (OR). Логічне АБО повертає true (істину) якщо хоча б один з виразів є істиною. В даному випадку другий вираз 1 завжди є істинною. Отже, в результати потраплять взагалі всі записи таблиці. У реальному веб-додатку можна досягти результату, коли будуть виведені дані всіх користувачів, незважаючи на те, що атакуючий не знав ні їх логіни, ні паролі.

У нашому прикладі після введеного значення Demo ми ставили одинарні лапки ('), щоб запит залишався правильним з точки зору синтаксису. Запит може бути написаний по-різному, наприклад, все такі форми повертають однаковий результат.

Для запитів з цифрою:

```
SELECT * FROM table_name WHERE id=1
SELECT * FROM table_name WHERE id='1'
SELECT * FROM table_name WHERE id="1"
SELECT * FROM table_name WHERE id=(1)
SELECT * FROM table_name WHERE id=('1')
SELECT * FROM table_name WHERE id=("1")
```

Для запитів з рядком:

```
SELECT * FROM table_name WHERE id = '1'
SELECT * FROM table_name WHERE id = "1"
SELECT * FROM table_name WHERE id = ('1')
SELECT * FROM table_name WHERE id = ("1")
```

Залежно від того, як буде складено запит, потрібно використовувати відповідні символи парні закривають символи, щоб не відбувалося порушення синтаксису. Наприклад, якби запит був записаний так (в ньому замість одинарних лапок, використовуються подвійні):

```
SELECT * FROM `members` WHERE name = "$name" AND password = "$password"
```

то ім'я користувача

```
Demo' #
```

не набуло б дії і не викликало б помилку. Для позначення кінця введеного імені потрібно використовувати закриває прямі подвійні лапки, тобто .:

```
Demo" #
```

Для такого запиту (використовуються одинарні лапки і круглі дужки):

```
SELECT * FROM `members` WHERE name = ('$name') AND password = ('$password')
```

потрібно також закривати круглі дужки, тобто для експлуатації SQL-ін'єкції потрібно ввести щось на зразок

```
Demo') #
```

Головними ознаками наявності SQL-ін'єкції є висновок помилки або відсутність виведення при введенні одинарної або подвійної лапки. Ці символи можуть викликати помилку і в самому додатку, тому щоб бути впевненим, що ви маєте справу саме з SQL-ін'єкцією, а не з іншого помилкою, потрібно вивчити виведене повідомлення [2].

Новою атакою впровадження коду є атака XPath-впровадження, використовує переваги вільної типізації (loose typing) і слабкість синтаксичних аналізаторів XPath, що дозволяють зловмисникам використовувати в своїх інтересах некоректні XPath-запити в URL формах або інших методах для отримання доступу до привілейованої інформації та її зміни. Вона виконується майже так само, як і атака прихованого SQL-впровадження, але, на відміну від

неї, не багато людей знають про атаки XPath-впровадження або вживають проти них якихось заходів. Найчастіше можна легко запобігти цій загрозі, якщо слідувати передового досвіду розробки захищених додатків.

Зазвичай більшість Web-додатків для зберігання даних і добування інформації використовує реляційні бази. Якщо, припустимо, у вас є Web-сайт, що вимагає аутентифікації, ймовірно, існує таблиця users з унікальним ID, ім'ям входу в систему, паролем, і (можливо) інформацією будь-якого іншого роду, наприклад, роллю. SQL-запит для отримання інформації про користувача з таблиці users може виглядати так, як показано в лістингу 1.

Новою атакою впровадження коду є атака XPath-впровадження, використовує переваги вільної типізації (loose typing) і слабкість синтаксичних аналізаторів XPath, що дозволяють зловмисникам використовувати в своїх інтересах некоректні XPath-запити в URL формах або інших методах для отримання доступу до привілейованої інформації та її зміни. Вона виконується майже так само, як і атака прихованого SQL-впровадження, але, на відміну від неї, не багато людей знають про атаки XPath-впровадження або вживають проти них якихось заходів. Найчастіше можна легко запобігти цій загрозі, якщо слідувати передового досвіду розробки захищених додатків.

Зазвичай більшість Web-додатків для зберігання даних і добування інформації використовує реляційні бази. Якщо, припустимо, у вас є Web-сайт, що вимагає аутентифікації, ймовірно, існує таблиця users з унікальним ID, ім'ям входу в систему, паролем, і (можливо) інформацією будь-якого іншого роду, наприклад, роллю. Шляхом проб і помилок зломщик може перевірити різні дочірні вузли XML-документа і зібрати інформацію, спостерігаючи за результатами XPath-виразів при успішній аутентифікації. Потім, в принципі, зломщик може написати простий сценарій, що передає різні XPath-впровадження та видобуває XML-документ із системи [3].

Аналіз існуючих методів та моделей захисту від DDoS атак. Одним з перших методів нейтралізації DDoS-атак є зведення до мінімуму розміру зони, яку можна атакувати, що обмежує можливості зловмисників для атаки і забезпечує можливість створення централізованого захисту. Необхідно переконатися, що доступ до додатка або ресурсів не було відкрито для портів, протоколів або додатків, звідки не повинно бути звернення до них. Таким чином, зведення до мінімуму можливих точок для атаки дозволяє зосередити зусилля на їх нейтралізації.

У деяких випадках можна домогтися цього, розмістивши свої обчислювальні ресурси за мережами розповсюдження контенту (CDN) або балансувальник навантаження і обмеживши прямий інтернет-трафік до певних частин своєї інфраструктури, таким як сервери баз даних. В інших випадках можна використовувати брандмауери або списки контролю доступу (ACL), щоб контролювати, який трафік надходить в додатки.

План масштабування. Двома основними елементами нейтралізації великомасштабних DDoS-атак є пропускна здатність (або транзитна потенціал) і продуктивність сервера, достатня для поглинання і нейтралізації атак.

Транзитний потенціал. При проектуванні додатків необхідно переконатися, що постачальник послуг хостингу надає надлишкову пропускну здатність підключення до Інтернету, яка дозволяє обробляти великі об'єми трафіку. Оскільки кінцева мета DDoS-атак - вплинути на доступність ресурсів / додатків, необхідно розміщувати їх поруч не тільки з кінцевими користувачами, а й з великими вузлами мережевого обміну трафіком, які легко забезпечать вашим користувачам доступ до додатка навіть при великому обсязі трафіку.

Крім того, для інтернет-додатків можна піти ще далі, скориставшись мережами розповсюдження контенту (CDN) і сервісами інтелектуального перетворення адрес DNS, які створюють додатковий рівень мережевої інфраструктури для обслуговування контенту і дозволу DNS-запитів з місць, які часто розташовані ближче до кінцевих користувачів. Продуктивність сервера.

Більшість DDoS-атак є об'ємними атаками, що споживають багато ресурсів; тому важливо мати можливість швидко збільшувати або зменшувати обсяг своїх обчислювальних ресурсів. Це можна забезпечити, використовуючи надлишковий обсяг обчислювальних ресурсів або ресурси зі спеціальними можливостями, такими як більш продуктивні мережеві інтерфейси або поліпшена мережева конфігурація, що дозволяють підтримувати обробку великих обсягів трафіку. Крім того, часто використовуються балансувальник навантаження для постійного контролю і розподілу навантажень між ресурсами та запобігання перевантаження будь-якого одного ресурсу.

Розуміння того, який трафік є типовим, а який – ні. Кожен раз, коли виявляється підвищення обсягу трафіку, що потрапляє на хост, в якості орієнтира можна брати максимально можливий обсяг трафіку, який хост може обробити без погіршення його доступності. Така концепція називається обмеженням швидкості. Більш просунуті методи захисту можуть йти ще далі і інтелектуально приймати тільки трафік, який дозволений, аналізуючи окремі пакети. Для цього необхідно буде визначити характеристики хорошого трафіку, який зазвичай отримує цільовий об'єкт, і мати можливість порівнювати кожен пакет з цим еталоном.

Розгортання брандмауерів для відображення складних атак рівня додатків. Проти атак, які намагаються використовувати уразливість в додатку, наприклад проти SQL-ін'єкцій або підробки міжсайтових запитів, рекомендується використовувати Web Application Firewall (WAF). Крім того, через унікальність цих атак ви повинні бути здатні самостійно нейтралізувати заборонені запити, які можуть мати певні характеристики, наприклад можуть визначатися як відмінні від хорошого трафіку або виходити з поганих IP-адрес, з несподіваних географічних регіонів і т. Д. Іноді для нейтралізації відбуваються атак може бути корисно отримати підтримку фахівців для вивчення характеристик трафіку і створення індивідуального захисту.

Оскільки XPath-впровадження багато в чому аналогічне SQL-впровадженню, від нього можна захиститися методами, схожими на ті, які використовуються для запобігання атакам SQL-впровадження [4].

Для захисту проти XPath-впровадження та інших форм впровадження коду необхідно перевіряти всі дані, передані від Web-сервера до служб системи зберігання даних. Наприклад, з Apache можна використовувати фільтр Mod_Security (наприклад, SecFilterSelective THE_REQUEST “(“/”)”), Щоб знайти в рядках одиночні та подвійні лапки і заборонити їх. Такий же підхід можна застосувати для фільтрації і заборони інших форм спеціальних символів (наприклад, (“* ^”; &> << /)), які можуть бути використані для різних атак з впровадженням коду. Цей підхід може бути правильним для деяких додатків, які, можливо, використовують основані на REST або SOAP XML-сервіси, але в інших випадках він може бути не можливий. Як завжди найкращим підходом є розумне проектування, починаючи з початкового дизайну і до реалізації програми.

Широке поширення нових платформ (наприклад, Ajax, RIA-платформи FLEX або Open Laszlo), а також інтеграція XML-сервісів від таких організацій як Google, змушує розробників враховувати загрози та ризики, створювані подібними методами роботи.

Давайте поглянемо на код, який зводить ймовірність вдалої SQL-ін’єкції до 0.

```
$order = isset($_GET['order']) ? $_GET['order'] : ''; // просто для повноти коду
$sort = isset($_GET['sort']) ? $_GET['sort'] : '';
$allowed = array("name", "price", "qty"); //перераховуємо варіанти
$key = array_search($sort,$allowed); // шукаємо серед них переданий параметр
$orderby = $allowed[$key]; //вибираємо найдений елемент.
$order = ($order == 'DESC') ? 'DESC' : 'ASC'; // обираємо напрямок сортування
$query = "SELECT * FROM `table` ORDER BY $orderby $order"; //запит 100% безпечний
```

Висновок

У роботі розглядається питання досягнення рівня захисту інформації від DDoS атак. Аналіз існуючих атак та наведені рекомендації дозволять покращити рівень захисту інформації у порівнянні з існуючим.

Перелік посилань

1. anti-malware.ru //Комплексная и многофункциональная защита информационных ресурсов рабочих станций и серверов. [Електронний ресурс]. Режим доступу: World Wide Web. –URL: https://www.anti-malware.ru/threats/websites-attacks

2. Електронний журнал «Хакер» //Sqlmap: SQL-инъекции. [Електронний ресурс].Режим доступу: World Wide Web. – URL: https://xakep.ru/2011/12/06/57950

3. anti-malware //Брутфорс [Електронний ресурс]. –Режим доступу: https://www.anti-malware.ru/threats/brute-force

4. DDoS атаки. https://aws.amazon.com/ru/shield/ddos-attack-protection/ [Електронний ресурс]. Режим доступу: https://aws.amazon.com/ru/shield/ddos-attack-protection/

Надійшла: 01.08.2019 Рецензент: д.т.н., доцент Гайдур Г.І.