[WAPT]

Оглавление

6.8 Security Misconfiguration

Security Misconfiguration или небезопасная конфигурация, что это? В нашем случае это неправильные настройки приложения, сервиса, фреймворка, базы данных, которые могут привести к утечке конфиденциальных данных или к полной компрометации системы (сервера). Зачастую небезопасная конфигурация возникает, когда параметры настроек безопасности были сохранены с дефолтными значениями или не были настроены должным образом. Для обеспечения безопасности сервера необходима детальная и развернутая настройка, также немаловажным фактором является своевременное обновление программного обеспечения на сервере и проверка настроек после обновлений, нередко с выходом обновлений изменяются и параметры настроек.

Общие наиболее часто встречающиеся ошибки конфигурации:

- Учетные записи с дефолтными значениями логин:пароль, включенные гостевые (анонимные) доступы.
- Неверно настроенные веб сервера web.config, .htacces
- Установлены и включены ненужные функции, приложения, службы (открыты порты)
- Использование не обновленных (не пропатченных) CMS, плагинов к CMS.

Все вышеперечисленное относится к общим ошибкам конфигурации, но помимо этого при проведении тестирования стоит обращать внимание на все приложения, работающие на сервере. Внимательно прочитать

документацию по приложению, узнать особенности настроек. Нужно понимать, что множество приложений и сервисов “из коробки” уже являются уязвимыми. Можно привести пример сетевых камер видеонаблюдения. В большинстве своем при установке производитель видеокамер просто напоминает администратору, что нужно заменить пароль, но само собой далеко не каждый это предупреждение воспринимает всерьез.

Поговорим о паре отдельных случаев misconfiguration.

Небезопасный CORS

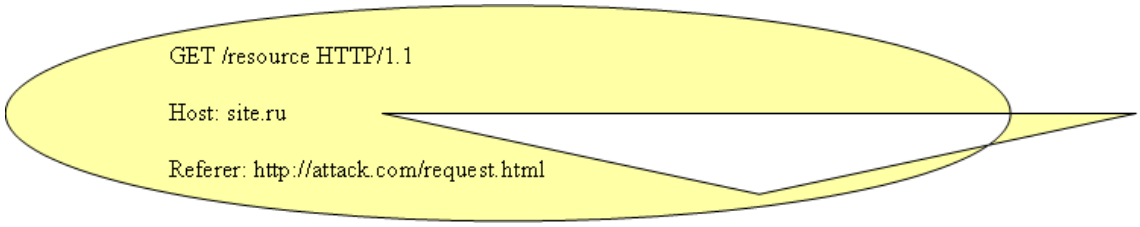
Cross-origin resource sharing – технология, которая позволяет предоставить веб-странице доступ к ресурсам другого домена. Подробнее про данную технологию можно прочитать здесь.

Первым шагом в понимании CORS является знание того, как работают некоторые функции безопасности веб-браузеров. По умолчанию веб-браузеры не разрешают AJAX-запросы на сайты, кроме сайта, который вы посещаете. Это называется политикой единого происхождения, и это важная часть модели веб-безопасности. Совместное использование ресурсов между разными источниками (cross-origin resource sharing) — это механизм HTML 5, который дополняет политику единого происхождения для упрощения совместного использования ресурсов домена между различными веб-приложениями.

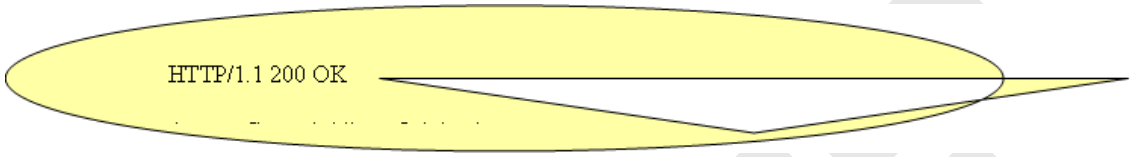
CORS определяет набор заголовков, которые позволяют серверу и браузеру определять, какие запросы для междоменных ресурсов разрешены, а какие нет.

Наиболее распространенная проблема безопасности при внедрении CORS — это отказ от проверки запроса белых списков. Зачастую разработчики устанавливают значение для AccessControl-Allow-Origin в '*'. Это позволяет любому домену в Интернете получать доступ к ресурсам этого сайта.

Запрос:



Ответ:



Основная проблема кроется в том, что многие компании размещают API в пределах домена, не ограничивая к нему доступ политикой "белого списка". Это порождает уязвимость.

Большинство веб-приложений использует файлы cookie для отслеживания информации о сеансе. При генерации, cookie ограничены определенным доменом. При каждом HTTP запросе к этому домену браузер подставлять значение cookie, созданные для этого домена. Это относится к каждому HTTP запросу — для получения изображений, страниц или AJAX-вызовов.

Что это означает на практике: при авторизации в example1.ru, cookie генерируются и хранятся для этого домена. Веб-приложение example2.ru содержит API на example1.ru/api для взаимодействия с помощью AJAX. Предположим, что вы просматриваете example2.ru, будучи авторизованным на example1.ru. Без ограничения Access-Control-Allow-Origin по белому списку (с указанием сайта) example2.ru может выполнить любой разрешенный аутентифицированный запрос к example1.ru, даже не имея прямого доступа к сессионной cookie. Это связано с тем, что браузер автоматически привязывает файлы cookie к example1.ru для любых HTTP запросов в этом домене, включая AJAX запросы от example2.ru в example1.ru. Таким образом, атакующий может взаимодействовать даже с внутренним ресурсом, недоступным в сети интернет и находящимся в корпоративной сети.

Сканер для поиска небезопасных конфигураций CORS

```
https://github.com/RUB-NDS/CORStest.git
```

XST - Cross Site Tracing

Межсайтовую трассировку тоже можно отнести к разряду небезопасной конфигурации, данный метод представляет из себя сочетания XSS + HTTP METHOD TRACE

XST позволяет получить пользовательские cookies даже если на сервере установлен флаг HttpOnly. Например, если мы заставим жертву просто выполнить наш XXS и на сервере будет включен HttpOnly нашему скрипту будут недоступны cookies. Но если XXS выполнит TRACE запрос, то файлы cookies вернутся вместе с TRACE.

Как обнаружить и проэксплуатировать XSS доступно в одноименной главе. Обнаружить доступность метода TRACE можно с помощьюскрипта nmap

```
nmap example.com -p 80 --script http-methods
```

Security Misconfiguration в системах управления контентом.

Сюда относится:

- Оставление установочных файлов CMS;
- Комментарии в коде;
- Листинг директорий;
- Определение CMS и ее версии по названию файлов, путей, плагинов;
- Права на чтение конфигурационных файлов;
- Неверно настроенные права доступов на файлы и папки.

Misconfiguration в сторонних сервисах

Не стоит забывать про такие службы как smb и ftp, часто на этих сервисах остаются включены анонимные и гостевые доступы, при определенном стечении обстоятельств и неправильной конфигурации

может быть возможность залить бекдор (вебшел), посмотреть код веб-приложений недоступный для браузера.

Security Misconfiguration - это даже не вектор атаки, а человеческий фактор, которым мы можем воспользоваться в основном из-за неправильной конфигурации. Появляются уязвимости LFI, RCE, SQLI. Каких- то определенных техник нет, каждый случай уникален. Проверяем дефолтные значения, читаем документацию по настройке.

Session fixed

Session fixed является подвидом Session hijacking, другими словами кража сессии пользователя. Представим себе ситуацию когда *http://example.com* принимает любой идентификатор сессии из строки запроса и при этом никак его не валидирует. Тогда злоумышленник может провести следующий трюк. Прислать пользователю, который имеет аккаунт на *http://example.com*, ссылку вида *http://example.com?SID=123456789*. Если пользователь пройдет по этой ссылке, то он попадает на страницу входа в аккаунт. После того как пользователь вводит свой логин и пароль и авторизуется, злоумышленник может воспользоваться *SID=123456789,* чтобы авторизоваться без ввода логина и пароля.

Вариант, при котором идентификатор сессии генерируется на сервере тоже можно использовать. Разница лишь в том, что прежде чем отправить ссылку с SID пользователю, злоумышленник сам делает так, чтобы сервер вернул ему сгенерированный ID сессии после чего порядок действий тот же, что и выше.

Полезные ссылки:

- Технология CORS Developer Mozilla
- Сканер CORS Github
- Словари с дефолтными данными для многих сервисов Github