Отчет по лабораторной работе №8 : Проект OWASP WebGoat

Дедков Сергей

2015

Содержание

1 Цель работы

2 Ход работы

2.1 Изучить описания десяти самых распространенных веб-уязвимости согласно рейтингу OWASP.

- A1 Внедрение кода - Injection

  Атака при которой данные выполняются как код, например SQL-иньекции.

  Примеры использования:

  ```
  String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("i
  ```

  ```
  http://example.com/app/accountView?id=' or '1'='1
  ```

- A2 Некорректная аутентификация и управление сессией - Broken Authentification and Session Management

  По умолчанию, индентификатор созданной сессии сохраняется в cookie браузера, за исключением случаев когда cookie в браузере откючены. В таком случае, они будут автоматом подставляться в каждый URL самим сервером

  ```
  index.php?PHPSESSID=jf843jgk4ogkfdjfgomg84o4og54mg
  ```

- A3 Межсайтовый скриптинг - Cross-Site scripting (XSS)

  Тип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с веб-сервером злоумышленника. Является разновидностью атаки «внедрение кода».

- A4 Небезопасные прямые ссылки на объекты - Insecure Direct Object References

  Данная уязвимость проявляется, когда разработчик указывает прямую ссылку на внутренний объект, например такой как файл, каталог или запись в базе данных, как параметр в URL. Это позволяет атакующему за счёт манипуляций с этим параметром получить несанкционированный доступ к системе.

  Примеры:

  – Прямая ссылка на приватную фотографию в закрытом альбоме

  - Открытый номер кошелька в GET-запросе
  - AJAX запрос-ответ по userID, возвращающий все данные о юзере в JSON, которые фильтруются на стороне клиента
  - Незакрытый просмотр/редактирование, например, своего профиля. В примере ниже, юзер с ID 3 может перейти на страницу профиля ID 1, и отредактировать его

- A5 Небезопасная конфигурация - Security Misconfiguration

  Недостаточно или неправильно настроенный конфигурационный файл сервера/PHP/... Сюда относятся:

  - Открытые логи
  - Не закрытый development mode (незакрытое профилирование запросов, stack trace)
  - Открытый конфиг. Не main.php, к примеру, а main.inc
  - Не иметь аккаунтов admin/admin. Даже на роутере в офисе.
  - Открытый доступ в папки images. Закрывается с помощью .htaccess или пустым index.html
  - И открытые папки вроде .git или .svn

- A6 Утечка чувствительных данных - Sensitive Data Exposure

  Как правило, данная уязвимость уже вытекает как следствие взлома сайта, в ходе которого были похищены данные, которые оказались легко читаемыми.

  Тест на уязвимость:

  - Хранятся ли у нас какие-либо данные о деньгах/номерах кошельков/состоянии здоровья в БД в открытом виде?
  - Также, в каком виде эти данные передаются по сети?
  - Свежие ли версии софта стоят?
  - Достаточно ли силён алгоритм шифрования, и часто ли происходит его обновление?

- A7 Отсутствие контроля доступа к функциональному уровню - Missing Function Level Access Control

  Заголовок подразумевает разграниченный доступ к определённым функциям приложения.

  - Не оставлять ссылки для тех, кому нет прав туда заходить
  - Иммунитет даст по дефолту снятие доступов для всех и со всего — принцип белого листа и deny(*).

  - Так же, не помешает проверка в важных местах кода в процессе выполнения действия — при вызове функции/метода поставить проверку ещё раз.

- A8 Подделка межсайтовых запросов (CSRF) - Cross-Site Request Forgery (CSRF)

  Вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер (например, на сервер платёжной системы), осуществляющий некую вредоносную операцию (например, перевод денег на счёт злоумышленника). Для осуществления данной атаки жертва должна быть аутентифицирована на том сервере, на который отправляется запрос, и этот запрос не должен требовать какого-либо подтверждения со стороны пользователя, который не может быть проигнорирован или подделан атакующим скриптом.

- A9 Использование компонентов с известными уязвимостями - Using Components with Known Vulnerabilities

  Для устранения - поддерживать все подключаемые части проекта в актуальном состоянии, обновлять до последних стабильных версий, не юзать малопопулярные или любительские модули. Если стоит выбор — не использовать их в принципе.

- A10 Невалидированные редиректы - Unvalidated Redirects and Forwards

  Доверяя сайту, пользователи могут переходить по любым ссылкам. Сообщение вроде «Вы покидаете наш сайт, переходя по ссылке ...», не что иное, как простейшая защита от подобного рода уязвимостей. Злоумышленник может воспользоваться подобного рода редиректами через сайт на угодные ему страницы.

  Профилактика:

  - Не злоупотреблять редиректами.
  - Если пришлось, не использовать пользовательские данные в запросе (вроде success.php&user_mail=eeee@eee.com)
  - Рекомендуется перезаписывать url средствами сервера.

2.2 Недостатки аутентификации

- Сложность пароля:

  Было предложено проверить сложность паролей с помощью сервиса https://www.cnlab.ch/codecheck. (Сейчас этот сервис не работает)

123456 - 0 seconds (dictionary based, one of top 100) abzfez - up to 5 minutes (26 chars on 6 positions = $26\hat{6}$ seconds) a9z1ez - up to 40 minutes (26+10 chars on 6 positions = $36\hat{6}$ seconds) aB8fEz - up to 16 hours (26+26+10 chars on 6 positions = $62\hat{6}$ seconds) z8!E?7 - up to 50 days (127 chars on 6 positions = $127\hat{6}$ seconds)

- Забыли пароль

  Пользователи могут восстановить их пароль если у них получится ответить на секретный вопрос. На странице восстановления нет никаких мезанизмов связанных с блокировкой аккаунтов. Ваше имя пользователя - 'webgoat', любимый цвет - красный (red). Цель урока - восстановить пароль к аккаунту другого пользователя.

  Results: Username: webgoat Color: red Password: webgoat

- Основная аутентификация

  Основная аутентификация используется для защиты ресурсов расположенных на стороне сервера. При получении запроса от пользователя веб-сервер отправляет ему ответ с кодом 401. Получив его браузер запрашивает у пользователя логин и пароль в специальном диалоговом окне. После ввода браузер кодирует полученные данные по алгоритму base64 и отсылает веб-серверу. Последний, в свою очередь, проверяет полученную информацию и, если всё правильно, отдаёт клиенту запрашиваемый документ. Указанные пользователем данные далее автоматически отсылаются браузером при каждом обращении к защищённым ресурсам.

  Для декодирования base64 используем сервис http://yehg.net/encoding/

  Congratulations, you have figured out the mechanics of basic authentication. - Now you must try to make WebGoat reauthenticate you as: - username: basic - password: basic. Use the Basic Authentication Menu to start at login page.

  После тго как данные введены в форму. Чистим куки и аутентификационные сессии.

  Далее прописываем следующий url:

  `http://basic:basic@localhost:8080/WebGoat/attack?Screen=187&menu=500`

  * Congratulations. You have successfully completed this lesson. * Error generating org.owasp.webgoat.lessons.BasicAuthentication

- Логирование через TAN (Transaction authentication number)

  1. В данном уроке хакер знает Имя и Партоль, а так же TAN1, но дело в том, что он уже использован. Поэтому будет предложено ввести второй, но мы можем поменять номер TAN в параметрах запроса.

2. Другой вариант, когда известен логин и TAN. Тогда, сначала залогинившись под своим пользователем, при отправке TAN можно заменить имя на другого. TAN будет воспринят корректно, а проля не потребуется.

2.3 Переполнение буфера

В данном варианте используется переполнение буфера - сайт ведет себя некорректно и предоставляет информацию обо всех пользователях вместо одного. Для предотвращение на клиенте можно добавть проверку на количество символов и на сервере корректность данных.

2.4 Качество кода

В данном варианте в комментариях на странице HTML были прописаны логин и пароль. Достаточно заглянуть в текст странички, чтобы вытащить их и пройти авторизацию.

2.5 Многопоточность

В данном случае при одновременном доступе к одним и тем же запросам разным пользователям приходят одни и те же ответы. За такими уязвимостями надо следить.

2.6 Межсайтовое выполнение сценариев

- В первом задании нужно ввести в поле поиска вредоносный код, например, такой:

```
<form>
<form name="stealer">
Username: <input type="text" name="username"><br>
Password: <input type="password" name="password">
<input type="submit" value="Submit" onclick=<script><img src="http://localhost:8
</script>
</form>
```

  После чего появяться поля, где нужно будет ввести логин и пароль, которые будут переданы на нужный сервер.

- Хранимые Stored XSS

  Хранимый XSS является наиболее разрушительным типом атаки. Хранимый XSS возможен, когда злоумышленнику удается внедрить на сервер вредоносный код, выполняющийся в браузере каждый раз при обращении к оригинальной странице. Классическим

примером этой уязвимости являются форумы, на которых разрешено оставлять комментарии в HTML формате без ограничений, а также другие сайты Веб 2.0 (блоги, вики, имиджборд), когда на сервере хранятся пользовательские тексты и рисунки. Скрипты вставляются в эти тексты и рисунки.

В данном случае в одном из полей, которое храниться в базе данных и выводиться у других пользоателей нужно указать вредоностный javascript код в тэге <script>.

Для предотвращения можно сделать валидацию по вооду данных и валидацию по выводу на сервере.

- Отраженные Reflected XSS

  Атака, основанная на отражённой уязвимости, на сегодняшний день является самой распространенной XSS-атакой. Эти уязвимости появляются, когда данные, предоставленные веб-клиентом, чаще всего в параметрах HTTP-запроса или в форме HTML, исполняются непосредственно серверными скриптами для синтаксического анализа и отображения страницы результатов для этого клиента, без надлежащей обработки. Отражённая XSS-атака срабатывает, когда пользователь переходит по специально подготовленной ссылке.

2.7 Неправильная обработка ошибок

Примером данной уязвимости служит возможность удалить из параметров, например, пароль и после этого удачно пройти аутентификацию.

2.8 Недостатки приводящие к осуществлению инъекций (SQL и прочее)

- Иньекция команд

  Атаки класса "Инъекция команд"представляют собой серьёзную угрозу для сайтов принимающих от пользователей какие-либо данные. Методика их использования достаточно тривиальна, но в тоже время они могут приводить к полной компрометации атакованной системы. Несмотря на это количество приложений имеющих подобные уязвимости неуклонно растёт.

  На самом деле подобные угрозы могут быть полностью устранены с помощью принятия разработчиками простейших мер направленных на обеспечение безопасности приложения.

  Проверка всех получаемых от пользователя данных, особенно тех, которые будут использоваться в командах ОС, скриптах или запросах к БД, является хорошей практикой.

Если вместо файла, который должен быть исполнен отправить на сервер такой код `" & ping 192.168.150.3` то сервер пропингует данный ip.

- Проведение числовых SQL-инъекций

  Всегда можно избежать появления уязвимостей этого класса если в процессе написания приложений соблюдать общие меры предосторожности. Например фильтровать все поступающие от пользователя данные. Особенно те, которые будут помещены в SQL-запросы.

  Пусть есть запрос:

  SELECT * FROM weather_data WHERE station = 101

  Где 101 - передается на сервер через запрос. если подменить на 101 or true. Получиться следующий запрос и будет выведена вся информация:

  SELECT * FROM weather_data WHERE station = 101 or true

  Даже так, котороая должна была быть скрыта отпользователя.

- Log-spoofing

  Целью этих атак является подделка записей лог-файла за счёт помещения в него специально сформированной строки. Это позволит атакующему запутать администратора и скрыть свои следы.

  Суть атаки в том, что помимо логина на сервер отправляется строка, которая предположительно записывается в лог, после чего администратор не понимает что произошло.

- XPATH иньекция

  По аналогии с SQL-инъекциями, XPath-инъекции возникают тогда, когда пользовательские данные без должной проверки попадают в запрос к XML-данным. Посылая приложению специльно сформированные запросы злоумышленник может раскрыть внутреннюю структуру XML-базы и получить доступ к той информации, к которой ему обращаться нельзя. Например он может повысить свои привилегии если ему удастся произвести XPath-инъекцию в отношении файла хранящего пользовательские аккаунты.

  Запросы к XML осуществляются с помощью XPath - не сложного языка, позволяющего определять местонахождения информации в XML-структуре. Как и в SQL, в нём вы можете устанавливать критерии поиска. В случаях когда данные приложения хранятся в виде XML-базы, пользователь с помощью одного или нескольких параметров запроса может определять что из неё будет извлечено

и отображено на сайте. Эти параметры должны тщательно проверяться, чтоб атакующий не смог изменить структуру изначального XPath-запроса и извлечь чувствительную информацию.

Если после ввода логина и пароля показывается информация об этом пользователе, то если прописать условие которОе всегда выполняется покажется информация о всех пользователях.

- Строковая sql иньекция

  По аналогии с цифровой, только в конце запроса добавляется кавычка, чтобы закрыть строку. Таким образом в условии можно например дописать следующий текст:

```
' or 'a'='a
```

SQL иньекции возможны благодаря тому, что переменные вставляются прямо в текст запроса, для того, чтобы избежать следует использовать параметризированные запросы.

Для поиска SQL иньекций существует add-on firefox - SQL Inject me, который проверяет формы на сайте на наличие SQL иньекций.

2.9 Отказ в обслуживании

Атаки класса "Отказ в обслуживании"являются главной проблемой веб-приложений. Ситуации, при которых конечный пользователь долгое время не может получить доступ к важному приложению или сервису, могут принести большие убытки.

В предложенной работе сайт позволяет нескольким пользователям авторизироваться одновременно. В то же время веб-приложение может устанавливать с БД только 2 соединения за раз. Нужно было получить список существующих пользователей и попытаться одновременно произвести вход от 3 логинов.

Для этого методом SQL иньекции выясняются пароли. После чего производится авторизация трех разных пользователей.

2.10 Небезопасное сетевое взаимодействие

Пользуясь данной атакой можно просматривать траффик(сниффинг) браузера, при этом пароли и прочие данные будут передаваться в незашифрованном виде и их легко будет обнаружить, например заставив трафик в сети с другой машины проходить через машину злоумышленника (например подменив в таблице ARP MAC роутера). Если же клиент использует защищенный протокл https эти данные ничего не скажут злоумышленнику, т.к. будут зашифрованы.

2.11 Небезопасная конфигурация

Эта техника используется хакерами для обращения к тем ресурсам, ссылок на которые на сайте нет, но доступ к которым никак не ограничен. Одним из примеров такой техники является затирание части URL для того чтоб просмотреть содержимое незащищённой директории.

Уязвимость вознкает в том случае если разработчик не предоставляет ссылку непривелигерованному пользователю, но при этом никаких проверок не делает. В таком случае можно угадать нужный url и просматреть интересующую информацию.

2.12 Небезопасное хранилище

Для хранения паролей и прочей секретной информации могут быть использованы алгоритмы шифрования, которые можно расшифровать и получить необходимые данные в первоначальном виде.

2.13 Исполнение злонамеренного кода

В данном случае на странице предлагается загрузить на сервер картинку. Зная то, что картинки грузятся в папку http://localhost:8080/WebGoat/uploads/. Можно загрузить туда исполняемый файл, например, exp.jsp. А потом запустить его перейдя в браузере по url http://localhost:8080/WebGoat/uploads/exp.jsp. Что и было сделано.

Код файла:

```
<html>
<% java.io.File file = new java.io.File("/root/WebGoat-5.4/tomcat/webapps/WebGoat/mfe
file.createNewFile(); %>
</html>
```

3 Вывод