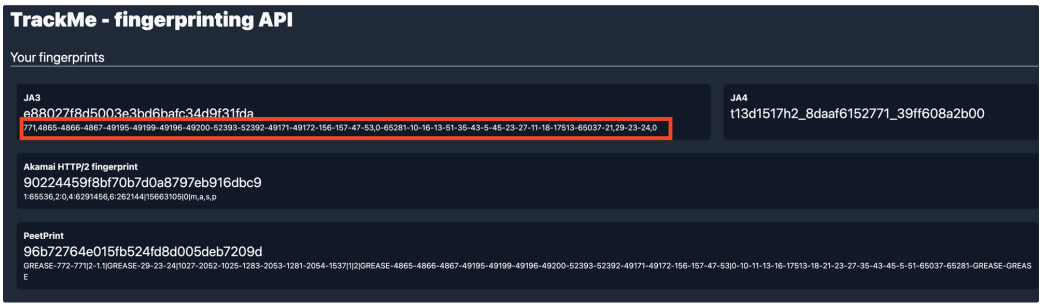
Chromium, доработка boringssl

У tls соединения есть фингерпринт, почитать что это такое можно вот тут
https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967/

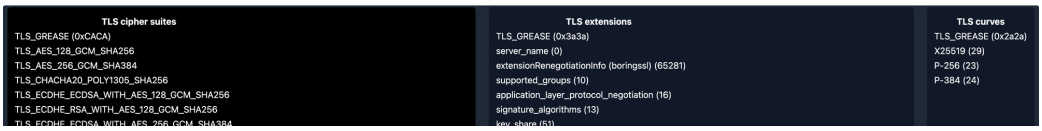
если коротко, то когда происходит инициализация https соединения, отправляется ClientHello пакет, в котором перечислены поддерживаемые клиентом алгоритмы шифрования + экстеншены, это нужно, чтобы договориться о шифровании с сервером. Тк у каждого браузера свой набор алгоритмов и последовательность в пакете, то это позволяет определить браузер или http клиент (его отпечаток/fingerprint). Этот фингепринт используется для защиты от парсинга и ботов. Мы парсим, поэтому есть задача сделать библиотеку, которая сможет эмулить любой набор фингерпринтов.

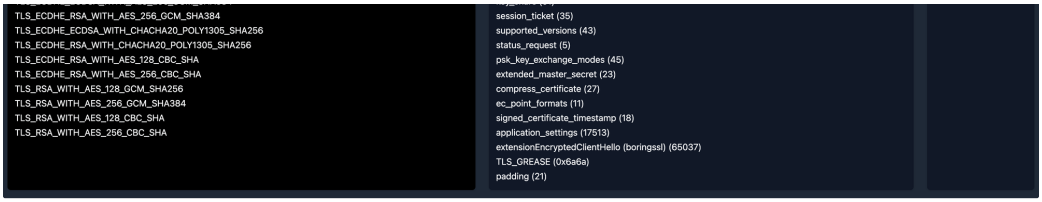
Вот тут можно посмотреть свой фингерпринт и почитать инфу дополнительно
https://tls.peet.ws/

речь только про ja3 фингерпринт (он самый первый)



вот тут список расширений и алгоритмов шифрования для моего браузера, как пример





те строка
771,4865-4866-4867-49195-49199-49196-49200-52393-52392-49171-49172-156-157-47-53,51-18-5-23-65037-45-10-27-17513-13-16-35-43-0-11-65281-41,29-23-24,0

с первого скриншота - это коды алгоритмов шифрования и расширений на втором скрине

мы уже делали что-то подобное для старой версии хромиум (внутри он использует boringssl либу для ssl соединения), это было давно, наша доработка позволяла выстраивать последовательность из уже поддерживаемых алгоритмов шифрования и расширений, весь этот код есть и будет передан исполнителю

тк хромиум тяжело и долго собирается, то мы сделали тестовый проект, который делает запрос по https (с использованием boringssl, которую использует хромиум) и возвращает результат в stdout, код тестового приложения тоже есть

Что нужно сделать:

1. взять наш код для старой версии хромиум и доработать тестовый проект, это позволит менять последовательность и состав пакета ClientHello, естественно, использоваться будут только поддерживаемые алгоритмы шифрования + расширения. Смысл там примерно такой - кладем рядом с проектом файл finger.txt вот такого содержания (ja3 фингерпринт)

   771,4865-4866-4867-49195-49199-49196-49200-52393-52392-49171-49172-156-157-47-53,51-18-5-23-65037-45-10-27-17513-13-16-35-43-0-11-65281-41,29-23-24,0

   после чего список алгоритмов и расширений в ClientHello выстраивается в соответствии с этим fingerprint'ом и использоваться при инициализации tls соединения

используется при инициализации tls соединения

2. написать тест, который будет делать следующее: кладем рядом файл fingerprint_test.txt
   в нем будет список фингерпринтов, каждый с новой строки, тест должен проверять - все ли алгоритмы и расширения из фингерпринта поддерживает boringssl, если какого-то нет - писать об этом, если все есть - делаем запрос вот сюда https://tls.peet.ws/api/tls и проверяем, что текущий фингерпринт соответствует полю ja3, если не соответствует, значит что-то не так - пишем лог

это все.

Это первый этап, следующий будет заключаться в реализации всех недостающих алгоритмов шифрования и экстеншенов, которые мы найдем в результате тестирования, они все есть в других опенсорс либах, те задача будет интегрировать их в проект. Следующий этап оплачивается отдельно.