Урок №20. Безопасность веб-приложений

План

1) Fingerprints.

2) Data leak (утечка данных).

3) Защищенное соединение HTTPS.

4) Манипуляции данными:

* XSS (cross site scripting)
* SQL injection
* CSRF (cross site request forgery)

5) Другие советы по безопасности.

[Видео 1](1.mp4)

[Видео 2](2.mp4)

[Видео 3](3.mp4)

[Видео 4](4.mp4)

Полезные ссылки:

<https://github.com/lukesky1/php-up/tree/1.0.12>

[Безопасность через неясность](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D0%BD%D0%B5%D1%8F%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C)

<https://httpd.apache.org/docs/2.4/mod/core.html#servertokens>

<https://habrahabr.ru/post/111489/>

<https://habrahabr.ru/post/111714/>

<https://habrahabr.ru/post/188042/>

<https://ru.wikipedia.org/wiki/OWASP>

Обязательно посмотрите это:

<https://youtu.be/FB8Y6ik7wnk>

Еще хочу разместить информацию, которую я нашел в комментарии на хабре (неплохо все систематизированно):

0. хватит шишек:

0.1. велосипедам — нет! используем известные библиотеки\фреймворки, где уже многое исправлено

0.2. своевременный апгрейд

0.3. логи и мониторинг подозрительной активности.

1. входные данные для каждого слоя:

1.1. слой контроллера (код): фильтрация данных из суперглобалов (всех)

1.1.1. инициализация данных, мусор может быть вредным (например, те же автоглобалы)

1.2. слой модели\данных (sql): используем prepared statements или иные placeholders, хотя бы экранизацию(escaping)

1.3. слой вида (html): используем escaping везде, где требуется вывод данных

1.4. слои не должны доверять друг другу и полагаться, что не придут вредные данные.

1.5. минимизация функциональности: делать что нужно и не больше (например, для отдачи файла НЕ надо его исполнять).

2. фильтрация данных:(xss+code\sql injection)

2.1. до максимально узкого значения. например, если должно быть число, пытаемся кастить в число.

2.2. до более широкого диапазона можно прогонять через регулярки. не забываем про длину.

2.3. если данные исчислимы (выбор из списка), то сопоставить реальным данным числовой идентификатор.

2.4. не перестараться, не портить данные «универсальными» чистками.

3. аутентичность:

3.1. минимальный срок жизни данных:

3.1.1. используем сессионные данные, для доступа используем трудноугадываемый идентификатор

3.1.2. для подтверждений: длинный ОДНОРАЗОВЫЙ ИД.

3.1.3. логин, повышение прав: регенерация ИД, подтверждение пароля (session hijacking)

3.1.4. формы и иные stateful запросы также имеют одноразовый ИД (csrf,request\form spoofing)

3.2. в зависимости от важности данных устанавливаем сложность проверки юзверя (двухфазный вход, генераторы паролей, заставляем сложные пароли)

3.3. права доступа (как ФС, так и системные)

4. client is evil:

4.1. ничего не храним на клиенте, даже в зашифрованном виде. максимум — session id в куках

4.2. валидация на клиенте бессмысленна, только для удобства пользователя

4.3. не полагаемся на данные об ОС\браузере\…

4.4. шифрование на клиенте бессмысленно, но необходимо (если вдруг что-то увели)

4.5. имена файлов, другие пароли и явки — ложь.

5. DoS предупреждение:

5.1. слишком длинные данные + размер загружаемых данных (картинки, например)

5.2. много неудачных попыток входа

5.3. много загрузок

5.4. много запросов, добавляющие данные (напр.комментарии)

6. О шифровании:

6.1. все пароли (в базе) — в виде хешей с солью, проверить на криптостойкость и коллизии.

6.2. для чувствительных данных — шифрование потока по HTTPS\SSL c коротким expiration

6.3. не храним данные в файлах, особенно — в открытом виде.

6.4. статика — отдельно от динамики, с соотв. правами доступа (либо читать, либо выполнять, либо писать)

6.4.1. желательно — статика очень отдельно от динамики (др. сервер, папка, за пределы корня) т.е. чтобы не пересекались

7. Лишнее:

7.1. ничего лишнего — например, никаких .svn, паролей, чеклистов, туду и вообще не жизненно важного для работы.

7.2. никаких инфо-заголовков — об используемом софте, версиях, путях, файлах, именах

7.3. никаких сообщений об ошибках, кроме подсказок юзверю, что он делает не так и что ему делать так.

7.4. источник должен быть виден, исключить возможность подделок, пример:

7.4.1. комментарии должны выглядеть как комментарии и не иначе

7.4.2. в заголовке всех окон ввода — четко имя сайта и функция, должно быть узнаваемо.

7.5. лучше использовать pretty urls — и для удобства, и сокрытия данных.

Безопасность системы определяется наиболее уязвимым местом!