Анализ приложения `baskerville`

Введение

Итак, разработано клиент-серверное приложение для защищенной пересылки файлов, названное `baskerville`. Файлы пересылаются односторонним образом: от клиента к серверу. Передаваемые файлы (их содержимое) подписываются при помощи электронной подписи. На сервере файл может быть размещен (по желанию клиента) в одной из *корзин*.

В этой секции будут описаны возможные векторы атаки на приложения и способы противодействия им. Здесь имеет смысл рассматривать только специфические атаки, связанные с заданным в условии задачи протоколом и особенностями реализации.

Здесь и далее мы считаем, что нарушитель частично контролирует канал передачи: может читать передаваемые сообщения, может формировать и отправлять на сервер собственные сообщения. Используемый в реализации протокол gRPC мы считаем “прозрачным” для нарушителя.

1. Конфиденциальность

Данные передаются открыто и могут быть прочитаны нарушителем (задача защиты конфиденциальности в условии не ставилась).

2. Подделка и нарушение целостности файлов при пересылке

Для аутентификации файлов и контроля их целостности используется схема подписи RSA в сочетании с хэш-функцией SHA-512, реализация взята из OpenSSL. На данный момент нет причин считать, что у выбранных криптопримитивов при использовании случайно выработанных секретных ключей длиной не менее 2048 битов есть эксплуатируемые уязвимости, если секретный ключ не скомпрометирован. Таким образом, содержимое передаваемых файлов можно считать подлинным (в том смысле, что их подписал обладатель секретного ключа).

3. Открытые каталоги

Режим листания работает без аутентификации, поэтому любой нарушитель может подключиться к серверу и ознакомиться с содержимым корзин. Идентификаторы корзин дописываются к имени заданного каталога, при этом получается абсолютный путь до корзины. Теоретически, используя специально подобранные идентификаторы корзин (например, “/../../../etc/”), нарушитель мог бы ознакомиться с содержимым всей файловой системы. В реализации, однако, обрабатываются только идентификаторы, прописанные в конфиге сервера, и атака невозможна.

4. DoS-атака

Благодаря тому, что запросы клиента не аутентифицируются, возможна DoS-атака путем посылки множества бессмысленных файлов, возможно, со случайно сгенерированными подписями, что заставит сервер тратить ресурсы на проверку неверных подписей.

5. Replay-атака

Нарушитель может повторно направить точную копию сообщения (вместе с ЦП), что приведет к перезаписи хранимого файла. Рассмотрим такую ситуацию:

- Легальный клиент посылает файл 1.txt, нарушитель сохраняет копию его сообщения.
- Легальный клиент изменяет файл 1.txt и отправляет его на сервер.
- Нарушитель воспроизводит сохраненное сообщение, файл 1.txt перезаписывается старой копией.

6. Атака с перезаписью файла

Нарушитель, перехвативший данные одного запроса, может послать его повторно, заменив в запросе поле имени имя файла, и

контент будет записан в файл с новым именем, возможно, перезаписав корректно размещенный файл. Для противодействия атаке нужно подписывать запрос на размещение файла.

Если имя файла содержит символы вида “../”, то в принципе можно было бы выйти за границы каталога и записать файл в произвольное место файловой системы. Чтобы этого не происходило, в реализации сервер обрезает имя файла при помощи *basename()*.

7. Атака с подменой корзины

Нарушитель, перехвативший данные одного запроса, может послать его повторно, заменив id корзины, и файл (вернее, его копия) будет записан не туда, куда предполагал отправитель.

8. Атака с подменой файла и корзины

Комбинация предыдущих двух атак: нарушитель подменяет одновременно корзину и имя файла. Итак, перехватив один запрос,

9. Человек посередине

Если рассмотреть более сильную модель нарушителя, “человека посередине”, который в состоянии перехватывать и изменять передаваемые сообщения, то такой нарушитель сможет, заменяя соответствующие поля в запросе, перенаправить запись файла в любую корзину и любой файл по своему желанию в режиме передачи файлов, а в режиме листания подделать вывод, заменив идентификатор корзины. Более того, поскольку канал в направлении сервер$\rightarrow$клиент никак не защищен от модификации сообщений, такой нарушитель сможет подделать ответы сервера произвольным образом.

Выводы

Против разработанного приложения можно применить целое семейство атак, связанных с тем, что проверяется лишь подпись под контентом файла. Для борьбы с ними можно предложить следующие общие методы:

- Скорректировать процесс отправки, подписывая не только контент файла, но и его имя, и корзину назначения.
- Сделать протокол листания аутентифицированным, подписывая запрос. Еще лучше сделать весь протокол общения клиент-сервер трехшаговым:
  1. Клиент инициирует соединение с сервером.
  2. Сервер вырабатывает одноразовое случайное значение $Nonce_S$, отправляет клиенту.
  3. Клиент готовит запрос, в который включает подпись от набора ($Nonce_S$, id корзины) для режима листания и ($Nonce_S$, id корзины, имя файла) для режима отправки, отдельно подписывая контент файла, и передает соответствующие поля и подпись (подписи) серверу.
  4. Сервер обрабатывает только те запросы, для которых подпись верна. При этом заодно уменьшится нагрузка на сервер при попытке DoS-атак.
- Для борьбы с “человеком посередине” можно выдать серверу собственный секретный ключ ЦП и аналогичным образом аутентифицировать сообщения сервера:
  1. Клиент обращается к серверу, посылая одноразовое случайное значение $Nonce_C$.
  2. Сервер вырабатывает одноразовое случайное значение $Nonce_S$, отправляет клиенту.
  3. Клиент готовит запрос, в который включает подпись от набора ($Nonce_S$, $Nonce_C$, id корзины) для режима листания и ($Nonce_S$, $Nonce_C$, id корзины, имя файла) для режима отправки, отдельно подписывая контент файла, и передает соответствующие поля и подпись (подписи) серверу.
  4. Сервер обрабатывает только те запросы, для которых подпись верна. К ответу сервера добавляются ($Nonce_S$, $Nonce_C$), ответ подписывается ЦП сервера.
- Обернуть весь трафик протокола в TLS.