Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Отчет по лабораторной работе №6**

**по курсу «СиМОИБ»**

**на тему: «Анализ уязвимостей WEB-серверов и приложений»**

|  |  |
| --- | --- |
| Выполнил студенты группы 821703: | Веренич К.О. |
| Проверил: | Захаров В. В. |

МИНСК

2020

**Задание 1:** Проанализировать историю взлома.

Изначально взломщик производил анализ сервера на предмет уязвимостей. С помощью порта 80 и уязвимости в POST X HTTP/1.0 он узнал ОС сервера и что он запущен с помощью Apache. Далее он пытался найти уязвимость в cgi скриптах, однако у него не вышло, так как сканер cgi-скриптов ничего не нашёл. Однако он обнаружил набор скриптов, который начал пытаться поломать.

На примере взлома сервиса, история показывает множество упущений, которые допускаются при создании и проектировании серверов. Допустим, при проверке данных на корректность, сервис фильтровал большинство полученных данных вместо того, чтобы сравнить получаемые данные с необходимыми по шаблону. Таким же образом происходит проверка gif файла, скрипт проверяет наличие “.gif” в конце файла, но не учитывает того, что в названии файла может быть нулевой байт, после которого все игнорируется в ядре. И точно также происходит проверка входных данных (проверяет лишь наличие символа возврата каретки).

Однако это не помогло и он решил использовать уязвимость в /tmp, которые позволяют взломать права администратора. Чтобы этого не допустить, необходимо придерживаться принципа минимальных прав доступа к другим системам. Например заводить столько пользователей с различными привилегиями, сколько это необходимо для реализации логики приложения.

Также в коде приложения используется метод get, который использовать не рекомендуется, так как это раскрывает логику работы.

**Задание 2:** Выявить ошибки администрирования и проектирования, которые позволили взломщику провести успешную атаку.

1. *POST X HTTP/1.0* – ошибочная директива.
2. Использование внешних скриптов.
3. Проверка входных данных(используется оптимистичных подход, а не пессимистичный).
4. При вызове функций open(), writen() передаваемыми параметрами являются идентификаторы, а не индексы, что позволяет модифицировать их и использовать функцию, со своим файлом.
5. Проверка входных данных происходит не по шаблону, что позволяет, обойти проверку на корректность, и, учитывая особенности ядра, передать ему необходимый файл.
6. Использование метода get в коде раскрывает логику программы.
7. Возможность получения пользователю прав администратора.

**Задание 3:** Найти интернет сайты с уязвимостями в дизайне, описанными в исследовании американских ученых.

http://драгоценности.рф/index.php?route=product/routecat&routecat\_id=12870 union all select 1,[t],3,4,5,6,7,8,9,10

http://walkthroughs.info/index.php?id=0 union all select 1,[t],3,4,5

https://www.perlmonks.org/bare/?node\_id=512343

**Вывод:**

Для обеспечения защиты от взлома достаточно следовать рекомендациям:

* Административная панель – только для администратора.
* Не переходите по непроверенным ссылкам и не допускайте их появления на сайте. Контроль над обновлением информации позволит значительно сократить количество нежелательных линков.
* Использовать сложные пароли. Чем больше символов использовано, тем дольше будет работать программа по автоматическому подбору. Собственное имя или год рождения – не самые подходящие данные для пароля.
* Нельзя давать слишком много информации в ваших сообщениях об ошибки. Например, вы пытаетесь залогиниться на вашем сайте. Вы должны использовать общие сообщения типа «Неправильное имя пользователя или пароль». Не надо уточнять, что именно, имя или пароль неверны, так как это позволит злоумышленнику понять, что он разгадал одно поле и может сконцентрироваться на другом.
* Запоминайте пароли или держите их в менеджерах. Лучше использовать специальные программы, которые обеспечат шифровку и сортировку паролей.
* Проверка данных должны быть, как в браузере, так и на стороне сервера. В браузере можно отловить простые ошибки и выделить поля, в которых допущены ошибки. Например, проверить на пустоту или на ввод только цифр. Однако, эти проверки могут быть легко пропущены и на сервер могут отправиться непроверенные данные. Если сервер не будет проверять входные данные, то это может привести к нежелательным последствиям.
* Пароль должен храниться в зашифрованном виде. Можно использовать такие алгоритмы, как SHA. Во время авторизации будут сравниваться только зашифрованные данные паролей. Для дополнительной безопасности можно добавлять «соль» в пароль. Для каждого пароля должна быть своя «соль».
* Позволять пользователям загружать файлы на сервер очень опасно, даже если они всего лишь меняют аватар. Риск в том, что загруженный файл может содержать скрипт, который может быть выполнен на сервере, если открыть его через браузер.