

Анализ приложения **baskerville**

Введение

Итак, разработано клиент-серверное приложение для защищенной пересылки файлов. 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*.

1. Атаки на приложение

В этой секции мы опишем возможные векторы атаки на приложения и способы противодействия им.

1.1. Конфиденциальность

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

1.2. 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 подписи RSA при использовании случайно выработанных секретных ключей длиной около 2048 битов есть эксплуатируемые уязвимости, если секретный ключ не скомпрометирован.

1.3. Открытые каталоги

Режим листания работает без аутентификации, поэтому любой нарушитель может подключиться к серверу и ознакомиться с содержимым корзины. Идентификаторы корзины дописываются к имени заданного каталога, при этом получается абсолютный путь до корзины. Теоретически, используя специально подобранные идентификаторы корзины (например, `“/../../etc/”`), нарушитель мог бы ознакомиться с содержимым всей файловой системы. В реализации, од-

нако, обрабатываются только идентификаторы, прописанные в конфиге сервера, и атака невозможна.

1.4. 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.

1.5. 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.

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

1.7. Атака с подменой корзины

Нарушитель, перехвативший данные одного запроса, может послать его повторно, заменив id корзины, и файл будет записан не туда, куда предполагал отправитель. Для противодействия атаке нужно подписывать запрос на размещение файла.

1.8. Атака с подменой файла и корзины

Комбинация предыдущих двух атак: нарушитель подменяет одновременно корзину и имя файла.

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 обращается к серверу за “квитанцией”.
 2. Сервер вырабатывает одноразовое случайное значение – “квитанцию” Nonce, отправляет клиенту.
 3. Клиент готовит запрос, в который включает подпись от набора (квитанция, id корзины) для режима листания и (квитанция, 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.
- Обернуть весь трафик протокола в TLS.