ОБРПО. Лекция 11 Пользовательская аутентификация

ТЮРИН КАЙ АНДРЕЕВИЧ

Аутентификация

Аутентификация — это процесс проверки подлинности чего-либо— это процесс проверки подлинности чего-либо

В случае с пользователями, это доказательство, что я – это тот, за кого я себя выдаю.

Самой распространённой системой аутентификации на сегодняшний день является проверка пароля.

Есть ли альтернатива паролям?

Идея с паролями выглядит очень плохой с самого начала:

- Низкая энтропия – их легко угадать
- Вопросы для восстановления паролей имеют ещё меньше энтропии

Информацио́нная энтропи́я — мера неопределённости некоторой системы, в частности, непредсказуемость появления какого-либо символа первичного алфавита.

Что такое вообще пароль?

Пароль – секрет, разделённый между пользователем и сервером

В простейшем реализация проверки пароля является таблицей соответствия «логин-пароль»

- Проблема: можно украсть базу

Решение: база логин-хеш пароля

- Проблема: атака предварительного вычисления

У паролей искажённое распределение

5000 паролей покрывают 20% пользователей

У пароля от 10 до 20 бит энтропии

Классические хеши оптимизированны, поэтому считаются быстро (2 миллиона SHA1 на ноутбуке, то есть 1 пароль в секунду при условии 20 бит энтропии)

Решение проблемы с утечкой базы

Дорогие по вычислению функции:

- Раньше рекомендовались PBKDF2, Bcrypt
- Сегодня рекомендуются scrypt, argon2

Тем не менее, это не спасает от использования радужных таблиц*

*Радужные таблицы – таблицы поиска для обращения криптографических хешей, оптимизирующие количество используемой памяти.

Решение проблемы с обращением хешей

Решение: соль (и перец!)

Нельзя построить радужную таблицу

Пароли в базе разные, даже если они одинаковые

Соль подлиннее, менять соль при смене пароля (потому что люди используют одинаковые пароли)

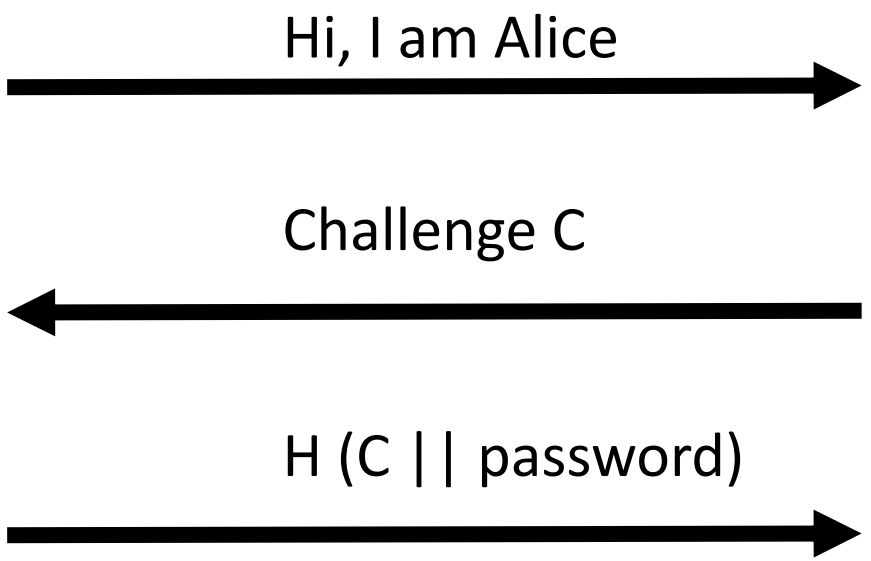
Но проблемы остаются

Проблема: передача пароля

1. Посылать в открытом виде
   - Понятно, почему это плохо
2. Чуть лучше: посылать пароль через зашифрованное соединение
   - Не защищает от MITM
3. Что если посылать хэш?
   - Хеш становится паролем, то есть атакующий может перехватить и посылать его

Что с этим делать?

Challenge/response protocol



Плюсы

Если сервер обманывает клиента, то он не получает пароля

Как защититься от перебора со стороны сервера?

- Дорогое хэширование + соль
- Позволить клиенту вносить свою случайность

Как предотвратить брутфорс?

Брутфорс – перебор пароля

Ограничение количества попыток

Таймауты между попытками (и группами попыток)

Уязвимы ли пароли к офлайн-атакам?

Например, система может возвращать что-то, зашифрованное паролем клиента. В таком случае атакующий может попробовать перебрать пароль посредством расшифровки данных и проверки их целостности

Такая ситуация была в Kerberos v4 (сетевой протокол аутентификации, который предлагает механизм взаимной аутентификации клиента и сервера перед установлением связи между ними)

Вопросы для восстановления пароля

О вопросах для восстановления думают меньше, чем о пароле

Вопросы для восстановления пароля часто

1. Имеют малую энтропию по умолчанию:
   - Любимый цвет
   - Имя лучшего друга
2. Утекают посредством социальных сетей:
   - Любимый фильм
3. Сгенерированные пользователем вопросы могут быть простыми
   - 5+5

Параметры оценки схем аутентификации

Статья «The Quest to Replace Passwords: A Framework for Comparative of Web Authentication Schemes» предлагает ряд параметров для оценки схем аутентификации

Параметры разделяются на три группы:

- Использование
- Возможность развёртывания
- Безопасность

Требования по Использованию

Простота изучения – люди, которые не знают схемы, должны легко понять её

Малая вероятность ошибки – легальный пользователь должен легко войти в систему

Масштабирование для пользователей – добавление нового ресурса не должно усложнять жизнь пользователю

Простота восстановления

Отсутствие чего-то, что нужно иметь (Nothing-to-Carry)

Возможность развёртывания

Совместимость с серверами

Совместимость с браузерами

Доступность – могут ли пользователи, использующие систему сейчас, использовать систему в будущем в различных условиях

Это важное свойство, потому что заставить пользователей и сервера массово обновиться – тяжело!

Безопасность

Устойчивость к физическому наблюдению

Устойчивость к целевой атаке

Устойчивость к ограниченному угадыванию

Устойчивость к неограниченному угадыванию

Устойчивость к внутреннему наблюдению – кейлогеры, сетевые снифферы…

Устойчивость к фишингу

Отсутствие доверенной третьей стороны

Устойчивость к утечкам от других ресурсов

Биометрия

Каковы размеры пространств?

- Отпечатки пальцев – 13.3 бита
- Сетчатка – 19.9 бит
- Распознавание голоса – 11.7 бит

То есть биометрия близка по устойчивости к паролям

Биометрия vs. Пароли: Использование

	Пароли	Биометрия
Простота изучения	+	+
Малая вероятность ошибки	+/-	-
Масштабирование для пользователей	-	+
Простота восстановления	+	-
Отсутствие чего-то, что нужно иметь	+	+

Биометрия vs. Пароли: Развёртывание

	Пароли	Биометрия
Совместимость с серверами	+	-
Совместимость с браузерами	+	-
Доступность	+	+/-

Биометрия vs. Пароли: Безопасность

	Пароли	Биометрия
Устойчивость к физическому наблюдению	-	+
Устойчивость к целевой атаке	+/-	-
Устойчивость к ограниченному угадыванию	-	+
Устойчивость к неограниченному угадыванию	-	-
Устойчивость к внутреннему наблюдению	-	-
Устойчивость к фишингу	-	-
Отсутствие доверенной третьей стороны	+	+
Устойчивость к утечкам от других ресурсов	-	-

Многофакторная аутентификация

Факторы:

- На основе знания
- На основе наличия
- На основе существования

Классическая многофакторная аутентификация – пароль и телефон

Многофакторная аутентификация – хорошая идея, но практика показывает, что люди ставят более простые пароли при её использовании!

PAKE

Password Authenticated Key Exchange

Шикарная штука, но почти не используемая

Позволяет проверять корректность пароля без пересылки его на сервер!

Диффи-Хеллман – протокол обмена ключами, но без аутентификации.

PAKE существуют с 1992 года, но почти не имеют распространения.

По двум причинам:

1. Мало хороших реализаций

2. Криптографы плохо умеют объяснять результаты своей работы, поэтому многие не знают про PAKE

SRP

Secure Remote Password

Разработан в 1998 году.

Стандартизован в TLS и реализован в OpenSSL, хоть его никто и не использует

Apple использует SRP в iCloud Key Vault.

Последний факт делает SRP одним из самых распространённых криптографических протоколов в мире!

Общая идея SRP

Имея параметр g (g = генератор по модулю N => для любого 0 < X < N существует и единственный x такой, что g^x % N = X) пользователь вычисляет

v = g^H(salt, password) % N

И присылает серверу логин, соль и полученный v (verifier)

Проверка пароля (1)

Пользователь присылает серверу логин, и

A = g^a % N, где a – случайное число

Сервер находит в базе два числа, соответствующих логину: соль (s) и верификатор пароля (v) и вычисляет

B=(k*v + g^b % N) % N, где b – случайное число

Клиенту присылается пара из соли s и вычисленного B.

Обе стороны вычисляют скремблер:

u = H(A, B).

Проверка пароля (2)

Клиент вводит свой пароль p, на основе которого вычисляется общий ключ сессии:

x = H(s, p)

S = ((B - k*(g^x % N)) ^ (a + u*x)) % N

K = H(S)

Сервер со своей стороны так же вычисляет общий ключ сессии:

S = ((A*(v^u % N)) ^ b) % N

K = H(S)

Начиная с этого момента, если клиент и сервер оба _действительно_ знают друг друга, есть идентичное число K у обоих.

Что вообще можно сказать про SRP?

1. SRP не требует хранить пароль в чистом виде, и это хорошо. Вместо этого нужно хранить «валидатор» – одностороннюю функцию от хеша пароля.

2. Текущая версия SRP (v6a) ещё не взломана

3. Но старые версии были взломаны. И доказательство безопасности SRP не доказывает ничего полезного.

4. SRP основана на целочисленной арифметике и по разным причинам плохо переносится на эллиптические кривые.

5. SRP уязвима к атакам предварительного вычисления.

6. SRP достаточно проста в использовании.

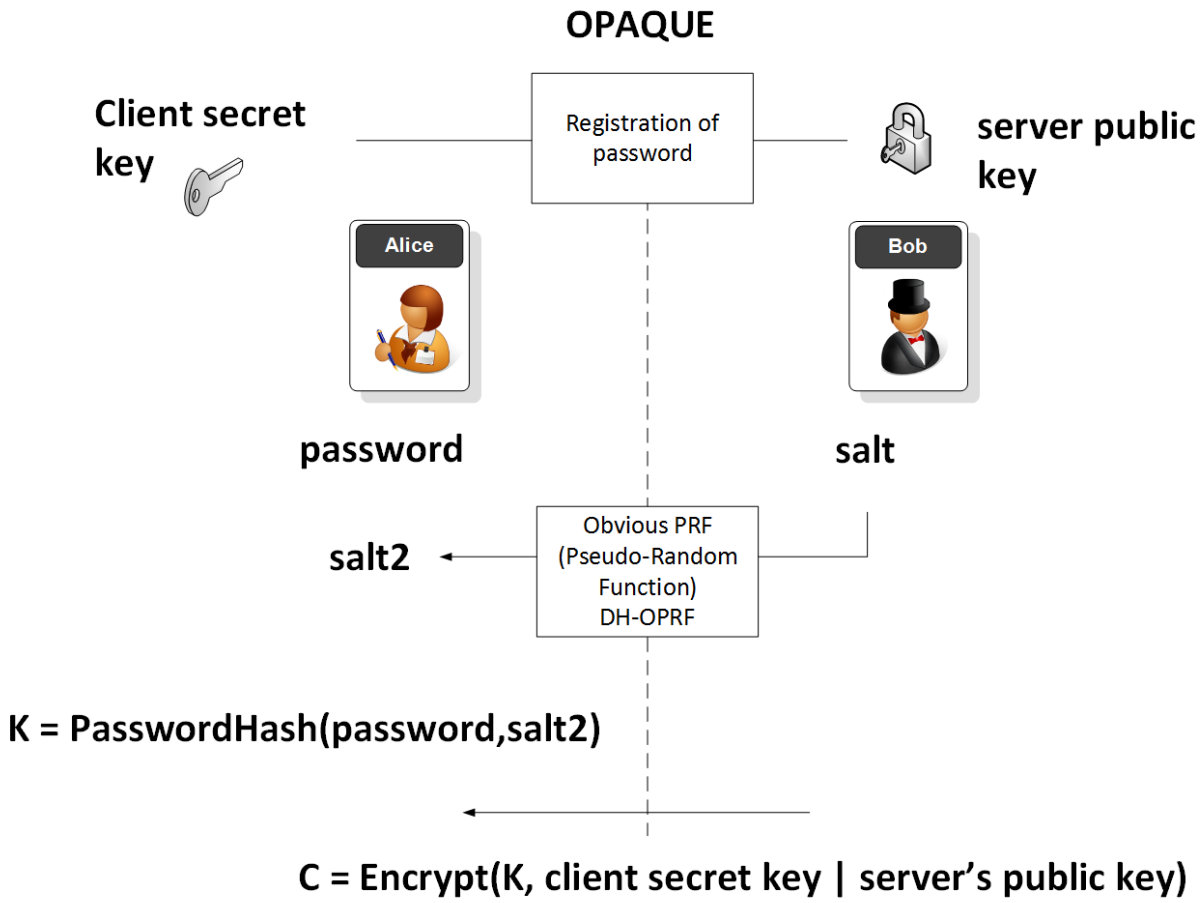
OPAQUE

OPAQUE – схема аутентификации, предложенная в 2018 году

На данный момент не имеет распространения, потому что почти нет реализаций.

Позволяет хранить не передавать пароль на сервер, а соль на клиент.

OPAQUE


OPAQUE
Client secret key
Registration of password
server public key
Alice
Bob
password
salt
Obvious PRF (Pseudo-Random Function) DH-OPRF
salt2
K = PasswordHash(password,salt2)
C = Encrypt(K, client secret key | server's public key)

Итоги

Парольная аутентификация всё ещё остаётся одной из ведущих.

И с аутентификацией всё не так просто.

Следует использовать многофакторную аутентификацию и OPAQUE/SRP для хранения и проверки пароля.