Администрирование многопользовательских систем управления баз данных

ФИО преподавателя: Смирнов Михаил Вячеславович

e-mail: smirnovmgupi@gmail.com

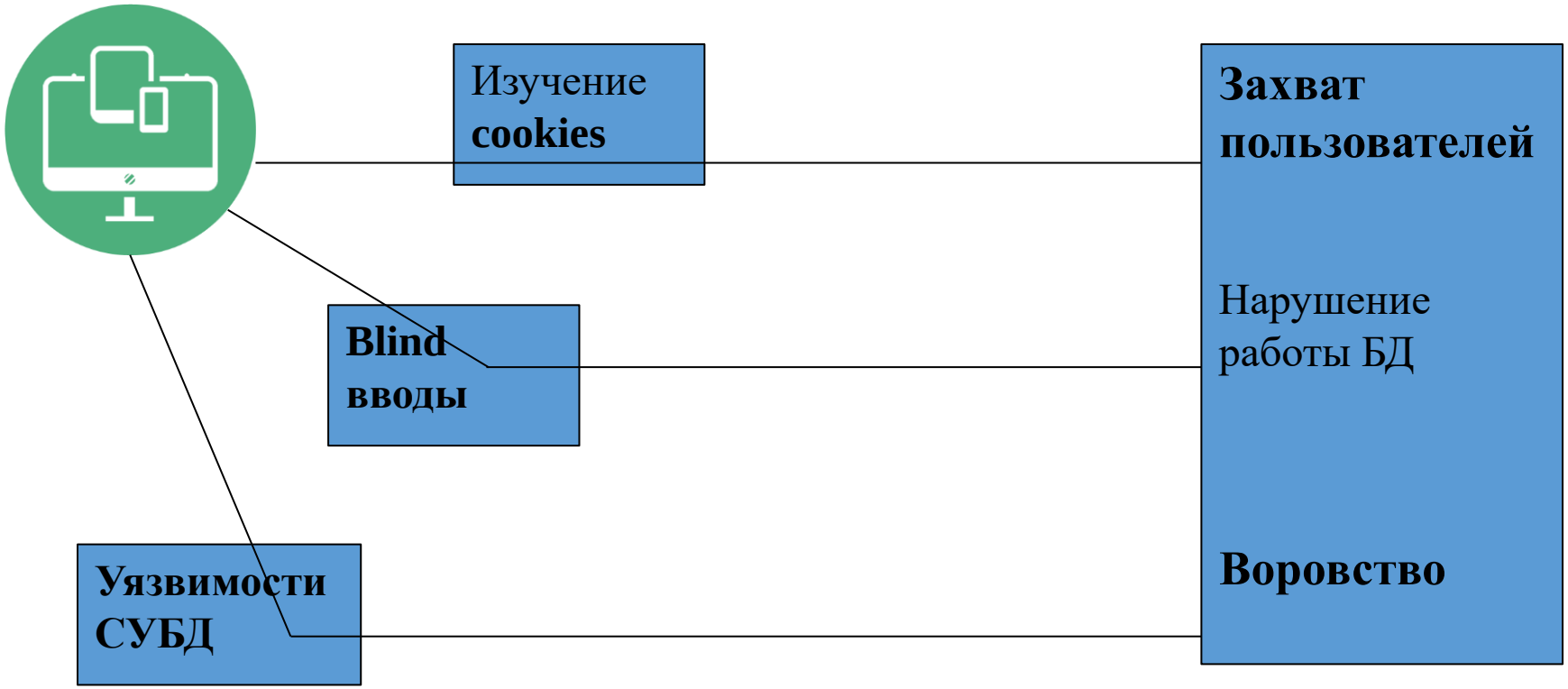
Лекция 7

Вредоносные атаки на реляционные СУБД.

Проблема обеспечения безопасности от вредоносных атак

- Наиболее распространенным типом хакерских атак на базы данных и на данные в них содержащиеся являются атаки с внедрением SQL кода (SQL injection).
- Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.
- Атака типа внедрения SQL может быть возможна из-за некорректной обработки входных данных, используемых в SQL-запросах.
- Разработчик прикладных программ, работающих с базами данных, должен знать о таких уязвимостях и принимать меры противодействия внедрению SQL

Схема типовых атак на реляционные СУБД



Атака с модификацией Cookies

- Cookie — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса.

- Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:
  - Аутентификации пользователя
  - Хранения персональных предпочтений и настроек пользователя
  - Отслеживания состояния сеанса доступа пользователя
  - Ведения статистики о пользователях

Принцип работы Cookie

- Запрашивая страницу, браузер отправляет веб-серверу короткий текст с HTTP-запросом. Например, для доступа к странице http://www.example.org/index.html, браузер отправляет на сервер www.example.org следующий запрос: GET /index.html HTTP/1.1 HOST: www.example.org

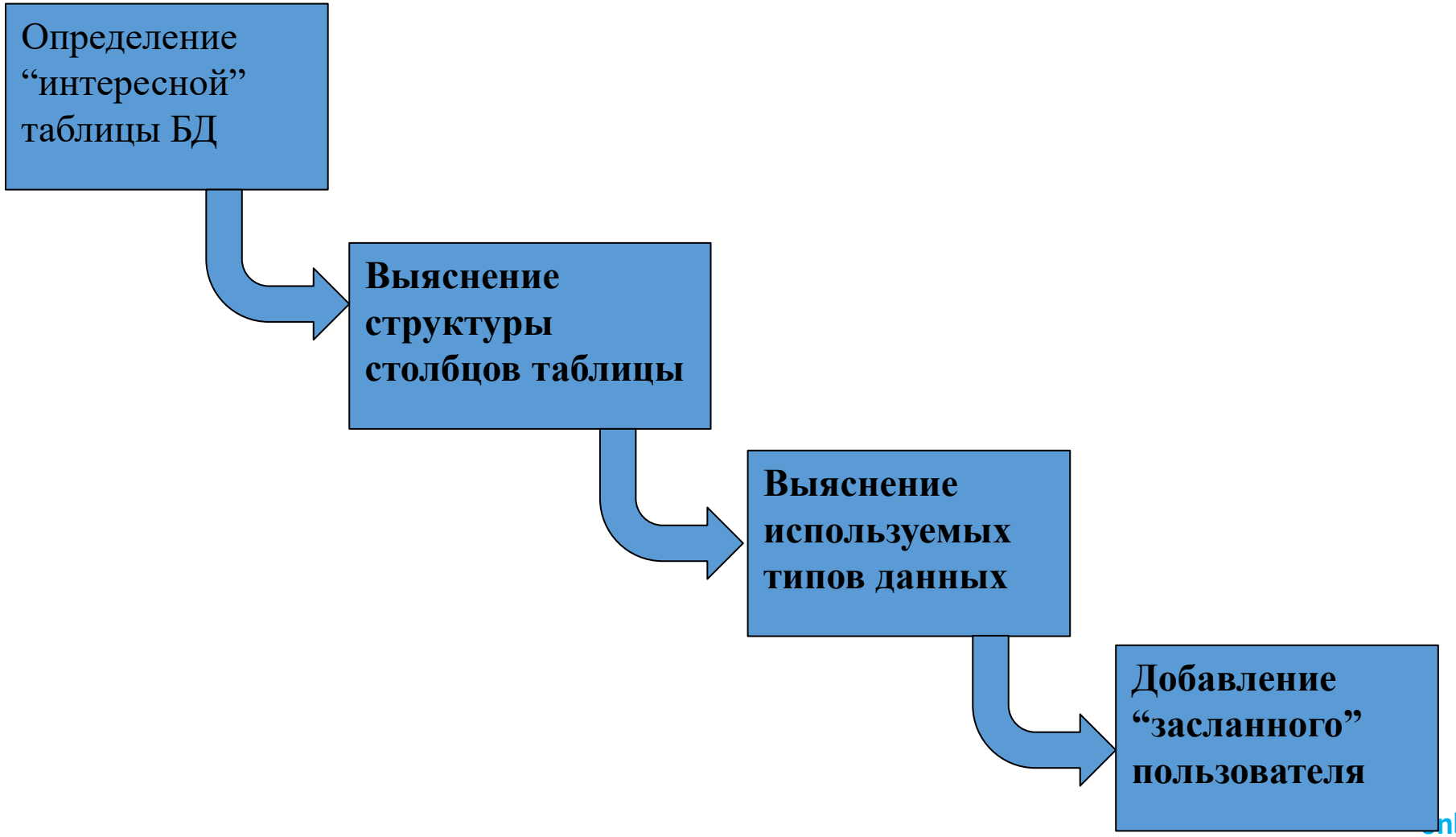
- Сервер отвечает, отправляя запрашиваемую страницу, с указание браузеру сохранить cookie: HTTP:/1.1 200 jr Content-type: text/html SET-cookie: name=value (содержимое страницы)

- При следующем обращении к серверу, браузер прикрепит cookie, вот так: GET /index.html HTTP/1.1 HOST: www.example.org cookie: name=value Accept:/

Уязвимость cookie

- Сервер отвечает, отправляя запрашиваемую страницу и, возможно, добавив новые cookie.
- Значение cookie может быть изменено сервером путём отправления новых строк SET- cookie: name=newvalue.
- cookie также могут устанавливаться программами на языках типа JavaScript, встроенными в текст страниц, или аналогичными скриптами, работающими в браузере.
- В JavaScript для этого используется объект document.cookie. Например, document.cookie = "temperature=20" создаст cookie под именем «temperature» и значением 20.
- isAdmin [0] -> [1]

Атака blind-перебором



Атака с использованием SQL Injection

- Это типы атак, которые могут быть выполнены с использованием **SQL-инъекции.**
- Атака нацеливается на динамические операторы **SQL**.
- Динамический оператор — это оператор, который создается во время выполнения на основе параметров из веб-формы или строки запроса **URI**.

Пример инъекции SQL

```
<form action='index.php' method="post">

<input type="email" name="email" required="required"/>

<input type="password" name="password"/>

<input type="checkbox" name="remember_me" value="Remember me"/>

<input type="submit" value="Submit"/>

</form>
```

```
SELECT * FROM users WHERE email = $_POST['email']
AND password = md5($_POST['password']);
```

Пример инъекции SQL

- SELECT * FROM users WHERE email = 'admin@admin.sys' AND password = md5('1234’);
- xxx@xxx.xxx' OR 1 = 1 LIMIT 1 -- ‘]
- SELECT * FROM users WHERE email = 'xxx@xxx.xxx' OR 1 = 1 LIMIT 1 -- '] AND password = md5('1234');

Методы борьбы с инъекциями

- Параметризованный запрос с плейсхолдерами – запрос с переменными, которые жестко прописаны в скрипте с кодом и не меняются в зависимости от данных.

- Escaping – функция, заложенная в языки программирования, которая заменит потенциально опасные SQL символы на безопасные escape-последовательности. Например, в PHP это: `mysql_real_escape_string`

- **Хранимые процедуры** — они могут инкапсулировать **SQL-запросы** и обрабатывать все входные данные в качестве параметров.

- **Регулярные выражения** — могут быть использованы для обнаружения потенциально вредоносного кода и его удаления перед выполнением операторов **SQL**.

- **Права доступа на подключение к базе данных** – чтобы **защититься от SQL инъекций**, учетным записям, которые используются для подключения к базе данных, должны предоставляться только необходимые права доступа. Это поможет ограничить действия, которые **SQL-операторы** могут выполнять на сервере.

- **Сообщения об ошибках** — не должны раскрывать конфиденциальную информацию. Простые пользовательские сообщения об ошибках, такие как «*Извините, возникла техническая ошибка. Служба поддержки уже уведомлена о ней. Повторите попытку позже*», можно использовать вместо отображения запросов **SQL**, вызвавших ошибку.

Чтение на дом

- OWASP Top-10 на русском: https://habr.com/ru/company/simplepay/blog/258499/
- Оригинал OWASP Top-10: https://owasp.org/www-project-top-ten/
- Методы и средства взлома БД MySQL: https://habr.com/ru/company/xakep/blog/256665/

Вопросы для самостоятельного изучения

- 1. Что такое GitHub? Для чего используется этот ресурс? Почему он является уязвимостью для СУБД?
- 2. Что такое HTTP-форма и как в общем виде формируется браузером HTTP-запрос?
- 3. Перечислите, какие бывают типы файлов cookie.

Спасибо за внимание!