Безопасность в Django

№ урока:	14	**Курс:**	Django Starter

Средства обучения:	Персональный компьютер с установленными: Python 3.8.2 Django 3.0.4

Обзор, цель и назначение урока

Целью данного урока является научиться тому, как на практике сделать свое приложение более безопасным и как избежать ошибок в защите Django-приложения на продакшене. Как избежать утечки данных, обезопасить данные пользователя и избежать случаев, когда приложение может быть захвачено третьими лицами.

Изучив материал данного занятия, учащийся сможет:

- Защитить свое приложение и приложение клиента от различных атак, таких как:
  - XSS-атаки
  - CSRF-атаки
  - SQL-атаки
- Сделать приложение более безопасным.
- Защитить данные пользователей.
- Использовать внешние сервисы для проверки безопасности .

Содержание урока

1) XSS-атаки и защита
2) CSRF-атаки и защита
3) Защита от SQL-атак
4) SSL и HTTPS
5) Проверка заголовка хоста
6) Пользовательский контент
7) Использование сеанса
8) Процесс загрузки файлов
9) Онлайн сервисы для проверки безопасности

Резюме

- **XSS** атаки позволяют пользователю вставить собственные JS скрипты в браузеры других пользователей. Это достигается с помощью сохранения вредоносных скриптов в базе данных, которые затем запрашиваются и отображаются браузерами других пользователей, или принуждением пользователя нажать на ссылку, которая позволит вредоносному скрипту выполниться в браузере пользователя. Однако, XSS атаки могут происходить из любого недоверенного источника данных, такого как куки или веб сервисы, в случае, когда данные были недостаточно очищены перед их размещением на странице.
- Использование шаблонов Django защищает вас от большинства XSS атак. Тем не менее, важно понимать, какую именно защиту они обеспечивают и где находится граница их возможностей.

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Шаблоны Django экранируют специальные символы, которые обычно создают проблемы для HTML. И хотя экранирование защищает пользователя от большинства видов вредоносного ввода, оно не является панацеей. Например, оно не защитит от такого: <style class={{ var }}>...</style>
- Если var содержит 'class1 onmouseover=javascript:func()', то это может вылиться в неавторизованный запуск JavaScript, здесь всё зависит от того, как браузер интерпретирует несовершенный HTML. (Установка кавычек вокруг значения атрибута решает данную проблему.)
- Важно обратить особое внимание на использование is_safe, совместно со сторонними шаблонными тегами, с шаблонным тегом safe, с mark_safe и когда автоматическое экранирование отключено.
- Дополнительно, если вы используйте шаблонную систему для вывода контента, отличного от HTML, то там могут быть отдельные символы и слова, которые требуют экранирования.
- Вы также должны быть очень осторожны при сохранении HTML в базе данных, особенно в случае, когда этот HTML будет отображаться впоследствии.
- **CSRF** атаки позволяют недобросовестному пользователю выполнять действия от имени другого пользователя, без ведома последнего или его согласия.
- Django обладает встроенной защитой против большинства типов CSRF атак, если вы активировали и использовали её там, где это необходимо. Однако, как это обычно бывает, существуют ограничения. Например, есть возможность отключить защиту CSRF глобально или на уровне отдельного представления. Такое можно делать, только если вы точно уверены в своих действиях. Существуют другие ограничения, если у вашего сайта есть поддомены, не находящиеся под вашим управлением.
- CSRF защита работает, проверяя метку с текущим временем в каждом POST запросе. Она не позволяет злоумышленнику просто сформировать POST запрос формы к вашему сайту и создать возможность другому авторизованному пользователю нечаянно отправить эту форму. Злоумышленнику потребуется знать содержимое метки, которое зависит от пользователя (используются куки).
- При работе через HTTPS, CsrfViewMiddleware будет проверять, что заголовок HTTP Referer установлен на URL того же источника (включая поддомен и порт). Так как HTTPS предоставляет дополнительную защиту, надо всегда проверять, что соединения используют его всегда, перенаправляя на HTTPS запросы по незащищённым соединениям и используя HSTS для браузеров, которые это поддерживают.
- Будьте очень внимательны, декорируя представления с помощью csrf_exempt, когда в этом нет явной необходимости.
- **Внедрение SQL** — это тип атаки, когда недобросовестный пользователь имеет возможность выполнить в базе данных определенный SQL запрос. Результатом выполнения такого запроса может быть удаление или даже утечка данных.
- При использовании Django ORM созданный SQL запрос будет правильно экранирован соответствующим драйвером базы данных. Однако, Django предоставляет разработчикам возможность писать запросы напрямую или выполнять собственные запросы. Эти возможности следует использовать умеренно и всегда обращать пристальное внимание на экранирование всех параметров, которые предоставлены пользователем. Также следует проявлять осторожность при использовании extra() и RawSQL.
- SSL/HTTPS. Применение этой защиты хорошо отражается на безопасности, хотя и не всегда уместно с практической точки зрения. При отсутствии HTTPS злоумышленник имеет возможность перехватывать аутентификационные данные или любую другую информацию, передаваемую между клиентом и сервером, а в случае активной атаки — может даже изменять данные, передаваемые в любом направлении.
- Если вам нужна защита, предоставляемая HTTPS, и на сервере произведена соответствующая настройка ПО, то надо выполнить ещё несколько шагов, чтобы быть уверенным в защите своей информации:

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- При необходимости, установите параметр конфигурации SECURE_PROXY_SSL_HEADER, чтобы показать, что вы поняли все предупреждения. Отказ от этого может привести к CSRF проблемам, а отказ сделать это правильно также может быть опасен!
- Настройте перенаправление HTTP запросов на HTTPS, указав True в SECURE_SSL_REDIRECT.
  Пожалуйста, обратите внимание на недостатки SECURE_PROXY_SSL_HEADER. Для случая обратного прокси может быть проще или более безопасно настроить главный веб сервер для перенаправление запросов по HTTPS.
- Использование 'безопасных' куки.
  Если браузер изначально подключается через HTTP, что характерно для большинства браузеров, есть возможность утечки существующих кук. По этой причине вам установить параметры SESSION_COOKIE_SECURE и CSRF_COOKIE_SECURE в True. Это заставит браузер отправлять такие куки только через HTTPS. Следует отметить, это сделает невозможным работу сессий через HTTP, а CSRF защита не будет принимать POST данные, полученные через HTTP (это решается с помощью перенаправления HTTP трафика через HTTPS).
- Использование HTTP Strict Transport Security (HSTS)
  HSTS является HTTP заголовком, который информирует браузер, что все следующие соединения к конкретному сайту всегда должны использовать HTTPS. Дополняя это перенаправлением HTTP запросов на HTTPS, мы получаем дополнительную защиту от использования SSL. HSTS может быть настроен с помощью настроек SECURE_HSTS_SECONDS и SECURE_HSTS_INCLUDE_SUBDOMAINS, или на web-сервере.

- Django использует заголовок Host, предоставляемый клиентом, для создания URL в определённых случаях. Несмотря на то, что эти данные безопасны с точки зрения Cross Site Scripting атак, поддельный заголовок Host может быть использован для атак CSRF, подмены кэша и для подмены ссылок в сообщениях электронной почты.
- Поскольку даже по-видимому безопасные конфигурации веб-сервера восприимчивы к поддельным заголовкам Host заголовки, Django проверяет этот заголовок относительно параметра конфигурации ALLOWED_HOSTS с помощью метода django.http.HttpRequest.get_host().
- Эта проверка применяется только через get_host(). Если ваш код получает содержимое заголовка Host напрямую из request.META, то вы игнорируете эту защиту.
- Предыдущие версии этого документа рекомендовали настраивать ваш веб сервер на проверку входящих HTTP заголовков Host. Рекомендация всё ещё в силе, на многих веб-серверах используется конфигурация, которая не проверяет заголовок Host, хотя по всем признакам должна это делать. Например, даже если Apache настроен таким образом, что ваш сайт работает на нестандартном виртуальном узле с установленным ServerName, всё ещё есть возможность предоставить поддельный заголовок для этого узла. Таким образом, Django теперь требует явного определения параметра конфигурации ALLOWED_HOSTS, не доверяя конфигурации самого веб сервера.
- Поддомены внутри сайта имеют возможность устанавливать куки на клиенте для всего домена. Это приводит к возможности фиксации сессии, если куки принимаются от поддоменов, которые не находятся под управлением доверенных пользователей.
- Например, атакующий может авторизоваться на good.example.com и получить достоверную сессию для своего аккаунта. Если у атакующего есть контроль над bad.example.com, он может использовать его для отправки своего ключа сессии вам, так как поддомену разрешено устанавливать куки для *.example.com. Когда вы посетите good.example.com, вы будете авторизованы как атакующий и можете непреднамеренно

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

внести важные персональные данные (например, номер кредитной карты) в аккаунт атакующего.

- Другой возможной атакой может быть ситуация, если сайт good.example.com имеет в параметре конфигурации SESSION_COOKIE_DOMAIN значение ".example.com", что может привести отправке сессионной куки на сайт bad.example.com.
- Если ваш сайт принимает файлы, настоятельно советуем ограничить размер таких загрузок в конфигурации веб сервера, для предотвращения атак на отказ сервиса (**DOS**). В случае Apache это легко можно сделать с помощью директивы LimitRequestBody.
- Если вы самостоятельно раздаете статические файлы, убедитесь, что обработчики, например mod_php в Apache, который будет выполнять статические файлы в виде кода, будут отключены. Вы же не хотите, чтобы пользователи могли выполнить произвольный код, путем загрузки и запроса специально созданного файла.
- Обработка загрузки медиа файлов в Django имеет некоторые уязвимости, если медиа файлы раздаются без учета некоторых правил безопасности. В частности, HTML файл может быть загружен в виде изображения, если этот файл содержит правильный заголовок PNG с последующим вредоносным HTML. Этот файл будет проходить проверку библиотеками, которые Django использует для обработки изображений (Pillow) в ImageField. Когда этот файл впоследствии отображается для пользователя, он может отображаться как HTML в зависимости от типа и конфигурации веб-сервера.
- Не существует идеального технического решения на уровне Django для безопасной проверки всех типов файлов, которые могут загрузить другие пользователи. Однако, можно предпринять некоторые другие способы, чтобы обезопасить себя от атаки:
- Один класс атак можно предотвратить путем раздачи медиа файлов с отдельного домена верхнего уровня или второго уровня. Это предотвращает любые эксплойты, которые могут быть заблокированы same-origin policy, таких как межсайтовый скриптинг. Например, если ваш сайт работает на example.com, вам следует раздавать медиа файлы (настройка MEDIA_URL) с другого домена, например usercontent-example.com. Но недостаточно раздавать медиа файлы с поддомена вида usercontent.example.com.
- Помимо этого, приложения могут определять белый список допустимых расширений файлов для загружаемых пользователем файлов и настроить веб-сервер чтобы раздавать только такие файлы.

Закрепление материала

- Какие существуют типы атак?
- Как избежать SQL атак?
- Какие есть методы защиты данных пользователя?
- Какие правильно загружать файлы от пользователя?
- Какие есть сервисы для проверки безопасности?

Дополнительное задание

Задание
Рассмотреть, безопасно ли приложение, которое создавалось в процессе уроков.

Самостоятельная деятельность учащегося

Задание 1
Изучить и понять все преимущества и недостатки инструментов, которые были рассмотрены на уроке.

Задание 2

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Запустить проверку приложения на сервисе для проверки безопасности.

Задание 3
Сделать ошибку в защите и попробовать взломать приложение.

Рекомендуемые ресурсы

Официальная документация Django:
https://www.djangoproject.com/

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com