Разработка Системы Аутентификации:

Диалог в четырех сценах

Исходный текст написан Биллом Брайантом (Bill Bryant) в феврале 1988.

Приведено в порядок и конвертировано в HTML Теодором Тсо (Theodore Ts'o)
в феврале 1987 года. Им же добавлено послесловие, описывающее изменения
в Версии 5 протокола Kerberos.

Перевод выполнен Дмитрием Стасюком <wintermute@undenied.ru> в январе 2011 года.
Оригинал доступен по адресу http://web.mit.edu/kerberos/www/dialogue.html

Краткое содержание

Диалог представляет собой вымышленную летопись разработки сетевой системы аутентификации с названием "Харон". В ходе диалога персонажи Афина и Еврипид обнаруживают проблемы в безопасности, характерные для открытых сетей. Каждая проблема должна быть учтена в конструкции Харона, и его конструкция соответственно видоизменяется. Афина и Еврипид не закончат работу пока диалог не завершится.

Когда они заканчивают разработку системы, Афина изменяет имя системы на "Kerberos" (Цербер), как бы случайно совпадающее с названием системы аутентификации, которая была разработана и реализована в проекте Афина в МИТ. Система Цербер имеет поразительное сходство с системой Kerberos: Сервисом Аутентификации для Систем в Открытых Сетях, представленной на зимней конференции USENIX в 1988 году в Далласе, штат Техас.

Оглавление

Действующие лица

Афина многообещающий разработчик систем

Еврипид опытный разработчик и местный критик

Сцена I

Маленькая комнатка. Афина и Еврипид работают за соседними терминалами.

Афина Знаешь, Рип, эта система с разделением времени просто тормоз. Я ничего не могу

сделать, потому что в нее зашел кто-то еще.

Еврипид Мне только не жалуйся. Я просто тут работаю.

Афина Знаешь что нам нужно? Нам нужно дать каждому его собственную рабочую станцию. Тогда им не придется заботится о разделении машинных циклов. Мы используем сеть чтобы соединить все рабочие станции, и люди смогут общаться друг с другом.

Еврипид Прекрасно. Значит нам нужны ... примерно тысяча рабочих станций?

Афина Плюс-минус.

Еврипид Ты видела какого размера диски на среднестатистической рабочей станции? Там не хватит места для всего софта, который у нас есть на машине с разделением времени.

Афина С этим я уже разобралась. Мы можем держать копии системного ПО на нескольких серверах. Когда входишь на рабочую станцию, она получает доступ к системному ПО устанавливая сетевое соединение с одним из серверов. Эта схема позволяет всем рабочим станциям использовать одну и ту же копию системного ПО и делает удобным его обновление. Не надо заходить на каждую рабочую станцию. Изменения вносятся только на серверах с системным ПО.

Еврипид Ну, хорошо. А что ты собираешься делать с личными файлами? В системе с разделением времени я могу войти и увидеть все свои файлы с любого терминала, подключенного к системе. У меня будет возможность подойти к любой рабочей станции и автоматически добраться до своих файлов? Или мне придется поступать как пользователю персоналки и хранить файлы на дискете? Надеюсь что нет.

Афина Думаю, что мы можем использовать другие машины для персонального файлового хранилища. Ты сможешь зайти на любую рабочую станцию и добраться до своих файлов.

Еврипид А как же печать? У каждой рабочей станции есть собственный принтер? Ты чьи деньги тратишь? Что насчет электронной почты? Как ты собираешься доставлять почту на все эти рабочие станции?

Афина Ну ... Ладно. Очевидно что у нас нет денег для того чтобы дать принтер каждому, но у нас могут быть машины, выделенные для сервиса печати. Отправляешь задание на сервер печати, и он его для тебя печатает. Нечто похожее можешь сделать и с почтой. Имея машину, выделенную для почтового сервиса. Тебе нужна почта - соединяешься с почтовым сервером и забираешь свою почту.

Еврипид Твоя система рабочих станций выглядит очень хорошо, Фина. Когда у меня будет рабочая станция, знаешь что я сделаю? Я выясню какое у тебя имя пользователя и заставлю свою рабочую станцию считать что я это ты. Потом я подключусь к почтовому серверу и заберу твою почту. Я подключусь к файл-серверу и удалю твои файлы, и ...

Афина Ты сможешь это сделать?

Еврипид Конечно! Как все эти сетевые серверы узнают что я не ты?

Афина Блин, я не знаю. Кажется, мне надо немного подумать.

Еврипид Похоже на то. Дай мне знать когда разберешься.

Сцена II

Офис Еврипида на следующее утро. Еврипид сидит за столом и читает почту. Афина стучит в дверь.

Афина Я поняла как обезопасить среду открытой сети настолько, чтобы недобросовестные товарищи вроде тебя не могли пользоваться сетевыми сервисами под именами других людей.

Еврипид Неужели? Садись.

Она садится.

Афина Перед тем как я это расскажу, могу я установить одно основополагающее правило для этой дискуссии?

Еврипид Что за правило?

Афина Предположим, я говорю что-то вроде: "Мне нужна моя электронная почта, поэтому я соединяюсь почтовым сервером и прошу его отправить почту на мою рабочую станцию". На самом деле не я та сущность, которая контактирует с почтовым сервером. Для соединения с почтовым сервером и извлечения своей почты я использую программу, являющуюся КЛИЕНТОМ программы, предоставляющей почтовый сервис.

Но я не хочу говорить "клиент делает то-то и то-то" каждый раз когда я ссылаюсь на взаимодействие пользователя и сервера в сети. Я хотела бы сказать просто "я делаю то-то и то-то", подразумевая, естественно, что программа-клиент совершает действия от моего лица. Тебя это устраивает?

Еврипид Конечно. Без проблем.

Афина Хорошо. Так, я начну с формулировки проблемы, которую решила. В открытой сетевой среде машины, предоставляющие сервисы, должны иметь возможность удостоверить личности людей, которые обращаются к сервису. Если я соединяюсь с почтовым сервером и требую свою почту, программа сервиса должна быть в состоянии проверить что я та, за кого себя выдаю, правильно?

Еврипид Да.

Афина Можно решить проблему "в лоб", заставив почтовый сервер запрашивать пароль перед тем как он разрешит себя использовать. Я докажу серверу кто я, предоставив ему свой пароль.

Еврипид Это точно решение "в лоб". В такой системе каждый сервер должен знать твой пароль. Если в сети тысяча пользователей, каждый сервер должен знать тысячу паролей. Если ты захочешь изменить свой пароль, тебе придется соединиться со всеми серверами и уведомить их об изменении. Полагаю, твоя система не настолько тупая.

Афина Моя система не тупая. Она работает так: пароли должны быть не только у людей, у сервисов они должны быть тоже. Каждый пользователь знает свой пароль, каждый сервис знает свой пароль, и существует СЕРВИС АУТЕНТИФИКАЦИИ, который знает ВСЕ пароли - пароль каждого пользователя и каждого сервиса. Сервис аутентификации хранит пароли в одной, централизованной базе данных.

Еврипид У тебя есть название для этого сервиса аутентификации?

Афина Я еще не думала об этом. У тебя есть идеи?

Еврипид Как звали того парня, который перевозил мертвых через реку Стикс?

Афина Харон?

Еврипид Ага, он самый. Он не перевезет тебя через реку пока ты не сможешь подтвердить свою личность.

Афина Ты сам туда отправишься, Рип, если попытаешься еще раз переписать греческую мифологию. Харона не интересует твоя личность. Ему нужно только убедиться в том, что ты мертвый.

Еврипид А у тебя есть имя получше?

Пауза.

Афина Вообще-то, нет.

Еврипид Ну, тогда пусть сервис аутентификации называется "Харон".

Афина Ладно. Наверное, мне следует описать систему?

Скажем, ты хочешь воспользоваться сервисом, почтовым сервисом. В моей системе ты не можешь пользоваться сервисом до тех пор пока, м-м-м, Харон не скажет сервису, что ты тот за кого себя выдаешь. И ты не получишь разрешение использовать сервис до тех пор, пока не удостоверишь себя перед Хароном. Когда ты запрашиваешь аутентификацию у Харона, ты должен сказать ему для какого сервиса тебе требуется подтверждение. Если хочешь использовать почтовый сервер, придется сказать Харону.

Харон попросит тебя подтвердить свою личность. Ты это делаешь предоставляя свой секретный пароль. Харон берет твой пароль и сравнивает его с тем, который зарегистрирован для тебя в базе данных Харона. Если два пароля совпадают, Харон считает, что твоя личность подтверждена.

Теперь Харон должен убедить почтовый сервер в том, что ты тот, кем ты себя называешь. Поскольку Харон знает пароли всех сервисов, он знает пароль почтового сервиса. Понятно что Харон мог бы дать тебе пароль, который ты мог бы передать почтовому сервису в качестве доказательства того, что ты подтвердил свою личность перед Хароном.

Проблема в том, что Харон не может дать тебе непосредственно пароль, потому что в этом случае ты его узнаешь. Когда в следующий раз тебе понадобится почта, ты сможешь обойти Харона и использовать почтовый сервер без точной идентификации себя. Ты сможешь даже притвориться кем-то другим и воспользоваться почтовым сервером под именем другого человека.

Так что, вместо того чтобы дать тебе пароль от сервера почты, Харон дает тебе БИЛЕТ к почтовому сервису. Этот билет содержит версию твоего имени пользователя, ЗАШИФРОВАННУЮ С ИСПОЛЬЗОВАНИЕМ ПАРОЛЯ ПОЧТОВОГО СЕРВЕРА.

Обладая билетом, ты можешь потребовать у почтового сервиса свою почту. Ты делаешь запрос, говоря почтовому серверу кто ты такой и предоставляя билет, который подтверждает что ты тот, кто, как ты говоришь, ты есть.

Сервер использует свой пароль чтобы расшифровать билет и, если билет расшифруется корректно, у сервера окажется имя пользователя, которое Харон поместил в билет.

Сервис сравнивает это имя с именем, которое ты послал вместе с билетом. Если имена совпадают, почтовый сервер считает что твоя личность подтверждена и приступает к отдаче тебе твоей почты.

Что думаешь обо всем этом?

Еврипид У меня есть кое-какие вопросы.

Афина Понятно. Задавай.

Еврипид Когда программа сервиса расшифровывает билет откуда она знает, что расшифровала билет правильно?

Афина Я не знаю.

Еврипид Может быть, тебе стоит включить в билет имя сервиса. Таким образом, когда сервис расшифрует билет, он сможет понять что у него получилось по факту того сможет он или нет найти свое имя в расшифрованном билете.

Афина Звучит неплохо. Значит, билет выглядит примерно так:

(Она вывела в блокноте:)

БИЛЕТ - {пользователь:сервис}

(TICKET - {username:servicename})

Еврипид Значит билет к сервису содержит только имя твоего пользователя и имя сервиса?

Афина Зашифрованные паролем сервиса.

Еврипид Не думаю, что этой информации достаточно, чтобы сделать билет защищенным.

Афина Что ты имеешь ввиду?

Еврипид Предположим, ты просишь у Харона билет к почтовому серверу. Харон составляет этот билет так, что в нем содержится твое имя пользователя "tina". Предположим, я копирую этот билет, в то время как он несется по сети от Харона к тебе. Предположим, я заставляю поверить мою незащищенную рабочую станцию в то, что мое имя пользователя - "tina". Программа почтового клиента на моей рабочей станции считает что я это ты. От твоего имени программа отправляет украденный билет почтовому серверу. Сервер расшифровывает билет и видит, что он правильный. Имя пользователя в билете совпадает с именем пользователя, который прислал билет. Почтовый сервер отдает мне твою почту ...

Афина О! Ну, это не очень здорово.

Еврипид Но, мне кажется, я знаю как исправить эту проблему. Или, как минимум, предоставить ее частичное решение. Мне кажется, Харону следует включать больше информации в создаваемые им билеты к сервисам. В дополнение к имени пользователя, в билет следует также включить СЕТЕВОЙ АДРЕС, с которого пользователь запросил у Харона билет. Это даст дополнительный уровень защиты.

Я поясню. Предположим, я сейчас краду твой билет к почте. Билет содержит внутри сетевой адрес твоей рабочей станции, и этот адрес не совпадает с адресом моей рабочей станции. От твоего имени я отправляю похищенный билет на почтовый сервер. Программа на сервере извлекает имя пользователя и сетевой адрес из билета и пытается найти соответствие этой информации с именем пользователя и сетевым адресом сущности, которая прислала билет. Имя пользователя совпадает, но сетевой адрес - нет. Сервер не принимает билет, потому что очевидно, что он был украден.

Афина Браво, браво! Вот если бы я до этого додумалась.

Еврипид Что ж, вот что я хотел сказать.

Афина Значит измененная структура билета выглядит как-то так:

Она вывела на доске:

БИЛЕТ - {пользователь:адрес_станции:сервис}

(TICKET - {username:ws_address:servicename})

Афина Я просто в восторге. Давай построим систему Харон и посмотрим работает ли она!

Еврипид Не так быстро. У меня есть еще несколько вопросов о твоей системе.

Афина Ладно. *(Афина наклонилась вперед на стуле)* Давай.

Еврипид Похоже что я должен получать новый билет каждый раз, когда хочу воспользоваться сервисом. Если я работаю полный рабочий день, я, вероятно, хочу получать почту больше одного раза. Должен ли я получать новый билет каждый раз когда я хочу забрать почту? Если это так, то мне не нравится твоя система.

Афина О ... Ну, не вижу причин почему билеты не могут быть повторно использованы. Если ты получил билет для почтового сервера, то обязан иметь возможность использовать его снова и снова. Например, когда почтовая программа делает запрос к сервису от твоего имени, она отправляет КОПИЮ билета почтовому серверу.

Еврипид Уже лучше. Но у меня все еще есть проблемы. Похоже, что ты подразумеваешь, что я должен предоставлять Харону свой пароль каждый раз когда захочу использовать сервис, для которого у меня нет билета. Я захожу на рабочую станцию и хочу получить доступ к своим файлам. Я выпаливаю Харону запрос соответствующего билета и это означает, что я должен использовать свой пароль. Потом я хочу прочитать почту. Еще один запрос Харону. Я снова должен ввести свой пароль. Теперь, предположим, я хочу отправить одно из писем на сервер печати. Еще запрос к Харону, и ... ну, ты поняла.

Афина О, да, поняла.

Еврипид И, если это все было еще не очень плохо, подумай об этом: похоже, когда ты подтверждаешь свою личность Харону, то посылаешь свой секретный пароль по сети открытым текстом. Умные люди, как твой покорный слуга, могут прослушивать сеть и украсть копии паролей людей. Если я получил твой пароль, то могу использовать любой сервис от твоего имени.

Афина вздыхает.

Афина Это серьезные проблемы. Кажется, мне нужно начать все с начала.

Сцена III

Следующее утро. Афина перехватывает Еврипида в буфете. Она хлопает его по плечу в то время как он наливает воду в чашку.

Оба идут к кофеварке.

Афина У меня есть новая версия Харона, которая решает наши проблемы.

Еврипид Неужели? Быстро.

Афина Ну, знаешь, эти проблемы мне всю ночь не давали спать.

Еврипид Наверное, угрызения совести. Может нам стоит пойти вон в ту переговорку?

Афина Почему бы и нет?

Оба перемещаются в маленькую комнату для переговоров.

Афина Я снова начну с изложения проблем, но переверну их так, что они станут требованиями к системе.

Афина прокашливается.

Афина Первое требование: Пользователи должны вводить свои пароли только один раз в начале сеанса работы на своих рабочих станциях. Это требование подразумевает, что тебе не надо будет вводить свой пароль каждый раз, когда тебе понадобится новый билет к сервису. Второе требование: пароли не должны пересылаться по сети открытым текстом.

Еврипид Хорошо.

Афина Начну с первого требования: пароль необходимо использовать только один раз. Я выполнила это требование путем изобретения нового сетевого сервиса. Он называется сервис "выдачи билетов". Этот сервис выдает билеты Харона пользователям, которые уже подтвердили свою подлинность Харону. Ты можешь пользоваться этим сервисом выдачи билетов если у тебя есть для него билет - билет для выдачи билета.

Сервис выдачи билетов, на самом деле, просто версия Харона ввиду того, что он имеет доступ к базе данных Харона. Он часть Харона, которая разрешает тебе подтвердить свою подлинность билетом вместо пароля.

Как бы там ни было, система аутентификации теперь работает следующим образом: ты заходишь на рабочую станцию и запускаешь программу с именем kinit чтобы установить соединение с сервером Харон. Ты подтверждаешь свою личность Харону, и программа kinit выдает тебе билет для выдачи билетов.

Теперь, скажем, ты хочешь забрать свою почту с почтового сервера. У тебя еще нет билета к почтовому серверу, так что ты используешь билет для "выдачи билетов" чтобы получить свой билет к почтовому серверу. Тебе не нужно использовать пароль для получения нового билета.

Еврипид Я должен получать новый билет для "выдачи билетов" каждый раз когда мне нужно подключиться к новому сетевому сервису?

Афина Нет. Помнишь, в прошлый раз мы договорились, что билеты могут повторно использоваться. Как только ты получил билет для выдачи билетов тебе не нужно получать еще один. Ты используешь билет для выдачи билетов для получения других билетов, которые тебе понадобятся.

Еврипид Хорошо. В этом есть смысл. И, поскольку можно повторно использовать билеты, как только сервис выдачи билетов выдал тебе билет для конкретного сервиса, тебе не нужно получать билет для этого сервиса снова.

Афина Ага. Разве не здорово?

Еврипид Да, пока ничего ... Если не пришлось отправлять пароль открытым текстом по сети при получении билета для выдачи билетов.

Афина Как я уже сказала, эту проблему я тоже решила. Суть вот в чем. Когда я говорю что ты должен связаться с Хароном чтобы получить билет для выдачи билетов, это звучит так как будто ты должен послать свой пароль открытым текстом по сети на сервер Харона. Но все не должно быть так.

Вот как все происходит на самом деле. Когда ты запускаешь программу kinit чтобы получить билет для выдачи билетов, она не отправляет твой пароль на сервер Харона. kinit посылает только твое имя пользователя.

Еврипид Прекрасно.

Афина Харон использует имя для поиска твоего пароля. Затем Харон собирает пакет данных, которые содержат билет для выдачи билетов. Перед тем как отправить тебе пакет Харон шифрует содержимое пакета используя твой пароль.

Твоя рабочая станция принимает пакет с билетом. Ты вводишь пароль. kinit пытается расшифровать билет с паролем, который ты ввел. Если ему удается, ты успешно подтвердил свою подлинность Харону. Теперь ты обладаешь билетом для выдачи билетов, и этот билет может добыть для тебя другие билеты, которые тебе потребуются.

Ну и как это, если рассудить с пристрастием?

Еврипид Не знаю ... Я пытаюсь сосредоточиться. Знаешь, мне кажется, что отдельные части системы, которую ты только что описала, работают довольно хорошо. Твоя система требует от меня подтвердить свою подлинность только один раз. Впоследствии Харон может выдать мне билеты к сервисам без моего вмешательства. Безупречно, безупречно в этом отношении. Но есть кое-что в конструкции сервиса билетов, что несколько меня беспокоит. Он имеет дело с билетами, которые могут быть повторно использованы. Я согласен с тем, что они должны повторно использоваться, но повторно используемые билеты, по своей природе, очень опасны.

Афина Что ты имеешь ввиду?

Еврипид Посмотри на них с такой точки зрения. Предположим, ты используешь небезопасную рабочую станцию. В ходе твоего сеанса входа на нее ты получаешь билет к сервису почты, билет к сервису печати и к файловому сервису. Предположим, что ты, по неосмотрительности, оставила эти билеты на рабочей станции когда выходила из системы.

Теперь предположим, что я вхожу на рабочую станцию и нахожу эти билеты. У меня есть желание сделать гадость, и я заставляют рабочую станцию думать что я это ты. Поскольку билеты выписаны на твое имя, я могу использовать программу почтового клиента для доступа к твоей почте, могу использовать клиента файлового сервиса для удаления твоих файлов или доступа к ним, могу пользоваться сервисом печати чтобы накрутить огромные расходы на твоем счете. И все потому, что эти билеты случайно остались там лежать.

Ничто не сможет удержать меня от копирования этих билетов себе. Я могу продолжать пользоваться ими вечно.

Афина Но это легко исправить. Мы просто пишем программу, которая уничтожает пользовательские билеты после каждого завершения сеанса работы. Ты не сможешь использовать билеты, которые были уничтожены.

Еврипид Очевидно, что твоя система должна иметь программу уничтожения билетов, но глупо заставлять пользователей полагаться на такие вещи. Ты не можешь рассчитывать на то, что пользователи не забудут уничтожить свои билеты при каждом завершении сеанса на рабочей станции. И, даже если ты полагаешься на то, что твои пользователи будут уничтожать свои билеты, подумай над следующим сценарием.

У меня есть программа, которая просматривает сеть и копирует билеты к сервисам когда они мчатся по сети. Предположим, я хочу сделать тебя жертвой. Я жду когда ты начнешь сеанс на рабочей станции, а потом запускаю свою программу и копирую пачку твоих билетов.

Я жду когда ты закончишь сеанс работы, и, в конце концов, выйдешь из системы и уйдешь. Я вожусь с сетевыми программами на своей рабочей станции и изменяю ее адрес так, что он совпадает с адресом рабочей станции, которую ты использовала при получении скопированных мной билетов. Я заставляю рабочую станцию поверить в то, что я это ты. У меня есть твои билеты, твое имя пользователя и правильный сетевой адрес. Я могу СНОВА ИСПОЛЬЗОВАТЬ эти билеты и пользоваться сервисами от твоего имени.

Не имеет значения, что ты уничтожила свои билеты перед окончанием сеанса на рабочей станции. Билеты, которые я украл, действительны до тех пор, пока я склонен ими пользоваться. Потому что твоя текущая структура билета не устанавливает ограничение на то, сколько раз можно повторно использовать билет, или как долго билет остается действующим.

Афина О, я понимаю о чем ты говоришь! Билеты не могут быть действительны вечно, потому что в этом случае они будут представлять серьезную угрозу безопасности. Мы должны ограничить время, в течение которого билет может быть использован. Возможно, добавить билету своего рода срок действия.

Еврипид Именно. Я думаю, каждый билет должен иметь два дополнительных блока данных: время жизни, показывающее длительность времени, в течение которого действует билет, и отметку времени, отмечающую дату и время, когда Харон выдал билет. Таким образом, билет будет выглядеть как-то так:

Еврипид идет к доске и выводит на ней:

БИЛЕТ {пользователь:адрес:сервис:время_жизни:отметка_времени}

(TICKET {username:address:servicename:lifespan:timestamp})

Еврипид Теперь когда сервис расшифровывает билеты, он сравнивает имя пользователя в билете и адрес с именем и адресом лица, приславшего билет. А также использует информацию об отметке времени и времени жизни чтобы определить не истек ли срок действия билета.

Афина Согласна. Какое время жизни должно быть у обычного билета к сервису?

Еврипид Я не знаю. Вероятно, такой же как средняя длительность сеанса на рабочей станции. Скажем, восемь часов.

Афина То есть, если я сижу за рабочей станцией больше чем восемь часов, у всех моих билетов закончится время действия. Включая мой билет для выдачи билетов. Значит, я должна снова подтвердить свою личность Харону после восьми часов.

Еврипид Это ведь не безосновательно, да?

Афина Думаю, нет. Значит, мы договорились - срок действия билетов истекает через восемь часов. теперь у меня есть к тебе вопрос. Предположим, я скопировала ТВОИ билеты из сети ...

Еврипид *(Хлопает глазами)*

О, Тина! Ты бы, на самом деле, не сделала этого, правда?

Афина Это только для доказательства. Я скопировала твои билеты. Теперь я жду когда ты выйдешь из системы. Предположим, у тебя назначен визит к врачу или занятие, которое нужно посетить. Ты заканчиваешь сеанс на рабочей станции через пару часов. Ты умный парень и уничтожил свои копии билетов перед выходом из системы.

Но я украла твои билеты, и они действуют еще примерно шесть часов. Это дает мне достаточно времени чтобы разграбить твои файлы и напечатать тысячу копий

неважно чего от твоего имени.

Видишь, вариант с временем жизни и отметкой времени хорошо работает в случае, если вор, укравший билет, решает воспользоваться билетом после того как его срок действия истек. Если же вор может воспользоваться им раньше ...

Еврипид Ну, что ж ... Конечно, ты права.

Афина Думаю, у нас будет серьезная проблема.

(Она вздыхает)

Пауза.

Еврипид Мне кажется, это означает, что ты будешь занята сегодня. Хочешь еще кофе?

Афина Почему бы и нет?

Сцена IV

Утро следующего дня в офисе Еврипида. Афина стучит в дверь.

Еврипид У тебя сегодня круги под глазами.

Афина Ну, знаешь. Еще одна длинная ночь.

Еврипид Ты решила проблему повторного использования билетов?

Афина Кажется, да.

Еврипид Садись.

Она садится.

Афина Как обычно, я считаю, что необходимо снова сформулировать проблему. Билеты могут повторно использоваться в пределах ограниченного отрезка времени, скажем восьми часов. Если кто-то украдет билеты и решит воспользоваться ими до того как истечет их срок действия, мы не сможем ничего сделать чтобы остановить его.

Еврипид Это проблема.

Афина Мы сможем решить проблему если спроектируем билеты таким образом, чтобы они не могли быть повторно использованы.

Еврипид Но тогда придется получать новый билет каждый раз когда понадобится использовать сетевой сервис.

Афина Точно. Это, в лучшем случае, корявое решение. *(Пауза.)* Я уже перешла к аргументации? *(На мгновение она задумалась.)*

Ладно. Я снова переформулирую проблему. На этот раз в форме требований. Сетевой сервис должен быть в состоянии проверить то, что человек, который использует билет тот же самый, кому этот билет был выдан.

Давай я пройду по шагам процесс аутентификации, и посмотрим, смогу ли я найти подходящий способ для объяснения моего решения этой проблемы.

Я хочу использовать определенный сетевой сервис. Я получаю доступ к этому сервису посредством запуска программы-клиента на своей рабочей станции. Клиент отправляет машине с сервисом три параметра: мое имя, сетевой адрес моей рабочей станции и соответствующий билет к сервису.

Билет содержит имя человека, которому он был выдан, и адрес рабочей станции, которую этот человек использовал в момент, когда он или она получал билет. Он

также содержит срок действия билета в виде времени жизни и отметки времени. Вся эта информация была зашифрована паролем Харона от сервиса.

Наша текущая схема аутентификации полагается на следующие тесты:

- Может ли сервис расшифровать билет?
- Истек ли срок действия билета?
- Совпадает ли имя и адрес рабочей станции, указанные в билете, с именем и адресом человека, который прислал билет?

Что проверяют эти тесты? Первый тест проверяет пришел данный билет от Харона или нет? Если билет не может быть расшифрован, то он пришел не от настоящего Харона. Настоящий Харон зашифровал бы билет паролем сервиса. Только Харон и сам сервис знают пароль сервиса. Если билет успешно расшифрован, сервис знает, что билет пришел от настоящего Харона. Этот тест не дает людям создавать поддельные билеты Харона.

Второй тест проверяет время жизни и отметку времени выдачи билета. Если срок действия билета истек, сервис не принимает билет. Этот тест удерживает людей от использования старых билетов, билетов которые, вероятно, были украдены.

Третий тест сверяет имя пользователя билета и адрес с именем и адресом человека, указанными в билете. Если тест проваливается, то пользователь билета получил (возможно тайком) билет другого человека. Билет, естественно, не принимается.

Если имена и адреса совпадают, то что подтвердил этот тест? Ничего. Мошенники могут украсть билеты в сети, изменить адреса своих рабочих станций и имена пользователей нужным образом и обшарить ресурсы других людей. Как я отметила вчера, билеты могут быть вновь использованы до тех пор пока не истек их срок действия. Они могут быть вновь использованы потому, что сервис не может определить является ли на самом деле человек приславший билет законным владельцем этого билета.

Сервис не может принять это решение потому, что он не поделился ключом с пользователем. Посмотри на ситуацию так. Если я на вахте в Элсиноре, ну, ты знаешь, в замке в шекспировском Гамлете, а ты должен сменить меня, я не должна позволить тебе занять мое место, если ты не сможешь сказать правильный пароль. Вот пример, когда у нас двоих есть общий ключ. И, вероятно, это ключ, который кто-то другой придумал для каждого, кто стоит на вахте.

И вот я думала сегодня ночью, почему бы Харону не создать пароль для законного владельца билета, который будет известен сервису? Харон дает одну копию этого сеансового ключа сервису, а вторую - пользователю. Когда сервис получает билет от пользователя, он может воспользоваться сеансовым ключом для того, чтобы проверить личность пользователя.

Еврипид Подожди секунду. Как Харон собирается отдавать обоим участникам сеансовый ключ?

Афина Владелец билета получает сеансовый ключ как часть ответа от Харона. Вот так:

Она выводит на доске:

ОТВЕТ ХАРОНА - [сеансовый_ключ|билет]

([sessionkey|ticket])

Копия ключа для сервиса приходит внутри билета. Сервис получает ключ когда

расшифровывает билет. Так что, билет выглядит вот так:

БИЛЕТ - {ключ:пользователь:адрес:сервис:время_жизни:отметка_времени}

(TICKET - {sessionkey:username:address:servicename:lifespan:timestamp})

Когда тебе хочется получить доступ к сервису, программа-клиент, которую ты запускаешь, создает то, что я называю АУТЕНТИФИКАТОР. Аутентификатор содержит твое имя и адрес рабочей станции. Клиент шифрует эту информацию сеансовым ключом. Копию сеансового ключа ты получил когда запросил билет.

АУТЕНТИФИКАТОР - {пользователь:адрес} зашифрованный сеансовым ключом

(AUTHENTICATOR - {username:address})

После создания аутентификатора клиент отправляет его и билет сервису. Сервис пока не может расшифровать аутентификатор, потому что у него нет сеансового ключа. Этот ключ есть в билете. Так что сначала сервис расшифровывает билет.

После расшифровки билета у сервиса появляется следующая информация:

- Время жизни и отметка времени создания билета;
- Имя владельца билета;
- Сетевой адрес владельца билета;
- Сеансовый ключ.

Сервис смотрит не истек ли срок действия билета. Если в этом отношении все хорошо, то следующим шагом сервис использует сеансовый ключ для расшифровки аутентификатора. Если расшифровка проходит без помех, у сервиса появляются имя пользователя и сетевой адрес. Сервис сверяет эту информацию с именем и адресом, найденными в билете, И с именем и адресом человека, который прислал билет и аутентификатор. Если все совпадает, сервис решает, что отправитель билета без сомнения является его настоящим владельцем.

Афина делает паузу, прокашливается, отпивает кофе.

Я думаю, что использование сеансового ключа-аутентификатора решает проблему вновь используемых билетов.

Еврипид Может быть. Но мне интересно ... Чтобы взломать эту версию системы у меня должен быть подходящий аутентификатор для сервиса.

Афина Нет. У тебя должен быть аутентификатор И билет для сервиса. Аутентификатор бесполезен без билета, потому что сервис не сможет расшифровать аутентификатор без предварительного получения соответствующего сеансового ключа. И сервис не сможет получить подходящий сеансовый ключ без предварительной расшифровки билета.

Еврипид Хорошо, это я понимаю, но разве ты не утверждаешь, что когда клиентская программа соединяется с сервером, она отправляет билет вместе с соответствующим ему аутентификатором?

Афина Да, кажется, это я говорила.

Еврипид Если это то, что происходит на самом деле, то что удержит меня от кражи билета и аутентификатора одновременно? Уверен, что я смог бы написать программу, которая это сделает. Если я получу билет и аутентификатор к нему, полагаю, что смогу использовать их обоих пока не истечет срок действия билета. Мне просто нужно будет соответствующим образом изменить адрес своей рабочей станции и имя пользователя. Так?

Афина *(Покусывая губу)* Так. Досадно.

Еврипид Стой, стой, стой! Это не такая большая проблема. Билеты могут быть повторно использованы пока не истек срок их действия, но это не значит, что аутентификаторы должны повторно использоваться. Предположим, что мы построили систему так, что аутентификаторы могут быть использованы только один раз. Это нам что-нибудь дает?

Афина Ну, может быть. Посмотрим. Программа-клиент создает аутентификатор и затем отправляет его вместе с билетом сервису. Ты копируешь и билет и аутентификатор в то время как они перемещаются от моей рабочей станции к серверу. Но билет и аутентификатор прибывают на сервер до того как ты сможешь отправить свои копии. Если аутентификатор может быть использован только один раз, то твоя копия этого аутентификатора уже не годна, и у тебя ничего не получится когда ты попытаешься повторно использовать свой билет и аутентификатор.

Что ж, уже легче. Итак, все что нам нужно сделать - это придумать способ сделать аутентификаторы одноразовыми.

Еврипид Никаких проблем. Давай просто поместим в них время жизни и отметку времени. Предположим, у каждого аутентификатора время жизни - пара минут. Когда ты хочешь воспользоваться сервисом, твоя программа-клиент создает аутентификатор, впечатывает в него текущее время, а затем отправляет его и билет на сервер.

Сервер принимает билет и аутентификатор и начинает их обрабатывать. Когда сервер расшифровывает аутентификатор, то проверяет время жизни и отметку времени аутентификатора. Если срок действия аутентификатора еще не истек и все остальное успешно проверено, сервер полагает, что ты прошла проверку подлинности.

Предположим, я скопировал аутентификатор и билет пока они пересекали сеть. Я должен изменить на своей рабочей станции сетевой адрес и имя пользователя. И я должен сделать все это за пару минут. Это довольно высокое требование. На самом деле, я не думаю что это возможно. Если не ...

Что ж, есть одна потенциальная проблема. Предположим, что вместо копирования билета и аутентификатора когда они путешествуют от твоей рабочей станции к серверу, я копирую пакет с исходным билетом, который идет от Харона. Того, который ты получаешь, когда просишь Харона выдать тебе билет.

Этот пакет, насколько я помню, содержит две копии сеансового ключа: один для тебя, а второй для сервиса. Копия для сервиса спрятана внутри билета, и я не могу до нее добраться. Но что насчет той копии, которую ты используешь чтобы создавать аутентификаторы?

Если я смогу получить эту копию сеансового ключа, то смогу создать свои собственные аутентификаторы, и, раз я могу создавать свои аутентификаторы, я могу взломать систему.

Афина Это то, о чем я думала прошлой ночью. Но потом я прошла по шагам процесс получения билетов и выяснила, что украсть аутентификаторы таким образом невозможно.

Ты сидишь за своей рабочей станцией и запускаешь программу kinit для получения билета для выдачи билетов. kinit запрашивает твое имя пользователя и, после того как ты его вводишь, отправляет имя Харону.

Харон использует это имя чтобы найти твой пароль, а затем начинает создавать для тебя билет для получения билетов. Как часть этого процесса, Харон создает сеансовый ключ, который ты будешь использовать вместе с сервисом выдачи билетов. Харон помещает копию сеансового ключа в билет для выдачи билетов, а твою копию - в пакет с билетом, который ты собираешься получить. Но перед тем как отправить этот пакет тебе, Харон шифрует все это твоим паролем.

Харон отправляет пакет по сети. Кто-то может скопировать пакет пока он передается, но не сможет с ним ничего сделать, потому что пакет был зашифрован твоим паролем. В связи с этим, никто не может украсть сеансовый ключ для получения билетов.

kinit получает пакет с билетом и запрашивает у тебя пароль, который ты вводишь. Если ты ввел правильный пароль, kinit может расшифровать пакет и предоставить тебе копию сеансового ключа.

Теперь, разобравшись с kinit, ты хочешь получить свою почту. Ты запускаешь программу почтового клиента. Эта программа ищет билет к почтовому сервису и не находит (ты же ведь еще не пытался получать почту). Клиент должен воспользоваться билетом для выдачи билетов чтобы запросить у сервиса выдачи билетов билет к почтовому сервису.

Клиент создает аутентификатор для запроса получения билета и шифрует аутентификатор своей копией сеансового ключа для получения билетов. Затем клиент отправляет Харону аутентификатор, билет для выдачи билетов, твое имя, адрес твоей рабочей станции и имя почтового сервиса.

Сервис выдачи билетов принимает все это и пропускает через набор проверок подлинности. Если все проверки успешно пройдены, у сервиса выдачи билетов появляется копия сеансового ключа, который он выдал тебе. Теперь сервис выдачи билетов создает для тебя билет к почтовому сервису и, в ходе этого процесса, создает новый сеансовый ключ, чтобы ты смог использовать его совместно с почтовым сервисом.

Сервис выдачи билетов подготавливает пакет с билетом, который отправит на твою рабочую станцию. Пакет содержит билет и твою копию сеансового ключа к почтовому сервису. Но, перед тем как отправить пакет, сервис выдачи билетов шифрует пакет своей копией сеансового ключа ДЛЯ ВЫДАЧИ БИЛЕТОВ. Дело сделано, пакет отправлен в путь.

Итак, у нас тут пакет с билетом к почтовому сервису, скачущий по сети. Предположим, некий сетевой огр копирует его пока тот движется. Огру не везет, потому что пакет зашифрован сеансовым ключом для выдачи билетов; только ты и сервис выдачи билетов знаете этот ключ. Поскольку огр не может расшифровать пакет с почтовым билетом, то не может и узнать ПОЧТОВЫЙ СЕАНСОВЫЙ КЛЮЧ. Без этого сеансового ключа огр не может воспользоваться ни одним билетом к почтовому сервису, которые ты можешь впоследствии отправить по сети.

Так что, думаю что мы в безопасности. Как считаешь?

Еврипид Возможно.

Афина Возможно! И это все что ты можешь сказать!

Еврипид *(смеется)* Не расстраивайся. Ты уже должна знать мои методы. Наверное, это низко с моей стороны, а ты не спала полночи.

Афина Пфффффф!

Еврипид Ну ладно, три четверти ночи. Вообще-то, система начинает выглядеть приемлемо. Использование сеансового ключа решает проблему, о которой я думал прошлой ночью: проблему взаимной аутентификации.

Пауза.

Не возражаешь если я минутку поговорю?

Афина *(Немного прохладно)* Да сколько хочешь.

Еврипид Ты так добра. *(Еврипид прокашливается.)* Прошлой ночью, пока в твоей голове плясали версии сеансовых ключей и аутентификаторов, я пытался обнаружить новые проблемы в системе. И я нашел одну, которая показалась мне довольно серьезной. Я поясню ее следующим сценарием.

Предположим, что тебя тошнит от твоей работы, и ты решила, что для тебя будет лучше уйти. Ты хочешь напечатать резюме на потрясном лазерном принтере компании, чтобы кадровые агентства и потенциальные наниматели могли отметить твой стиль.

И вот, ты набираешь команду для печати и указываешь ей отправить резюме на соответствующий сервер печати. Команда получает нужный билет к сервису, если у тебя его еще нет, а затем отправляет от твоего имени билет на соответствующий сервер печати. По крайней мере, ты думаешь, что он отправлен туда. Ты не знаешь наверняка, что запрос отправлен на нужный сервер печати.

Предположим, что некий бессовестный хакер - скажем, твой босс - так изнасиловал систему, что перенаправил твой запрос и билет к почтовому серверу в свой кабинет. Его программу сервиса печати не интересует билет и его содержимое. Она выбрасывает билет и отправляет сообщение на твою рабочую станцию, в котором говорится, что билет прошел проверку, и что сервер готов и жаждет распечатать твое задание. Команда печати отправляет задание поддельному серверу печати и твое резюме оказывается у врага.

Я изложу проблему с другой стороны. Без сеансовых ключей и аутентификаторов Харон может защитить свои сервера от фальшивых пользователей, но не может защитить своих пользователей от фальшивых серверов. Системе нужен способ, которым программы-клиенты проверят подлинность сервера перед тем, как отправить важную информацию сервису. Система должна предусматривать взаимное подтверждение подлинности.

Но сеансовый ключ решает эту проблему, если ты должным образом проектируешь программы-клиенты. Возвращаясь к сценарию с сервером печати. Для печати мне нужна программа-клиент, которая следит за тем чтобы сервис, которому она отправляет задания, был легальным сервисом.

Вот что делает такая программа. Я ввожу команду печати и передаю ей имя файла, содержащего мое резюме. Будем считать, что у меня есть билет к сервису печати и сеансовый ключ. Программа-клиент использует сеансовый ключ для построения аутентификатора, а затем отправляет аутентификатор и билет на "предполагаемый" сервер печати. Клиент пока НЕ отправляет само резюме; он ждет ответа от сервиса.

Настоящий сервис принимает билет и аутентификатор, расшифровывает билет и извлекает сеансовый ключ, затем использует сеансовый ключ для расшифровки аутентификатора. Когда все сделано сервис запускает все необходимые проверки подлинности.

Будем считать, что тесты подтвердили мою личность. Теперь сервер

подготавливает пакет с ответом таким образом, чтобы он подтвердил подлинность сервера программе-клиенту. Он использует свою копию сеансового ключа для шифрования пакета с ответом, а затем отправляет пакет ожидающему клиенту.

Клиент принимает пакет и пытается расшифровать его при помощи моей копии сеансового ключа. Если пакет расшифровывается правильно и в нем содержится корректное ответное сообщение, моя программа-клиент знает, что сервер, который зашифровал пакет, - настоящий сервер. Теперь клиент отправляет задание печати резюме сервису печати.

Предположим, мой босс изнасиловал систему так, что его сервер печати представляется тем, который мне нужен. Моя программа-клиент отправляет аутентификатор и билет "сервису печати" и ждет ответа. Поддельный сервис печати не может сгенерировать корректный ответ, потому что не может расшифровать билет и получить сеансовый ключ. Моя программа-клиент не будет отправлять задание до тех пор, пока не получит корректный ответ. В конце концов, клиент перестанет ждать и завершит работу. Мое задание печати не будет завершено, но, по крайней мере, мое резюме не попадет на стол врагу.

Знаешь, я думаю что у нас есть прочная основа для создания системы аутентификации Харон.

Афина Возможно. Тем не менее, мне не нравится название "Харон".

Еврипид Не нравится? С каких это пор?

Афина Мне оно никогда не нравилось, потому что в нем нет смысла. Я вчера говорила об этом со своим дядей Аидом, и он предложил другое название, имя его трехголового сторожевого пса.

Еврипид О, ты имеешь ввиду Cerberus (Цербер).

Афина Прикуси язык, Рип! Как же, Cerberus ...

Еврипид Ну, разве у него не такое имя?

Афина Такое, если ты римлянин! Я - греческая богиня, он - греческий сторожевой пес, и его имя "Kerberos" (Цербер). "Kerberos" с буквы K.

Еврипид Ладно, ладно, не надо метать молнии. Я соглашусь на имя. Вообще-то, оно сюда хорошо подходит. Прощай Харон и здравствуй Kerberos.

Послесловие

Диалог был написан в 1988 году с целью помочь его читателям понять фундаментальные причины того, почему протокол Kerberos V4 был таким каким он был. Годы он выполнял свою работу очень хорошо.

Когда я конвертировал этот документ в HTML, я был поражен насколько этот документ все еще был применим к протоколу Kerberos V5. Несмотря на то, что многое изменилось, базовые идеи ядра протокола остались те же. На самом деле, Kerberos V5 отличается от протокола "Kerberos" в этом диалоге только двумя изменениями.

Первое изменение появилось после признания того, что использования короткого пятиминутного сдвига времени не всегда было достаточно для предотвращения атак повторного использования билетов от атакующего, который использовал программу для автоматического захвата билета и аутентификатора в то время как они пересекают сеть, а затем немедленно отправил их снова для начала атаки.

В протоколе Kerberos V5 аутентификаторы сделаны по настоящему "одноразовыми" благодаря тому, что серверы, принимающие билеты, получили "кэш повторного

использования", который хранит информацию об аутентификаторах, недавно представленных серверу. Если атакующий пытается утащить аутентификатор и повторно его использовать, даже в течение разрешенного пятиминутного интервала кэш повторного использования будет в состоянии определить, что аутентификатор уже был представлен серверу.

Второе важное изменение в протоколе в том, что билет больше не шифруется паролем пользователя при пересылке его с сервера Kerberos программе kinit во время первоначального обмена билетами. Билет уже зашифрован секретным ключом сервера выдачи билетов; более того, когда он фактически используется для получения других билетов, то все равно отправляется по сети без шифрования. Исходя из этого, нет причин тому, чтобы билет был еще раз зашифрован паролем пользователя. (Все остальные данные в ответе сервера Kerberos пользователю, содержащие, например, пользовательскую копию сеансового ключа билета, конечно, по прежнему шифруются паролем пользователя.)

Похожее изменение было также внесено в протокол сервиса выдачи билетов (TGS - ticket granting service); билеты, возвращаемые TGS также больше не шифруются сеансовым ключом билета для выдачи билетов, поскольку билеты приложения уже зашифрованы секретным ключом сервера приложения. Так что, например, пакет, который в Kerberos V4 выглядел бы вот так:

ОТВЕТ_KDC* = {БИЛЕТ, клиент, сервер, K_сеанса}K_пользователя

где "{X}K_Y" читается как "X зашифровано ключом K_Y" и

БИЛЕТ = {клиент, сервер, время_начала, время_жизни, K_сеанса}K_сервера

в Kerberos V5 ОТВЕТ_KDC теперь будет выглядеть так:

ОТВЕТ_KDC* = БИЛЕТ, {клиент, сервер, K_сеанса}K_пользователя

*(*KDC - Key Distribution Center (Центр Выдачи Ключей) - прим. переводчика)*

Конечно, в Kerberos V5 также много нововведений. Пользователи теперь могут безопасно пересылать свои билеты, благодаря чему они могут быть использованы из другого места в сети; дополнительно, пользователи также могут делегировать подмножество своих полномочий серверу, чтобы этот сервер мог выступать в качестве уполномоченного представителя пользователя (proxy). Другие новшества включают в себя возможность заменить DES более безопасным алгоритмом шифрования, таким как тройной DES (triple-DES). Читатели, которым интересно больше узнать о различиях между Kerberos V4 и V5, приглашаются к прочтению Эволюции Системы Аутентификации Kerberos (The Evolution of the Kerberos Authentication System), написанной Клиффом Ньюманом (Cliff Neumann) и Теодором Тсо (Theodore Ts'o).

Надеюсь, вам понравилось это краткое введение в протокол Kerberos. Желаю удачи в его дальнейшем изучении!

Теодор Тсо (Theodore Ts'o), Февраль 1997.